■ Arquitectura y Tecnologías del Proyecto SEGA-Cockpit

■■ Arquitectura General

Tipo de Arquitectura: Microservicios con API Gateway

- **Backend:** Servicio independiente con API REST
- **Frontend:** Aplicación web SPA (Single Page Application)
- **Comunicación:** HTTP/REST con proxy interno
- **Separación:** Totalmente desacoplados, desplegables por separado

---

■ Backend

■ *Stack Tecnológico*

Tecnología	Versión	Propósito
TypeScript	Latest	Tipado estático sobre JavaScript
Node.js	LTS	Entorno de ejecución JavaScript
Fastify	Latest	Framework web de alto rendimiento
Microsoft SQL Server	2019+	Base de datos relacional
mssql	Latest	Conector oficial para Node.js

■■ *Arquitectura Backend*

backend/
■■■ src/
■ ■■■ routes/ # Endpoints API
■ ■ ■■■ productivity.ts # Ruta principal de productividad
■ ■■■ db.ts # Configuración de conexión a BD
■ ■■■ server.ts # Punto de entrada del servidor
■ ■■■ config.ts # Variables de entorno

■■■ package.json # Dependencias y scripts

■■■ tsconfig.json # Configuración TypeScript

■ *Características Principales*

- **Tipado fuerte** - TypeScript para seguridad y mantenibilidad
- **Rápidas respuestas** - Fastify optimizado para alto rendimiento
- **Pool de conexiones** - Gestión eficiente de conexiones a BD
- **Caching integrado** - Cache en memoria para respuestas frecuentes
- **Manejo de errores** - Middleware centralizado de errores
- **Logging estructurado** - Logs con niveles y metadatos

■ *API Backend*

Endpoint Principal

GET /productividad?operacion=PICKING&fromDate;=2025-01-01&toDate;=2025-01-10

Parámetros Soportados

- **operacion:** PICKING, CROSSDOCKING, EXTRACCION, REPOSICION, ALMACENAJE, RECEPCION
- **fromDate/toDate:** Rango de fechas (inclusivo)
- **Timeout:** 60 segundos para queries complejas

Estructura de Respuesta

{

daily: Array<{

fecha_operativa: string,

unidades: number,

movimientos: number,

segundos: number,

uni_x_h: number,

mov_x_h: number

}>,

perOperator: Array<{

usuario_id: number,

operario: string,

legajo: string,

horas: number,

unidades: number,
movimientos: number,
cajas: number,
unidades_uom: number,
packs: number,
uni_x_h: number,
mov_x_h: number,
productividad_media: number
}>,
dailyDetailGrid: Array<{
fecha_operativa: string,
usuario_id: number,
operario: string,
bultos: number,
minutos: number,
productividad: number
}>,
cards: {
cajas: number,
unidades_uom: number,
packs: number,
pallets: number,
operarios: number,
horas_promedio_por_operario: number
}
}

■ Frontend

■ *Stack Tecnológico*

Tecnología	Versión	Propósito
TypeScript	Latest	Tipado estático
Next.js	13+	Framework React con App Router

| **React** | 18 | Component-based UI |

| **Tailwind CSS** | 3+ | Utility-first CSS |

| **Recharts** | Latest | Librería de gráficos React |

| **Webpack** | Integrado | Build tool (Next.js) |

■■ *Arquitectura Frontend*

frontend/

■■■ src/

■ ■■■ app/

■ ■ ■■■ dashboard/

■ ■ ■ ■■■ productivity/

■ ■ ■ ■■■ page.tsx # Página principal

■ ■ ■■■ api/

■ ■ ■■■ productividad/

■ ■ ■■■ route.ts # Proxy API

■ ■■■ components/

■ ■ ■■■ dashboard/

■ ■ ■■■ ExpandableGrid.tsx # Componente reutilizable

■ ■■■ app/

■ ■■■ layout.tsx # Layout principal

■ ■■■ page.tsx # Home

■■■ package.json # Dependencias

■■■ tailwind.config.js # Configuración Tailwind

■■■ next.config.js # Configuración Next.js

■ *Características Principales*

- **Server-Side Rendering** - Next.js App Router
- **Component-based** - Arquitectura de componentes React
- **Tipado end-to-end** - TypeScript en frontend y backend
- **Responsive Design** - Tailwind CSS mobile-first
- **Dark Mode** - Soporte completo para temas claro/oscuro
- **API Proxy** - Proxy interno para seguridad y CORS

■ *Componentes Principales*

1. Página Principal (`productivity/page.tsx`)

Secciones:

- **6 KPI Cards** - Métricas principales con iconos
- **2 Gráficos** - Tendencias temporales con Recharts
- **1 Grilla Expandible** - Componente personalizado con niveles
- **1 Grilla Detallada** - Tabla tradicional con datos brutos
- **Filtros** - Fechas y operación con validación

KPI Cards:

1. ■ Cajas - Conteo de cajas procesadas
2. ■ Packs - Conteo de packs procesados
3. ■ ULs - Conteo de unidades individuales
4. ■■ Pallets - Conteo de pallets movidos
5. ■ Operarios - Número de operarios activos
6. ■■ Horas Promedio - Promedio de horas por operario

2. Componente ExpandableGrid (`ExpandableGrid.tsx`)

Características:

- **Niveles jerárquicos** - Resumen → Detalles
- **Carga asíncrona** - Soporte para Promise en getDetailData
- **Estados de carga** - Spinners y skeletons
- **Renderizado personalizado** - Columnas con render functions
- **Reusable** - Componente genérico para cualquier data

Columnas del Resumen:

- Operario (con legajo)
- ULs
- Movimientos
- Horas
- ULs/Hora
- Movimientos/Hora
- Productividad Media

Columnas del Detalle:

- Fecha
- Bultos

- Minutos
- Productividad

■ Comunicación y Flujo de Datos

■ *Arquitectura de Comunicación*

Usuario → Frontend → API Proxy → Backend → Base de Datos
↓ ↓ ↓ ↓ ↓
Interfaz Next.js Route.ts Fastify MSSQL
React Proxy CORS REST SQL Queries

Flujo de Datos Detallado

1. Frontend → API Proxy
// Frontend llama a proxy interno
const response = await fetch(`/api/productividad?operacion=PICKING&fromDate;=...&toDate;=...`);

2. API Proxy → Backend
// Proxy reenvía a backend real
const backendResponse = await fetch(`${BACKEND_URL}/productividad?${searchParams}`);

3. Backend → Base de Datos
// Backend ejecuta queries SQL
const result = await connection.request()
.input('operacion', sql.NVarChar, operacion)
.query(complexSQLQuery);

■ Seguridad y Optimización

■ *Seguridad*

- **API Proxy** - Oculta URL del backend
- **CORS Management** - Control de orígenes permitidos
- **Parameter Validation** - Validación estricta en backend
- **SQL Injection Protection** - Queries parametrizadas

■ *Optimización*

- **Connection Pooling** - Reuso de conexiones BD
- **Response Caching** - Cache en memoria (TTL configurable)
- **Lazy Loading** - Carga bajo demanda en grilla expandible
- **Code Splitting** - Next.js optimiza bundles automáticamente

---

■ Tecnologías por Capa

Capa	Tecnología	Propósito
Datos	Microsoft SQL Server	Almacenamiento persistente
Backend	Node.js + TypeScript + Fastify	API REST de alto rendimiento
Frontend	Next.js + React + TypeScript	SPA con Server-Side Rendering
Estilos	Tailwind CSS	Utility-first CSS responsive
Gráficos	Recharts	Visualización de datos
Build	Webpack (Next.js)	Bundling y optimización

■ Ventajas de esta Arquitectura

■ *Escalabilidad*

- **Microservicios independientes** - Backend y frontend pueden escalar por separado
- **Desacoplamiento total** - Cambios en uno no afectan al otro
- **Despliegue independiente** - Actualizaciones sin downtime del otro servicio

■ *Mantenibilidad*

- **TypeScript tipado fuerte** - Detección temprana de errores
- **Componentes reutilizables** - DRY principle
- **Código modular** - Fácil de entender y modificar

■ *Performance*

- **Fastify** - Framework optimizado para alto rendimiento
- **Next.js** - Server-Side Rendering y optimización automática
- **Caching** - Múltiples niveles de cache para respuestas rápidas

■ *UX (User Experience)*

- **React + Tailwind** - Interfaz moderna y responsive
- **Dark Mode** - Accesibilidad y preferencias del usuario
- **Componentes interactivos** - Grillas expandibles, gráficos dinámicos

■ *Seguridad*

- **API Proxy** - Capa adicional de seguridad
- **Validaciones estrictas** - Protección contra datos maliciosos
- **Queries parametrizadas** - Prevención de SQL injection

■ *Flexibilidad*

- **Componentes genéricos** - ExpandableGrid reusable
- **Múltiples operaciones** - 6 tipos de operaciones soportadas
- **Filtros dinámicos** - Fechas y operación seleccionables

■ Funcionalidades Implementadas

■ *Dashboard de Productividad*

Operaciones Soportadas:

1. **PICKING** - Preparación de pedidos
2. **CROSSDOCKING** - Transferencia directa

3. **EXTRACCION** - Retiro de mercancía
4. **REPOSICION** - Reabastecimiento de stock
5. **ALMACENAJE** - Gestión de inventario
6. **RECEPCION** - Ingreso de mercancía

Métricas Calculadas:
- **Unidades por hora** - Productividad individual
- **Movimientos por hora** - Eficiencia operativa
- **Horas trabajadas** - Tiempo real basado en inicio/fin
- **Productividad media** - Promedio del período
- **Tendencias diarias** - Evolución temporal

Visualizaciones:
- **KPI Cards** - Métricas clave con iconos
- **Gráficos de líneas** - Tendencias temporales
- **Grilla expandible** - Resumen con detalles anidados
- **Tabla detallada** - Datos brutos completos

■ Resumen de Tecnologías

Backend Stack
■ Node.js (LTS)
■ TypeScript
■ Fastify
■■ Microsoft SQL Server
■ mssql driver
■ Connection Pooling
■ In-Memory Cache
■ Complex SQL Queries

Frontend Stack
■■ React 18
■ Next.js 13 (App Router)
■ TypeScript

■ Tailwind CSS

■ Recharts

■ Webpack

■ Responsive Design

■ Dark Mode

■ API Proxy

DevOps & Tools

■ npm/yarn

■ ESLint + Prettier

■■ TypeScript Compiler

■ Hot Reload

■ DevTools

■ Performance Monitoring

■ Conclusión

SEGA-Cockpit es una aplicación moderna de productividad que combina:

- **Backend robusto** con TypeScript y Fastify para alto rendimiento
- **Frontend moderno** con Next.js y React para excelente UX
- **Base de datos potente** con SQL Server para análisis complejos
- **Arquitectura escalable** con microservicios independientes
- **Seguridad integral** con múltiples capas de protección
- **Experiencia de usuario superior** con componentes interactivos y responsive design

Es una solución completa y optimizada para el análisis de productividad en operaciones de almacén! ■■

Documento generado el 20 de febrero de 2026

Proyecto: SEGA-Cockpit - Dashboard de Productividad