Esta es la "joya visual" del Módulo 5. Para lograr ese histograma profesional (la "Curva de Campana" o Distribución Normal) que mostraste en la imagen, usaremos **Recharts**.

Es la librería estándar en el ecosistema React por ser ligera, declarativa y muy fácil de animar.

Aquí tienes la implementación completa: desde el motor matemático (Simulación de Monte Carlo) hasta la visualización gráfica.

Paso 1: Instalación

Si no tienes Recharts instalado en tu frontend, ejecuta:

Bash
npm install recharts

Paso 2: El Motor Matemático (`simulationEngine.js`)

No podemos graficar sin datos. Necesitamos una función que "tire los dados" 5,000 veces rápidamente. Crea este archivo de utilidad en `src/utils/simulationEngine.js` (o dentro del componente si prefieres).

JavaScript
```
// Generador de Ruido Gaussiano (Campana de Gauss)
// Simula la variabilidad natural del mercado (no es plano, se agrupa al centro)
const getGaussianRandom = (mean, stdDev) => {
  let u = 0, v = 0;
  while(u === 0) u = Math.random();
  while(v === 0) v = Math.random();
  // Transformada de Box-Muller
  const num = Math.sqrt( -2.0 * Math.log( u ) ) * Math.cos( 2.0 * Math.PI * v );
  return num * stdDev + mean;
};

export const runMonteCarlo = (baseRevenue, opex, capex, uncertaintyPercent, runs = 2000) =>
{
  const results = [];

  // 1. Ejecutar N simulaciones
  for (let i = 0; i < runs; i++) {
    // Aplicamos incertidumbre al Ingreso y al OPEX
    // Si incertidumbre es 20%, el valor variará +/- 20% siguiendo una curva normal
    const volatility = uncertaintyPercent / 100;
```

```
    const simRevenue = getGaussianRandom(baseRevenue, baseRevenue * volatility);
    const simOpex = getGaussianRandom(opex, opex * (volatility * 0.5)); // OPEX suele variar
menos

    // Cálculo simplificado de Beneficio Neto (Net Profit)
    const netProfit = simRevenue - simOpex - (capex / 10); // Amortización simple a 10 años
    results.push(netProfit);
  }

  // 2. Crear los "Bins" (Agrupar resultados para el histograma)
  results.sort((a, b) => a - b);
  const min = results[0];
  const max = results[results.length - 1];
  const binCount = 20; // Número de barras en el gráfico
  const binSize = (max - min) / binCount;

  const histogramData = [];

  for (let i = 0; i < binCount; i++) {
    const binStart = min + (i * binSize);
    const binEnd = binStart + binSize;

    // Contar cuántas simulaciones cayeron en este rango
    const frequency = results.filter(r => r >= binStart && r < binEnd).length;

    histogramData.push({
      range: `${(binStart / 1000).toFixed(0)}k - ${(binEnd / 1000).toFixed(0)}k`, // Etiqueta eje X
      profit: (binStart + binEnd) / 2, // Valor medio numérico
      frequency: frequency, // Eje Y (Altura de la barra)
      fill: frequency > 0 ? "#2563eb" : "#94a3b8" // Azul Nexo
    });
  }

  // Calculamos riesgo: % de veces que el beneficio fue negativo
  const lossRuns = results.filter(r => r < 0).length;
  const riskPercent = ((lossRuns / runs) * 100).toFixed(1);

  return { histogramData, riskPercent };
};
```

Paso 3: El Componente del Gráfico (`MonteCarloChart.jsx`)

Este es el componente visual que usarás dentro de tu `RiskSimulator`.

```
JavaScript
import React from 'react';
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer,
ReferenceLine } from 'recharts';

const MonteCarloChart = ({ data }) => {
  if (!data || data.length === 0) return <div className="text-gray-400">Esperando
simulación...</div>;

  // Personalización del Tooltip (La cajita negra al pasar el mouse)
  const CustomTooltip = ({ active, payload, label }) => {
    if (active && payload && payload.length) {
      return (
        <div className="bg-slate-800 text-white p-3 rounded shadow-lg border
border-slate-700">
          <p className="font-bold mb-1">Rango: {label} €</p>
          <p className="text-emerald-400">Probabilidad: {payload[0].value} escenarios</p>
        </div>
      );
    }
    return null;
  };

  return (
    <div style={{ width: '100%', height: 300 }}>
      <ResponsiveContainer>
        <BarChart data={data} margin={{ top: 20, right: 30, left: 20, bottom: 5 }}>
          <CartesianGrid strokeDasharray="3 3" vertical={false} stroke="#e2e8f0" />

          <XAxis
            dataKey="range"
            tick={{ fontSize: 10, fill: '#64748b' }}
            interval={2} // Mostrar etiquetas salteadas para no saturar
          />
          <YAxis hide /> {/* Ocultamos eje Y para limpieza visual */}

          <Tooltip content={<CustomTooltip />} cursor={{ fill: 'transparent' }} />

          {/* La Barra Azul */}
          <Bar
            dataKey="frequency"
            fill="#3b82f6"
            radius={[4, 4, 0, 0]} // Bordes redondeados arriba
            animationDuration={1500}
```

```
        />

        {/* Línea de "Punto de Equilibrio" (Cero) */}
        <ReferenceLine x="0" stroke="red" strokeDasharray="3 3" label="Break Even" />
      </BarChart>
    </ResponsiveContainer>
  </div>
 );
};

export default MonteCarloChart;
```

Paso 4: Integración Final en `RiskSimulator.jsx`

Ahora conectamos todo. Usamos `useEffect` para recalcular el gráfico cada vez que el usuario mueve los sliders.

JavaScript

```
import React, { useState, useEffect } from 'react';
import { useProject } from '../context/ProjectContext';
import { runMonteCarlo } from '../utils/simulationEngine'; // Importamos el motor
import MonteCarloChart from './MonteCarloChart'; // Importamos el gráfico
import { Card, Slider, Statistic, Alert } from 'antd'; // O tu librería UI

const RiskSimulator = () => {
  const { simulationState } = useProject();

  // Inputs Financieros
  const [priceBiochar, setPriceBiochar] = useState(300);
  const [uncertainty, setUncertainty] = useState(15); // % de Incertidumbre (Slider)

  // Estado del Resultado
  const [chartData, setChartData] = useState([]);
  const [riskMetric, setRiskMetric] = useState(0);

  // Datos base (Sinergia o Default)
  const productionVol = simulationState.isSynced ?
simulationState.technicalData.productionVolume : 1000;
  const baseRevenue = productionVol * priceBiochar;
  const baseOpex = 150000; // Valor ejemplo
  const baseCapex = 1000000; // Valor ejemplo

  // --- EFECTO: RECALCULAR MONTE CARLO ---
```

```
  useEffect(() => {
    // Usamos un pequeño timeout para no colgar el navegador mientras mueves el slider
(debounce)
    const timer = setTimeout(() => {
      const { histogramData, riskPercent } = runMonteCarlo(baseRevenue, baseOpex,
baseCapex, uncertainty);
      setChartData(histogramData);
      setRiskMetric(riskPercent);
    }, 100);

    return () => clearTimeout(timer);
  }, [baseRevenue, baseOpex, baseCapex, uncertainty]);


  return (
    <div className="p-6 grid grid-cols-1 md:grid-cols-2 gap-6">

      {/* COLUMNA IZQUIERDA: CONTROLES */}
      <div className="space-y-6">
        <Card title="Parámetros de Incertidumbre">
          <div className="mb-6">
            <label>Precio Biochar (€/ton)</label>
            <Slider min={100} max={800} value={priceBiochar} onChange={setPriceBiochar} />
          </div>
          <div className="mb-6">
            <label>Volatilidad de Mercado (+/- %)</label>
            <Slider min={5} max={50} value={uncertainty} onChange={setUncertainty} />
            <p className="text-xs text-gray-400">Cuanto mayor es la volatilidad, más ancha será
la campana.</p>
          </div>
        </Card>

        {/* METRICAS DE RESULTADO */}
        <div className="grid grid-cols-2 gap-4">
          <Card>
            <Statistic
              title="Riesgo de Pérdida"
              value={riskMetric}
              suffix="%"
              valueStyle={{ color: riskMetric > 20 ? '#cf1322' : '#3f8600' }}
            />
          </Card>
          {/* ... otras métricas ... */}
        </div>
```

```
      </div>

      {/* COLUMNA DERECHA: EL GRÁFICO VISUAL */}
      <Card title="Distribución de Resultados (Monte Carlo)" className="h-full">
        <div className="mb-4">
          {/* ALERTA INTELIGENTE */}
          {riskMetric < 5 ? (
            <Alert type="success" message="Inversión Segura: Probabilidad de éxito muy alta."
showIcon />
          ) : riskMetric > 30 ? (
            <Alert type="error" message="Alto Riesgo: Se recomienda asegurar contratos de
venta." showIcon />
          ) : (
            <Alert type="warning" message="Riesgo Moderado: Dependiente de variables
externas." showIcon />
          )}
        </div>

        {/* AQUI ESTÁ EL GRÁFICO AZUL */}
        <MonteCarloChart data={chartData} />

        <p className="text-center text-xs text-gray-400 mt-2">
          Eje X: Beneficio Neto Estimado (€) | Altura: Probabilidad de ocurrencia
        </p>
      </Card>
    </div>
  );
};

export default RiskSimulator;
```

El Resultado Visual

1. **Interactividad:** Al mover el slider de `Incertidumbre`, verás cómo la "montaña azul" se aplana y se ensancha (mayor riesgo, cola más larga). Si bajas la incertidumbre, la montaña se vuelve un pico alto y estrecho (seguridad).
2. **Color:** Las barras son azules (`#3b82f6`), idénticas a tu referencia visual.
3. **Animación:** Recharts anima las barras subiendo y bajando suavemente.

¡Ahora tienes un Simulador Financiero de grado profesional integrado en tu Nexo!