Guía Rápida: Estructura, Ramas y Flujo de Push/PR

Proyecto videojuego (Flutter · Flame · Firebase)

1) Estructura del repositorio (monorepo)

```
/flutter/ — Cliente móvil Flutter (UI, widgets, assets, integración)
/flame/ — Lógica del juego (paquetes, escenas, sprites, assets de juego)
/firebase/ — Funciones, reglas de seguridad, hosting, configuraciones
/docs/ — GDD, decisiones técnicas, actas, diagramas
```

Archivo .gitignore recomendado (raíz)

```
# Flutter/Dart
.dart_tool/
build/
ios/Pods/
android/.gradle/
android/app/build/
# Firebase
firebase/functions/node_modules/
firebase/.firebase/
# IDEs
.vscode/
.idea/
```

2) Esquema de ramas

- main → estable (releases). No se hace push directo; solo merge por PR.
- dev → integración diaria. Recibe PR desde team/* tras verificación.
- team/flutter, team/flame, team/firebase → trabajo continuo por equipo.
- feature// → ramas cortas que nacen desde su rama de equipo.

Flujo recomendado

```
Flujo: feature/* → team/ → dev → main (release)
```

3) Convenciones de commits y nombres de ramas

Formato de commit (Conventional Commits):

Formato	Ejemplo
<tipo>(<área>): <cambio>	feat(flutter): vista detalle de carta

Tipos útiles: feat, fix, refactor, docs, chore, test, perf.

Nombres de ramas cortos y descriptivos:

```
team/flutter, team/flame, team/firebase
feature/flutter/detalle-carta
feature/flame/sistema-combate
feature/firebase/reglas-lectura
```

4) Push y Pull Requests (Sublime Merge y CLI)

1. Trabaja tus cambios en una rama de feature de tu equipo.
2. Haz commits pequeños, con mensajes claros y verificables.
3. Haz push de la rama a origin la primera vez con 'Track' (Sublime Merge) o '-u' (CLI).
4. Abre un PR hacia team/ para revisión del equipo; realiza ajustes.
5. Luego abre PR de team/ → dev; tras validación cruzada, se integra.
6. Para publicar, se hace PR de dev → main y se taguea la versión (release).

Comandos útiles (CLI):

```
# Crear y subir rama de equipo (una sola vez por equipo)
git checkout -b team/flutter dev
git push -u origin team/flutter
# Crear una feature (Flutter)
git checkout team/flutter
git pull
git checkout -b feature/flutter/detalle-carta
# ... commits ...
git push -u origin feature/flutter/detalle-carta
```

5) Reglas en GitHub (recomendado)

- Protección de ramas: exigir PR, al menos 1–2 approvals y checks de CI en main y dev.
- CODEOWNERS por carpeta (/flutter, /flame, /firebase) para asignar revisores automáticos.
- Plantillas de PR y CI por carpeta (linter/pruebas de Flutter y Functions).

6) Ejemplos de commits y PR

- Commit: feat(flame): sistema-combate básico con sprites y cooldown
- Commit: fix(firebase): corrige reglas de lectura para colección scores
- PR título (feature→team): Flutter – Detalle de Carta (UI + navegación)
- PR descripción: Resumen, capturas, 'Cómo probar', checklist y riesgos.

Mantén ramas pequeñas, PRs frecuentes y mensajes de commit claros. ¡Buen desarrollo!