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 que permite almacenar proyectos que usan el sistema de control de versiones Git. Es el servicio más popular del mundo para compartir y colaborar en proyectos de software, tanto de código abierto como privados.

Almacena los repositorios en la nube, lo que facilita el trabajo colaborativo entre desarrolladores ubicados en diferentes partes del mundo.

Además de servir como un "espacio remoto" donde subir tu código, GitHub incluye herramientas para organizar, discutir, revisar y aprobar cambios, lo que lo convierte en una solución integral para el desarrollo de software moderno.

Es una comunidad donde nosotros podemos compartir nuestros repositorios de forma pública o privada.

Es una plataforma donde los programadores guardan, comparten y colaboran en proyectos de código de manera eficiente.

➢ **• ¿Cómo crear un repositorio en GitHub?**

Primero hay que registrarse.
Posteriormente loguearse.
Seguido a eso Seleccionar New Repository
Dentro colocamos el nombre del Repositorio, definimos si será Público o Privado y demás configuraciones.
Finalmente presionamos el botón “Create repository”.

➢ **• ¿Cómo crear una rama en Git?**

En la Terminal escribimos lo siguiente:
`git branch nombre_de_la_rama`

También podemos hacerlo con:
`git checkout -b nombre_de_la_rama`
Esto permite crear la rama y moverse a ella.

➢ **• ¿Cómo cambiar a una rama en Git?**

Con el comando git checkout.
Ejemplo: **`git checkout nombre_de_la_rama`**

➢ **• ¿Cómo fusionar ramas en Git?**

Primero nos posicionamos en la rama que va a recibir los cambios
Ejemplo: git checkout main
Luego para fusionar utilizamos el comando merge y seleccionamos la rama desde la que queremos traer los cambios.
Ej.: **`git merge ejercicio2`**

➢ **• ¿Cómo crear un commit en Git?**

Utilizamos el comando commit seguido de -m para agregarle un mensaje significativo.
Ej: **`git commit -m "Primer Commit creado ejercicio2.py"`**

➢ **• ¿Cómo enviar un commit a GitHub?**

Primero tenemos que realizar:
La configuración global de usuario y correo electrónico
Ej: git config –global user.name “Usuario”
Ej: git config –global user.email “usuario@gmail.com”

Luego tenemos que agregar los cambios y haber hecho un commit local
Ej: git add .
Ej: git commit -m “Primer Commit”

Ahora tenemos que vincular el Repositorio Remoto de GitHub al que vamos a subir
Ej: git remote add origin “https://github.com/usuarioGitHub/nombre_del_repo.git”

Finalmente usamos el comando push para subir a GitHub
Ej: **`git push -u origin nombre_de_la_rama`**
El -u se utiliza sólo la 1º vez, luego podemos sólo usar **git push**

➢ **• ¿Qué es un repositorio remoto?**

Es un repositorio que se aloja en un servidor externo como GitHub.
Nos sirve para no depender de nuestro equipo local y también para poder trabajar de manera colaborativa.

➢ **• ¿Cómo agregar un repositorio remoto a Git?**

Para Vincular el repositorio remoto usamos el comando:
git remote add origin <url>
Ej: **git remote add origin https://github.com/userGitHub/nombre_repo.git**

➢ **• ¿Cómo empujar cambios a un repositorio remoto?**

Usamos el comando para la 1º vez:
Ej: **`git push -u origin nombre_de_la_rama`**

Luego podemos utilizar sólo:
Ej: **`git push`**

➢ **• ¿Cómo tirar de cambios de un repositorio remoto?**

Para traer los cambios de un Repositorio Remoto al local hacemos lo siguiente:
Ej: **`git pull origin nombre_de_la_rama`**
Luego podemos usar directamente **`git pull`**

➢ **• ¿Qué es un fork de repositorio?**

Un Fork es una copia completa de un repositorio de GitHub que se crea dentro de nuestra propia cuenta de GitHub.

Te permite:

- Probar cambios sin afectar el proyecto original.
- Proponer mejoras enviando los cambios mediante un pull request.
- Trabajar con repositorios a los que no se tiene permisos directos.

➢ **• ¿Cómo crear un fork de un repositorio?**

Para crear un Fork de un repositorio los pasos son:

- Ingresar a un repositorio público de GitHub
- Estar logueado en nuestra cuenta de GitHub
- Hacer click en el botón “Fork”
- Elegir el nombre y la configuración
- Hacer click en el botón “Crear el Fork”

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

Pasos para enviar una solicitud de extracción (pull request):

- Primero ingresar al fork de nuestra cuenta de GitHub con los cambios subidos.
- Elegir la rama y seleccionar “New pull request”
- Completar título, descripción con los cambios realizados y ramas a fusionar
- Finalmente presionar el botón “Create pull request”

➢ **• ¿Cómo aceptar una solicitud de extracción?**

Pasos para aceptar una solicitud de extracción:

- Ingresar al repositorio de GitHub.
- Hacer click en “pull request”
- Elegir el pull request a revisar
- Revisar los cambios propuestos
- Hacer click en “Mergue pull request” para aceptar los cambios
- Finalmente click en “Confirm merge”

➢ **• ¿Qué es una etiqueta en Git?**

Es como un marca que se le pone a un commit importante.
Generalmente se utiliza para marcar versiones de un proyecto.
Ej: versión 1.0.1

Sirve para indicar un punto específico en el historial del código para poder encontrarlo fácilmente. Por ejemplo una versión estable del proyecto

Ej: v2.0 release 2025.

➢ **• ¿Cómo crear una etiqueta en Git?**

Para crear una etiqueta en Git utilizamos el comando **`git tag`**.

Ej: git tag v1.0.0

Si queremos crear un mensaje junto a la etiqueta usamos el -a y -m
Ej: git tag -a v1.0.0 -m “Versión 1.0 Estable”

➢ **• ¿Cómo enviar una etiqueta a GitHub?**

Para enviar una etiqueta a GitHub usamos el comando **`git push origin`**.
Ej: git push origin v1.0.0

Si queremos enviar Todas las etiquetas creadas localmente usamos:
Ej: **`git push –tags`**

➢ **• ¿Qué es un historial de Git?**

Es el registro de todos los cambios que se hicieron en el repositorio.
Cada vez que se hace un commit Git guarda:

- Los archivos cambiaron
- Las líneas que se agregaron o eliminaron
- Quién lo hizo
- Cuándo lo hizo
- El mensaje del commit

➢ **• ¿Cómo ver el historial de Git?**

Para poder ver el historial usamos el comando: **`git log`**
También podemos usar **`git log –oneline`** para ver más resumido el historial.

➢ **• ¿Cómo buscar en el historial de Git?**

Se puede buscar de varias maneras dependiendo que estemos necesitando.

Algunos ejemplos pueden ser:

- Buscar por palabra clave en los mensajes de commit.

  `git log --grep="palabra"`

- Buscar cambios en un archivo específico

  `git log ruta/nombre_archivo.extensión`

- Buscar un texto dentro del contenido de los commits

  `git log -S"texto a buscar"`

- Buscar por Autor

  `git log --autor="nombre o email"`

- Buscar quién fue el último en modificar una línea con su commit

  `git blame archivo.extensión`

- También podemos unir varias opciones o parámetros para mejorar la visual

  `git log –grep="palabra" --oneline –grep –all –decorate`

➢ **• ¿Cómo borrar el historial de Git?**

Se puede borrar simplemente eliminando el archivo .git de la carpeta del Repositorio.

Si queremos continuar con el Proyecto tenemos que volver a hacer un git init, agregar todo o lo que necesitemos y luego realizar el commit.

➢ **• ¿Qué es un repositorio privado en GitHub?**

Es un Proyecto que está oculto para el público.
Sólo pueden verlo y trabajar con el proyecto las personas que tengan acceso.
No se puede hacer un fork ni enviar un pull request sin los permisos adecuados.

➢ **• ¿Cómo crear un repositorio privado en GitHub?**

Cuando creamos el repositorio en GitHub debemos seleccionar la opción “Private” dentro de Visibilidad.

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

Para invitar a alguien a un repositorio privado en GitHub debemos:

- Entrar al repositorio privado
- Seleccionar la opción Settings (ícono de la rueda)
- Buscar “Collaborators” dentro del menú lateral
- Presionar botón “Add people”
- Escribir el username o email de la persona a invitar
- Seleccionarla y presionar “Add ..”

➢ **• ¿Qué es un repositorio público en GitHub?**

Es un Proyecto que cualquier persona puede ver, explorar o clonar inclusive si no tiene cuenta en GitHub.
Son los que llamamos comúnmente “proyecto de código abierto”.

➢ **• ¿Cómo crear un repositorio público en GitHub?**

Para crear un repositorio público en GitHub se deben seguir estos pasos:

- Entrar a la web de GitHub
- Ingresar a la cuenta de GitHub previa registración
- Hacer click en “New Repository”
- Completar los datos solicitados
- Seleccionar la opción “**Public**”
- Clickear en “Create Repository”

➢ **• ¿Cómo compartir un repositorio público en GitHub?**

Se comparte simplemente enviando el enlace de la URL del repositorio creado.

Ej: https://github.com/sbruselario/UTN-TUPaD-P1