

Trabajo Colaborativo en GitHub - Guía Completa

1. Crear el Repositorio (Una sola persona)

En GitHub:

1. Accede a github.com y haz clic en "New repository"
2. Dale un nombre descriptivo (ej: `(proyecto-conjunto-clase)`)
3. Añade una descripción
4. Elige "Public" o "Private" según prefieras
5. **Importante:** Inicializa con README, .gitignore (elige tu lenguaje) y licencia
6. Crea el repositorio

Invitar al compañero:

1. En la página del repositorio, ve a "Settings"
2. Busca "Collaborators" en el menú izquierdo
3. Haz clic en "Add people"
4. Escribe el username de GitHub de tu compañero
5. Elige el nivel de acceso (recomendado: Maintain)
6. Envía la invitación

2. Primeras Configuraciones (Los dos)

Clonar el repositorio:

```
bash  
git clone https://github.com/usuario/proyecto-conjunto-clase.git  
cd proyecto-conjunto-clase
```

Configurar Git localmente (si no lo has hecho):

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

3. Trabajar con Ramas - El Flujo Básico

Crear tu rama personal:

Cada uno debe tener su propia rama. Nombres recomendados:

- `feature/nombre-del-compañero` (ej: `feature/juan`)
- `dev/nombre-del-compañero` (ej: `dev/maria`)

bash

Ver todas las ramas

`git branch -a`

Crear una nueva rama basada en main

`git checkout -b feature/tu-nombre`

O si usas Git moderno

`git switch -c feature/tu-nombre`

Enviar tu rama al repositorio remoto:

bash

`git push -u origin feature/tu-nombre`

El `-u` vincula tu rama local con la remota automáticamente.

4. Flujo de Trabajo Diario

Trabajar en tu rama:

bash

Asegúrate de estar en tu rama

git branch

Hacer cambios en los archivos...

Ver qué ha cambiado

git status

Preparar los cambios para guardar

git add .

O añadir archivos específicos

git add archivo1.js archivo2.js

Guardar los cambios con un mensaje descriptivo

git commit -m "Añado función de login"

Enviar los cambios al servidor

git push

Mensajes de commit buenos:

- Añado validación de formulario
- Corrijo bug en la función calcular
- Actualizo estilos CSS del header
- cambios (muy vago)
- asdf (inaceptable)

5. Mantener tu Rama Actualizada

Ver cambios de tu compañero:

bash

Traer cambios del servidor sin aplicarlos

git fetch

Ver cambios en la rama principal

git log origin/main --oneline

Actualizar tu rama local con cambios de main:

```
bash
```

```
# Opción 1: Rebase (recomendado, mantiene el historial limpio)
```

```
git fetch origin
```

```
git rebase origin/main
```

```
# Opción 2: Merge (más fácil si hay conflictos)
```

```
git fetch origin
```

```
git merge origin/main
```

Si hay conflictos, Git te lo dirá. Resuélvelos manualmente en los archivos.

6. Crear un Pull Request (Fusionar tu trabajo)

Cuando termines una tarea importante:

1. Asegúrate de que todo esté guardado:

```
bash
```

```
git status
```

2. Sube los últimos cambios:

```
bash
```

```
git push
```

3. Ve a GitHub y verás un botón "Compare & pull request"
4. Añade una descripción clara de qué cambios hiciste
5. Pide que tu compañero revise el código (code review)
6. Una vez aprobado, haz clic en "Merge pull request"
7. Después, baja los cambios nuevos:

```
bash
```

```
git pull origin main
```

7. Comandos Útiles de Rescate

Ver el historial de cambios:

bash

Últimos 5 commits en tu rama actual

git log --oneline -5

Historial visual de todas las ramas

git log --graph --oneline --all

Deshacer cambios antes de commit:

bash

Ver cambios no guardados

git diff

Deshacer cambios en un archivo específico

git checkout -- archivo.js

O con Git moderno

git restore archivo.js

Deshacer todos los cambios no guardados

git reset --hard HEAD

Deshacer el último commit (si aún no lo subiste):

bash

git reset --soft HEAD~1

Ver qué cambió en un commit específico:

bash

git show nombre-del-commit

8. Puntos Importantes a Recordar

- ✓ **Buenas prácticas:**

1. **Siempre trabaja en tu rama**, nunca directamente en `main`
2. **Haz commits frecuentes** (cada tarea pequeña completada)
3. **Sube tu trabajo regularmente** con `git push` para no perderlo
4. **Actualiza tu rama** antes de hacer merge con cambios de `main`
5. **Escribe mensajes de commit claros** (el futuro tú te lo agradecerá)
6. **Comunica con tu compañero** sobre qué parte cada uno está desarrollando
7. **Revisa los cambios ajenos** antes de hacer merge (code review)

⚠ Evita:

- Hacer cambios directamente en `main` (crea conflictos)
- Trabajar en archivos simultáneamente sin comunicación
- Hacer commits enormes sin mensajes claros
- Olvidar hacer `git push` (tus cambios se quedan solo en tu computadora)
- Renombrar o eliminar archivos sin avisar a tu compañero

9. Si Surge un Conflicto

Los conflictos ocurren cuando dos personas modifican la misma línea de código.

Ejemplo de conflicto en un archivo:

```
javascript

<<<<<< HEAD
console.log("Cambios de Juan");
=====
console.log("Cambios de María");
>>>>> feature/maria
```

Cómo resolverlo:

1. Abre el archivo y elimina los marcadores `<<<<<<`, `=====`, `>>>>>>`
2. Decide qué código mantener
3. Guarda el archivo
4. Ejecuta:

bash

```
git add archivo-resuelto.js  
git commit -m "Resuelvo conflicto en archivo-resuelto.js"  
git push
```

10. Resumen: Secuencia Típica de una Sesión

bash

Inicio

```
git switch feature/tu-nombre    # Cambia a tu rama  
git pull                         # Descarga cambios recientes
```

Desarrollo

(editas archivos)

```
git status                      # Revisa qué cambió  
git add .                        # Prepara cambios  
git commit -m "Tu mensaje"      # Guarda localmente  
git push                          # Envía al servidor
```

Fin de sesión

```
git log --oneline -3            # Verifica que todo está guardado
```

11. Contactos Útiles

- **Documentación oficial:** <https://docs.github.com>
- **Git cheat sheet:** <https://github.github.com/training-kit/>
- **Resuelve dudas juntos:** Comunícate con tu compañero frecuentemente