Taller de Git y Github

**¿Diferencia entre Git y GitHub?**

* **Git:** Es el software, la herramienta que se instala en la computadora para rastrear los cambios en los archivos de un proyecto. Es como tener un registro detallado de cada modificación.
* **GitHub:** Es una plataforma en línea (una página web) que aloja repositorios de Git. Facilita la colaboración, el respaldo en la nube y la gestión de proyectos.

**Instalación de Git**

* **Windows**: Descargar el instalador desde la página oficial: [Git - Downloads](https://git-scm.com/downloads) y ejecutalo.
* **MacOs**: debes de instalar primero homebrew: [Homebrew — The Missing Package Manager for macOS (or Linux)](https://brew.sh/), ejecutando en la terminal el comando que está en esa página.
* Luego abriendo la terminal, y colocando el comando: $ brew install git, y listo.
* **Linux**: Usar el comando: sudo apt-get install git.

**Creación de cuenta en Github**

* Accede a este link: [GitHub](https://github.com/), crea una cuenta con tu correo principal, y elige un nombre de usuario que te sea fácil recordar.

**Configuración inicial de Git.**

* Abre la terminal, colocando en tu buscador “cmd”, y por si acaso ejecútalo como administrador. Ejecuta los siguientes comandos, sustituyéndolos por tu nombre de usuario y correo de Github.
* git config --global user.name "Tu Nombre"
* git config --global user.email "tu\_correo@ejemplo.com"

**Ejercicio individual: creando el primer repositorio**

1. Creen en el escritorio una carpeta llamada mi-primer-repo
2. Dentro de esa carpeta den click derecho y elijan Open Git Bash Here, eso abrirá una terminal
3. Ejecuten el comando: **git init**
4. Creen un bloc de notas dentro de la carpeta, puedes colocar su top 5 de canciones favoritas o series. No olviden guardar dentro de la carpeta el bloc.
5. Ahora ejecuta el siguiente comando: **Git status** en la misma terminal que se abrió al dar click a Open Git Bash. Verán su bloc de notas como ‘’untracked files’’ en rojo.
6. Para agregar el archivo al área de preparación debes ejecutar el comando: **git add <nombre\_del\_archivo>, o git add . , este ultimo es para agregar todos los archivos que existen en una carpeta.**
7. Para guardar los cambios en el repositorio debes ejecutar el comando: **git commit -m "Mi primer commit", lo que esta en comillas lo puedes cambiar.**
8. Si quieres ver el historial de commits que se han hecho puedes ejecutar: **git log**
9. **Ahora a subir a github el repositorio, entraras a tu cuenta y crearas un nuevo repositorio:**

A black rectangular object with a white border

AI-generated content may be incorrect.

1. Le colocas un nombre, puede ser “mi primer repo”

A screenshot of a computer program

AI-generated content may be incorrect.

1. Ahora ejecutaras los siguientes comandos en la terminal:
2. git branch -M main (se crea la rama principal en la que se subirá el repo)
3. git remote add origin (aquí ira el link del repo de github, sin parentesis)
4. git push -u origin main (este comando es para subir el repo).

Si haces cambios en la carpeta o documento que subiste al repositorio, puedes seguir los siguientes pasos para actualizarlo en github:

1. git status
2. git add .
3. git commit -m “actualización de repo”
4. git push origin main

**Trabajo colaborativo-opción 1**

1. accede al siguiente repositorio: <https://github.com/Laroguee/Taller-ejercicio-colaborativo-AEB.git>
2. le darás click a for, y luego a **create new fork**, esto hará una copia en tus repositorios

A screenshot of a computer

AI-generated content may be incorrect.

1. ahora en tu escritorio darás click derecho y elegirás de nuevo la opción **Open git bash Here**
2. ejecutaras el comando **git clone** (aquí el link del repo a clonar, sin paréntesis)
3. esto creara una carpeta en tu escritorio con el nombre del repositorio
4. Ahora abrirás una terminal de nuevo con Open git bash here, en la carpeta que se genero
5. Y ejecutaras los siguientes comandos:
6. **git branch mi-aporte** (puedes colocar tu nombre de usuario en vez de mi-aporte)
7. **git checkout mi-aporte**
8. ya puedes realizar cambios al documento de texto que esta en el repositorio
9. una vez finalices los cambies ejecuta los comandos:
10. **git push origin mi-aporte**
11. para enviar los cambios al repositorio del que hiciste fork, darás click a la **opción crear pull request**, esto lo puedes hacer desde la versión tuya de ese repo

A screenshot of a computer

AI-generated content may be incorrect.

1. luego de haber generado el pull request, el dueño del repositorio original fusionara los cambios.

**Trabajo colaborativo -opción 2 agregando colaboradores**

**Añadir Colaboradores:**

* 1. Ve a la pestaña Settings de tu repositorio.
  2. En el menú de la izquierda, haz clic en Collaborators and teams.
  3. Presiona el botón Add people.
  4. Busca el nombre de usuario de GitHub de cada participante, selecciónalo y asígnale un rol (generalmente Write)
  5. Cada participante recibirá una invitación por correo electrónico y una notificación en GitHub que deberán aceptar.

**Clonar el repositorio**

1. Una vez que han aceptado la invitación, cada colaborador debe clonar el repositorio principal directamente a su máquina local. No necesitan hacer un *fork*.
2. Usando el comando en una terminal abierta con open git bash en el escritorio:
3. git clone <https://github.com/tu-usuario/proyecto-colaborativo-equipo.git>

**Crea una rama personal**

1. Esta es la regla de oro: **nunca trabajar directamente en la rama main**. Cada colaborador debe crear su propia rama para trabajar de forma aislada.
2. Usarás el comando:
3. git checkout -b nombre-del-colaborador/su-aporte

**Desarrollar y guardar cambios**

1. hacer el trabajo en su propia rama, cada colaborador creará su archivo o modificará los existentes.
2. Guardar los cambios localmente usando los comandos básicos que son
3. Git status
4. Git add .
5. Git commit -m “cambio que se hizo”

**Sincronizar y compartir el trabajo**

1. **Actualizar con la Rama Principal:** Antes de subir sus cambios, es una buena práctica asegurarse de que su rama personal está actualizada con los últimos cambios de la rama main.
2. **Usando los siguiente comandos:**
3. Git checkout main
4. Git pull origin main
5. Git checkout nombre-del-colaborador/su-aporte
6. Git merge main

**Subir la rama a Github**

1. Usando los comandos
2. Git push origin HEAD
3. Git push origin nombre-del-colaborador/su-aporte

**Revisión y fusión**

Crear el Pull Request (Rol: Cada Colaborador):

1. Cada colaborador debe ir a la página del repositorio en GitHub.
2. Aparecerá una notificación amarilla sugiriendo crear un Pull Request para la rama que acaban de subir.
3. Hacen clic y se abrirá un formulario donde deben asegurarse de que la fusión sea de su rama (nombre-del-colaborador/su-aporte) hacia la rama main.
4. Añaden un título y una descripción clara, y crean el Pull Request.

Revisar y Fusionar el Pull Request (Rol: Dueño):

1. En la pestaña Pull Requests del repositorio, verás todas las solicitudes.
2. Puedes abrir cada una, revisar los cambios (Files changed), dejar comentarios o solicitar modificaciones.
3. Si todo está correcto, presionas Merge pull request.
4. **Tip:** Después de fusionar, GitHub te dará la opción de eliminar la rama remota. Es una buena práctica hacerlo para mantener limpio el repositorio.

Actualización Final (Rol: Todos):

1. Una vez que varias ramas han sido fusionadas a main, todos los colaboradores deben actualizar su copia local de main para tener la versión más reciente del proyecto.
2. Generated bash
3. git checkout main
4. git pull origin main