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?
- ¿Cómo crear un repositorio en GitHub?
- ¿Cómo crear una rama en Git?
- ¿Cómo cambiar a una rama en Git?
- ¿Cómo fusionar ramas en Git?
- ¿Cómo crear un commit en Git?
- ¿Cómo enviar un commit a GitHub?
- ¿Qué es un repositorio remoto?
- ¿Cómo agregar un repositorio remoto a Git?
- ¿Cómo empujar cambios a un repositorio remoto?
- ¿Cómo tirar de cambios de un repositorio remoto?
- ¿Qué es un fork de repositorio?
- ¿Cómo crear un fork de un repositorio?

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- ¿Cómo aceptar una solicitud de extracción?
- ¿Qué es una etiqueta en Git?
- ¿Cómo crear una etiqueta en Git?
- ¿Cómo enviar una etiqueta a GitHub?
- ¿Qué es un historial de Git?
- ¿Cómo ver el historial de Git?
- ¿Cómo buscar en el historial de Git?
  - ¿Cómo borrar el historial de Git?
- ¿Qué es un repositorio privado en GitHub?
- ¿Cómo crear un repositorio privado en GitHub?
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
- ¿Qué es un repositorio público en GitHub?
- ¿Cómo crear un repositorio público en GitHub?
- ¿Cómo compartir un repositorio público 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.

Respuestas de las 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 que permite a los usuarios almacenar, administrar y colaborar en proyectos de software. A continuación, se presentan algunos aspectos clave de GitHub:

Control de versiones: GitHub utiliza Git, un sistema de control de versiones que permite realizar un seguimiento de los cambios en el código a lo largo del tiempo. Esto facilita la colaboración, ya que varios desarrolladores pueden trabajar en el mismo proyecto sin conflictos.

Repositorios: Los proyectos de software se almacenan en repositorios, que son directorios que contienen todos los archivos y el historial de cambios del proyecto.

Colaboración: GitHub facilita la colaboración a través de funciones como solicitudes de extracción (pull requests), que permiten a los desarrolladores proponer cambios en el código y discutirlos antes de fusionarlos.

Seguimiento de problemas: GitHub ofrece herramientas para realizar un seguimiento de los errores y las mejoras pendientes en un proyecto.

Alojamiento de código: GitHub proporciona un lugar centralizado para alojar el código, lo que facilita el acceso y la distribución del software.

★ ¿Cómo crear un repositorio en GitHub?

1. Creación de un repositorio a través de la interfaz web:

- **Paso 1: Inicia sesión en GitHub.**
  - Dirígete a la página web de GitHub (github.com) e inicia sesión con tu cuenta.

- **Paso 2: Accede a la opción "Nuevo repositorio".**
    - En la esquina superior derecha de cualquier página, haz clic en el icono "+" y selecciona "Nuevo repositorio".

- **Paso 3: Configura el repositorio.**
    - **Nombre del repositorio:** Escribe un nombre corto y descriptivo para tu repositorio.
    - **Descripción (opcional):** Agrega una descripción que explique el propósito del repositorio.
    - **Visibilidad:** Elige si quieres que el repositorio sea público (visible para todos) o privado (visible solo para ti y las personas que invites).
    - **Inicializar con un README (opcional):** Marca esta casilla si deseas crear un archivo README.md inicial. Este archivo se utiliza para proporcionar información sobre tu proyecto.
    - **Añadir .gitignore (opcional):** Puedes seleccionar un archivo .gitignore para especificar qué archivos y carpetas deben ignorarse en el control de versiones.
    - **Elegir una licencia (opcional):** Selecciona una licencia para tu proyecto. Esto define cómo otros pueden usar tu código.

- **Paso 4: Crea el repositorio.**
    - Haz clic en el botón "Crear repositorio".

¿Cómo crear una rama en Git?

Usando la línea de comandos de Git:

- **Paso 1: Asegúrate de estar en la rama correcta.**
    - Antes de crear una nueva rama, es importante que estés en la rama desde la cual quieres ramificar. Por lo general, esto es la rama principal (normalmente llamada "main" o "master"). Para verificar en qué rama estás, usa el comando:
        - `git status`
    - Si necesitas cambiar a otra rama, usa el comando:
        - `git checkout <nombre-de-la-rama>`

- **Paso 2: Crea la nueva rama.**
    - Para crear una nueva rama, usa el comando `git branch` seguido del nombre que quieres darle a la rama:
        - `git branch <nombre-de-la-nueva-rama>`
    - Este comando crea la rama, pero no te cambia a ella.

- **Paso 3: Cambia a la nueva rama.**
  - Para cambiar a la nueva rama que acabas de crear, usa el comando `git checkout`:
    - `git checkout <nombre-de-la-nueva-rama>`
  - También puedes combinar los pasos 2 y 3 en un solo comando usando la opción `-b`:
    - `git checkout -b <nombre-de-la-nueva-rama>`

- **Paso 4: Empuja la rama al repositorio remoto (opcional).**
  - Si deseas que la rama sea visible para otros colaboradores en un repositorio remoto (como GitHub), deberás "empujarla" (push) al repositorio remoto. Usa el siguiente comando:
    - `git push -u origin <nombre-de-la-nueva-rama>`
    - La opción `-u` establece una relación de seguimiento entre tu rama local y la rama remota, lo que facilita futuras operaciones de `push` y `pull`.

¿Cómo cambiar a una rama en Git?

Usando la línea de comandos de Git:

- **Paso 1: Abre la línea de comandos.**
  - Abre la terminal o la línea de comandos en tu ordenador.
- **Paso 2: Cambia a la rama deseada.**
  - Utiliza el siguiente comando, reemplazando `<nombre-de-la-rama>` con el nombre de la rama a la que deseas cambiar:
    - `git checkout <nombre-de-la-rama>`
  - Por ejemplo, si quieres cambiar a una rama llamada "desarrollo", el comando sería:
    - `git checkout desarrollo`

¿Cómo fusionar ramas en Git?

1. Cambiar a la rama de destino:

- Primero, debes cambiar a la rama en la que deseas fusionar los cambios. Esta suele ser la rama principal (normalmente "main" o "master").
  - Utiliza el comando: `git checkout <rama_destino>`
  - Ejemplo: `git checkout main`

2. Fusionar la rama de origen:

- A continuación, utiliza el comando `git merge` para fusionar la rama de origen en la rama de destino.
    - Utiliza el comando: `git merge <rama_origen>`
    - Ejemplo: `git merge feature/nueva-funcionalidad`

3. Resolver conflictos (si los hay):

- Si hay cambios conflictivos entre las dos ramas, Git te pedirá que los resuelvas manualmente.
    - Git marcará los archivos con conflictos. Deberás editar estos archivos para elegir qué cambios conservar.
    - Después de resolver los conflictos, utiliza `git add <archivos_conflictivos>` para marcar los archivos como resueltos.
    - Finalmente, utiliza `git commit` para completar la fusión.

4. Empujar los cambios (opcional):

- Si estás trabajando en un repositorio remoto (como GitHub), deberás empujar los cambios fusionados al repositorio remoto.
    - Utiliza el comando: `git push origin <rama_destino>`
    - Ejemplo: `git push origin main`

¿Cómo crear un commit en Git?

1. Preparar los cambios (staging):

- Antes de crear un commit, debes indicar a Git qué cambios quieres incluir en él. Esto se hace utilizando el comando `git add`.
    - Para añadir un archivo específico, utiliza: `git add <nombre_del_archivo>`
    - Para añadir todos los cambios en el directorio actual, utiliza: `git add .`

2. Crear el commit:

- Una vez que hayas preparado los cambios, utiliza el comando `git commit` para crear el commit.
    - Es muy importante añadir un mensaje descriptivo al commit para explicar los cambios que has realizado. Puedes hacerlo de dos maneras:
        - `git commit -m "Mensaje del commit"`: Esta opción te permite escribir el mensaje directamente en la línea de comandos.
        - `git commit`: Si utilizas este comando sin la opción `-m`, Git abrirá un editor de texto donde podrás escribir un mensaje de commit más extenso.

3. Consideraciones importantes:

- **Mensajes de commit claros y concisos:** Escribir buenos mensajes de commit es crucial para mantener un historial de cambios claro y fácil de entender. Intenta ser lo más descriptivo posible sobre los cambios que has realizado.
- **Revisar los cambios preparados:** Antes de crear un commit, puedes utilizar el comando `git status` para revisar los cambios que has preparado.
- **Evitar commits grandes:** Es recomendable crear commits pequeños y enfocados en cambios específicos. Esto facilita la revisión y el seguimiento de los cambios.

⋆ ¿Cómo enviar un commit a GitHub?

1. Verifica que tu repositorio local esté configurado con un repositorio remoto:

- Utiliza el comando `git remote -v` para verificar si tienes un repositorio remoto configurado. Deberías ver una lista de los remotos y sus URLs.
- Si no tienes un remoto configurado, puedes añadirlo con el comando: `git remote add origin <URL_del_repositorio_de_GitHub>`. Reemplaza `<URL_del_repositorio_de_GitHub>` con la URL de tu repositorio en GitHub.

2. Asegúrate de que tus cambios estén confirmados (commited):

- Antes de enviar los cambios, debes confirmarlos localmente con el comando `git commit`. Si no lo has hecho, realiza los pasos necesarios para crear el commit.

3. Envía los commits a GitHub:

- Utiliza el comando `git push origin <nombre_de_la_rama>` para enviar los commits a la rama correspondiente en GitHub.
    - `origin` es el nombre del repositorio remoto (generalmente se usa "origin" por defecto).
    - `<nombre_de_la_rama>` es el nombre de la rama que quieres enviar (por ejemplo, "main" o "develop").
- Si es la primera vez que envías una rama nueva, puedes usar `git push -u origin <nombre_de_la_rama>`. La opción `-u` establece una relación de seguimiento entre tu rama local y la rama remota, lo que facilita futuras operaciones de `push` y `pull`.

4. Consideraciones importantes:

- **Autenticación:** Es posible que GitHub te pida que te autentiques antes de permitirte enviar los cambios. Esto puede implicar ingresar tu nombre de usuario y contraseña o utilizar un token de acceso personal.
- **Conflictos:** Si otros colaboradores han realizado cambios en el repositorio remoto que entran en conflicto con tus cambios locales, Git te impedirá enviar tus commits. Deberás resolver los conflictos localmente antes de poder enviar los cambios.
- **Actualizar tu repositorio local:** Antes de enviar tus commits, es recomendable actualizar tu repositorio local con los últimos cambios del repositorio remoto utilizando el comando `git pull origin <nombre_de_la_rama>`. Esto ayuda a evitar conflictos.

★ ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de tu proyecto Git que está alojada en un servidor externo, generalmente en Internet. A diferencia de tu repositorio local, que reside en tu computadora, el repositorio remoto sirve como un punto centralizado para que los desarrolladores colaboren y compartan código.

Funciones principales de un repositorio remoto:

- **Colaboración:**
  - Permite que varios desarrolladores trabajen en el mismo proyecto simultáneamente.
  - Facilita el intercambio de código y la revisión de cambios.
- **Copia de seguridad:**
  - Actúa como una copia de seguridad del código, protegiéndolo de la pérdida de datos en caso de fallas en la computadora local.
- **Distribución de código:**
  - Permite compartir el código con otros desarrolladores o con el público en general.
- **Control de versiones:**
  - Al igual que un repositorio local, un repositorio remoto mantiene un historial de todos los cambios realizados en el código.

★ ¿Cómo agregar un repositorio remoto a Git?

1. Verifica si ya tienes un repositorio remoto configurado:

- Antes de agregar un nuevo repositorio remoto, es útil verificar si ya tienes alguno configurado. Puedes hacerlo con el siguiente comando:
  - `git remote -v`
- Este comando mostrará una lista de los repositorios remotos configurados y sus URLs. Si no ves nada, significa que aún no tienes ningún repositorio remoto configurado.

2. Agrega el repositorio remoto:

- Para agregar un nuevo repositorio remoto, utiliza el comando `git remote add`. La sintaxis es la siguiente:
  - `git remote add <nombre_del_remoto> <URL_del_repositorio_remoto>`
- Donde:
  - `<nombre_del_remoto>`: Es el nombre que le darás al repositorio remoto. Por convención, se suele utilizar "origin", pero puedes usar cualquier nombre.
  - `<URL_del_repositorio_remoto>`: Es la URL del repositorio remoto. Esta URL puede ser SSH o HTTPS.
- Ejemplo:
  - `git remote add origin https://github.com/usuario/repositorio.git`

3. Verifica que el repositorio remoto se haya agregado correctamente:

- Después de agregar el repositorio remoto, puedes verificar que se haya configurado correctamente utilizando el comando `git remote -v` nuevamente. Deberías ver el nombre del remoto y su URL en la lista.

★ ¿Cómo empujar cambios a un repositorio remoto?

1. Verifica que tu repositorio local esté configurado con un repositorio remoto:

- Utiliza el comando `git remote -v` para verificar si tienes un repositorio remoto configurado. Deberías ver una lista de los remotos y sus URLs.
- Si no tienes un remoto configurado, puedes añadirlo con el comando: `git remote add origin <URL_del_repositorio_de_GitHub>`. Reemplaza `<URL_del_repositorio_de_GitHub>` con la URL de tu repositorio en GitHub.

2. Asegúrate de que tus cambios estén confirmados (commited):

- Antes de enviar los cambios, debes confirmarlos localmente con el comando `git commit`. Si no lo has hecho, realiza los pasos necesarios para crear el commit.

3. Envía los commits a GitHub:

- Utiliza el comando `git push origin <nombre_de_la_rama>` para enviar los commits a la rama correspondiente en GitHub.
  - `origin` es el nombre del repositorio remoto (generalmente se usa "origin" por defecto).
  - `<nombre_de_la-rama>` es el nombre de la rama que quieres enviar (por ejemplo, "main" o "develop").

- Si es la primera vez que envías una rama nueva, puedes usar `git push -u origin <nombre_de_la-rama>`. La opción `-u` establece una relación de seguimiento entre tu rama local y la rama remota, lo que facilita futuras operaciones de `push` y `pull`.

4. Consideraciones importantes:

- **Autenticación:** Es posible que GitHub te pida que te autentiques antes de permitirte enviar los cambios. Esto puede implicar ingresar tu nombre de usuario y contraseña o utilizar un token de acceso personal.
- **Conflictos:** Si otros colaboradores han realizado cambios en el repositorio remoto que entran en conflicto con tus cambios locales, Git te impedirá enviar tus commits. Deberás resolver los conflictos localmente antes de poder enviar los cambios.
- **Actualizar tu repositorio local:** Antes de enviar tus commits, es recomendable actualizar tu repositorio local con los últimos cambios del repositorio remoto utilizando el comando `git pull origin <nombre-de-la-rama>`. Esto ayuda a evitar conflictos.

★ ¿Cómo tirar de cambios de un repositorio remoto?

1. Usando `git fetch`:

- El comando `git fetch` descarga los cambios del repositorio remoto pero no los fusiona automáticamente con tu rama local.
- Esto te permite revisar los cambios antes de fusionarlos.
- Para usar `git fetch`, simplemente ejecuta el siguiente comando:
  - `git fetch origin`
    - `origin` es el nombre del repositorio remoto (generalmente se usa "origin" por defecto).
- Después de ejecutar `git fetch`, puedes revisar los cambios utilizando el comando `git log origin/<nombre-de-la-rama>`.
- Para fusionar los cambios con tu rama local, puedes utilizar el comando `git merge origin/<nombre-de-la-rama>`.

2. Usando `git pull`:

- El comando `git pull` descarga los cambios del repositorio remoto y los fusiona automáticamente con tu rama local.
- Es una forma más rápida de actualizar tu repositorio local, pero puede generar conflictos si hay cambios conflictivos entre tu rama local y la rama remota.
- Para usar `git pull`, ejecuta el siguiente comando:
  - `git pull origin <nombre-de-la-rama>`
    - `origin` es el nombre del repositorio remoto.
    - `<nombre-de-la-rama>` es el nombre de la rama que quieres actualizar.

★ ¿Qué es un fork de repositorio?

En el contexto de Git y plataformas como GitHub, un "fork" de un repositorio se refiere a la creación de una copia personal de ese repositorio en tu propia cuenta. Es como hacer una bifurcación del proyecto original, lo que te permite trabajar en él de forma independiente.

★ ¿Cómo crear un fork de un repositorio?

Crear un fork de un repositorio en GitHub es un proceso sencillo que te permite tener una copia personal del repositorio en tu propia cuenta. Aquí te explico cómo hacerlo:

Pasos para crear un fork:

1. Navega al repositorio:
    - Ve a la página del repositorio que deseas forkar en GitHub.
2. Haz clic en el botón "Fork":
    - En la esquina superior derecha de la página del repositorio, encontrarás un botón que dice "Fork". Haz clic en él.
3. Selecciona tu cuenta:
    - GitHub te preguntará dónde deseas crear el fork. Selecciona tu cuenta de usuario personal o una organización en la que tengas permisos para crear repositorios.
4. Espera a que se cree el fork:
    - GitHub creará una copia del repositorio en tu cuenta. Este proceso puede tardar unos segundos.
5. ¡Listo!
    - Ahora tienes una copia del repositorio en tu cuenta. Puedes acceder a ella y realizar cambios sin afectar el repositorio original.

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

Enviar una solicitud de extracción (pull request) a un repositorio es un proceso fundamental para colaborar en proyectos de código abierto o en equipos de desarrollo que utilizan Git. Aquí te explico los pasos para hacerlo:

1. Haz un fork del repositorio (si no tienes permisos de escritura):

- Si no eres colaborador directo del repositorio al que quieres contribuir, debes hacer un fork. Esto crea una copia del repositorio en tu propia cuenta de GitHub.
- Ve a la página del repositorio original y haz clic en el botón "Fork".

2. Clona tu fork localmente:

- Clona el repositorio que has forkado a tu computadora usando el comando `git clone <URL_de_tu_fork>`.

3. Crea una rama para tus cambios:

- Crea una nueva rama para tus cambios usando el comando `git checkout -b <nombre-de-tu-rama>`. Utiliza un nombre descriptivo para la rama, como `feature/nueva-funcionalidad` o `bugfix/arreglar-error-de-login`.

4. Realiza tus cambios:

- Realiza los cambios que deseas aportar al proyecto.

5. Confirma tus cambios (commits):

- Prepara tus cambios para el commit usando `git add <archivos-modificados>`.
- Crea un commit con un mensaje descriptivo usando `git commit -m "Mensaje descriptivo de los cambios"`.

6. Envía tu rama a tu repositorio en GitHub:

- Envía tu rama a tu repositorio remoto en GitHub usando `git push origin <nombre-de-tu-rama>`.

7. Crea la solicitud de extracción (pull request) en GitHub:

- Ve a tu repositorio en GitHub.
- GitHub debería mostrar un aviso de que tu rama ha sido enviada y ofrecerte la opción de "Comparar y pull request". Haz clic en ese botón.
- Si no ves el aviso, puedes ir a la pestaña "Pull requests" y hacer clic en "New pull request".
- Selecciona la rama en tu repositorio que quieres fusionar y la rama en el repositorio original a la que quieres enviar tus cambios.
- Escribe un título y una descripción clara para tu pull request. Explica los cambios que has realizado y por qué son necesarios.
- Haz clic en "Create pull request".

★ ¿Cómo aceptar una solicitud de extracción?

Aceptar una solicitud de extracción (pull request) en GitHub implica revisar los cambios propuestos y fusionarlos en la rama principal del repositorio. Aquí te explico los pasos para hacerlo:

1. Revisa la solicitud de extracción:

- **Ve a la pestaña "Pull requests"**: En el repositorio de GitHub, haz clic en la pestaña "Pull requests".
- **Selecciona la solicitud de extracción**: Haz clic en la solicitud de extracción que deseas revisar.

- **Revisa los cambios**:
  - Lee la descripción de la solicitud de extracción para comprender los cambios propuestos.
  - Haz clic en la pestaña "Files changed" para revisar los cambios de código.
  - Revisa los commits individuales para comprender la progresión de los cambios.
  - Asegúrate de que los cambios sigan las pautas de codificación y las mejores prácticas del proyecto.
- **Deja comentarios (opcional)**: Si tienes comentarios o sugerencias, deja comentarios en líneas de código específicas o en la conversación general de la solicitud de extracción.

2. Aprueba la solicitud de extracción:

- **Haz clic en "Review changes"**: En la parte superior de la pestaña "Files changed", haz clic en el botón "Review changes".
- **Selecciona "Approve"**: Selecciona la opción "Approve" para indicar que apruebas los cambios.
- **Agrega un comentario (opcional)**: Agrega un comentario para explicar tu aprobación o para proporcionar comentarios adicionales.
- **Haz clic en "Submit review"**: Haz clic en el botón "Submit review" para enviar tu aprobación.

3. Fusiona la solicitud de extracción:

- **Verifica los conflictos**: Antes de fusionar, asegúrate de que no haya conflictos entre la rama de la solicitud de extracción y la rama principal. Si hay conflictos, deberás resolverlos.
- **Haz clic en "Merge pull request"**: Si no hay conflictos, haz clic en el botón "Merge pull request".
- **Confirma la fusión**: Confirma la fusión haciendo clic en el botón "Confirm merge".
- **Elimina la rama (opcional)**: Después de fusionar la solicitud de extracción, puedes eliminar la rama de la solicitud de extracción haciendo clic en el botón "Delete branch".

★ ¿Qué es un etiqueta en Git?

En Git, una etiqueta (tag) es una instantánea estática de un punto específico en el historial de tu repositorio. A diferencia de las ramas, que están diseñadas para evolucionar con nuevos commits, las etiquetas son inmutables y se utilizan para marcar versiones de lanzamiento, puntos de referencia importantes o cualquier otro momento significativo en el desarrollo de tu proyecto.

★ ¿Cómo crear una etiqueta en Git?

1. Tipos de etiquetas:

Git admite dos tipos de etiquetas:

- **Etiquetas ligeras (lightweight tags):** Son simples punteros a un commit específico.
- **Etiquetas anotadas (annotated tags):** Almacenan información adicional, como el nombre del etiquetador, la fecha y un mensaje.

2. Crear una etiqueta ligera:

Para crear una etiqueta ligera, utiliza el siguiente comando:

Bash

git tag <nombre-de-la-etiqueta>

Por ejemplo:

Bash

git tag v1.0

3. Crear una etiqueta anotada:

Para crear una etiqueta anotada, utiliza el siguiente comando:

Bash

git tag -a <nombre-de-la-etiqueta> -m "<mensaje>"

Por ejemplo:

Bash

git tag -a v1.0 -m "Versión de lanzamiento 1.0"

4. Etiquetar un commit anterior:

Si deseas etiquetar un commit anterior, puedes especificar el hash del commit:

Bash

git tag -a <nombre-de-la-etiqueta> <hash-del-commit> -m "<mensaje>"

★ ¿Cómo crear una etiqueta en Git?

5. Enviar etiquetas al repositorio remoto:

Las etiquetas no se envían automáticamente al repositorio remoto. Debes enviarlas explícitamente utilizando el siguiente comando:

Bash

git push origin <nombre-de-la-etiqueta>

Para enviar todas las etiquetas:

Bash

git push origin --tags

6. Ver etiquetas:

Para ver la lista de etiquetas en tu repositorio, utiliza el siguiente comando:

Bash

git tag

Para ver información sobre una etiqueta específica:

Bash

git show <nombre-de-la-etiqueta>

★ ¿Qué es un historial de Git?

En Git, el "historial" se refiere al registro completo de todos los cambios que se han realizado en un repositorio a lo largo del tiempo. Este historial es una de las características más poderosas de Git, ya que permite a los desarrolladores rastrear, revertir y comprender la evolución de su código.

★ ¿Cómo ver el historial de Git?

Para ver el historial de Git, puedes utilizar el comando `git log`. Este comando muestra una lista de los commits realizados en el repositorio, ordenados del más reciente al más antiguo. Aquí te explico cómo usarlo y algunas opciones útiles:

Comando básico:

- El comando `git log` por sí solo muestra el historial de commits con información como:
  - El hash del commit (identificador único).
  - El autor del commit.
  - La fecha y hora del commit.
  - El mensaje del commit.

★ ¿Cómo buscar en el historial de Git?

Git ofrece varias formas de buscar en el historial de un repositorio, lo que te permite encontrar commits específicos, cambios de archivos o patrones de texto. Aquí te presento algunas de las técnicas más útiles:

1. Buscar commits por mensaje, autor o fecha:

- **`git log --grep="patrón"`**:
  - Busca commits cuyos mensajes contengan el "patrón" especificado.
  - Ejemplo: `git log --grep="arreglar error de login"`
- **`git log --author="nombre"`**:
  - Filtra los commits por el autor especificado.
  - Ejemplo: `git log --author="Juan Pérez"`
- **`git log --since="fecha"`** y **`git log --until="fecha"`**:
  - Filtran los commits por un rango de fechas.
  - Ejemplo: `git log --since="2023-01-01" --until="2023-03-15"`
- **`git log -S"cadena"`**:
  - Busca los cambios que incluyan esa cadena de caracteres.
  - Ejemplo: `git log -S"contraseña"`

★ ¿Cómo borrar el historial de Git?

Borrar el historial de Git puede ser un proceso complejo y potencialmente peligroso, especialmente si estás trabajando en un repositorio compartido. Es importante comprender las implicaciones antes de proceder. Aquí te explico las diferentes formas de borrar el historial de Git, desde eliminar commits recientes hasta reescribir el historial completo.

1. Eliminar commits recientes:

- **`git reset --soft HEAD~n`**:
  - Esta opción revierte el HEAD a "n" commits anteriores, pero conserva los cambios en el área de preparación. Esto significa que puedes modificar los commits y volver a confirmarlos.
  - Ejemplo: `git reset --soft HEAD~2` (revierte los últimos 2 commits).
- **`git reset --mixed HEAD~n`**:
  - Esta opción revierte el HEAD a "n" commits anteriores y elimina los cambios del área de preparación. Tendrás que volver a preparar los cambios antes de volver a confirmarlos.
  - Esta es la opción por defecto.
- **`git reset --hard HEAD~n`**:
  - Esta opción revierte el HEAD a "n" commits anteriores y elimina todos los cambios, tanto del área de preparación como del directorio de trabajo. ¡Esta opción es peligrosa y debe usarse con precaución!

★ ¿Cómo borrar el historial de Git?

Un repositorio privado en GitHub es un repositorio de código al que solo pueden acceder los propietarios y colaboradores explícitamente invitados. A diferencia de los repositorios públicos, que son visibles para cualquier persona en Internet, los repositorios privados ofrecen un mayor control sobre quién puede ver y modificar el código.

★ ¿Cómo crear un repositorio privado en GitHub?

Crear un repositorio privado en GitHub es un proceso sencillo que te permite mantener tu código y proyectos protegidos. Aquí te explico los pasos para hacerlo:

1. Inicia sesión en GitHub:

- Dirígete a la página web de GitHub (github.com) e inicia sesión con tu cuenta.

2. Accede a la opción "Nuevo repositorio":

- En la esquina superior derecha de cualquier página, haz clic en el icono "+" y selecciona "Nuevo repositorio".

3. Configura el repositorio:

- **Nombre del repositorio:** Escribe un nombre corto y descriptivo para tu repositorio.
- **Descripción (opcional):** Agrega una descripción que explique el propósito del repositorio.
- **Visibilidad:** Aquí es donde eliges si quieres que el repositorio sea público o privado. Selecciona la opción "Privado".
- **Inicializar con un README (opcional):** Marca esta casilla si deseas crear un archivo README.md inicial. Este archivo se utiliza para proporcionar información sobre tu proyecto.
- **Añadir .gitignore (opcional):** Puedes seleccionar un archivo .gitignore para especificar qué archivos y carpetas deben ignorarse en el control de versiones.
- **Elegir una licencia (opcional):** Selecciona una licencia para tu proyecto. Esto define cómo otros pueden usar tu código.

4. Crea el repositorio:

- Haz clic en el botón "Crear repositorio".

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

Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo que te permite colaborar con otros desarrolladores de forma segura. Aquí te explico los pasos para hacerlo:

1. Navega a la página del repositorio:

- Ve a la página principal del repositorio privado al que deseas invitar a alguien.

2. Accede a la configuración del repositorio:

- Debajo del nombre del repositorio, haz clic en la pestaña "Settings" (Configuración).

3. Ve a la sección de colaboradores:

- En la barra lateral izquierda, haz clic en "Collaborators & teams" (Colaboradores y equipos).

4. Invita a un colaborador:

- Haz clic en el botón verde "Add people" (Agregar personas).
- Escribe el nombre de usuario de GitHub, el nombre completo o la dirección de correo electrónico de la persona que deseas invitar.
- Selecciona el nombre de usuario de la lista de resultados.
- Opcionalmente, puedes asignar un rol especifico a ese colaborador.
- Haz clic en "Add [nombre del usuario] to [nombre del repositorio]".

5. El colaborador recibe la invitación:

- El usuario invitado recibirá una notificación por correo electrónico con la invitación para colaborar en el repositorio.
- Una vez que acepte la invitación, tendrá acceso al repositorio según los permisos que le hayas otorgado.

★ ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio de código accesible para cualquier persona en Internet. Esto significa que cualquier usuario de GitHub puede ver, clonar, bifurcar y, en algunos casos, contribuir a un repositorio público.

★ ¿Cómo crear un repositorio público en GitHub?

Crear un repositorio público en GitHub es un proceso sencillo que te permite compartir tu código y proyectos con la comunidad. Aquí te explico los pasos para hacerlo:

1. Inicia sesión en GitHub:

- Dirígete a la página web de GitHub (github.com) e inicia sesión con tu cuenta.

2. Accede a la opción "Nuevo repositorio":

- En la esquina superior derecha de cualquier página, haz clic en el icono "+" y selecciona "Nuevo repositorio".

3. Configura el repositorio:

- **Nombre del repositorio:** Escribe un nombre corto y descriptivo para tu repositorio.
- **Descripción (opcional):** Agrega una descripción que explique el propósito del repositorio.
- **Visibilidad:** Aquí es donde eliges si quieres que el repositorio sea público o privado. Selecciona la opción "Público".
- **Inicializar con un README (opcional):** Marca esta casilla si deseas crear un archivo README.md inicial. Este archivo se utiliza para proporcionar información sobre tu proyecto.
- **Añadir .gitignore (opcional):** Puedes seleccionar un archivo .gitignore para especificar qué archivos y carpetas deben ignorarse en el control de versiones.
- **Elegir una licencia (opcional):** Selecciona una licencia para tu proyecto. Esto define cómo otros pueden usar tu código.

4. Crea el repositorio:

- Haz clic en el botón "Crear repositorio".

★ ¿Cómo compartir un repositorio público en GitHub?

Compartir un repositorio público en GitHub es bastante sencillo, ya que están diseñados para ser accesibles a cualquier persona. Aquí te explico las principales formas de hacerlo:

1. Compartir la URL del repositorio:

- **Copiar la URL:**
  - La forma más directa es copiar la URL del repositorio desde la barra de direcciones de tu navegador.
  - Puedes compartir esta URL a través de correo electrónico, mensajes, redes sociales o cualquier otro medio digital.
- **Facilitar el acceso:**
  - Cualquier persona que tenga la URL podrá acceder al repositorio, ver el código, clonarlo o bifurcarlo.