Esta es la "neurotransmisión" que le falta a tu sistema. Para conectar el **Módulo Técnico** (Pyrolysis Hub) con el **Módulo Financiero** (Risk Simulator), no usaremos una base de datos compleja todavía. Usaremos el **React Context API** para crear una memoria de sesión compartida.

Esto permitirá que cuando el usuario haga clic en el botón, los datos "vuelen" instantáneamente a la siguiente pantalla sin recargar la página.

Aquí tienes la implementación en 3 pasos:

Paso 1: Crear la "Memoria del Proyecto" (ProjectContext)

Este archivo actuará como el cerebro compartido. Guardará los resultados de la pirólisis para que cualquiera (Finanzas, Creatividad) pueda leerlos.

Crea `src/context/ProjectContext.js`:

JavaScript

```
import React, { createContext, useState, useContext } from 'react';

// Creamos el contexto
const ProjectContext = createContext();

export const ProjectProvider = ({ children }) => {
  // Aquí guardamos el estado "vivo" del proyecto actual
  const [simulationState, setSimulationState] = useState({
    technicalData: null, // Aquí caerán los datos del Hub de Pirólisis
    financialData: null,
    isSynced: false
  });

  // Función para que el Pyrolysis Hub guarde sus datos
  const syncTechnicalData = (data) => {
    setSimulationState((prev) => ({
      ...prev,
      technicalData: data,
      isSynced: true
    }));
  };

  return (
    <ProjectContext.Provider value={{ simulationState, syncTechnicalData }}>
      {children}
    </ProjectContext.Provider>
```

```
  );
};

// Hook personalizado para usarlo fácil en cualquier componente
export const useProject = () => useContext(ProjectContext);
```

(Nota: Asegúrate de envolver tu aplicación en `_app.js` o `layout.js` con `<ProjectProvider>...</ProjectProvider>`).

Paso 2: El "Botón de Puente" en el Pyrolysis Hub

Vamos a modificar tu `PyrolysisSimulator.jsx`. Reemplaza o añade el botón de acción final con esta lógica de navegación y guardado.

```
JavaScript
import { useRouter } from 'next/router';
import { useProject } from '../context/ProjectContext'; // Importamos el cerebro
import { TrendingUp, ArrowRightCircle } from 'lucide-react'; // Iconos

// ... dentro de tu componente PyrolysisSimulator ...

const PyrolysisSimulator = () => {
  const router = useRouter();
  const { syncTechnicalData } = useProject(); // Hook del contexto

  // ... tus estados de simulación (yields, feedstock, etc.) ...

  const handleBridgeToFinance = () => {
    // 1. Empaquetamos la "Verdad Técnica"
    const technicalPayload = {
      sourceId: "SIM-" + Math.floor(Math.random() * 10000),
      feedstockName: feedstock, // ej: "Spirulina"
      productionVolume: 5000, // Supongamos que tu simulador calcula esto (Ton/año)
      biocharYield: yields.char, // ej: 25 (%)
      bioOilYield: yields.oil,   // ej: 55 (%)
      efficiency: efficiency
    };

    // 2. Guardamos en la memoria global (Sincronización)
    syncTechnicalData(technicalPayload);
```

```
    // 3. Navegamos al Módulo Financiero
    router.push('/risk-simulator');
  };

  return (
    // ... en tu sección de botones ...
    <div className="flex gap-4 mt-6">

      {/* Botón existente para Creatividad */}
      <Button className="...">Crear Espacio Creativo</Button>

      {/* EL NUEVO BOTÓN PUENTE */}
      <Button
        onClick={handleBridgeToFinance}
        className="flex-1 py-4 bg-emerald-600 hover:bg-emerald-500 text-white font-bold
rounded-lg shadow-md flex items-center justify-center gap-2 border border-emerald-400"
      >
        <TrendingUp size={20} />
        <span>Analizar Viabilidad Financiera</span>
        <ArrowRightCircle size={18} className="opacity-70"/>
      </Button>

    </div>
  );
};
```

Paso 3: El Receptor en el Risk Simulator

Ahora, en tu `RiskSimulator.jsx`, hacemos que "escuche" si hay datos entrantes. Si los hay, bloquea los inputs de volumen y calcula automáticamente.

JavaScript

```
import { useProject } from '../context/ProjectContext';
import { Alert, Card, Slider, Statistic } from 'antd';

const RiskSimulator = () => {
  // Recuperamos los datos del contexto
  const { simulationState } = useProject();
  const { technicalData, isSynced } = simulationState;

  // Estado financiero local
  const [priceBiochar, setPriceBiochar] = useState(300); // €/ton
```

```
  // LÓGICA DE SINERGIA:
  // Si venimos del Hub, usamos sus datos. Si no, usamos default (1000 tons).
  const productionVolume = isSynced ? technicalData.productionVolume : 1000;
  const feedstockName = isSynced ? technicalData.feedstockName : "Genérico";

  // Cálculo automático: (Volumen * %Rendimiento * Precio)
  // Nota: biocharYield viene en % (ej 25), lo dividimos por 100
  const biocharRevenue = productionVolume * (isSynced ? technicalData.biocharYield / 100 :
0.3) * priceBiochar;

  return (
    <div className="p-6">
      <h1 className="text-2xl font-bold mb-4">Módulo 5: Simulador de Riesgos</h1>

      {/* COMPONENTE VISUAL DE "PUENTE ACTIVO" */}
      {isSynced && (
        <Alert
          message="Sinergia Activa: Datos Técnicos Vinculados"
          description={`Simulación basada en rendimiento real de ${feedstockName} (ID:
${technicalData.sourceId}). El volumen de producción está fijado por la física del reactor.`}
          type="success"
          showIcon
          className="mb-6 border-emerald-500 bg-emerald-50"
        />
      )}

      <div className="grid grid-cols-2 gap-6">
        {/* PANEL DE CONTROL */}
        <Card title="Variables de Mercado">

          {/* CAMPO DE SOLO LECTURA (Read-Only) - Sinergia */}
          <div className="mb-4 opacity-75">
            <label className="text-xs font-bold uppercase text-gray-500">Volumen de Producción
(Fijado por Hub)</label>
            <div className="text-xl font-mono font-bold text-slate-700">
              {productionVolume.toLocaleString()} Ton/año
            </div>
          </div>

          <div className="mb-4">
            <label className="text-xs font-bold uppercase text-gray-500">Precio Biochar
(€/ton)</label>
            <Slider
```

```
            min={100} max={1000}
            value={priceBiochar}
            onChange={setPriceBiochar}
          />
          <div className="text-right font-bold text-blue-600">{priceBiochar} €</div>
        </div>
      </Card>

      {/* PANEL DE RESULTADOS */}
      <Card className="bg-slate-900 text-white">
        <Statistic
          title={<span className="text-gray-400">Ingresos Proyectados (Biochar)</span>}
          value={biocharRevenue}
          precision={0}
          prefix="€"
          valueStyle={{ color: '#3f8600', fontWeight: 'bold' }}
        />
        <p className="text-xs text-gray-500 mt-2">
          *Cálculo dinámico basado en rendimiento técnico x condiciones de mercado.
        </p>
      </Card>
    </div>
  </div>
 );
};
```

¿Qué acabas de construir?

1. **Continuidad de Datos:** El usuario siente que está en una sola plataforma ("Nexo"), no en herramientas sueltas.
2. **Integridad de Ingeniería:** El financiero no puede "inventar" que produce 1 millón de toneladas si el reactor técnico dice que solo puede hacer 5,000. Esto fuerza el realismo.
3. **Feedback Visual:** La alerta verde "Sinergia Activa" refuerza la confianza en el sistema.