PDF SUB-PROCESO · TRELLO EDITOR

Flow: Escalado Formatos · 5 similares + 1 arriesgado (factory_v4.html)

Station ID	trello_editor
Skill canónica	creador-trello-editor-videos
Agent .md	.claude/agents/escalado_formatos/creador-trello-editor-videos.md
Verifier L1	verify_trello_card_complete.py (6 attachments + descripción no vacía + label correcto)
Supervisor L2 skill	Detectar si la tarjeta está incompleta o mal formateada
Modelo default	claude-opus-4-7 (msg 4658)
Master PDF flow	_documentacion/PDF_MAESTRO_ESCALADO_VIDEO_v16.pdf · Flow D "Concepto ganador
Generado	2026-05-09 20:26

Especificación canónica de la skill (verbatim del agente .md)

Esto es lo que el agente DEBE hacer literalmente. El supervisor de control de calidad valida cada output contra esta spec.

Agente CREADOR-TRELLO-EDITOR-VIDEOS

Misión

Empaquetar los 6 finalistas en una tarjeta Trello clara y completa para que el editor humano de video la procese.

Inputs

- `finalists_path`: `_finalists.json` con los 6 elegidos.
- `analysis_path`: `_analysis.json` del ganador (contexto).
- `product`: nombre del producto.
- `producto_imagen_path`: foto del producto para cover.

Output

JSON `_trello_card.json`:

```
{
  "card_id": "...",
  "short_url": "https://trello.com/c/...",
  "board_id": "rH2kazG2",
  "title": "<Producto> · Escalado Formatos · 6 referentes (5 similares + 1 arriesgado) · YYYY-MM-DD",
  "attachments_count": 13,
  "ping_telegram_sent": true
}
```

Procedimiento

1. Cargar `_finalists.json` y `_analysis.json`.

2. Para CADA finalista, generar PDF brief con estructura canónica REGLA brief_editor_3_secciones:

   • **Sección 1 — Video referencia**: URL + brand + país + días activos + escalado evidencia + thumbnail.

   • **Sección 2 — Explicación**: por qué este ref para este escalado (awareness match + sof + cluster driver + transmutación necesaria).

   • **Sección 3 — Speech como prosa continua**: sin bullets/timestamps/notas, abre con 13 palabras hook nativo, incluye garantía + mecanismo + 1-2 reviews en prosa natural, cierra con CTA COD canónica "pagas en la puerta de tu casa cuando llega", cero precio.

3. Crear tarjeta en Trello board `rH2kazG2` (Editor de Videos):

`python

POST https://api.trello.com/1/cards

params: {idList: <input list id>, name: <title>, desc: <descripción detallada>, pos: top}

`

4. Adjuntar:

   • Cover: producto_imagen.jpg con `setCover=true`

   • 6 mp4: video paths de los finalistas

   • 6 PDFs brief: uno por finalista

5. Enviar ping Telegram CEO con short_url + resumen de los 6 finalistas.

REGLAs canónicas a respetar

- REGLA #71: card.name MUST contener nombre del producto actual.
- REGLA #61: tono peninsular neutro en el speech, sin tacos.
- REGLA #63: COD canónico literal "pagas en la puerta de tu casa cuando llega".
- REGLA brief_editor_3_secciones: estructura PDF estricta.

Verifier L1

`verify_trello_card_complete.py`:

- card.name contiene producto
- ≥ 13 attachments (cover + 6 mp4 + 6 PDFs)
- desc no vacía
- short_url devuelta no vacía
- Telegram ping enviado

Restricciones

- NO crear card si <6 finalistas válidos (debería haber fallado en COMPARADOR ya).
- NO subir mp4 corruptos (validar antes con ffprobe).
- NO modificar el speech original del ganador.
- Modelo Opus 4.7 OBLIGATORIO (msg 4658 Fer 2026-05-09): aunque la tarea de crear card es mecánica, generar los 6 PDFs brief con estructura canónica 3 secciones requiere razonamiento de calidad. En duda, siempre Opus.

CHECKLIST DE AUDITORÍA · Supervisor de Control de Calidad

El supervisor de este sub-proceso recorre estos checks por cada ejecución. Si CUALQUIERA falla → BREAK con razón concreta y el agente reintenta con el feedback inline (REGLA #66, max_retries=5).

1. El output existe en la sección caja.trello_editor y NO está vacío.
2. Verifier L1 determinista pasa: `verify_trello_card_complete.py (6 attachments + descripción no vacía + label correcto)`.
3. Supervisor L2 olfato (Haiku): Detectar si la tarjeta está incompleta o mal formateada.
4. REGLAs duras del agente .md NO violadas (ver sección 'Restricciones' del agente).
5. Si paralelo: el shared_state del cluster fue consultado antes de actuar y actualizado después.
6. Default Opus 4.7 obligatorio (msg 4658) — sin fallback Haiku para razonamiento.
7. Si BREAK → escalación L3 Opus + escribir caso al CBR del supervisor.

Cascada de escalación (Andon Tier · REGLA #146)

• L1 verifier determinista (Python · 0 tokens) — runs SIEMPRE.

• L2 supervisor Haiku olfato — runs SIEMPRE en paralelo, captura lo nuevo.

• L3 supervisor sub-proceso Opus — runs si L1 FAIL o L2 FLAG. Resuelve y añade al CBR del sub-proceso.

• L4 Master Mejora Continua Opus — runs si ≥3 L3 escaladas en 24h o cron diario. Cross-pattern detection.

• L5 Fer humano — runs solo si L4 propone REGLA nueva o caso sin precedente.

Referencia al Master PDF

Este sub-proceso opera dentro del flow definido en **_documentacion/PDF_MAESTRO_ESCALADO_VIDEO_v16.pdf · Flow D "Concepto ganador - 5 similares + 1 fuera de caja"**. Cualquier conflicto entre este PDF y el master → manda el master.

Si Fer modifica el master (ej: añade flag "saturated", cambia "5+1" a "4+2", etc.), el cambio se propaga a este PDF en la próxima generación. Este PDF NO es source-of-truth — es destilado del agent.md + factory_metadata.json + master.