RESUMEN EJECUTIVO - SISTEMA DE INVENTARIO

Guía Rápida para Presentación al Profesor

¿QUÉ ES ESTE SISTEMA?

Sistema Web Full-Stack de Gestión de Inventario para "Rectificadora de Repuestos" en Tarapoto, Perú.

Funciones principales:

- ☑ Gestión de productos, categorías, proveedores
- ☑ Registro de ventas con carrito de compras
- ☑ Control de inventario (entradas/salidas)
- ☑ Reportes y gráficos estadísticos
- ☑ Exportación a PDF y Excel
- ☑ Sistema de usuarios (Admin/Vendedor)
- ☑ Autenticación segura con JWT

LENGUAJES DE PROGRAMACIÓN UTILIZADOS

#	Lenguaje	Uso Principal	Archivos	Tamaño	% Proyecto
1	**JavaScript**	Backend - API REST, lógica de negocio	9,892	66.80 MB	**84.5%**
2	**JSX**	Frontend - Componentes React	24	0.41 MB	**0.5%**
3	**CSS3**	Estilos, animaciones, diseño responsive	50	0.18 MB	**0.2%**
4	**HTML5**	Estructura de páginas web	11	0.41 MB	**0.5%**
5	**SQL**	Base de datos MySQL	4	0.04 MB	**0.05%**
6	**PowerShell**	Scripts de automatización Windows	22	0.03 MB	**0.04%**
7	**JSON**	Configuración, datos, animaciones	550	11.69 MB	**14.7%**

TOTAL: 10,553 archivos = 79.56 MB

ARCHIVOS CLAVE DEL SISTEMA

Backend (JavaScript):

- `index.js` **(1784 líneas)** - Servidor Express + 58 endpoints API REST
- `db.js` (36 líneas) - Conexión MySQL con pool
- `emailService.js` (73 líneas) - Envío de emails con Nodemailer

Frontend (React - JSX):

- `Dashboard.jsx` **(3741 líneas)** - Interfaz principal completa
- `Auth.jsx` (181 líneas) - Login/Registro con animaciones
- 15 componentes modulares (modales, formularios, etc.)

Estilos (CSS):

- `styles.css` **(2507 líneas)** - Estilos globales completos
- `lightMode.css` (174 líneas) - Tema claro/oscuro
- `lottieIcons.css` (290 líneas) - Animaciones iconos

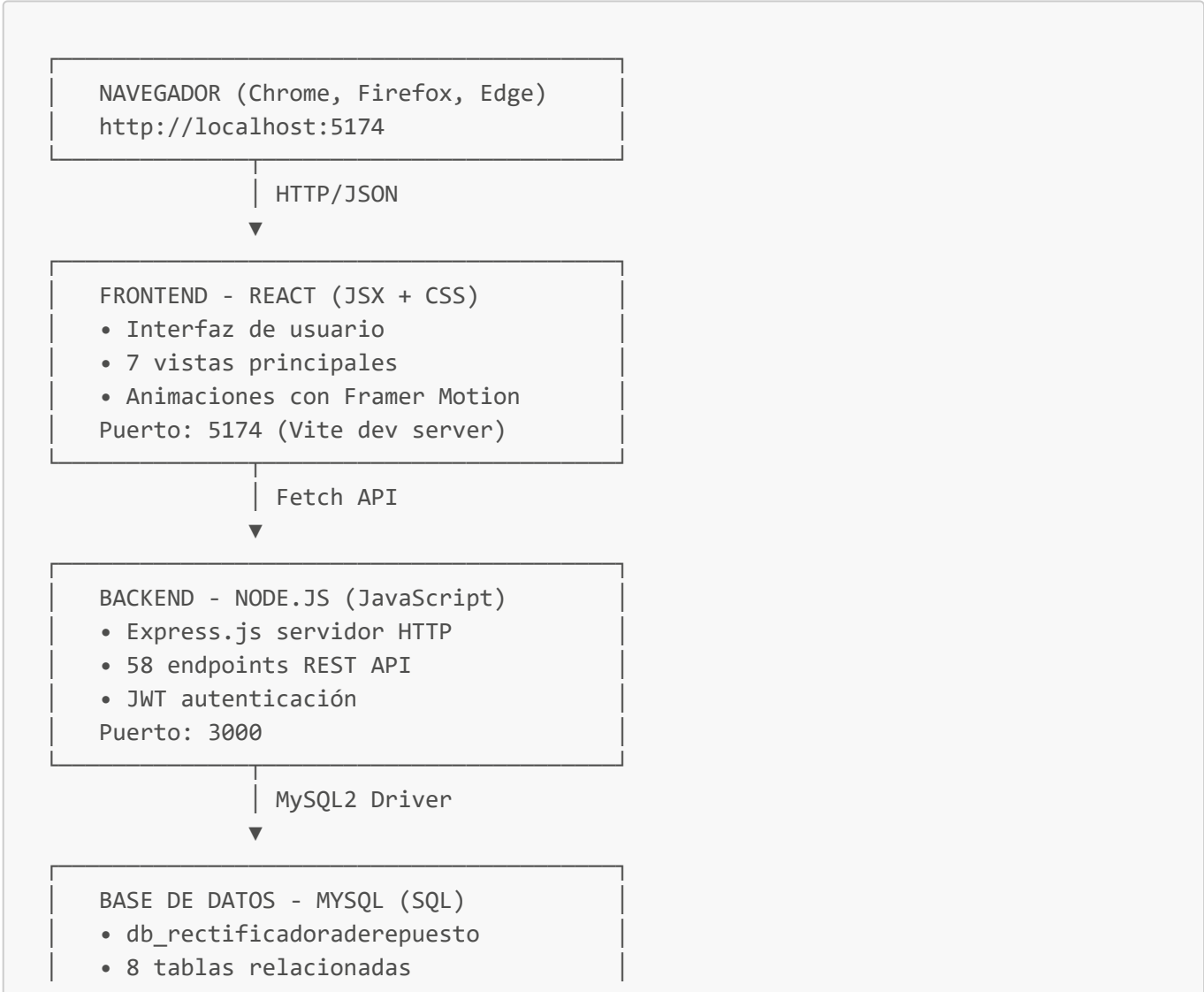
Base de Datos (SQL):

- `ESTRUCTURA_BD_COMPLETA.sql` - 8 tablas MySQL
- `DATOS_PRUEBA.sql` (228 líneas) - Datos iniciales

Automatización (PowerShell):

- `iniciar.ps1` - Inicia backend + frontend
- `cargar-todo.ps1` - Carga base de datos completa

🏗 ARQUITECTURA



```
│   Puerto: 3306                          │
└─────────────────────────────────────────┘
```

🎨 DETALLE DE CADA LENGUAJE

1. JAVASCRIPT (Backend)

¿Qué hace?

- Servidor web con Express.js
- API REST con 58 endpoints
- Autenticación JWT
- Consultas a MySQL
- Envío de emails
- Validación de datos

Ejemplo:

```
// index.js - Endpoint login
app.post('/api/auth/login', async (req, res) => {
  const { email, password } = req.body;

  // Buscar usuario
  const [users] = await pool.query(
    'SELECT * FROM Usuario WHERE email = ?',
    [email]
  );

  // Verificar contraseña
  const valid = await bcrypt.compare(password, users[0].password);

  // Generar token JWT
  const token = jwt.sign({ id: users[0].idUsuario }, SECRET, { expiresIn: '7d' });

  res.json({ token, usuario: users[0] });
});
```

---

2. JSX (Frontend React)

¿Qué hace?

- Componentes de interfaz
- Lógica de interacción
- Manejo de estado (useState, useEffect)
- Formularios dinámicos

Ejemplo:

```
// Dashboard.jsx - Componente de tarjeta de producto
function ProductCard({ producto, onEdit, onDelete }) {
  return (
    <motion.div
      className="product-card"
      whileHover={{ scale: 1.02 }}
    >
      <h3>{producto.nombre}</h3>
      <p>Stock: {producto.stock}</p>
      <p>Precio: S/. {producto.precio}</p>

      <div className="actions">
        <button onClick={() => onEdit(producto)}>Editar</button>
        <button onClick={() => onDelete(producto.idProducto)}>Eliminar</button>
      </div>
    </motion.div>
  );
}
```

---

3. CSS3 (Estilos y Animaciones)

¿Qué hace?

- Diseño visual completo
- Animaciones suaves
- Tema claro/oscuro
- Diseño responsive (móvil/tablet/desktop)
- Efectos hover, transiciones

Archivos principales:

- `styles.css` - 2507 líneas de estilos globales
- `lightMode.css` - Variables de color por tema
- `lottieIcons.css` - Animaciones de iconos
- `searchInput.css` - Estilos del buscador
- `animatedIcons.css` - Animaciones CSS puras

Ejemplo:

```
/* styles.css - Botón con gradiente animado */
.btn-primary {
  background: linear-gradient(135deg, #f97316 0%, #ea580c 100%);
  padding: 12px 24px;
  border-radius: 8px;
  transition: all 0.3s ease;
}

.btn-primary:hover {
```

```
  transform: translateY(-2px);
  box-shadow: 0 8px 20px rgba(249, 115, 22, 0.4);
}

/* Animación de pulso */
@keyframes pulse {
  0%, 100% {
    transform: scale(1);
    opacity: 1;
  }
  50% {
    transform: scale(1.05);
    opacity: 0.8;
  }
}

/* Tema oscuro */
body {
  background: #0b0b0c;
  color: #eaeaea;
}

/* Tema claro */
body.light-mode {
  background: #f5f5f5;
  color: #1a1a1a;
}
```

---

4. HTML5 (Estructura)

¿Qué hace?

- Estructura semántica de las páginas
- Punto de entrada de la aplicación
- Meta tags para SEO
- Etiquetas de accesibilidad

Ejemplo:

```
<!-- frontend-react/index.html -->
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Sistema de Inventario Rectificadora">
  <title>Sistema de Inventario</title>
  <link rel="icon" href="/assets/favicon.png">
</head>
<body>
```

```
  <!-- React se monta aquí -->
  <div id="root"></div>

  <!-- Script principal -->
  <script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

---

5. SQL (Base de Datos)

¿Qué hace?

- Define estructura de 8 tablas
- Relaciones con Foreign Keys
- Datos iniciales de prueba
- Consultas para reportes

Tablas (8):

1. Usuario (login, roles)
2. Categoria (motor, frenos, etc.)
3. Proveedor (contactos, RUC)
4. Producto (stock, precios)
5. Venta (registro de ventas)
6. DetalleVenta (items por venta)
7. MovimientoInventario (entradas/salidas)
8. HistorialPassword (reset password)

Ejemplo:

```
-- ESTRUCTURA_BD_COMPLETA.sql
CREATE TABLE Producto (
  idProducto INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(100) NOT NULL,
  descripcion TEXT,
  precio DECIMAL(10,2) NOT NULL,
  stock INT DEFAULT 0,
  stockMinimo INT DEFAULT 5,
  idCategoria INT,
  idProveedor INT,
  estado TINYINT DEFAULT 1,
  fechaCreacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (idCategoria) REFERENCES Categoria(idCategoria),
  FOREIGN KEY (idProveedor) REFERENCES Proveedor(idProveedor)
);

-- Consulta con JOIN
SELECT
```

```
  p.nombre AS producto,
  c.nombre AS categoria,
  p.stock,
  p.precio
FROM Producto p
INNER JOIN Categoria c ON p.idCategoria = c.idCategoria
WHERE p.stock < p.stockMinimo;
```

---

6. PowerShell (Automatización)

¿Qué hace?

- Inicia el sistema (backend + frontend)
- Carga base de datos automáticamente
- Scripts de mantenimiento
- Colores en consola

Ejemplo:

```
# iniciar.ps1
Write-Host "`n╔═══════════════════════════════════════════╗" -ForegroundColor Cyan
Write-Host "║   SISTEMA DE INVENTARIO - INICIANDO     ║" -ForegroundColor Cyan
Write-Host "╚═══════════════════════════════════════════╝`n" -ForegroundColor Cyan

# Iniciar Backend (Puerto 3000)
Write-Host "→ Iniciando Backend..." -ForegroundColor Green
Start-Process powershell -ArgumentList "-NoExit", "-Command", "cd '$PSScriptRoot';
node index.js"

Start-Sleep -Seconds 2

# Iniciar Frontend (Puerto 5174)
Write-Host "→ Iniciando Frontend..." -ForegroundColor Cyan
Start-Process powershell -ArgumentList "-NoExit", "-Command", "cd
'$PSScriptRoot\frontend-react'; npm run dev"

Write-Host "`n✓ Sistema iniciado correctamente" -ForegroundColor Green
Write-Host "→ Accede en: http://localhost:5174`n" -ForegroundColor Yellow
```

---

7. JSON (Configuración)

¿Qué hace?

- Configuración de dependencias (package.json)
- Animaciones Lottie (iconos animados)
- Configuración de Vite, ESLint
- Datos estructurados

Ejemplo:

```
// package.json (Backend)
{
  "name": "backend-inventario",
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "node index.js",
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^5.1.0",
    "mysql2": "^3.15.2",
    "jsonwebtoken": "^9.0.2",
    "bcrypt": "^6.0.0",
    "nodemailer": "^7.0.9",
    "express-validator": "^8.0.1",
    "express-rate-limit": "^8.1.0",
    "helmet": "^8.1.0",
    "cors": "^2.8.5"
  }
}
```

🎓 RESPUESTAS RÁPIDAS PARA EL PROFESOR

¿Qué lenguaje usaron?

7 lenguajes: JavaScript (principal), JSX, CSS, HTML, SQL, PowerShell, JSON

¿Por qué JavaScript?

- Full-Stack: mismo lenguaje frontend y backend
- Ecosistema: 1.3 millones de paquetes NPM
- Comunidad: Lenguaje más usado del mundo
- Performance: Motor V8 muy rápido
- Asíncrono: Ideal para APIs y base de datos

¿Cómo funciona?

1. Usuario entra a `http://localhost:5174` (HTML/CSS/JSX)
2. Frontend hace peticiones a `http://localhost:3000/api` (JavaScript)
3. Backend consulta MySQL (SQL)
4. Backend responde JSON al frontend
5. Frontend muestra datos con animaciones (CSS/Framer Motion)

¿Qué hace cada lenguaje?

- **JavaScript:** Lógica del servidor (58 endpoints)

- **JSX:** Interfaz React (24 componentes)
- **CSS:** Diseño visual y animaciones (2507 líneas principales)
- **HTML:** Estructura de páginas (11 archivos)
- **SQL:** Base de datos (8 tablas relacionadas)
- **PowerShell:** Automatización de inicio (4 scripts)
- **JSON:** Configuración y datos (550 archivos)

¿Qué bibliotecas usan?

Backend: Express, MySQL2, JWT, Bcrypt, Nodemailer
Frontend: React, Framer Motion, Recharts, jsPDF, xlsx
Total: 44 dependencias (26 backend + 18 frontend)

¿Cómo se inicia?

1. Doble click en `iniciar.ps1` (PowerShell)
2. Se abren 2 terminales automáticamente
3. Backend inicia en puerto 3000
4. Frontend inicia en puerto 5174
5. Navegar a `http://localhost:5174`

¿Cómo funcionan las animaciones?

- **Framer Motion** (librería React): Modales, transiciones de página
- **CSS @keyframes**: Iconos, botones, efectos hover
- **Lottie JSON**: Animaciones vectoriales complejas

¿Cuántas líneas de código?

- `index.js`: 1784 líneas (backend)
- `Dashboard.jsx`: 3741 líneas (frontend)
- `styles.css`: 2507 líneas (estilos)
- **Total estimado:** ~8000 líneas de código propio

📚 DOCUMENTOS COMPLETOS

Has preparado **6 documentos técnicos** que cubren TODO el sistema:

1. **00-RESUMEN-EJECUTIVO.md** (este documento) - Vista general
2. **01-INTRODUCCION-GENERAL.md** - Arquitectura, lenguajes, tecnologías
3. **02-ESTRUCTURA-ARCHIVOS.md** - Árbol completo de archivos
4. **03-API-ENDPOINTS.md** - 58 endpoints documentados
5. **04-COMPONENTES-FRONTEND.md** - 15 componentes React
6. **05-GRAFICOS-REPORTES.md** - Recharts, PDF, Excel
7. **06-ANIMACIONES.md** - Framer Motion, CSS animations

☑ CHECKLIST PARA PRESENTACIÓN

- ☑ **Lenguajes:** JavaScript, JSX, CSS, HTML, SQL, PowerShell, JSON
- ☑ **Arquitectura:** Cliente-Servidor, REST API
- ☑ **Frontend:** React con 24 componentes JSX + 50 archivos CSS
- ☑ **Backend:** Node.js con 58 endpoints JavaScript
- ☑ **Base de Datos:** MySQL con 8 tablas SQL
- ☑ **Automatización:** 4 scripts PowerShell
- ☑ **Animaciones:** Framer Motion + CSS + Lottie JSON
- ☑ **Seguridad:** JWT, Bcrypt, Rate Limiting, Helmet
- ☑ **Exportación:** PDF (jsPDF) y Excel (xlsx)
- ☑ **Gráficos:** Recharts (LineChart, BarChart, PieChart)

¡TODO LISTO PARA CUALQUIER PREGUNTA DEL PROFESOR! 🎓✨