Flujo de ramas (simple y práctico)

Este proyecto usa un flujo sencillo basado en dos ramas permanentes:

- main: producción (lo estable). Solo se mergea vía PR.
- develop: integración (trabajo diario). Se mergean features aquí.

Y ramas temporales:

- feature/nombre: para cada mejora o bug no crítico.
- hotfix/nombre: corrección urgente que parte desde main.
- release/x.y.z (opcional): preparación de versión.

Requisitos previos:

- Remoto principal se llama `main` (ya configurado)
- Tu rama local base es `main` y existe `develop`

Comandos básicos (PowerShell)

- Actualizar tu entorno antes de empezar trabajo:

```
# Traer cambios del remoto
git fetch main
# Ir a develop y actualizarlo
git checkout develop
git pull --ff-only
```

- Crear una feature desde develop:

```
git checkout -b feature/mi-mejora develop
# ... trabaja, commitea en esta rama ...
# Publica la rama para abrir PR
git push -u main feature/mi-mejora
```

- Abrir Pull Request
    - En GitHub abre un PR: base = `develop`, compare = `feature/mi-mejora`.
    - Pide revisión, ajusta si es necesario.
- Terminar la feature (tras aprobar PR):
    - Mergea en GitHub (Squash o Merge normal). Luego:

```
# Actualiza localmente
git fetch main
git checkout develop
git pull --ff-only
# Opcional: borrar rama local y remota si ya se mergeó
git branch -d feature/mi-mejora
git push main :refs/heads/feature/mi-mejora
```

- Preparar un release (opcional):

```
git checkout -b release/1.2.0 develop
# Ajusta versión, changelog, pruebas, etc.
# Publica
git push -u main release/1.2.0
# Abre PR: base = main, compare = release/1.2.0
```

- Hotfix urgente desde producción:

```
# Parte desde main
git fetch main
git checkout main
git pull --ff-only
# Crea la rama de hotfix
git checkout -b hotfix/critico-500 main
# ... arregla, commit ...
# Publica
git push -u main hotfix/critico-500
# Abre PR: base = main, compare = hotfix/critico-500
# Tras mergear a main, también mergea a develop (PR o fast-forward):
git checkout develop
git pull --ff-only
git merge --ff-only main
git push
```

Consejos rápidos

- Usa `--ff-only` al hacer pull/merge para evitar merges innecesarios.
- Un commit pequeño y claro > un mega commit.
- Nombra ramas con prefijos consistentes: `feature/`, `hotfix/`, `release/`.
- Antes de empezar una rama, asegúrate que `develop` o `main` están actualizadas.

Regla local para proteger `main` (pre-push)

Para evitar pushes accidentales a `main`, este repo incluye un hook pre-push que bloquea el push directo a `main`.

1. Activar hooks (una sola vez por clon):

```
npm run hooks:enable
```

1. Verificar estado:

```
npm run hooks:status
```

1. Si realmente necesitas forzar un push a main (no recomendado), define la variable y reintenta:

```
$env:ALLOW_PUSH_TO_MAIN=1; git push
```

1. Desactivar hooks (si lo requieres):

```
npm run hooks:disable
```

Cheatsheet mínimo

```
# Crear rama de trabajo
git checkout -b feature/nombre develop
# Publicarla
git push -u main feature/nombre
# Volver a develop y actualizar
git checkout develop; git pull --ff-only
# Borrar rama remota tras merge
git push main :refs/heads/feature/nombre
```