Aquí tienes el **Diagrama de Flujo Técnico** detallado que ilustra la arquitectura del "Nexo Sinérgico".

Este flujo muestra cómo transformamos los datos crudos de la **Imagen 5 (Electricidad de Proceso)** en un activo editorial inteligente.

Fase 1: El Disparador y la Carga de Datos (The Payload)

Cuando el usuario hace clic en "Analizar Huella Energética" o "Generar Reporte", el frontend (React/Vue) captura el estado actual del componente y empaqueta un objeto JSON.

Entrada: Objeto de Datos Crudos (`raw_telemetry.json`) Este es el objeto exacto extraído de los campos de la interfaz en la Imagen 5.

JSON

```
{
  "timestamp": "2023-10-27T14:30:00Z",
  "source_module": "PROCESS_ELECTRICITY_V1",
  "project_id": "PYRO-X-99",
  "data_context": {
    "inputs": {
      "demand_power_kw": 385,            // De "Demanda de Potencia"
      "electricity_rate_eur_kwh": 0.06,  // De "Costo Electricidad"
      "annual_hours": 8000               // De "Horas Anuales"
    },
    "calculated_results": {
      "hourly_cost_eur": 23.10,          // De "Costo por Hora"
      "annual_cost_eur": 184800.00       // De "Costo Anual"
    }
  },
  "user_intent": {
    "action": "GENERATE_EXECUTIVE_REPORT",
    "target_audience": "CFO_INVESTORS",  // Define el tono (Financiero/Estratégico)
    "tone": "ANALYTICAL_OPTIMISTIC"
  }
}
```

Fase 2: Lógica del Nexo (Pseudocódigo)

Este es el "cerebro" (Backend/Python) que recibe el JSON anterior. No se limita a pasar datos a la IA; primero los **enriquece** semánticamente para asegurar que el reporte tenga valor de negocio.

Python

```
class NexoSinergicoEngine:

    def process_request(self, payload):
        # 1. Extracción de Variables Clave
        costs = payload['data_context']['calculated_results']
        inputs = payload['data_context']['inputs']

        # 2. Enriquecimiento Semántico (Insights pre-IA)
        # El sistema calcula métricas derivadas antes de pedirle el texto a la IA
        insights = self.calculate_benchmarks(costs, inputs)

        # 3. Selección de Plantilla de Prompt Dinámica
        # Selecciona la estructura del prompt basada en la audiencia (CFO)
        prompt_template = self.get_template(payload['user_intent']['target_audience'])

        # 4. Construcción del Prompt Final
        final_prompt = prompt_template.format(
            kpi_annual=costs['annual_cost_eur'],
            kpi_hourly=costs['hourly_cost_eur'],
            efficiency_rating=insights['rating'], # Ej: "Alta Eficiencia"
            context_data=inputs
        )

        # 5. Llamada a la API Generativa (OpenAI/Anthropic/Local)
        ai_response = self.call_llm_api(final_prompt)

        # 6. Post-Procesamiento (Formato JSON para el Frontend)
        return self.format_output(ai_response)

    def calculate_benchmarks(self, costs, inputs):
        # Lógica interna para determinar si el costo es bueno o malo
        # Supongamos que el benchmark de la industria es 0.10 €/kWh
        market_rate = 0.10
        saving_pct = ((market_rate - inputs['electricity_rate_eur_kwh']) / market_rate) * 100

        return {
            "market_comparison": "BELOW_MARKET",
            "savings_percentage": f"{saving_pct}%", // "40% de ahorro vs mercado"
            "rating": "EXCELLENT"
        }
```

Fase 3: La Ingeniería del Prompt (El "Puente")

Aquí es donde ocurre la magia. El sistema construye un prompt complejo invisible para el usuario.

Prompt Generado Internamente (String):

> "Actúa como un Consultor Senior de Eficiencia Energética Industrial.
>
> **CONTEXTO:** Estamos analizando el consumo eléctrico de un proceso de pirólisis.
> **DATOS PUROS:**
>
> - Demanda: 385 kW
> - Tarifa lograda: 0.06 €/kWh (Excelente, 40% por debajo del mercado estándar).
> - Costo Operativo Anual: 184,800.00 €.
> - Operación: Continua (8000 horas/año).
>
> **OBJETIVO:** Escribe un 'Resumen Ejecutivo' para el Director Financiero (CFO).
>
> **RESTRICCIONES:**
>
> 1. No uses jerga técnica innecesaria.
> 2. Enfatiza el ahorro logrado gracias a la tarifa de 0.06.
> 3. Utiliza una metáfora visual para explicar la escala del consumo.
> 4. Formato de salida: Markdown estructurado con títulos."

Fase 4: La Respuesta Final al Frontend

El frontend recibe esto y lo renderiza en una tarjeta elegante en el dashboard, o genera el PDF.

JSON


```
{
  "status": "success",
  "generated_content": {
    "title": "Informe de Viabilidad Financiera: Operaciones Eléctricas 2024",
    "summary_text": "La unidad de proceso actual demuestra una eficiencia de costes excepcional. Con una tarifa negociada de 0,06 €/kWh, estamos operando un 40% por debajo de los estándares del mercado energético industrial.",
    "key_highlight": "El costo anual proyectado de 184.800€ es altamente competitivo para un régimen de 8000 horas.",
    "visual_metaphor_prompt": "Generar imagen de una turbina dorada pequeña sosteniendo una estructura industrial grande, simbolizando alta eficiencia a bajo costo.",
    "suggested_actions": [
```

```
    "Congelar tarifa actual por 24 meses.",
    "Evaluar ampliación de carga bajo el mismo esquema de costes."
   ]
  }
}
```

¿Por qué este flujo es mejor?

1. **Enriquecimiento:** No solo pasa los números (184k), pasa el *significado* (40% de ahorro) gracias a la función `calculate_benchmarks`.
2. **Modularidad:** Puedes cambiar la audiencia en el JSON de entrada (`target_audience`: "ENGINEER") y el sistema generará un reporte técnico totalmente diferente sin cambiar el código central.
3. **Seguridad:** Los datos sensibles se procesan en el backend antes de construir el prompt final.