📊 GRÁFICOS Y REPORTES

Sistema de Inventario - Visualización de Datos

📈 LIBRERÍA: Recharts

Propósito: Crear gráficos interactivos en React

Instalación:

```
npm install recharts
```

¿Por qué Recharts?

- ☑ Basada en componentes React
- ☑ Gráficos responsivos automáticamente
- ☑ Animaciones suaves incluidas
- ☑ Tooltips interactivos
- ☑ Personalización con props
- ☑ No requiere canvas, usa SVG

📊 GRÁFICO DE LÍNEAS - Ventas del Mes

Propósito: Mostrar tendencia de ventas diarias

Código en Dashboard.jsx:

```
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer } from 'recharts'

const renderLineChart = () => {
  // Datos del endpoint /api/dashboard/grafico-ventas-mes
  const ventasMes = [
    { fecha: '01/01', total: 850, cantidad: 5 },
    { fecha: '02/01', total: 1250, cantidad: 8 },
    { fecha: '03/01', total: 950, cantidad: 6 },
    { fecha: '04/01', total: 1450, cantidad: 10 },
    { fecha: '05/01', total: 780, cantidad: 4 },
    // ... más datos
  ]

  return (
    <div className="chart-card">
      <h3>Ventas del Mes</h3>
      <ResponsiveContainer width="100%" height={300}>
```

```
          <LineChart
            data={ventasMes}
            margin={{ top: 5, right: 30, left: 20, bottom: 5 }}
          >
            {/* Rejilla de fondo */}
            <CartesianGrid strokeDasharray="3 3" stroke="#333" />

            {/* Eje X (fechas) */}
            <XAxis
              dataKey="fecha"
              stroke="#9ca3af"
              style={{ fontSize: '12px' }}
            />

            {/* Eje Y (totales) */}
            <YAxis
              stroke="#9ca3af"
              style={{ fontSize: '12px' }}
              tickFormatter={(value) => `S/ ${value}`}
            />

            {/* Tooltip al hacer hover */}
            <Tooltip
              contentStyle={{
                backgroundColor: '#1f2937',
                border: 'none',
                borderRadius: '8px',
                color: '#fff'
              }}
              formatter={(value, name) => {
                if (name === 'total') return [`S/ ${value.toFixed(2)}`, 'Total']
                if (name === 'cantidad') return [value, 'Ventas']
              }}
            />

            {/* Leyenda */}
            <Legend />

            {/* Línea de Total (naranja) */}
            <Line
              type="monotone"
              dataKey="total"
              stroke="#f97316"
              strokeWidth={3}
              dot={{ fill: '#f97316', r: 5 }}
              activeDot={{ r: 8 }}
              name="Total (S/)"
            />

            {/* Línea de Cantidad (azul) */}
            <Line
              type="monotone"
              dataKey="cantidad"
              stroke="#3b82f6"
```

```
              strokeWidth={2}
              dot={{ fill: '#3b82f6', r: 4 }}
              name="Cantidad de Ventas"
            />
          </LineChart>
        </ResponsiveContainer>
      </div>
    )
  }
```

Características:

- 📊 Dos líneas: Total en soles y Cantidad de ventas
- 🎨 Colores: Naranja (#f97316) y Azul (#3b82f6)
- 🖱️ Hover: Muestra valores exactos
- 📱 Responsivo: Se adapta al tamaño del contenedor
- ✨ Animación: Líneas se dibujan al cargar

📊 GRÁFICO DE BARRAS - Productos Más Vendidos

Propósito: Ranking de productos top 5

Código:

```
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Cell } from 'recharts'

const renderBarChart = () => {
  const topProductos = [
    { nombre: 'Filtro de Aceite', cantidad: 145, ingresos: 3625 },
    { nombre: 'Pistón 125cc', cantidad: 98, ingresos: 8379 },
    { nombre: 'Cadena 420', cantidad: 87, ingresos: 6090 },
    { nombre: 'Bujía NGK', cantidad: 76, ingresos: 912 },
    { nombre: 'Pastillas de Freno', cantidad: 65, ingresos: 3250 }
  ]

  // Colores degradados para cada barra
  const COLORS = ['#f97316', '#fb923c', '#fdba74', '#fed7aa', '#ffedd5']

  return (
    <div className="chart-card">
      <h3>Top 5 Productos Más Vendidos</h3>
      <ResponsiveContainer width="100%" height={350}>
        <BarChart
          data={topProductos}
          margin={{ top: 20, right: 30, left: 20, bottom: 80 }}
        >
          <CartesianGrid strokeDasharray="3 3" stroke="#333" />

          <XAxis
```

```
              dataKey="nombre"
              stroke="#9ca3af"
              angle={-45}
              textAnchor="end"
              height={100}
              style={{ fontSize: '11px' }}
            />

            <YAxis
              stroke="#9ca3af"
              label={{ value: 'Cantidad Vendida', angle: -90, position: 'insideLeft'
}}
            />

            <Tooltip
              contentStyle={{
                backgroundColor: '#1f2937',
                border: 'none',
                borderRadius: '8px'
              }}
              formatter={(value, name) => {
                if (name === 'cantidad') return [value, 'Unidades']
                if (name === 'ingresos') return [`S/ ${value}`, 'Ingresos']
              }}
            />

            <Legend />

            {/* Barras de Cantidad con colores diferentes */}
            <Bar
              dataKey="cantidad"
              name="Cantidad Vendida"
              radius={[8, 8, 0, 0]}
            >
              {topProductos.map((entry, index) => (
                <Cell key={`cell-${index}`} fill={COLORS[index]} />
              ))}
            </Bar>
          </BarChart>
        </ResponsiveContainer>
      </div>
    )
}
```

Características:

- 📊 Barras con colores degradados (naranja oscuro → naranja claro)
- 🔄 Nombres rotados 45° para legibilidad
- 📏 Esquinas redondeadas en barras
- 🎯 Tooltip muestra cantidad e ingresos
- 🧮 Altura fija para mantener proporción

GRÁFICO DE PASTEL - Ventas por Categoría

Propósito: Distribución de ventas por tipo de producto

Código:

```
import { PieChart, Pie, Cell, Tooltip, Legend, ResponsiveContainer } from
'recharts'

const renderPieChart = () => {
  const ventasPorCategoria = [
    { categoria: 'Repuestos Motor', valor: 45680, porcentaje: 35 },
    { categoria: 'Sistema Eléctrico', valor: 28950, porcentaje: 22 },
    { categoria: 'Frenos', valor: 21400, porcentaje: 16 },
    { categoria: 'Transmisión', valor: 19850, porcentaje: 15 },
    { categoria: 'Accesorios', valor: 15680, porcentaje: 12 }
  ]

  // Paleta de colores
  const COLORS = ['#f97316', '#fb923c', '#fdba74', '#3b82f6', '#60a5fa']

  // Etiquetas personalizadas
  const renderLabel = ({ cx, cy, midAngle, innerRadius, outerRadius, porcentaje })
=> {
    const RADIAN = Math.PI / 180
    const radius = innerRadius + (outerRadius - innerRadius) * 0.5
    const x = cx + radius * Math.cos(-midAngle * RADIAN)
    const y = cy + radius * Math.sin(-midAngle * RADIAN)

    return (
      <text
        x={x}
        y={y}
        fill="white"
        textAnchor={x > cx ? 'start' : 'end'}
        dominantBaseline="central"
        style={{ fontSize: '14px', fontWeight: 'bold' }}
      >
        {`${porcentaje}%`}
      </text>
    )
  }

  return (
    <div className="chart-card">
      <h3>Ventas por Categoría</h3>
      <ResponsiveContainer width="100%" height={400}>
        <PieChart>
          <Pie
            data={ventasPorCategoria}
            cx="50%"
            cy="50%"
            labelLine={false}
```

```
              label={renderLabel}
              outerRadius={120}
              fill="#8884d8"
              dataKey="valor"
            >
              {ventasPorCategoria.map((entry, index) => (
                <Cell
                  key={`cell-${index}`}
                  fill={COLORS[index % COLORS.length]}
                />
              ))}
            </Pie>

            <Tooltip
              formatter={(value) => `S/ ${value.toFixed(2)}`}
              contentStyle={{
                backgroundColor: '#1f2937',
                border: 'none',
                borderRadius: '8px'
              }}
            />

            <Legend
              verticalAlign="bottom"
              height={36}
              formatter={(value, entry) => `${entry.payload.categoria}
(${entry.payload.porcentaje}%)`}
            />
          </PieChart>
        </ResponsiveContainer>
      </div>
    )
}
```

Características:

- Gráfico circular con porcentajes
- Etiquetas dentro de cada sector
- 5 colores diferentes
- Hover muestra valor en soles
- Leyenda con categoría y porcentaje

EXPORTAR A PDF

Librería: jsPDF + jsPDF AutoTable

Código (export.js):

```
import { jsPDF } from 'jspdf'
import 'jspdf-autotable'
```

```
import { COMPANY_INFO, nowPE } from './company'

export async function exportToPDF({ title, columns, rows, filename = 'reporte' })
{
  const doc = new jsPDF({
    unit: 'pt',  // puntos
    format: 'a4'  // tamaño carta
  })

  const marginX = 40
  let cursorY = 40

  // ===== ENCABEZADO EMPRESA =====
  doc.setFont('helvetica', 'bold')
  doc.setFontSize(14)
  doc.text(COMPANY_INFO.name, marginX, cursorY)

  doc.setFont('helvetica', 'normal')
  doc.setFontSize(10)
  const { fecha, hora } = nowPE()
  const infoLines = [
    `RUC: ${COMPANY_INFO.ruc}`,
    COMPANY_INFO.address,
    `Tel: ${COMPANY_INFO.phone}  |  Email: ${COMPANY_INFO.email}`,
    COMPANY_INFO.website,
    `Fecha: ${fecha}   Hora: ${hora}`
  ]

  cursorY += 16
  infoLines.forEach((line) => {
    doc.text(line, marginX, cursorY)
    cursorY += 14
  })

  // ===== TÍTULO DEL REPORTE =====
  cursorY += 10
  doc.setFont('helvetica', 'bold')
  doc.setFontSize(13)
  doc.text(title, marginX, cursorY)

  // ===== TABLA DE DATOS =====
  cursorY += 10
  doc.autoTable({
    startY: cursorY,
    head: [columns],  // ['Código', 'Nombre', 'Precio', 'Stock']
    body: rows,       // [['REP-001', 'Pistón', '85.50', '45'], ...]
    styles: {
      fontSize: 9,
      cellPadding: 8
    },
    headStyles: {
      fillColor: [249, 115, 22],  // Naranja
      textColor: [255, 255, 255],
      fontStyle: 'bold'
```

```
    },
    theme: 'striped',  // Filas alternadas
    alternateRowStyles: {
      fillColor: [245, 245, 245]
    }
  })

  // ===== PIE DE PÁGINA =====
  const pageHeight = doc.internal.pageSize.height
  doc.setFontSize(9)
  doc.setTextColor(150)
  doc.text(
    `${COMPANY_INFO.name} - Generado automáticamente`,
    marginX,
    pageHeight - 20
  )

  // ===== GUARDAR PDF =====
  doc.save(`${filename}.pdf`)
}
```

Uso en Dashboard:

```
const handleExportProductsPDF = async () => {
  const columns = ['Código', 'Nombre', 'Categoría', 'Precio', 'Stock']
  const rows = products.map(p => [
    p.codigo,
    p.nombre,
    p.nombreCategoria,
    `S/ ${p.precioUnitario.toFixed(2)}`,
    p.stock
  ])

  await exportToPDF({
    title: 'Reporte de Productos',
    columns,
    rows,
    filename: `productos_${new Date().toISOString().split('T')[0]}`
  })

  toast.notify('PDF generado exitosamente', { type: 'success' })
}
```

Resultado:

```
┌──────────────────────────────────────────────┐
│ Rectificación de Repuestos en Tarapoto SAC │
│ RUC: 20123456789                             │
│ Tarapoto, San Martín - Perú                 │
```

```
| Tel: +51 942 123 456                          |
| Fecha: 21/01/2025   Hora: 14:30:25           |
|                                               |
| REPORTE DE PRODUCTOS                          |
|                                               |
| ┌─────────┬──────────────┬──────────┬───────┐ |
| | Código  | Nombre       | Precio   | Stock | |
| ├─────────┼──────────────┼──────────┼───────┤ |
| |REP-001  | Pistón 125cc | S/ 85.50 |  45   | |
| |REP-002  | Filtro Aceite| S/ 25.00 |  100  | |
| └─────────┴──────────────┴──────────┴───────┘ |
|                                               |
| Rectificación de Repuestos - Generado auto    |
└───────────────────────────────────────────────┘
```

📗 EXPORTAR A EXCEL

Librería: xlsx (SheetJS)

Código (export.js):

```
import * as XLSX from 'xlsx'

export async function exportToExcel({ title, columns, rows, filename = 'reporte'
}) {
  // Crear workbook
  const wb = XLSX.utils.book_new()

  // Crear hoja con encabezados + datos
  const header = [columns]
  const data = header.concat(rows)
  const ws = XLSX.utils.aoa_to_sheet(data)  // Array of arrays to sheet

  // Ajustar ancho de columnas automáticamente
  const colWidths = columns.map((col, idx) => {
    const maxLength = Math.max(
      col.length,
      ...rows.map(row => String(row[idx] || '').length)
    )
    return { wch: Math.max(maxLength, 12) }
  })
  ws['!cols'] = colWidths

  // Estilo de encabezados (primera fila)
  const range = XLSX.utils.decode_range(ws['!ref'])
  for (let C = range.s.c; C <= range.e.c; ++C) {
    const address = XLSX.utils.encode_col(C) + '1'
    if (!ws[address]) continue

    ws[address].s = {
```

```
      font: { bold: true, color: { rgb: 'FFFFFF' } },
      fill: { fgColor: { rgb: 'F97316' } },
      alignment: { horizontal: 'center', vertical: 'center' }
    }
  }

  // Agregar hoja al workbook
  XLSX.utils.book_append_sheet(wb, ws, title.slice(0, 31))  // Max 31 chars

  // Guardar archivo
  XLSX.writeFile(wb, `${filename}.xlsx`)
}
```

Uso:

```
const handleExportProductsExcel = async () => {
  const columns = ['Código', 'Nombre', 'Categoría', 'Precio', 'Stock']
  const rows = products.map(p => [
    p.codigo,
    p.nombre,
    p.nombreCategoria,
    p.precioUnitario,
    p.stock
  ])

  await exportToExcel({
    title: 'Productos',
    columns,
    rows,
    filename: `productos_${new Date().toISOString().split('T')[0]}`
  })

  toast.notify('Excel generado exitosamente', { type: 'success' })
}
```

Resultado: Archivo `.xlsx` compatible con Excel, Google Sheets, LibreOffice Calc

---

DATOS DE LA EMPRESA (company.js)

```
export const COMPANY_INFO = {
  name: 'Rectificación de Repuestos en Tarapoto S.A.C.',
  ruc: '20123456789',
  address: 'Tarapoto, San Martín - Perú',
  phone: '+51 942 123 456',
  email: 'contacto@rectificadora.com',
  website: 'www.rectificadora-tarapoto.com',
  logo: '/assets/logo.png'
}
```

```
// Función para obtener fecha y hora actual en Perú (UTC-5)
export function nowPE() {
  const now = new Date()
  const pe = new Date(now.toLocaleString('en-US', { timeZone: 'America/Lima' }))

  const fecha = pe.toLocaleDateString('es-PE', {
    day: '2-digit',
    month: '2-digit',
    year: 'numeric'
  })

  const hora = pe.toLocaleTimeString('es-PE', {
    hour: '2-digit',
    minute: '2-digit',
    second: '2-digit'
  })

  return { fecha, hora }
}
```

📈 DASHBOARD COMPLETO - Vista Estadísticas

Código en Dashboard.jsx (renderDashboardView):

```
const renderDashboardView = () => {
  return (
    <div className="dashboard-view">
      {/* ===== CARDS DE RESUMEN ===== */}
      <div className="stats-grid">
        <div className="stat-card">
          <div className="stat-icon">
            <LottieIcon name="products" size={32} />
          </div>
          <div className="stat-info">
            <h3>{stats.totalProductos}</h3>
            <p>Productos Totales</p>
          </div>
        </div>

        <div className="stat-card highlight">
          <div className="stat-icon">
            <LottieIcon name="sales" size={32} />
          </div>
          <div className="stat-info">
            <h3>S/ {stats.ingresoHoy.toFixed(2)}</h3>
            <p>Ingresos Hoy</p>
          </div>
        </div>
```

```
          <div className="stat-card">
            <div className="stat-icon">
              <LottieIcon name="calendar" size={32} />
            </div>
            <div className="stat-info">
              <h3>{stats.ventasMes}</h3>
              <p>Ventas del Mes</p>
            </div>
          </div>

          <div className="stat-card warning">
            <div className="stat-icon">
              <LottieIcon name="inventory" size={32} />
            </div>
            <div className="stat-info">
              <h3>{stats.productosBajoStock}</h3>
              <p>Productos Bajo Stock</p>
            </div>
          </div>
        </div>

        {/* ===== GRÁFICOS ===== */}
        <div className="charts-grid">
          <div className="chart-container">
            {renderLineChart()}
          </div>

          <div className="chart-container">
            {renderBarChart()}
          </div>

          <div className="chart-container full-width">
            {renderPieChart()}
          </div>
        </div>

        {/* ===== TABLAS DE DATOS ===== */}
        <div className="tables-grid">
          <div className="table-card">
            <h3>Ventas Recientes</h3>
            <table>
              <thead>
                <tr>
                  <th>ID</th>
                  <th>Fecha</th>
                  <th>Total</th>
                  <th>Vendedor</th>
                </tr>
              </thead>
              <tbody>
                {ventasRecientes.map(venta => (
                  <tr key={venta.idVenta}>
                    <td>#{venta.idVenta}</td>
                    <td>{formatDate(venta.fecha)}</td>
```

```
                  <td>S/ {venta.total.toFixed(2)}</td>
                  <td>{venta.nombreVendedor}</td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>

        <div className="table-card">
          <h3>Productos Bajo Stock</h3>
          <table>
            <thead>
              <tr>
                <th>Producto</th>
                <th>Stock Actual</th>
                <th>Stock Mínimo</th>
                <th>Estado</th>
              </tr>
            </thead>
            <tbody>
              {productosBajoStock.map(producto => (
                <tr key={producto.idProducto}>
                  <td>{producto.nombre}</td>
                  <td className="text-danger">{producto.stock}</td>
                  <td>{producto.stockMinimo}</td>
                  <td>
                    <span className="badge badge-warning">⚠ Bajo</span>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>
    </div>
  )
}
```

CSS Grid para layout:

```
.stats-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
  margin-bottom: 30px;
}

.charts-grid {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 20px;
  margin-bottom: 30px;
```

```
}

.chart-container.full-width {
  grid-column: 1 / -1;
}

.tables-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(500px, 1fr));
  gap: 20px;
}
```

🎨 ANIMACIONES DE CARGA

Skeleton Screens mientras cargan datos:

```
const SkeletonCard = () => (
  <div className="skeleton-card">
    <div className="skeleton-icon"></div>
    <div className="skeleton-text"></div>
    <div className="skeleton-text short"></div>
  </div>
)

// En el render
{loading ? (
  <div className="stats-grid">
    <SkeletonCard />
    <SkeletonCard />
    <SkeletonCard />
    <SkeletonCard />
  </div>
) : (
  renderStatsCards()
)}
```

CSS:

```
.skeleton-card {
  background: var(--color-bg-secondary);
  border-radius: 12px;
  padding: 20px;
  animation: skeleton-loading 1s infinite;
}

@keyframes skeleton-loading {
  0%, 100% { opacity: 0.6; }
  50% { opacity: 1; }
```

```
}

.skeleton-icon {
  width: 48px;
  height: 48px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 50%;
  margin-bottom: 12px;
}

.skeleton-text {
  height: 20px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 4px;
  margin-bottom: 8px;
}

.skeleton-text.short {
  width: 60%;
}
```

Último documento: 06-ANIMACIONES.md