📐 Arquitectura del Proyecto

Principios fundamentales para el desarrollo del juego

🎯 Principios Fundamentales

1. Separación Simulación ↔ Visualización

Regla de Oro:

> *La lógica del juego NUNCA debe depender de nodos visuales*

```
❌ MAL: EnergyPulse (nodo 3D) → detecta colisión → Compressor actúa
✅ BIEN: EnergyManager calcula → Compressor recibe dato → Visual se actualiza
```

Consecuencia:

- La simulación corre con números puros
- Los visuales solo representan el estado
- Si un visual desaparece antes de tiempo → no afecta la lógica

Verificación (visuales opcionales):

- **Pulsos (bolas):** `EnergyManager.spawn_pulse_visual()` solo se llama si `MOSTRAR_VISUAL_PULSO` ; la entrega de energía es siempre `EnergyManager.register_flow()` → `EnergyFlow._entregar()` → `recibir_energia_numerica()` . Sin visual, la lógica sigue igual.
- **Haces (beams):** `beam_emitter.dibujar_haz()` solo dibuja cilindros; no entrega energía. La entrega es numérica (EnergyFlow). `recibir_luz_instantanea()` en prismas actualiza solo dirección/color del haz dibujado, no la lógica de flujo.
- **Dónde se spawnean visuales:** siphon_logic, god_siphon, prism_logic, compressor, merger (spawn_pulse_visual + dibujar_haz en cada uno). En todos, `register_flow` / flujo numérico es independiente del visual.

---

2. Managers Centrales

Arquitectura recomendada:

```
GridManager       → Qué hay en cada celda, validación de colocación
EnergyManager     → Flujos de energía (números, no nodos)
BuildingManager   → Registro de edificios activos
```

Comunicación:

```
Building → Manager (registrarse, reportar estado)
Manager → Building (callbacks, actualizar estado)

❌ Building ↔ Building (nunca directamente)
```

---

3. Sistema de Energía (Target)

Modelo antiguo (eliminado en ROADMAP 3.2):

```
# Antes: energy_pulse.tscn como nodo físico. Ya no existe; solo modelo numérico.
```

Modelo objetivo (NUMÉRICO):

```
# Clase pura de datos
class EnergyFlow:
    var from: Building
    var to: Building
    var amount: float = 10.0
    var tick_rate: float = 1.0
    var timer: float = 0.0

    func update(delta):
        timer += delta
        if timer >= tick_rate:
            to.receive_energy(amount)
            timer = 0.0
            # Opcional: trigger visual
```

🔄 Flujo de Trabajo

Al añadir un nuevo edificio:

1. Crear escena `.tscn` (visual)
2. Crear script de lógica (extiende `Building` base)
3. Registrar en `BuildingManager` en `_ready()`
4. Implementar `receive_energy()` / `produce_energy()`
5. NO comunicarse directamente con otros edificios

Al modificar mecánicas:

1. Cambiar SOLO la lógica (managers)
2. Verificar que funciona con `print()` / debugger
3. Actualizar visuales si es necesario
4. Nunca mezclar ambos pasos

📏 Reglas de Oro

1. **Nada existe si no está documentado**
2. **Los edificios NO se comunican directamente**
3. **La simulación manda, el render obedece**
4. **Nunca mezclar lógica con nodos visuales**
5. **Preferir sistemas aburridos pero claros**
6. **Si algo "ya se arreglará después" → parar**

🎯 Estado Actual vs Objetivo

Aspecto	Actual	Objetivo

Energía	Nodos físicos	Datos numéricos
Comunicación	Building ↔ Building	Building → Manager
Validación	En múltiples lugares	Centralizada en managers
Escalabilidad	Limitada (~50 edificios)	Ilimitada (1000+)

📚 Documentos Relacionados

- `docs/README.md` - Índice de documentación
- `PROJECT_STATE.md` - Estado general
- `REFACTORING_PLAN.md` - Plan de migración (completado)
- `ENERGY_SYSTEM.md` - Sistema energía detallado (implementado)