Alumna: María Victoria Volpe

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

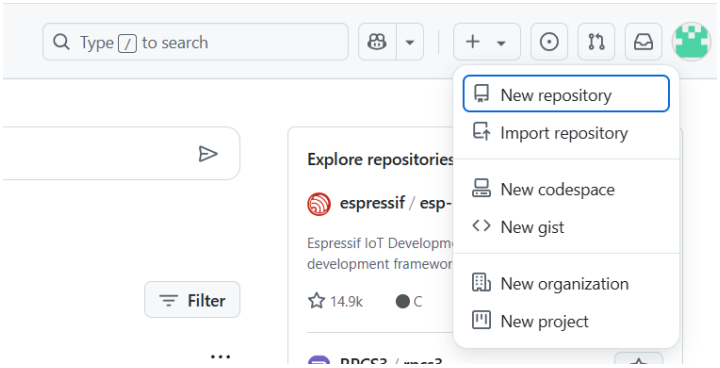
1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?

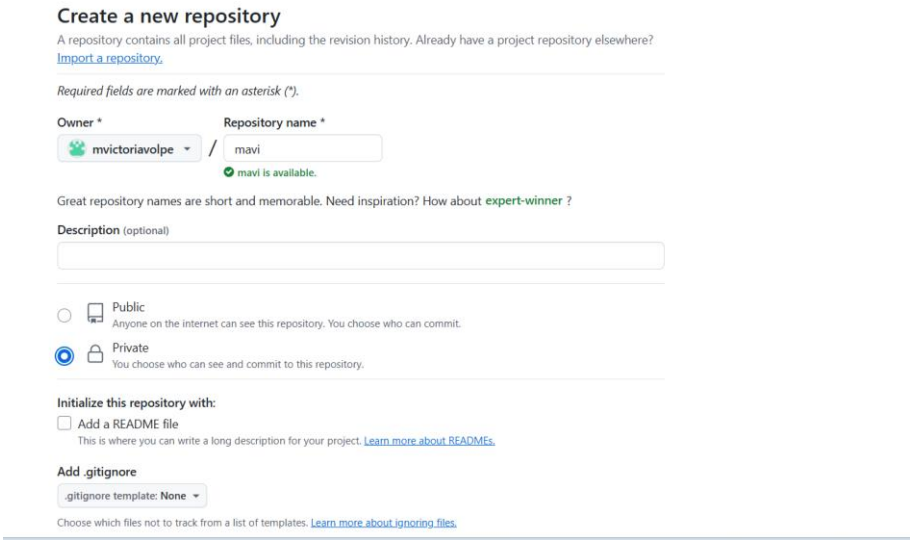
GitHub Es una plataforma en línea donde los desarrolladores pueden guardar, compartir y modificar códigos de manera colaborativa. Funciona como una comunidad de código abierto en la que se desarrollan proyectos de software en equipo. Permite guardar y organizar códigos en repositorios, rastrear y gestionar cambios en los proyectos. Explorar otros proyectos, temas y perfiles de desarrolladores.

- ¿Cómo crear un repositorio en GitHub?

1. Iniciar sesión en https://github.com/
2. Hacer click en el ícono **"+"** en la esquina superior derecha y selecciona **"New repository"**.



3. Escribir un nombre para el repositorio y agrega una descripción opcional, elegir si será público o privado.



(Opcionalmente, se puede inicializarlo con un **README**, un **.gitignore** o una licencia.)

4. Hcer click en botón verde “Create repository”.

- ¿Cómo crear una rama en Git?

  Las ramas nos permite trabajar en cambios sin afectar el código principal, para crear una nueva rama se puede usar la palabra *branch*

git branch nombre-de-la-rama

- ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama existente se usa la palabra *checkout*

git checkout nombre-de-la-rama

- ¿Cómo fusionar ramas en Git?

Para fusionar se usa la palabra *merge*

git merge nombre-de-la-rama

- ¿Cómo crear un commit en Git?

1. Verificar el estado de los arhivos con: git status
2. Preparar los cambios para el commit: git add *nombre-del-archivo*
3. Crear el commit: git commit –m “Descripción”
4. Para verificar se puede usar: git log
5.

- ¿Cómo enviar un commit a GitHub?

  Se usa: git push origin nombre-de-la-rama

- ¿Qué es un repositorio remoto?

Un **repositorio remoto** es una versión del proyecto alojado en un servidor (como GitHub). Permite colaboración y respaldo en la nube.

- ¿Cómo agregar un repositorio remoto a Git?

  Se usa: git remote add origin URL-del-repositorio

  Origin: nombre del remoto (puede ser otro)

  La URL es la del repositorio en GitHub

- ¿Cómo empujar cambios a un repositorio remoto?

Enviar cambios al remoto: se utiliza la palabra PUSH

Luego de usar commit, escribir:

git push origin nombre-de-la-rama

- ¿Cómo tirar de cambios de un repositorio remoto?

Traer cambios del repositorio remoto, se usa la palabra PULL

git pull origin nombre-de-la-rama

- ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio en otra cuenta de GitHub. Permite modificar el código sin afectar el repositorio original.

- ¿Cómo crear un fork de un repositorio?

1. Ir a un repositorio en GitHub
2. Hacer click en botón “Fork” (Arriba a la derecha)
3. GithHub crea una copia en la cuenta

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

Si quiero colaborar en un proyecto de otra persona en GitHub, no puedo modificar el código directamente. Primero, tengo que hacer un **fork**, que es básicamente una copia del repositorio en mi cuenta. Ahí sí puedo hacer cambios y después pedir que los agreguen al proyecto original con un **Pull Request (PR)**.

- Subir los cambios a fork en Github (git push)
- Ve a GitHub y entra en tu fork (tu copia del repositorio).
- Verás un mensaje que dice algo como "Compare & pull request".
- Haz clic en ese botón o ve a la pestaña Pull Requests del repositorio original y selecciona "New Pull Request".
- Selecciona la rama con tus cambios (mi-nueva-rama).
- Escribe un mensaje explicando qué cambios hiciste y por qué.
- Haz clic en "Create Pull Request".
- Esperar a que el dueño del repositorio revise la solucutud

- ¿Cómo aceptar una solicitud de extracción?

Si sos el propietario del repositorio:

1. Ir a la pestaña Pull Requests.
2. Abre la solicitud de extracción.
3. Revisa los cambios y haz clic en Merge pull request.
4. Confirma la fusión y elimina la rama si es necesario.

- ¿Qué es un etiqueta en Git?

Las etiquetas en Git sirven para marcar momentos importantes en el historial del proyecto, como versiones específicas (ejemplo: v1.0.0). Son como señales que ayudan a identificar cambios clave, por ejemplo, cuando se lanza una nueva versión del software.

- ¿Cómo crear una etiqueta en Git?

Las etiquetas se crean:

git tag -a v1.0 -m "Versión 1.0"

git tag v1.0

- ¿Cómo enviar una etiqueta a GitHub?

Se puede enviar a GitHub:

git push origin v1.0

Se pueden enviar varias:

git push origin --tags

- ¿Qué es un historial de Git?

El historial de Git es como registro detallado de todos los cambios relizados en un proyecto. Cada vez que se guardan los cambios con un commit, Git guarda un "punto de control" que incluye detalles como quién hizo el cambio, cuándo lo hizo y qué cambios específicos se hicieron. Este historial es importante porque te permite revisar cómo ha evolucionado el proyecto a lo largo del tiempo, y si algo sale mal, se puede volver a una versión anterior del código.

Con el historial de Git, es posible:

- Ver los cambios anteriores: Usando comandos como git log, puedes ver todo lo que se ha hecho en el repositorio, ordenado cronológicamente.
- Comparar versiones: comparar diferentes versiones del código para ver qué ha cambiado entre dos puntos en el tiempo.
- Deshacer cambios: retroceder a un commit anterior y deshacer los cambios que has hecho desde entonces.
- Rastrear el progreso: mantener un registro de qué características o arreglos se han implementado y cuándo.
- Buscar en el historial: se puede buscar por palabra clave, fecha o autor.

- ¿Cómo ver el historial de Git?

Se puede ver con:

git log

- ¿Cómo buscar en el historial de Git?

Se puede buscar en los mensajes de los *commits* utilizando `git log` con la opción `--grep`. Por ejemplo, si se busca un *commit* cuyo mensaje contiene la palabra "programación": git log --grep="progrmación"

Se puede buscar por autor del commit: git log --author="Juan"

Se puede buscar *commits* hechos en un rango de fechas usando las opciones `--since` y `--until`. Por ejemplo: git log --since="2025-01-01" --until="2025-03-01"

- ¿Cómo borrar el historial de Git?

  Se puede usar el comando: git rebase -i –root

  Para borrar un comando específico: git reset --hard <commit_id>

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado en GitHub es un repositorio cuyo acceso está restringido. Solo las personas a quienes se les haya otorgado permiso pueden ver o modificar el contenido de ese repositorio.

- ¿Cómo crear un repositorio privado en GitHub?

  1. Iniciar sesión
  2. 2. Crear un nuevo repositorio (haciendo click en “+”), seleccionar “New repository)
  3. Configurar el repositorio, visibilidad elegir “Privado”.

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

  1. Ir al repositorio a compartir/invitar
  2. Abrir configuraciones “Settings”
  3. En el menú lateral izquierdo, seleccionar **"Manage access"**.
  4. Hacer clic en el botón "Invite a collaborator".
  5. Escribir el nombre de usuario de GitHub de la persona que quieres invitar.
  6. Hacer click en "Add".

- ¿Qué es un repositorio público en GitHub?

Un **repositorio público** en GitHub es un repositorio que está abierto para todo el mundo. Cualquier persona con el enlace puede ver, clonar o bifurcar (hacer un *fork*) el repositorio. Los repositorios públicos son ideales para proyectos de código abierto o para compartir trabajo con la comunidad en general.

- ¿Cómo crear un repositorio público en GitHub?

  1. Iniciar sesión
  2. Crear un nuevo repositorio (haciendo click en “+”), seleccionar “New repository)
  3. Configurar el repositorio, visibilidad elegir “Publico”.

¿Cómo compartir un repositorio público en GitHub?

1. Abrir el repositorio en GitHub
2. Copiar la URL del repositorio: https://github.com/tu-usuario/nombre-del-repositorio
3. Comparte el enlace

Realizar la siguiente actividad:

Crear un repositorio.

- Dale un nombre al repositorio.
- Elije el repositorio sea público.
- Inicializa el repositorio con un archivo.
  Link del repositorio público:
  https://github.com/mvictoriavolpe/mi-primer-repo

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).
  https://github.com/mvictoriavolpe/mi-primer-repo

- Creando Branchs
    - Crear una Branch
    - Realizar cambios o agregar un archivo
    - Subir la Branch

    https://github.com/mvictoriavolpe/mi-primer-repo/branches

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

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.