

Guía Profesional v5.0 – Flujo Completo con Ejemplos por Proyecto

Complemento del flujo v4.0 con ejemplos concretos para cada proyecto (Signal Mapper, Canalizaciones, control_documental_mvp y SSAA). Cada comando incluye una explicación breve para usar esta guía como manual universal.

Reglas universales

- **Un repo por proyecto:** Cada carpeta/proyecto mantiene su propio historial y releases.
- **Un .venv por proyecto:** Evita conflictos de librerías. VS Code debe apuntar al intérprete de ese proyecto.
- **main = estable:** En main vive lo validado. Los tags estables se crean aquí.
- **develop = integración:** Aquí se integran branches. Cuando develop está validado, se promueve a main.

Capítulo 1 – Flujo completo universal (para cualquier proyecto)

Receta de punta a punta: tarea → PR → main estable → tag → Release en GitHub.

Comando	Qué hace / por qué
<code>git status</code>	Muestra rama actual y cambios pendientes.
<code>git checkout develop</code>	Te cambia a develop (base de trabajo recomendada).
<code>git pull --rebase origin develop</code>	Trae lo último desde GitHub y mantiene historial limpio (evita merges innecesarios).
<code>git checkout -b feature/nombre-claro</code>	Crea rama nueva para aislar la tarea.
<code>git diff</code>	Revisa cambios antes de hacer commit.
<code>git add .</code>	Agrega cambios al stage (listos para commit).
<code>git commit -m "feat(modulo): descripcion"</code>	Crea un snapshot trazable del cambio.
<code>git pull --rebase origin develop</code>	Te alineas con el remoto antes de publicar/integrar.
<code>git push -u origin feature/nombre-claro</code>	Sube la rama al remoto y configura seguimiento.
GitHub: Pull Request (base: develop)	Integra tu rama a develop con revisión y control.
<code>git checkout develop</code>	Vuelves a develop luego del merge del PR.
<code>git pull --rebase origin develop</code>	Actualizas develop local con lo mergeado.
<code>git checkout main</code>	Vas a la rama estable.
<code>git pull --rebase origin main</code>	Sincronizas main local con GitHub antes de promover.
<code>git merge --no-ff develop</code>	Promueves develop a main dejando un merge explícito (trazabilidad de release).
<code>git push origin main</code>	Publicas la versión estable en GitHub.
<code>git tag -a vX.Y.Z -m "Release vX.Y.Z"</code>	Marca el commit exacto como versión estable (tag anotado).
<code>git push origin vX.Y.Z</code>	Publica el tag en GitHub.
GitHub: Releases → Draft a new release	Crea el Release oficial asociado al tag, con notas y publicación.

Capítulo 2 – Signal Mapper (conservador / estable)

Prioriza estabilidad. Taggea versiones validadas con frecuencia moderada.

Cómo decides “estable”: Estable = abrir app, cargar/guardar plantillas JSON, operaciones de mapeo clave sin crash.

Comandos típicos y explicación:

Comando	Qué hace / por qué
<code>git checkout -b feature/remap-signals</code>	Aísla una nueva función de remapeo.
<code>git commit -m "feat(signals): add remap rules"</code>	Deja trazado el cambio por módulo.
<code>git tag -a v1.2.0 -m "Signal Mapper estable v1.2.0"</code>	Marca versión estable (solo en main).

Esquema de versiones recomendado: Versiones típicas: v1.2.0 (estable), v1.2.1 (hotfix), v1.3.0 (features).

Capítulo 3 – Canalizaciones (altamente iterativo)

Iteración rápida. Usa pre-releases alpha/beta antes de llegar a v1.0.0.

Cómo decides “estable”: Estable = no crashea al dibujar/editar, guardar/cargar escena OK, export PDF OK.

Comandos típicos y explicación:

Comando	Qué hace / por qué
<code>git checkout -b feature/canvas-background</code>	Cambio de alto impacto (scene/view) aislado.
<code>git commit -m "feat(canvas): add locked background image"</code>	Snapshot para revertir rápido si algo falla.
<code>git tag -a v0.9.0-alpha -m "Canalizaciones alpha (canvas bg)"</code>	Pre-release para pruebas antes de estabilidad total.

Esquema de versiones recomendado: Versiones típicas: v0.9.0-alpha → v0.9.0-beta → v1.0.0 (estable).

Capítulo 4 – control_documental_mvp (entregables / trazabilidad)

Alta trazabilidad. Tags por cada entrega/hotfix.

Cómo decides “estable”: Estable = carga Excel, filtros/semáforos correctos, gráficos correctos, ordenamientos OK.

Comandos típicos y explicación:

Comando	Qué hace / por qué
<code>git tag -a v2.6.3 -m "Entrega v2.6.3"</code>	Marca exactamente lo entregado.
<code>git checkout -b hotfix/fix-chart-percent</code>	Rama de corrección urgente.
<code>git commit -m "fix(chart): correct percent calculation"</code>	Registra el hotfix de forma clara.
<code>git tag -a v2.6.4 -m "Hotfix v2.6.4"</code>	Nueva versión estable tras corrección.

Esquema de versiones recomendado: Versiones típicas: v2.6.3 (entrega) → v2.6.4 (hotfix) → v2.7.0 (mejora).

Capítulo 5 – SSAA (crítico / flujo estricto)

Control total. Branches obligatorios y tags solo tras validación completa.

Cómo decides “estable”: Estable = persistencia OK (nombres de escenarios), cálculos consistentes, exportables OK, UI sin errores.

Comandos típicos y explicación:

Comando	Qué hace / por qué
<code>git checkout -b refactor/model-ui-separation</code>	Refactor aislado (alto riesgo) sin mezclar con features.
<code>git commit -m "refactor(model): separate UI from calculations"</code>	Commit atómico de cambio estructural.
<code>git tag -a v1.0.0 -m "SSAA v1.0.0 estable"</code>	Marca versión estable validada (solo en main).

Esquema de versiones recomendado: Versiones típicas: v1.0.0 (estable) → v1.0.1 (hotfix) → v1.1.0 (features).

Capítulo 6 – Plantilla universal de Release en GitHub

Para que el Release sirva siempre, usa un formato consistente de notas.

Comando	Qué hace / por qué
Título (Release title)	Usa el mismo nombre del tag, por ejemplo: v1.0.0.
Resumen	1–2 líneas: qué habilita esta versión y para quién.
Added / Changed / Fixed	Lista corta de cambios (3–10 bullets).
Impacto / migración	Si hay cambios de estructura o config, indica pasos concretos.
Validación	Qué probaste: abrir, guardar, exportar, casos críticos.

Ejemplo de notas:

Added: imagen base bloqueada en canvas

Fixed: crash al conectar wires desalineados

Improved: export PDF con fondo y items

Validación: crear proyecto → dibujar → guardar/cargar → export PDF