📚 DOCUMENTACIÓN COMPLETA DEL SISTEMA DE INVENTARIO

Rectificadora de Repuestos - Tarapoto

🎯 RESUMEN EJECUTIVO

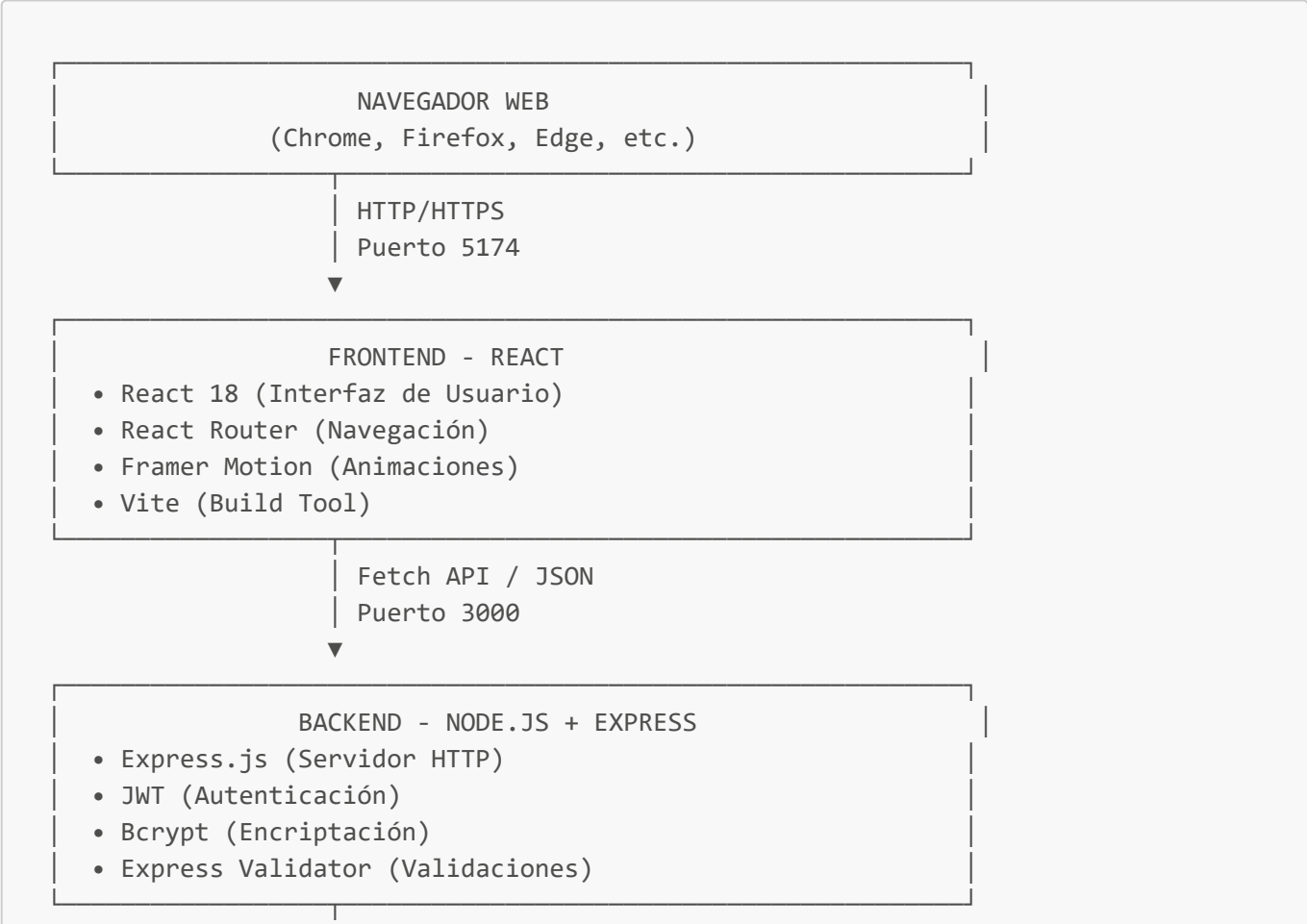
Este es un **Sistema Web Full-Stack** de gestión de inventario para la empresa "Rectificadora de Repuestos" ubicada en Tarapoto, Perú. El sistema permite administrar productos, categorías, proveedores, ventas y movimientos de inventario con dos tipos de usuarios: **Administradores** y **Vendedores**.

🏗️ ARQUITECTURA DEL SISTEMA

Tipo de Aplicación

- **Arquitectura**: Cliente-Servidor (Client-Server)
- **Patrón**: MVC (Model-View-Controller) adaptado para APIs REST
- **Comunicación**: HTTP/HTTPS con JSON
- **Base de Datos**: Relacional (MySQL)

Componentes Principales



```
                    | MySQL2 Driver
                    | Puerto 3306
                    ▼
┌──────────────────────────────────────────────────────────┐
│              BASE DE DATOS - MYSQL                       │
│  • db_rectificadoraderepuesto                            │
│  • 8 Tablas principales                                  │
│  • Relaciones con Foreign Keys                           │
└──────────────────────────────────────────────────────────┘
```

💻 LENGUAJES DE PROGRAMACIÓN UTILIZADOS

El sistema utiliza **7 lenguajes diferentes** trabajando en conjunto:

Lenguaje	Uso	Archivos	Peso Total
JavaScript	Backend - API REST y lógica	9,892 archivos `.js`	66.80 MB
JSX	Frontend - Componentes React	24 archivos `.jsx`	0.41 MB
CSS3	Estilos y animaciones	50 archivos `.css`	0.18 MB
HTML5	Estructura de páginas web	11 archivos `.html`	0.41 MB
SQL	Base de datos MySQL	4 archivos `.sql`	0.04 MB
PowerShell	Scripts de automatización	22 archivos `.ps1`	0.03 MB
JSON	Configuración y datos	550 archivos `.json`	11.69 MB

---

1. JavaScript (Lenguaje Principal)

Backend: JavaScript (Node.js) - 9,892 archivos

- **Versión**: ECMAScript 2015+ (ES6+)
- **Runtime**: Node.js v16+
- **Módulos**: ESM (import/export) - Especificado en package.json con `"type": "module"`
- **Archivos clave**: `index.js` (1784 líneas), `db.js`, `emailService.js`, `crearAdmin.js`

¿Por qué JavaScript en el backend?

- Node.js permite usar JavaScript tanto en frontend como backend (Full-Stack JavaScript)
- Alto rendimiento para operaciones I/O (lectura/escritura de base de datos)
- Gran ecosistema de paquetes NPM (1.3 millones de paquetes)
- Ideal para APIs REST y aplicaciones en tiempo real

Frontend: JSX (React) - 24 archivos

- **Versión**: ECMAScript 2015+ (ES6+)
- **Framework**: React 18 (librería de componentes)
- **Sintaxis**: JSX (JavaScript XML) - permite escribir HTML dentro de JavaScript

- **Archivo principal**: `Dashboard.jsx` (3741 líneas)

Ejemplo de código backend (JavaScript puro):

```
import express from 'express';
const app = express();

app.get('/api/productos', async (req, res) => {
  const [productos] = await pool.query('SELECT * FROM Producto');
  res.json({ productos });
});
```

Ejemplo de código frontend (JSX - React):

```
export default function ProductCard({ producto }) {
  return (
    <div className="card">
      <h3>{producto.nombre}</h3>
      <p>Stock: {producto.stock}</p>
    </div>
  );
}
```

---

2. CSS3 (Estilos y Animaciones) - 50 archivos

- **Versión**: CSS3
- **Uso**: Estilos visuales, animaciones, diseño responsive
- **Archivos principales**:
  - `styles.css` (2507 líneas) - Estilos globales completos
  - `lightMode.css` - Tema claro/oscuro
  - `lottieIcons.css` - Animaciones de iconos Lottie
  - `searchInput.css` - Estilos del componente buscador
  - `animatedIcons.css` - Animaciones CSS puras

Características CSS usadas:

- Flexbox y Grid Layout (diseño flexible)
- Animaciones con `@keyframes` (iconos, transiciones)
- Variables CSS (`:root { --color-primary: #f97316; }`)
- Media queries (diseño responsive para móviles)
- Gradientes lineales (`linear-gradient`)
- Transformaciones (`transform`, `rotate`, `scale`)
- Transiciones suaves (`transition`)

Ejemplo:

```
.button {
  background: linear-gradient(135deg, #f97316 0%, #ea580c 100%);
  animation: pulse 2s infinite;
  transition: all 0.3s ease;
}

@keyframes pulse {
  0%, 100% { transform: scale(1); }
  50% { transform: scale(1.05); }
}

/* Tema oscuro con variables */
:root {
  --bg-primary: #0b0b0c;
  --text-color: #eaeaea;
}
```

---

3. HTML5 (Estructura Web) - 11 archivos

- **Uso**: Estructura semántica del frontend
- **Archivos**:
  - `frontend-react/index.html` - Aplicación React (SPA)
  - `public/index.html` - Login sistema antiguo
  - `public/dashboard.html` - Dashboard antiguo

Características HTML5 usadas:

- Etiquetas semánticas (`<header>`, `<nav>`, `<main>`, `<section>`, `<article>`)
- Atributos de accesibilidad (`aria-label`, `aria-hidden`, `role`)
- Meta tags para SEO y viewport responsive
- Formularios HTML5 (`type="email"`, `required`, `pattern`)

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sistema de Inventario</title>
</head>
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

---

4. SQL (Base de Datos) - 4 archivos

- **Base de Datos**: MySQL 8.0+
- **Uso**: Consultas, creación de tablas, datos de prueba
- **Archivos**:
  - `ESTRUCTURA_BD_COMPLETA.sql` - 8 tablas con relaciones
  - `DATOS_PRUEBA.sql` - 228 líneas con datos iniciales
  - `DATOS_VENTAS_MOVIMIENTOS.sql` - Datos de ventas ejemplo
  - `AGREGAR_CAMPOS_RESET_PASSWORD.sql` - Migración reset password

Características SQL usadas:

- DDL: `CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`
- DML: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- Constraints: `PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, `NOT NULL`
- Funciones: `NOW()`, `COUNT()`, `SUM()`, `AVG()`
- Joins: `INNER JOIN`, `LEFT JOIN`

Ejemplo:

```sql
-- Crear tabla
CREATE TABLE Producto (
  idProducto INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(100) NOT NULL,
  stock INT DEFAULT 0,
  idCategoria INT,
  FOREIGN KEY (idCategoria) REFERENCES Categoria(idCategoria)
);

-- Consulta con JOIN
SELECT p.nombre, c.nombre AS categoria, p.stock
FROM Producto p
INNER JOIN Categoria c ON p.idCategoria = c.idCategoria
WHERE p.stock < 10;
```

---

5. PowerShell (Automatización) - 22 archivos

- **Versión**: PowerShell 5.1+ (Windows)
- **Uso**: Scripts para iniciar el sistema y cargar datos
- **Archivos principales**:
  - `iniciar.ps1` - Inicia backend (puerto 3000) + frontend (puerto 5174)
  - `iniciar-simple.ps1` - Inicia solo el backend
  - `cargar-todo.ps1` - Ejecuta todos los SQL en orden
  - `cargar-ventas.ps1` - Carga solo datos de ventas

Características PowerShell usadas:

- Cmdlets nativos (`Start-Process`, `Write-Host`)

- Ejecución de comandos externos (`npm`, `mysql`)
- Colores en consola (`-ForegroundColor`)
- Manejo de rutas (`Get-Location`, `Set-Location`)

Ejemplo:

```
# iniciar.ps1
Write-Host "Iniciando Backend..." -ForegroundColor Green
Start-Process powershell -ArgumentList "-NoExit", "-Command", "cd backend; npm run
dev"

Write-Host "Iniciando Frontend..." -ForegroundColor Cyan
Start-Process powershell -ArgumentList "-NoExit", "-Command", "cd frontend-react;
npm run dev"

Write-Host "Sistema listo en http://localhost:5174" -ForegroundColor Yellow
```

---

6. JSON (Configuración y Datos) - 550 archivos

- **Uso**: Configuración de paquetes, animaciones Lottie, datos estructurados
- **Archivos principales**:
  - `package.json` - Dependencias backend (26 paquetes)
  - `frontend-react/package.json` - Dependencias React (18 paquetes)
  - `assets/lottie/dashboard.json` - Animación Lottie dashboard
  - `assets/lottie/products.json` - Animación Lottie productos
  - `assets/lottie/sales.json` - Animación Lottie ventas

Ejemplo package.json:

```
{
  "name": "backend",
  "type": "module",
  "scripts": {
    "dev": "node index.js"
  },
  "dependencies": {
    "express": "^5.1.0",
    "mysql2": "^3.15.2",
    "jsonwebtoken": "^9.0.2"
  }
}
```

📊 RESUMEN DE LENGUAJES

Aspecto	JavaScript/JSX	CSS	HTML	SQL	PowerShell	JSON

Aspecto	JavaScript/JSX	CSS	HTML	SQL	PowerShell	JSON
Archivos	9,916	50	11	4	22	550
Tamaño	67.21 MB	0.18 MB	0.41 MB	0.04 MB	0.03 MB	11.69 MB
% del proyecto	84.5%	0.2%	0.5%	0.05%	0.04%	14.7%
Propósito	Lógica + UI	Estilos	Estructura	Datos	Scripts	Config

Conclusión: El sistema es principalmente **JavaScript** (84.5%) con **CSS** para estilos, **HTML** para estructura, **SQL** para base de datos, **PowerShell** para automatización y **JSON** para configuración.

🔧 TECNOLOGÍAS Y HERRAMIENTAS

BACKEND (Node.js + Express)

---

2. SQL (Structured Query Language)

- **Base de Datos**: MySQL 8.0+
- **Uso**: Consultas, creación de tablas, relaciones

Ejemplo:

```sql
CREATE TABLE Producto (
  idProducto INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(100) NOT NULL,
  stock INT DEFAULT 0
);

SELECT * FROM Producto WHERE stock < 10;
```

3. CSS (Cascading Style Sheets)

- **Versión**: CSS3
- **Uso**: Estilos visuales, animaciones, responsive design
- **Características usadas**:
  - Flexbox y Grid
  - Animaciones (@keyframes)
  - Variables CSS (--color-primary)
  - Media queries (responsive)

Ejemplo:

```css
.button {
  background: linear-gradient(135deg, #f97316 0%, #ea580c 100%);
  animation: pulse 2s infinite;
}
```

```
@keyframes pulse {
  0%, 100% { transform: scale(1); }
  50% { transform: scale(1.05); }
}
```

4. HTML5

- **Uso**: Estructura semántica del frontend
- **Características**:
  - Semántica moderna (
    ,
    ,
    ,
)
  - Atributos de accesibilidad (aria-*)
  - Meta tags para SEO

🛠️ TECNOLOGÍAS Y FRAMEWORKS

FRONTEND (Cliente)

Tecnología	Versión	Propósito
React	18.3.1	Framework principal de UI - Crea interfaces interactivas con componentes reutilizables
React Router DOM	6.26.2	Navegación entre páginas (SPA - Single Page Application) sin recargar
Vite	5.4.8	Build tool ultra-rápido - Compila y optimiza el código
Framer Motion	12.23.24	Animaciones fluidas y transiciones entre componentes
Styled Components	6.1.19	CSS-in-JS - Estilos con JavaScript para componentes dinámicos
Recharts	3.3.0	Gráficos interactivos (líneas, barras, pastel) para dashboard
jsPDF	3.0.3	Generación de reportes en formato PDF
jsPDF AutoTable	5.0.2	Tablas automáticas en PDFs
XLSX	0.18.5	Exportación a Excel (.xlsx)
Lottie React	2.4.1	Animaciones JSON (íconos animados)
LDRS	1.1.7	Loaders y spinners animados

BACKEND (Servidor)

Tecnología	Versión	Propósito
Node.js	v16+	Runtime de JavaScript en el servidor
Express.js	5.1.0	Framework web - Maneja rutas HTTP (GET, POST, PUT, DELETE)
MySQL2	3.15.2	Driver para conectar Node.js con MySQL - Soporta Promises
Bcrypt	6.0.0	Encriptación de contraseñas con hash seguro (10 rounds)
JsonWebToken (JWT)	9.0.2	Tokens de autenticación - Sesiones sin cookies
Dotenv	17.2.3	Variables de entorno (.env) para configuración segura
CORS	2.8.5	Control de acceso entre dominios (Frontend ↔ Backend)
Helmet	8.1.0	Headers de seguridad HTTP (XSS, clickjacking, etc.)
Express Rate Limit	8.1.0	Limitar intentos de login (anti fuerza bruta)
Express Validator	7.2.1	Validación y sanitización de datos de entrada
Nodemailer	7.0.9	Envío de emails (recuperación de contraseña)

📊 BASE DE DATOS

Motor: MySQL 8.0

- **Servidor**: XAMPP (localhost:3306)
- **Usuario**: root (sin contraseña por defecto)
- **Nombre BD**: `db_rectificadoraderepuesto`
- **Charset**: utf8mb4 (soporta emojis y caracteres especiales)

8 Tablas Principales:

1. **Rol** - Define permisos (Administrador, Vendedor)
2. **Usuario** - Datos de usuarios del sistema
3. **Categoria** - Clasificación de productos (Repuestos, Accesorios, etc.)
4. **Proveedor** - Empresas que suministran productos
5. **Producto** - Inventario de repuestos para motos
6. **Venta** - Registro de transacciones de venta
7. **DetalleVenta** - Items individuales de cada venta
8. **MovimientoInventario** - Entradas y salidas de stock

👥 ROLES Y PERMISOS

1. Administrador (idRol = 1)

- ☑ Acceso total al sistema (58 endpoints)
- ☑ Gestión de usuarios (crear, editar, desactivar)
- ☑ Gestión de productos, categorías, proveedores
- ☑ Registro de ventas

- ☑ Movimientos de inventario
- ☑ Ver reportes y estadísticas
- ☑ Exportar datos (PDF, Excel)

2. Vendedor (idRol = 2)

- ☑ Ver dashboard con estadísticas (50 endpoints)
- ☑ Gestión de productos, categorías, proveedores
- ☑ Registro de ventas
- ☑ Movimientos de inventario
- ☑ Ver su propio perfil
- ❌ NO puede gestionar usuarios (crear, editar, desactivar otros usuarios)

🚀 FLUJO DE TRABAJO DEL SISTEMA

1. Usuario accede al sistema:

```
Usuario → Navegador (localhost:5174) → Frontend React
```

2. Login / Autenticación:

```
1. Usuario ingresa email y contraseña
2. Frontend envía POST /api/usuarios/login
3. Backend verifica en base de datos
4. Backend encripta con bcrypt y compara
5. Backend genera token JWT (válido 8 horas)
6. Frontend guarda token en localStorage
7. Usuario redirigido al Dashboard
```

3. Navegación protegida:

```
1. Usuario intenta acceder a /dashboard
2. ProtectedRoute verifica token JWT
3. Si válido → Muestra Dashboard
4. Si inválido/expirado → Redirige a Login
```

4. Operación CRUD (ejemplo: Crear producto):

```
1. Usuario llena formulario "Nuevo Producto"
2. Frontend valida datos (precio > 0, stock >= 0)
3. Frontend envía POST /api/productos con token JWT
4. Backend verifica token
5. Backend valida datos con Express Validator
```

```
6. Backend inserta en base de datos MySQL
7. Backend responde con producto creado
8. Frontend actualiza lista de productos
9. Muestra notificación toast "Producto creado"
```

🎨 CARACTERÍSTICAS DE LA INTERFAZ

Diseño Visual:

- ✨ **Tema Dual**: Modo oscuro (predeterminado) y modo claro
- 🎭 **Animaciones**: Transiciones suaves con Framer Motion
- 📱 **Responsive**: Adaptable a móviles, tablets y escritorio
- 🎯 **Iconos**: SVG animados con CSS personalizado
- 🌈 **Paleta de colores**: Naranja (#f97316) como color principal

Animaciones Implementadas:

1. **Página de Login/Registro:**
    - Transición lateral entre Login ↔ Registro
    - Fade in de formularios
    - Pulse en botones al hover
2. **Dashboard:**
    - Sidebar que se expande/contrae
    - Transición de contenido al cambiar de vista
    - Cards con hover elevado
    - Gráficos animados (Recharts)
3. **Modales:**
    - Backdrop con fade in
    - Modal que escala desde 0.9 a 1
    - Animación de salida suave
4. **Notificaciones Toast:**
    - Deslizamiento desde abajo (Y: 20 → 0)
    - Fade in/out automático
    - Auto-desaparición en 3 segundos
5. **Switch de Tema:**
    - Sol/Luna animado con CSS
    - Nubes flotantes en modo día
    - Estrellas parpadeantes en modo noche
    - Rotación del icono al cambiar
6. **Iconos de Menú:**

- Rotación al hacer hover
- Escala y brillo en hover
- Animaciones específicas por icono

7. **Loaders:**

- Spinner circular con LDRS
- Skeleton screens en carga de datos
- Progress bars en operaciones largas

SEGURIDAD IMPLEMENTADA

1. Autenticación JWT:

- Token firmado con clave secreta (JWT_SECRET)
- Expiración de 8 horas
- Renovación automática antes de expirar

2. Encriptación de Contraseñas:

- Bcrypt con 10 rounds de salt
- Nunca se almacenan contraseñas en texto plano

3. Rate Limiting:

- Máximo 5 intentos de login en 15 minutos
- Previene ataques de fuerza bruta

4. Validaciones:

- Express Validator en backend
- Validación de formato de emails
- Sanitización de inputs (previene XSS)

5. Headers de Seguridad (Helmet.js):

- X-Content-Type-Options: nosniff
- X-Frame-Options: DENY
- X-XSS-Protection: 1; mode=block
- Strict-Transport-Security (HSTS)

6. CORS Configurado:

- Solo permite requests desde localhost:5174
- Credenciales habilitadas para cookies

ESTRUCTURA DE ARCHIVOS

Ver documento: **02-ESTRUCTURA-ARCHIVOS.md**

API ENDPOINTS

Ver documento: **03-API-ENDPOINTS.md**

COMPONENTES FRONTEND

Ver documento: **04-COMPONENTES-FRONTEND.md**

GRÁFICOS Y REPORTES

Ver documento: **05-GRAFICOS-REPORTES.md**

ANIMACIONES DETALLADAS

Ver documento: **06-ANIMACIONES.md**

PREGUNTAS FRECUENTES DEL PROFESOR

1. ¿Qué lenguaje usaron?

JavaScript - Tanto en backend (Node.js) como frontend (React)

2. ¿Por qué eligieron JavaScript?

- Full-Stack: Un solo lenguaje para todo el proyecto
- React: Interfaz moderna y reactiva
- Node.js: Alto rendimiento para APIs
- Gran comunidad y librerías disponibles

3. ¿Cómo se conecta el frontend con el backend?

- Frontend (puerto 5174) hace peticiones HTTP al backend (puerto 3000)
- Usa Fetch API para enviar/recibir JSON
- Token JWT en header Authorization

4. ¿Cómo funcionan las animaciones?

- Framer Motion: Animaciones de componentes React
- CSS: Animaciones de iconos con @keyframes
- Styled Components: Estilos dinámicos con props

5. ¿Cómo es el tema claro/oscuro?

- Switch con Styled Components y CSS
- Guarda preferencia en localStorage
- Cambia clases CSS en

6. ¿Qué pasa si el token expira?

- Middleware verifica token en cada request

- Si expiró: Backend responde 401
- Frontend detecta 401 y redirige a login

7. ¿Cómo se generan los PDFs?

- Librería jsPDF en el navegador
- jsPDF AutoTable para tablas
- Se genera en cliente, no en servidor

🎓 CONCEPTOS CLAVE PARA EXPLICAR

1. Single Page Application (SPA)

No hay recarga de página. React Router cambia URL y componentes sin pedir nuevos HTML al servidor.

2. REST API

Arquitectura donde cada endpoint representa un recurso:

- GET /api/productos → Leer todos
- POST /api/productos → Crear uno
- PUT /api/productos/5 → Actualizar producto 5
- DELETE /api/productos/5 → Eliminar producto 5

3. JWT (JSON Web Token)

Token firmado que contiene datos del usuario. Se envía en cada request para verificar identidad sin cookies.

4. Async/Await

Manejo moderno de operaciones asíncronas (base de datos, APIs):

```
async function getProductos() {
  const response = await fetch('/api/productos');
  const data = await response.json();
  return data;
}
```

5. React Hooks

Funciones especiales de React:

- useState: Mantener estado local
- useEffect: Ejecutar código al montar/desmontar
- useContext: Compartir datos globalmente

Continúa en los siguientes documentos...