Guía

Git & GitHub Cheat Sheet — Control de Versiones para Equipos de 20 Personas

Sistema Blockchain Modular | Feature Branch Workflow

SECCIÓN 0 — DEFINICIÓN CRÍTICA: QUÉ ES UN PULL REQUEST

Un **Pull Request (PR)** es una solicitud formal y revisada para fusionar código de una rama de módulo hacia `main`. No es un comando de Git: es un mecanismo de control en GitHub que interpone una capa de revisión humana antes de cualquier fusión.

Por qué es el único mecanismo seguro para este equipo:

Problema sin PR	Solución con PR
`git merge` directo sobreescribe cambios de otras parejas sin advertencia	GitHub bloquea la fusión si hay conflictos no resueltos
No hay registro de quién aprobó qué cambio	Cada PR genera un hilo de revisión auditado y permanente
Dos parejas pueden fusionar el mismo archivo simultáneamente	GitHub detecta conflictos entre ramas antes de que el daño ocurra en `main`
Un módulo roto contamina `main` para los 20 integrantes	`main` permanece intacto hasta aprobación explícita

Regla absoluta: `main` es de solo lectura para todos. La única escritura permitida hacia `main` es mediante PR aprobado.

SECCIÓN 1 — CONFIGURACIÓN INICIAL (Ejecutar una sola vez por máquina)

Concepto	Comando exacto	Explicación técnica
Identidad de autor	`git config --global user.name "Nombre Apellido"`	Asocia cada commit al autor real; requerido para trazabilidad en GitHub
Email de autor	`git config --global user.email "email@dominio.com"`	Debe coincidir con el email de la cuenta GitHub del integrante
Verificar configuración	`git config --global --list`	Lista todos los valores de configuración global activos
Clonar repositorio	`git clone https://github.com/org/repo.git`	Descarga el repositorio remoto completo y configura `origin` automáticamente

---

SECCIÓN 2 — ESTRUCTURA DE RAMAS (Nomenclatura obligatoria)

```
main
├── modulo-01-identidad
├── modulo-02-consenso
├── modulo-03-transacciones
├── modulo-04-bloques
├── modulo-05-red
├── modulo-06-contratos
├── modulo-07-almacenamiento
├── modulo-08-api
├── modulo-09-seguridad
└── modulo-10-interfaz
```

Patrón de nombre: `modulo-[NN]-[descriptor-kebab-case]` **Prohibición explícita:** Ningún integrante ejecutará `git push origin main` ni `git merge` directo a `main` bajo ninguna circunstancia.

SECCIÓN 3 — OPERACIONES DIARIAS (Flujo de trabajo por pareja)

3.1 — Inicio de sesión de trabajo

Concepto	Comando exacto	Explicación técnica
Posicionarse en rama del módulo	`git switch modulo-NN-descriptor`	Cambia el directorio de trabajo a la rama del módulo asignado
Sincronizar `main` remoto local	`git fetch origin`	Descarga cambios remotos sin modificar ningún archivo local
Actualizar rama del módulo con `main`	`git merge origin/main`	Integra cambios aprobados de `main` en la rama local del módulo
Verificar estado actual	`git status`	Muestra archivos modificados, en staging y conflictos activos
Verificar rama activa	`git branch --show-current`	Confirma en qué rama se está operando antes de cualquier commit

3.2 — Registro de cambios (Commit)

Concepto	Comando exacto	Explicación técnica
Ver diferencias no staged	`git diff`	Muestra línea por línea qué cambió en archivos aún no agregados al staging
Ver diferencias en staging	`git diff --staged`	Muestra qué está preparado exactamente para el próximo commit
Agregar archivo específico	`git add ruta/al/archivo.ext`	Agrega un archivo concreto al staging area; evita agregar archivos no relacionados
Agregar todos los cambios	`git add .`	Agrega todos los archivos modificados del directorio actual al staging
Crear commit	`git commit -m "modulo-NN: descripción técnica del cambio"`	Registra el snapshot con mensaje estructurado; el prefijo del módulo es obligatorio
Verificar historial	`git log --oneline --graph`	Muestra el árbol de commits de forma compacta para detectar divergencias

3.3 — Publicación de cambios

Concepto	Comando exacto	Explicación técnica
Primer push de rama nueva	`git push --set-upstream origin modulo-NN-descriptor`	Publica la rama por primera vez y vincula el tracking remoto
Push subsiguientes	`git push origin modulo-NN-descriptor`	Envía commits locales al remoto en la misma rama del módulo

---

SECCIÓN 4 — CREACIÓN Y GESTIÓN DE RAMAS

Concepto	Comando exacto	Explicación técnica
Crear rama de módulo desde `main`	`git switch main && git pull origin main && git switch -c modulo-NN-descriptor`	Garantiza que la rama nueva parte del estado más reciente de `main`
Listar todas las ramas (locales y remotas)	`git branch -a`	Muestra el mapa completo de ramas existentes para evitar duplicados
Eliminar rama local ya fusionada	`git branch -d modulo-NN-descriptor`	Elimina rama local únicamente si ya fue fusionada; Git rechaza la operación si no lo está
Eliminar rama remota ya fusionada	`git push origin --delete modulo-NN-descriptor`	Elimina la rama del servidor remoto tras aprobación del PR correspondiente

---

SECCIÓN 5 — RESOLUCIÓN DE CONFLICTOS

Concepto	Comando exacto	Explicación técnica
Identificar archivos en conflicto	`git status`	Los archivos en conflicto aparecen marcados como `both modified`
Abrir archivo en conflicto	*(editor de texto)*	Los marcadores `<<<<<<<`, `=======`, `>>>>>>>` delimitan las versiones en disputa

Concepto	Comando exacto	Explicación técnica
Marcar conflicto como resuelto	`git add ruta/al/archivo.ext`	Indica a Git que el conflicto fue resuelto manualmente; requerido antes del commit
Completar merge tras resolución	`git commit -m "modulo-NN: resuelve conflicto con main - descripción"`	Finaliza el merge con registro explícito de la resolución
Abortar merge en curso	`git merge --abort`	Revierte el estado al punto anterior al inicio del merge; no deja cambios residuales

Anatomía de un conflicto:

```
<<<<<<< HEAD  (tu versión local en la rama del módulo)
código de tu pareja
=======
código proveniente de main o de otra rama
>>>>>>> origin/main
```

Eliminar los marcadores y conservar el código correcto es responsabilidad de la pareja propietaria del módulo.

SECCIÓN 6 — PROCEDIMIENTO DE PULL REQUEST (GitHub)

Estos pasos se ejecutan en la interfaz web de GitHub, no en la terminal.

Paso	Acción en GitHub	Criterio obligatorio
1	Ir a la pestaña **Pull Requests** → **New Pull Request**	Base: `main` | Compare: `modulo-NN-descriptor`
2	Completar el título con formato: `[MOD-NN] Descripción técnica del cambio`	El título debe identificar el módulo sin ambigüedad
3	Asignar mínimo 1 revisor externo a la pareja (integrante de otro módulo)	El autor del PR no puede ser su propio aprobador
4	Esperar aprobación explícita mediante botón **Approve**	Un comentario sin Approve no habilita la fusión
5	Resolver todos los comentarios marcados como **Request Changes**	El estado debe ser **Approved** sin cambios pendientes

Paso	Acción en GitHub	Criterio obligatorio
6	Fusionar con botón **Squash and Merge** o **Merge Pull Request**	Usar **Squash and Merge** para mantener historial de `main` limpio
7	Eliminar la rama del módulo desde GitHub tras la fusión	Botón **Delete Branch** visible inmediatamente tras la fusión

---

SECCIÓN 7 — COMANDOS DE DIAGNÓSTICO Y RECUPERACIÓN

Concepto	Comando exacto	Explicación técnica
Ver quién modificó cada línea de un archivo	`git blame ruta/al/archivo.ext`	Muestra commit y autor por cada línea; útil para identificar origen de conflictos
Buscar en qué commit apareció un texto	`git log -S "texto_buscado" --oneline`	Localiza el commit donde se introdujo o eliminó una cadena de texto específica
Ver cambios de un commit específico	`git show [hash-del-commit]`	Muestra el diff completo y los metadatos del commit identificado por su hash
Deshacer último commit sin perder cambios	`git reset --soft HEAD~1`	Revierte el commit pero mantiene los cambios en staging; no altera el historial remoto
Descartar cambios locales de un archivo	`git restore ruta/al/archivo.ext`	Restaura el archivo al estado del último commit; los cambios locales se pierden permanentemente
Guardar cambios sin hacer commit	`git stash push -m "modulo-NN: descripción"`	Almacena cambios en progreso para cambiar de contexto sin perder trabajo
Recuperar cambios guardados en stash	`git stash pop`	Restaura el último stash guardado al directorio de trabajo

---

SECCIÓN 8 — REGLAS DE OPERACIÓN DEL EQUIPO (No negociables)

```
PROHIBIDO:  git push origin main
PROHIBIDO:  git merge [cualquier-rama] estando en main local
PROHIBIDO:  git commit --amend en commits ya pusheados
PROHIBIDO:  git push --force  (sin --force-with-lease como mínimo)
OBLIGATORIO: git fetch origin + git merge origin/main al inicio de cada
sesión
OBLIGATORIO: PR con mínimo 1 aprobador antes de fusionar cualquier módulo
OBLIGATORIO: Nomenclatura de rama: modulo-NN-descriptor
OBLIGATORIO: Prefijo de módulo en cada mensaje de commit
```

Referencia de sintaxis: Git 2.45+ | GitHub.com | Estándar Feature Branch Workflow (Atlassian/GitHub Flow)