

Este es el "Plano Maestro". Este archivo **JSON de Configuración Global** (`nexo_config.json`) actúa como el sistema nervioso central. Define las reglas del juego para que tus scripts de Python (Auditoría, Academia, Gestor de Tareas) sepan cómo comportarse, qué umbrales activar y cómo llenar el COPRESET.

Archivo: `nexo_config.json`

Este archivo unifica los parámetros del PDF "Diagnóstico General" (para reglas técnicas) y los datos de tus capturas de pantalla (para reglas económicas).

JSON

```
{
  "system_info": {
    "app_name": "Nexo Sinérgico Core",
    "version": "2.1.0",
    "environment": "production_local",
    "update_cycle": "realtime"
  },

  "modules": {
    "economic_auditor": {
      "enabled": true,
      "parameters": {
        "electricity_rate_kwh": 0.12,
        "currency_symbol": "USD",
        "industrial_hours_per_year": 7920,
        "maintenance_downtime_percent": 0.05
      },
      "thresholds": {
        "min_savings_for_alert": 1000.00,
        "high_impact_savings": 10000.00
      }
    },

    "academic_researcher": {
      "enabled": true,
      "parameters": {
        "confidence_level": 0.95,
        "alpha_significance": 0.05,
        "test_type": "t_test_ind"
      },
      "thresholds": {
        "min_sample_size": 3,
```

```

    "variance_warning_limit": 15.0
  }
},

"material_diagnostics": {
  "enabled": true,
  "description": "Reglas derivadas del PDF Z-2",
  "critical_limits": {
    "mesopore_volume_min": 0.02,
    "coke_resistance_min": 60.0,
    "stability_thermal_min": 90.0
  }
},

"titan_debate": {
  "enabled": true,
  "agents": [
    { "id": "titan_engineer", "focus": "viabilidad_operativa" },
    { "id": "titan_scientist", "focus": "rigor_teorico" },
    { "id": "titan_artist", "focus": "impacto_visual" }
  ]
},

"orchestration_rules": [
  {
    "id": "rule_economic_boom",
    "trigger_condition": "annual_savings >
modules.economic_auditor.thresholds.high_impact_savings",
    "action": {
      "target": "Prompt_Creator",
      "copreset_template": "investor_pitch",
      "priority": "HIGH",
      "tone_override": "Persuasivo y Financiero"
    }
  },
  {
    "id": "rule_scientific_breakthrough",
    "trigger_condition": "p_value <
modules.academic_researcher.parameters.alpha_significance",
    "action": {
      "target": "Prompt_Creator",
      "copreset_template": "academic_paper_abstract",
      "priority": "MEDIUM",

```

```

        "tone_override": "Formal y Objetivo"
    }
},
{
    "id": "rule_material_failure",
    "trigger_condition": "mesopore_volume <
modules.material_diagnostics.critical_limits.mesopore_volume_min",
    "action": {
        "target": "Prompt_Creator",
        "copreset_template": "failure_analysis_visualization",
        "priority": "CRITICAL",
        "extra_context": "Advertencia: Bloqueo de poros inminente."
    }
}
],

"copreset_templates": {
    "investor_pitch": {
        "role": "Consultor Financiero Industrial",
        "objective": "Visualizar el retorno de inversión masivo.",
        "presentation": "Gráfico de crecimiento exponencial, estilo Fintech."
    },
    "failure_analysis_visualization": {
        "role": "Experto en Nanotecnología",
        "objective": "Mostrar la obstrucción física a nivel molecular.",
        "presentation": "Render 3D sección transversal, contraste alto."
    }
}
}
}

```

Integración en Python: El "Cargador Maestro"

Ahora, necesitamos un script pequeño que lea este JSON e inicialice tus clases (`EconomicAuditor`, `AcademicResearcher`) con estos valores exactos. Esto garantiza que si sube la luz (costo kWh), solo cambias el JSON, no el código.

```

Python
import json
from economic_auditor import EconomicAuditor
from academic_researcher import AcademicResearcher

class NexoCoreLoader:

```

```

def __init__(self, config_path='nexo_config.json'):
    self.config = self._load_config(config_path)
    self.modules = {}

def _load_config(self, path):
    with open(path, 'r') as f:
        return json.load(f)

def initialize_system(self):
    print(f"Iniciando {self.config['system_info']['app_name']} v{self.config['system_info']['version']}...")

    # 1. Inicializar Módulo Económico
    if self.config['modules']['economic_auditor']['enabled']:
        econ_params = self.config['modules']['economic_auditor']['parameters']
        self.modules['auditor'] = EconomicAuditor(
            electricity_rate=econ_params['electricity_rate_kwh'],
            currency=econ_params['currency_symbol']
        )
        print(" [OK] Auditor Económico en línea.")

    # 2. Inicializar Módulo Académico
    if self.config['modules']['academic_researcher']['enabled']:
        acad_params = self.config['modules']['academic_researcher']['parameters']
        self.modules['researcher'] = AcademicResearcher(
            confidence_level=acad_params['confidence_level']
        )
        print(" [OK] Investigador Académico en línea.")

def check_rules(self, data_context):
    """
    El Cerebro: Compara resultados entrantes contra las reglas del JSON.
    """
    rules = self.config['orchestration_rules']
    triggered_actions = []

    # Ejemplo: Verificando Regla de Materiales (Caso Z-2)
    # Mesoporo < 0.02 es crítico
    limit_meso =
self.config['modules']['material_diagnostics']['critical_limits']['mesopore_volume_min']

    if 'mesopore_volume' in data_context:
        if data_context['mesopore_volume'] < limit_meso:
            # Buscar la regla correspondiente en el JSON

```

```

        rule = next(r for r in rules if r['id'] == 'rule_material_failure')
        triggered_actions.append(rule['action'])
        print(f" [ALERTA] Regla Activada: {rule['id']}")

    return triggered_actions

# --- PRUEBA DE CONCEPTO ---

# Simulamos que llega el dato del PDF Z-2
dato_entrada_pdf = {
    "mesopore_volume": 0.012 # Este valor es menor que el límite 0.02 definido en JSON
}

# Ejecución
core = NexoCoreLoader()
core.initialize_system()
acciones = core.check_rules(dato_entrada_pdf)

print("\n--- ACCIONES ENVIADAS AL GESTOR DE TAREAS ---")
print(json.dumps(acciones, indent=2))

```

¿Por qué este JSON es potente?

1. **Escalabilidad Científica:** Si mañana el Dr. Pirolis decide que la resistencia al coque crítica es 70.0 en lugar de 60.0, solo editas `material_diagnostics` en el JSON. No tocas el código.
2. **Ajuste de Mercado:** Si el precio de la energía cambia, ajustas `electricity_rate_kwh` y todos tus reportes de ROI se actualizan automáticamente.
3. **Lógica Desacoplada:** Las reglas de negocio (`orchestration_rules`) están separadas del código. Esto permite crear nuevas "recetas" de automatización (ej. "Si p-value < 0.01 Y Ahorro > \$5000, crear video promocional") simplemente editando el texto del archivo de configuración.