# Post LinkedIn: Cómo construí DataPM (desde fuera del software tradicional)

Este canvas contiene: (A) propuestas de ángulos para el post, (B) estructura base recomendada y micro‑guía de estilo, (C) prompt ideal para pedirle a Cursor/GPT la descripción técnica completa del flujo de DataPM, y (D) tres borradores listos para LinkedIn (corto, medio y largo).

# A. Ángulos propuestos (elige 1 o combina)

1. **La mentalidad “no-developer” que entrega producto** — foco en cómo decisiones de producto, herramientas de IA y disciplina de ingeniería ligera permitieron construir un sistema sólido sin ser ingeniero de carrera.
2. **Herramientas y arquitectura práctica** — detalle técnico: Copilot + Cursor, Gemini/Ollama, CSV local, normalización y pruebas. Ideal para audiencia técnica-práctica.
3. **De Make.com a código reproducible** — narrar la transición: automatización low-code → programa Python independiente (portabilidad, reproducibilidad, control de costes).
4. **Calidad de datos y normalización en el centro** — foco en por qué un buen “banco” de normalización reduce unknowns y cómo lo expandiste.
5. **Lecciones y anti‑patrones** — errores reales (rate limiting, unknowns, mappings insuficientes) y cómo los resolviste; útil para product people y managers.
6. **Impacto y métricas** — resultados cuantificables (nº registros procesados, % unknown inicial vs tras mejora, mejoras en pipeline) para convencer a stakeholders.
7. **Llamado a colaboración / open-source** — invitar a desarrolladores, data scientists o reclutadores a probar, aportar o contratarte.

# B. Estructura base recomendada (órdenes, longitud y propósito)

* **Hook / primer renglón (1 frase, 20–30 palabras):** idea fuerte y contraria (p. ej. “No soy ingeniero, y aún así lancé un pipeline de datos que reemplaza Make.com”).
* **Contexto + problema (1–2 párrafos cortos):** por qué el problema importa (descripciones de trabajo sucias, escalabilidad, dependencia de plataformas low-code).
* **Qué construí (1–2 párrafos):** solución en alto nivel — DataPM: qué hace y qué partes lo componen.
* **Cómo lo hice (bullets técnicos, 4–7 líneas):** herramientas (Copilot, Cursor), LLMs (Gemini/Ollama), arquitectura (CSV local → LLM → normalización → CSV limpio), pruebas y manejo de rate limiting.
* **Resultados (1 párrafo con números concretos):** registros procesados (50), problemas iniciales (alta tasa de Unknowns), acciones de mejora (banco de normalización expandido), estado actual.
* **Lecciones clave (3 bullets):** prácticas replicables — prompts estrictos, human‑in‑loop, tests, despliegue reproducible.
* **CTA (1 línea):** pedir conexión, feedback, contribuciones o invitación a demo.
* **Hashtags sugeridos:** #DataEngineering #Product #AI #PromptEngineering #NoCodeToCode #DataPM

**Tono:** formal, directo, escéptico y práctico. Evita grandilocuencia. Prioriza evidencia y llamadas a la acción concretas.

# C. Prompt ideal (single-shot) para Cursor / LLM — “Describir flujo completo de DataPM”

**Objetivo:** Obtener una descripción completa, reproducible y técnica del flujo de operaciones de DataPM para usarla en el post y en la documentación técnica.

Actúa como un ingeniero de software senior y redactor técnico. Tienes acceso al proyecto DataPM (pipeline que convierte descripciones de trabajo en CSV normalizado). Entrega 3 salidas separadas: (A) "Resumen ejecutivo" en 3–5 frases; (B) "Diagrama de flujo operativo" en pasos enumerados con detalles técnicos suficientes para que un desarrollador reproduzca el pipeline; (C) "Sección técnica" con: dependencias exactas (paquetes y versiones sugeridas), comandos para ejecutar, estructura de carpetas, muestras de prompts (system + user) usados, manejo de errores y rate limiting, tests y cómo replicarlos, ejemplos de entrada (1 registro) y salida (JSON/CSV final), métricas observadas (nº registros procesados, % Unknowns antes/después si está disponible), y recomendaciones de mejoras.  
  
Requisitos estrictos:  
- Formatea (A) como texto plano.  
- Formatea (B) y (C) en bloques de código Markdown y listas numeradas.  
- Incluye ejemplos concretos de prompts (system + user) que se usaron en producción para la normalización.  
- No inventes métricas si no las tienes; si faltan, indica claramente "Métrica no disponible".  
- Al final, genera un "LinkedIn Post Draft" en español (versión de 150–220 palabras) usando el resumen y las métricas, con tono formal y CTA.  
  
Fin del prompt.

# D. Borradores para LinkedIn (3 variantes)

## 1) Borrador corto (ideal para copy rápido — ~120–160 palabras)

No soy ingeniero de software de carrera. Y aun así, en los últimos meses diseñé y desplegué **DataPM**: un pipeline que transforma descripciones de trabajo desordenadas en CSV limpio y listo para análisis.

Lo hice combinando herramientas prácticas (GitHub Copilot, Cursor) y LLMs (Gemini/Ollama) para extraer, normalizar y validar campos críticos. Empecé desde una automatización en Make.com y la transformé en un programa Python reproducible que lee CSVs locales, llama a LLMs para extracción y aplica un banco de normalización extensible.

Resultado real: proceso un set inicial de **~50 registros**; detecté una tasa alta de Unknown que reduje ampliando el catálogo de normalización y mejorando prompts y tests. Lecciones concretas: prompts estrictos, human‑in‑loop y pruebas unitarias valen más que la teoría. Si te interesa ver el código, un demo o colaborar, escríbeme.

## 2) Borrador medio (más técnico, ~250–350 palabras)

Empecé con una pregunta práctica: ¿puedo construir un pipeline de datos robusto sin ser un “desarrollador” formal? La respuesta fue sí, pero no sin disciplina.

**Qué resolví.** DataPM nace para transformar descripciones de trabajo (CSV) en un CSV normalizado listo para Power BI y análisis. Migré la lógica de Make.com a un programa Python que integra LLMs (Gemini/Ollama) y un banco de normalización para títulos, skills y software.

**Cómo lo hice (resumen técnico).** Uso GitHub Copilot y Cursor para acelerar la implementación; el flujo es: CSV local → prompt system/user → LLM → JSON estructurado → post‑processing de normalización → CSV final. Implementé tests unitarios, manejo de rate limiting y un modo offline con Ollama.

**Resultados y problemas reales.** Procesé **~50 registros**; inicialmente hubo muchos Unknown porque el vocabulario normalizado era limitado. Expandí el banco de normalización y ajusté prompts; la tasa de Unknowns disminuyó notablemente (métricas precisas en la doc). El programa ahora es reproducible y permite experimentar con distintos LLMs.

**Lo que recomiendo a otros product people:** prioriza prompts estrictos, mantenimiento del catálogo de normalización, y una capa humana de verificación. Si quieres revisar el repo o coordinar un demo, contáctame.

## 3) Borrador largo (post técnico‑narrativo, ~450–650 palabras)

**No hacía falta un título formal para construir algo útil.** Empecé DataPM sin el grado típico de ingeniería, con una pregunta: ¿qué pasa si aplico disciplina de producto + herramientas de IA para resolver un problema repetible en RRHH/reclutamiento?

El problema era sencillo en la descripción y complejo en la práctica: las descripciones de trabajo están llenas de ruido (títulos variados, formato libre, ubicaciones inconsistentes y jergas de software). Mi objetivo: convertir esas descripciones en un CSV uniforme y listo para análisis.

Construí DataPM como un conjunto de programas Python que replican y mejoran una automatización previa en Make.com. La arquitectura es intencionalmente simple y reproducible:

* Entrada: CSV local con columnas title, company, location, description.
* Procesamiento: prompts system+user (estrictos) enviados a LLMs (Gemini por defecto, Ollama como fallback local).
* Normalización: banco ampliable de títulos, skills y software.
* Salida: CSV limpio para Power BI / análisis.

Herramientas clave: GitHub Copilot (aceleró implementación), Cursor (historial y reproducibilidad), Google Generative AI (Gemini) y Ollama para entornos offline.

Resultado real y lecciones: procesé **~50 registros** durante una fase de prueba. Al principio la tasa de Unknown fue significativa — un indicador de que el banco de normalización no cubría la diversidad real del mercado laboral. La respuesta fue pragmática: expandir el catálogo, endurecer los prompts y añadir pruebas automatizadas y revisión humana puntual. El resultado: un pipeline reproducible y mucho más fiable.

Si eres product manager, analista o desarrollador: no confundas la falta de un título con la falta de criterio técnico. Lo que marca la diferencia son los principios: reproducibilidad, pruebas, y métricas claras. Si quieres ver el código, la guía de despliegue o coordinar un demo, contáctame.

# Notas finales

* En el canvas ya incluí las muestras de posts y el prompt para Cursor.
* Si quieres que deje uno de los borradores pulido y listo para publicar (optimizado para SEO/engagement o versionado EN/ES), lo dejo directamente en este canvas.

*Fin del documento.*