Aquí tienes el script en **Python** diseñado para integrarse en tu backend (probablemente Django o Flask, dado que usas `localhost:3000`).

Este módulo, al que llamaremos `EconomicAuditor` (El Auditor Económico), no hace solo una multiplicación simple. Realiza un **Análisis de Opex (Operational Expenditure)** comparativo, calculando cuánto dinero estás perdiendo con el reactor ineficiente (R-004) frente al eficiente (R-003) y proyectándolo anualmente.

El Script: `economic_auditor.py`

Python

```
class EconomicAuditor:
    def __init__(self, electricity_rate=0.15, currency="USD"):
        """
        Configuración inicial con tarifas industriales promedio.
        :param electricity_rate: Costo por kWh (o unidad de energía del sistema).
        :param currency: Moneda para el reporte.
        """
        self.rate = electricity_rate
        self.currency = currency
        # Horas operativas anuales estándar en industria (aprox 330 días)
        self.annual_operating_hours = 7920

    def calculate_run_cost(self, energy_units):
        """Calcula el costo directo de una sola ejecución experimental."""
        return round(energy_units * self.rate, 2)

    def analyze_efficiency_gap(self, winner_data, loser_data):
        """
        Compara el mejor reactor (R-003) contra el peor (R-004) para hallar el 'Costo de Ineficiencia'.
        """

        # 1. Costos Individuales
        cost_winner = self.calculate_run_cost(winner_data['energy'])
        cost_loser = self.calculate_run_cost(loser_data['energy'])

        # 2. Delta (Diferencia)
        energy_saved = loser_data['energy'] - winner_data['energy']
        money_saved_per_run = cost_loser - cost_winner
        efficiency_gain_pct = (energy_saved / loser_data['energy']) * 100

        # 3. Proyección Anual (Escalabilidad Industrial)
        # Asumiendo que cada "run" dura 1 hora para efectos de estandarización
```

```
        projected_annual_savings = money_saved_per_run * self.annual_operating_hours

        return {
            "comparison_id": f"{winner_data['id']} vs {loser_data['id']}",
            "winner_cost": cost_winner,
            "loser_cost": cost_loser,
            "savings_per_run": round(money_saved_per_run, 2),
            "efficiency_gain_percent": round(efficiency_gain_pct, 2),
            "annual_projected_savings": round(projected_annual_savings, 2),
            "financial_verdict": self._generate_verdict(projected_annual_savings)
        }

    def _generate_verdict(self, savings):
        """Genera una frase de impacto para el reporte o el prompt."""
        if savings > 10000:
            return "ALTO IMPACTO: La optimización justifica inversión inmediata."
        elif savings > 1000:
            return "IMPACTO MEDIO: Mejora operativa significativa."
        else:
            return "BAJO IMPACTO: Optimización marginal."

# --- SIMULACIÓN DE DATOS (Basado en tus Capturas) ---

# Datos extraídos de tu "Forja de Innovación"
reactor_winner = {
    "id": "R-003 (Grupo A - Gaia)",
    "energy": 803.76  # El valor mínimo (MIN)
}

reactor_loser = {
    "id": "R-004 (Grupo C - Rápida)",
    "energy": 868.37  # El valor máximo (MAX)
}

# --- EJECUCIÓN DEL ANÁLISIS ---

# Instanciamos el auditor con un precio industrial de $0.12/kWh
auditor = EconomicAuditor(electricity_rate=0.12)

# Ejecutamos el análisis comparativo
financial_report = auditor.analyze_efficiency_gap(reactor_winner, reactor_loser)

# --- SALIDA PARA EL GESTOR DE TAREAS ---
```

```
print("--- REPORTE FINANCIERO AUTOMÁTICO ---")
print(f"Ganador: {reactor_winner['id']}")
print(f"Costo Operativo Ganador: ${financial_report['winner_cost']}")
print(f"Costo Operativo Perdedor: ${financial_report['loser_cost']}")
print(f"Ahorro por ciclo: ${financial_report['savings_per_run']}")
print(f"Mejora de Eficiencia: {financial_report['efficiency_gain_percent']}%")
print("-" * 30)
print(f"PROYECCIÓN ANUAL DE AHORRO:
${financial_report['annual_projected_savings']:,.2f}")
print(f"VERDICTO: {financial_report['financial_verdict']}")

# --- EXTENSIÓN: GENERACIÓN DE CONTEXTO PARA COPRESET ---
# Este texto se inyectaría automáticamente en el campo "Contexto" del Creador de Prompt

copreset_context_string = (
    f"Datos Financieros: Al cambiar del proceso {reactor_loser['id']} al {reactor_winner['id']}, "
    f"reducimos el costo operativo un {financial_report['efficiency_gain_percent']}%. "
    f"Esto proyecta un ahorro anual de ${financial_report['annual_projected_savings']:,.0f} USD, "
    f"validando económicamente el método Gaia."
)

print("\n--- STRING PARA INYECCIÓN EN COPRESET (CONTEXT) ---")
print(copreset_context_string)
```

Explicación de la Lógica Financiera

1. **Deltas en lugar de Absolutos:** El script no solo calcula cuánto cuesta operar, sino la *diferencia* entre hacerlo bien (Gaia) y hacerlo "estándar" (Rápida). Ese es el valor real para un inversor.
2. **Proyección Anual:** Un ahorro de $7.75 por ciclo parece poco, pero al multiplicarlo por 7,920 horas (un año industrial 24/7 menos mantenimiento), se convierte en una cifra macro ($61,000+). Esto cambia la narrativa de "un experimento interesante" a "una decisión de negocios".
3. **Inyección Contextual:** La última parte del script genera una cadena de texto (`copreset_context_string`) lista para ser consumida por el **Gestor de Tareas**.

¿Cómo se ve esto en tu flujo?

Cuando el experimento termina en la pantalla "Resultados Detallados":

1. El sistema detecta el **MIN (803.76)** y el **MAX (868.37)**.
2. Ejecuta este script en segundo plano.

3. Si el ahorro proyectado > $10,000 (configurable), se crea automáticamente una tarea urgente en el **Gestor de Tareas**: *"Crear presentación de Ahorro Energético para Gerencia"*.
4. Al abrir esa tarea en el **Creador de Prompt**, el campo contexto ya dice: *"Al cambiar del proceso R-004 al R-003, reducimos el costo un 7.44%..."*.