

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.

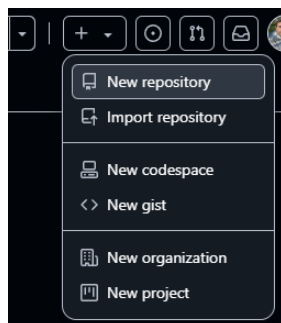
Resultados de aprendizaje:

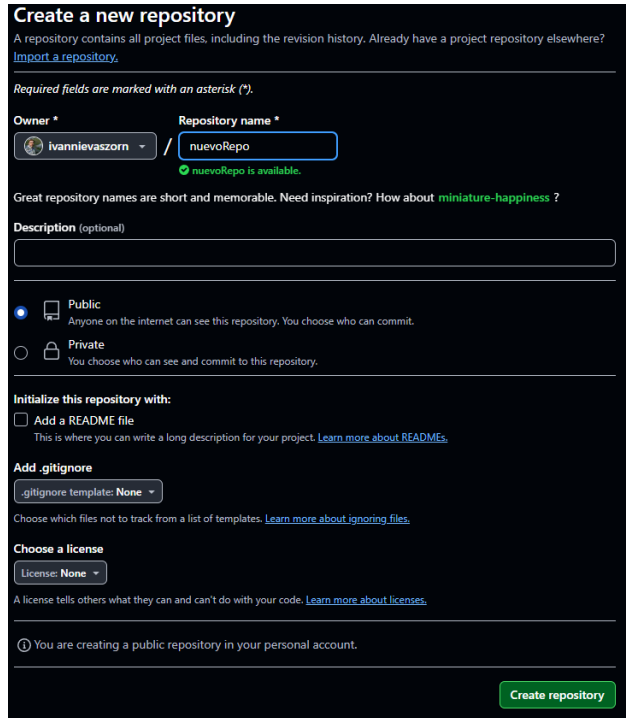
1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, 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 que ofrece alojamiento de repositorios de control de versiones, que permite a los desarrolladores almacenar y gestionar sus proyectos de software. Es una herramienta gratuita y de código abierto que permite el trabajo en equipo en proyectos, el intercambio de código y el trabajo conjunto de forma eficiente
- ¿Cómo crear un repositorio en GitHub?





- ¿Cómo crear una rama en Git?
`git branch nombreDeLaNuevaRama`
- ¿Cómo cambiar a una rama en Git?
`git checkout nombreDeLaRama`
- ¿Cómo fusionar ramas en Git?
Posicionados en la rama de destino: `git merge nombreDeLaRama`
- ¿Cómo crear un commit en Git?
`git add .` (Agrega los cambios al area de preparacion)
`git commit -m "mensaje"` (Hace el commit con el mensaje)
- ¿Cómo enviar un commit a GitHub?
`git add .`
`git commit -m "mensaje"`
`git push origin ramaActual`
- ¿Qué es un repositorio remoto?
Los repositorios remotos son versiones del proyecto que están hospedadas en Internet o en cualquier otra red.
- ¿Cómo agregar un repositorio remoto a Git?
`git remote add "nombre" "url"`
- ¿Cómo empujar cambios a un repositorio remoto?
`git push origin "rama"`

- ¿Cómo tirar de cambios de un repositorio remoto?
`git pull origin main`
- ¿Qué es un fork de repositorio?
Es una copia de un repositorio de otra cuenta en mi propia cuenta GitHub
- ¿Cómo crear un fork de un repositorio?



- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
En la solapa Pull request.
- ¿Cómo aceptar una solicitud de extracción?
En mi cuenta veo los Pull request y puedo aceptarlos.
- ¿Qué es un etiqueta en Git?
Se utiliza para marcar puntos específicos del historial
- ¿Cómo crear una etiqueta en Git?
Con etiqueta ligera o anotada.
- ¿Cómo enviar una etiqueta a GitHub?
`git tag "etiqueta"`
`git push origin "etiqueta"`
`git push origin --tags` (Push a todas las etiquetas creadas)
- ¿Qué es un historial de Git?
Secuencia de todos los cambios realizados en un repositorio de Git.
- ¿Cómo ver el historial de Git?
`git log`
- ¿Cómo buscar en el historial de Git?
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"`

Commits que han modificado un archivo específico, usa git log seguido del nombre del archivo: `git log --nombre_del_archivo`

Commits en un rango de fechas específico, usa las opciones `--since` y `--until`:

`git log --since="2024-01-01" --until="2024-01-31"`

Commits hechos por un autor específico, usa `--author`:

`git log --author="Nombre del Autor"`

- ¿Cómo borrar el historial de Git?

`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.

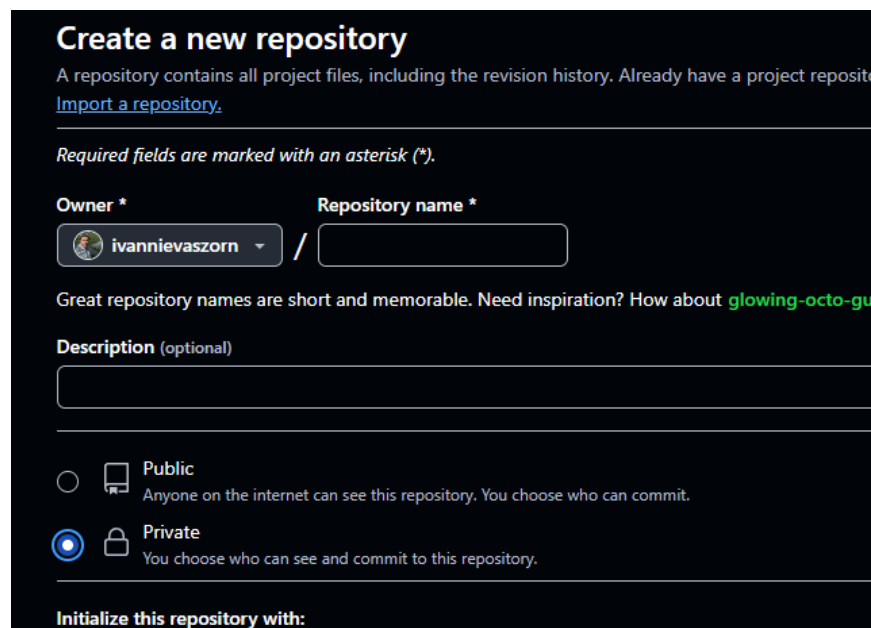
`git reset nombreCarpeta/nombreArchivo ->` Quita ese archivo del stage (que a la vez está dentro de una carpeta).

`git reset nombreCarpeta/*.extensión ->` Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

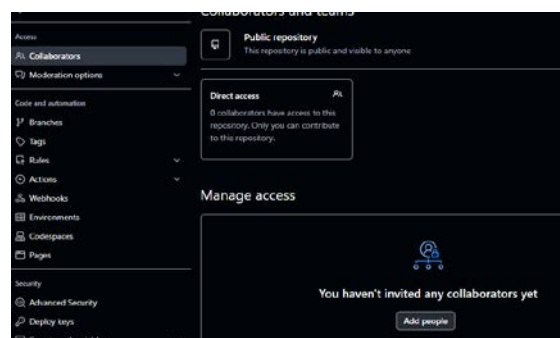
- ¿Qué es un repositorio privado en GitHub?

Solo es accesible para usuarios específicos que han sido autorizados.

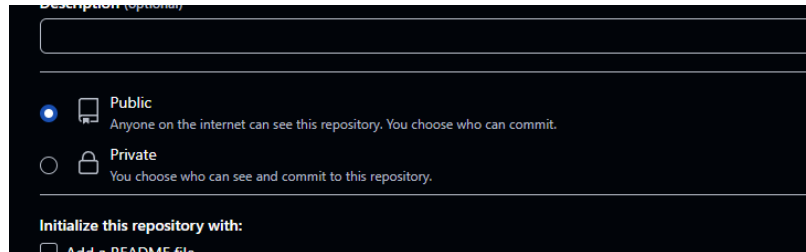
- ¿Cómo crear un repositorio privado en GitHub?



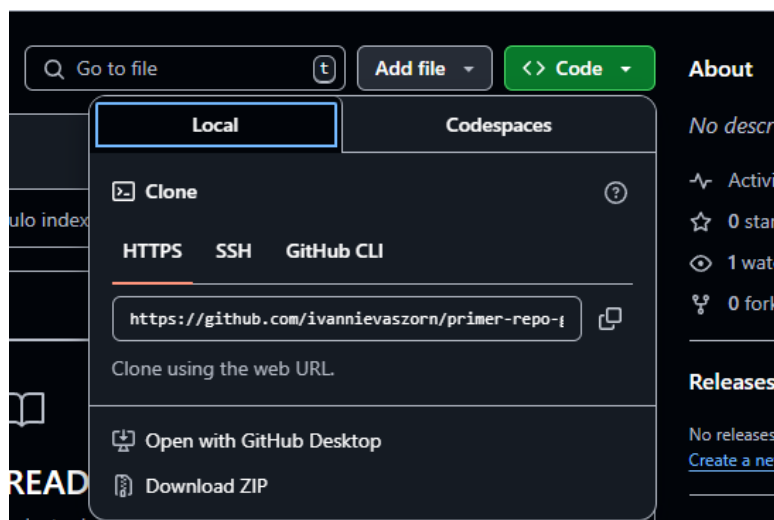
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?



- ¿Qué es un repositorio público en GitHub?
Su contenido es accesible a cualquier persona en Internet
- ¿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.