Minimanual de uso de GitHub para desarrollo de aplicaciones web

1. ¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo que permite almacenar y gestionar proyectos de código fuente, utilizando **Git** como sistema de control de versiones. GitHub facilita el trabajo en equipo, el control de cambios y la colaboración en proyectos.

2. Crear una cuenta en GitHub

1. Visita GitHub.
2. Haz clic en "Sign up" (Registrarse).
3. Rellena los campos necesarios: nombre de usuario, correo electrónico y contraseña.
4. Verifica tu correo electrónico para completar el registro.

3. Crear tu primer repositorio

1. Una vez que inicies sesión, en la página principal haz clic en el botón verde **"New"** para crear un nuevo repositorio.
2. Escribe un nombre para tu repositorio (ejemplo: `mi-proyecto-web`).
3. Opcionalmente, puedes agregar una descripción.
4. Elige la visibilidad del repositorio:
    - **Public**: Cualquiera puede ver tu código.
    - **Private**: Solo tú y las personas que invites pueden verlo.
5. Si es un proyecto nuevo, marca **"Initialize this repository with a README"**.
6. Haz clic en **"Create repository"**.

4. Configurar Git en tu computadora

1. **Instala Git**:
    - En Windows, puedes descargarlo desde git-scm.com.
    - En macOS, usa el terminal con el comando `brew install git`.
    - En Linux, usa el comando `sudo apt install git`.
2. **Configura tu nombre y correo** (estos serán los identificadores de tus cambios):

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu@correo.com"
```

3. **Clonar tu repositorio** (hacer una copia local en tu máquina):
   En GitHub, dentro de tu repositorio, haz clic en el botón verde **"Code"** y copia la URL (HTTPS).
   En tu terminal:

   ```
   git clone https://github.com/tu-usuario/mi-proyecto-web.git
   ```

5. Hacer cambios en tu proyecto

1. Navega hasta la carpeta del repositorio en tu máquina:

   ```
   cd mi-proyecto-web
   ```

2. Crea o edita archivos, por ejemplo:
   - Crea un archivo `index.html`.
   - Haz modificaciones en el `README.md`.

6. Guardar y subir cambios a GitHub

1. **Agregar cambios al staging area**:
   Después de editar tus archivos, debes agregar los cambios al área de preparación (staging):

   ```
   git add .
   ```

2. **Confirmar los cambios (commit)**:
   Una vez que los cambios estén listos, realiza un commit con un mensaje descriptivo:

   ```
   git commit -m "Agregado el archivo index.html"
   ```

3. **Subir los cambios a GitHub (push)**:
   Finalmente, sube tus cambios al repositorio en GitHub:

   ```
   git push origin main
   ```

   (Si estás trabajando en una rama diferente, reemplaza `main` por el nombre de tu rama).

7. Colaboración en equipo

Cómo trabajar con ramas

1. **Crear una nueva rama**:
   Si estás trabajando en una nueva funcionalidad, crea una nueva rama:

   ```
   git checkout -b nueva-funcionalidad
   ```

2. **Hacer cambios y subirlos**:
   Realiza tus cambios y luego sigue el proceso de `git add`, `git commit` y `git push` como se mostró antes.

3. **Crear un Pull Request (PR)**:
   - Ve a tu repositorio en GitHub.

- Haz clic en **"Compare & pull request"** para abrir un PR.
- Agrega una descripción de lo que cambiaste.
- Solicita revisión de tus compañeros o profesor.

Cómo revisar y aceptar Pull Requests

Si alguien hace un PR, puedes revisarlo, ver los cambios y luego **fusionarlo (merge)** a la rama principal (generalmente `main` o `master`).

8. Otras acciones importantes

Actualizar tu repositorio local (pull)

Si estás trabajando con otras personas y quieres traer los últimos cambios del repositorio de GitHub:

```
git pull origin main
```

Ver el historial de cambios (log)

Puedes ver los commits que se han hecho:

```
git log
```

Deshacer cambios

Si cometiste un error, puedes deshacer cambios específicos:

- Para deshacer un archivo agregado al staging:

  ```
  git reset archivo.html
  ```

- Para deshacer el último commit (pero conservar los cambios):

  ```
  git reset --soft HEAD~1
  ```

9. Consejos útiles

- **Commits pequeños y frecuentes**: Es mejor hacer commits con cambios pequeños y descriptivos.
- **No subas archivos grandes**: Usa `.gitignore` para evitar subir archivos innecesarios (como dependencias, archivos temporales, etc.).
- **Documentación**: Es importante que escribas buenos mensajes de commit y documentes tu proyecto (por ejemplo, en el archivo `README.md`).

10. Recursos adicionales

- Documentación oficial de GitHub
- Tutorial de Git