- **Curso Introduccion a Javascript**
- **Profesora: Sachetti Sofia**

Actividades Clase Numero 6:

Git y GitHub

¡Hola, chicas! 👏

Hoy nos enfocaremos en consolidar todo lo que hemos aprendido hasta ahora sobre **Git** y **GitHub**, dos herramientas fundamentales para el control de versiones y la colaboración en proyectos de desarrollo.

Esta ejercitación está diseñada para que refuercen sus conocimientos mediante una serie de actividades que simulan tareas reales en proyectos de software. Con estos ejercicios, podrán corregir errores, afianzar conceptos y ganar más confianza para usar Git y GitHub en futuros proyectos. ¡Vamos a trabajar juntas y aprender mucho! 💪

Objetivos 📝

- Crear un repositorio remoto en **GitHub**, un repositorio local en nuestra computadora, y luego trabajar con archivos locales sincronizando los cambios en GitHub.
- ¡Atención! 🔔 Este ejercicio se realiza en **parejas**, así que es hora de buscar a tu compañera de código.
- Podemos usar la **terminal** o una **interfaz gráfica**, pero les recomiendo usar la terminal para familiarizarnos más con las herramientas.

Requisitos 🔧

1. **Cuenta en GitHub:** Si no tienes una, puedes crearla en GitHub. Si ya tienes una, simplemente ingresa con tu usuario. ✨
2. **Git instalado en tu computadora:** Si aún no lo has instalado, sigue la guía de instalación de Git. 💻
3. Para este ejercicio, identificaremos a dos participantes: **Participante A** y **Participante B**.

Pasos a seguir 👣

1. Crear un repositorio remoto en GitHub 🌐

Participante A:

1. Accede a GitHub con tu cuenta.
2. Haz clic en el botón + en la esquina superior derecha y selecciona **"Nuevo repositorio"**.
3. Completa los campos con el **nombre** del proyecto. Es muy importante que **NO marques la opción de README.md** ni otras opciones. ¡Queremos un repositorio vacío!
4. Haz clic en **Crear repositorio**.

🎉 ¡Listo! Ahora tienes un repositorio remoto en GitHub.

---

2. Invitar a colaboradoras 🤝

Participante A:

1. En tu nuevo repositorio, ve a la pestaña **Settings** (Configuración) y busca la sección **Colaboradores** o **Manage Access**.
2. Haz clic en **Invitar colaborador** e ingresa el nombre de usuario o correo electrónico de **Participante B**.
3. Envía la invitación y espera a que **Participante B** la acepte.

Tip: Si **Participante B** acaba de crear su cuenta en GitHub, tal vez tarde unos minutos en aparecer en la búsqueda. ¡Paciencia! ⏳

---

3. Crear un repositorio local y vincularlo 🔗

Participante A:

1. Abre la **terminal** en tu computadora.
2. Navega hasta la carpeta donde quieres crear el repositorio local con

   cd nombre-de-carpeta.

3. Clona el repositorio remoto a tu computadora usando:

   git clone https://github.com/tu-usuario/nombre-del-repositorio.git

4. Entra en la carpeta del repositorio con:

   cd nombre-del-repositorio

5. Crea un archivo **README.md**:

   echo "# Nombre del Repositorio" > README.md

6. Agrega el archivo al repositorio local:

   git add README.md

7. Haz un **commit** con un mensaje descriptivo:

   git commit -m "Añadir archivo README.md inicial"

8. Sube los cambios al repositorio remoto:

   git push origin main

🎉 ¡Tu repositorio local está listo y sincronizado con GitHub!

Info Importante sobre el comando **echo "# Nombre del Repositorio" > README.md**

- **echo**: Es un comando de la terminal que se usa para mostrar un mensaje o texto en la pantalla. En nuestro caso, está mostrando el texto "# Nombre del Repositorio".

- **>**: Este símbolo se utiliza para redirigir la salida de un comando a un archivo. En lugar de mostrar el texto en la pantalla, el símbolo > lo redirige al archivo que sigue. Si el archivo no existe, lo crea; si ya existe, lo sobrescribe.

- **README.md**: Es el nombre del archivo donde se va a guardar el texto. Es un archivo muy común en proyectos de software, utilizado para describir el propósito y contenido

del proyecto. El .md indica que es un archivo de **Markdown**, que es un formato de texto que permite dar formato (como títulos, listas, enlaces, etc.).

---

4. Clonar el repositorio remoto 🦾

Participante B:

1. Abre la **terminal** en tu computadora.
2. Navega hasta la carpeta donde quieres clonar el repositorio con

   ```
   cd nombre-de-carpeta.
   ```

3. Clona el repositorio remoto con el comando:

   ```
   git clone https://github.com/usuario-participante-a/nombre-del-repositorio.git
   ```

4. Entra en la carpeta clonada:

   ```
   cd nombre-del-repositorio
   ```

🎉 ¡Ya tienes el repositorio de **Participante A** en tu computadora!

---

5. Trabajar en archivos diferentes (Parte 1) 📄

Ambos participantes:

1. Cada una creará **3 archivos** con nombres diferentes (¡a ponerse de acuerdo! 😉). Ejemplo:

   ```
   echo "Contenido de archivo" > pikachu.txt
   echo "Contenido de archivo" > bulbasaur.txt
   echo "Contenido de archivo" > squirtle.txt
   ```

2. Agreguen el contenido que quieran a cada archivo.
3. Agreguen los archivos al repositorio local:

   ```
   git add nombre-del-archivo.txt
   ```

4. Hagan **commit** con un mensaje descriptivo:

```
git commit -m "Agregar archivos iniciales"
```

5. Sincronicen los cambios con el repositorio remoto:

```
git push origin main
```

6. Después de subir sus cambios, cada una debe descargar los cambios de su compañera:

```
git pull origin main
```

💐 Al finalizar este paso, ambas deberían tener **7 archivos** en total (README.md y 3 archivos de cada una).

---

6. Trabajar en archivos diferentes (Parte 2) ✏️

Ambos participantes:

1. Modifiquen alguno de los archivos que hayan creado previamente.
2. Agreguen los cambios al repositorio local:

```
git add nombre-del-archivo-modificado.txt
```

3. Hagan **commit** de los cambios:

```
git commit -m "Modificar archivo"
```

4. Sincronicen los cambios con el repositorio remoto:

```
git push origin main
```

5. Descarguen las modificaciones realizadas por su compañera:

```
git pull origin main
```

💐 ¡Ahora cada una debería ver las modificaciones realizadas por su compañera!

---

7. Trabajar en el mismo archivo ⚔️

Ambos participantes:

1. Seleccionen el **mismo archivo** (por ejemplo, **pikachu.txt**) y agreguen algunas líneas de texto.
2. Cada una debe intentar subir el archivo modificado al repositorio remoto.

```
git add pikachu.txt
git commit -m "Modificación en pikachu.txt"
git push origin main
```

El primer participante en sincronizar no tendrá problemas, pero el segundo verá un mensaje de conflicto. ¡Es hora de resolverlo! ⚠️

---

8. Resolver un conflicto 🛠️

1. El segundo participante verá un mensaje indicando que hay un conflicto.
2. Ahora ambas deben decidir cómo resolver el conflicto. Hay tres opciones:
   - Dejar solo el contenido de **Participante A**.
   - Dejar solo el contenido de **Participante B**.
   - **Unificar** el contenido de ambas.
3. Modifiquen el archivo para resolver el conflicto.
4. Luego, el participante encargado de resolverlo hará un **commit** y subirá el archivo actualizado:

```
git add pikachu.txt
git commit -m "Resolver conflicto en pikachu.txt"
git push origin main
```

🎉 El conflicto está resuelto. El otro participante debe descargar los cambios para tener la versión final.

---

9. ¡Hagamos más conflictos! ❄️

1. Creen nuevos archivos o modifiquen los existentes, generen conflictos y resuélvanlos.
2. Documenten los conflictos y cómo los resolvieron en un archivo llamado conflictos.txt.

---

10. Crear dos ramas (branch) 🌱

1. Cada una creará una nueva **rama** a partir de main:

```
git branch nombre-de-la-rama
git checkout nombre-de-la-rama
```

2. Hagan cambios en los archivos de la nueva rama y súbanlos:

```
git add archivo.txt
git commit -m "Cambios en la nueva rama"
git push origin nombre-de-la-rama
```

🎉 ¡Listo! Ahora han aprendido a trabajar con ramas en Git.

¡Felicitaciones, chicas! 🎉 Han completado una práctica súper importante sobre Git y GitHub. Estas herramientas serán clave en su camino como desarrolladoras, así que sigan practicando y perfeccionando su uso. ¡Buen trabajo y mucho éxito! 🚀

Desde este punto, nos olvidamos de Figma, **¡y subiremos todo lo que hagamos en clase (ya sean ejercicios o ejemplos) Directamente a GitHub!**