Este documento es **la columna vertebral de todo lo que construiremos**. Lo leerán arquitectos, desarrolladores, auditores de seguridad y jefes de TI industrial.

Debe responder:

- ¿Cómo se conectan las piezas?
- ¿Qué tecnologías usamos y por qué?
- ¿Dónde corre cada componente?
- ¿Cómo fluyen los datos?
- ¿Cómo escala y se protege?

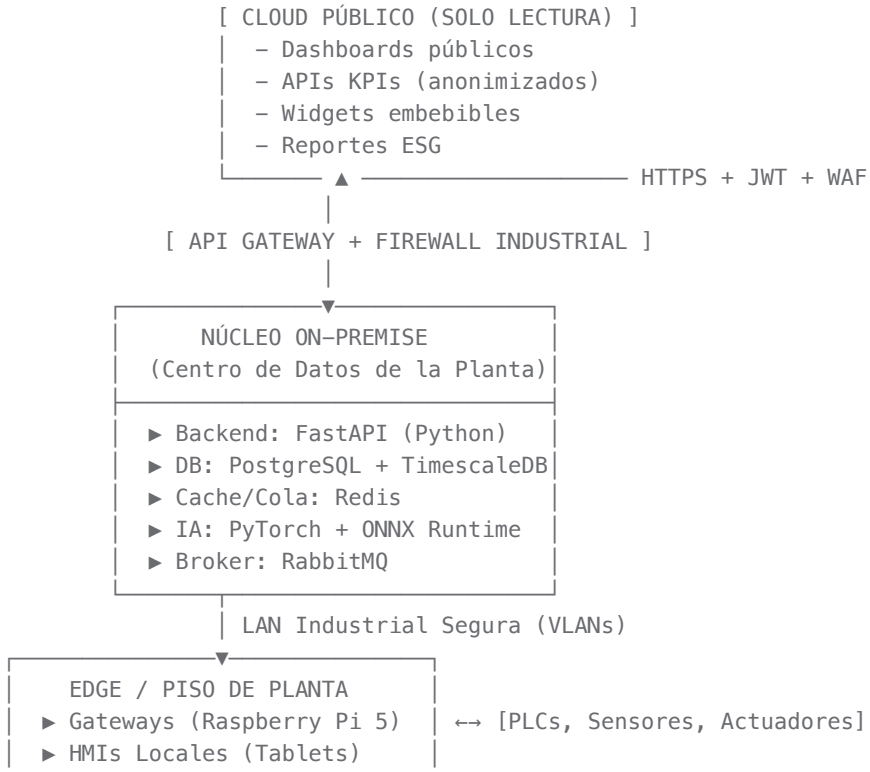
📄 `02-arquitectura-global.md` — LISTO PARA COPIAR Y PEGAR

```
# 🧩 Arquitectura Global — Orquestador Industrial 5.0

> **Una arquitectura híbrida, modular y segura: on-premise para control industrial, cloud solo para v

---

## 🖥️ Diagrama de Alto Nivel (Texto + ASCII Art)
```



```
│  ▶ App Móvil (React Native)   │ → Funciona OFFLINE
└──────────────────────────────┘
```

📦 Componentes Clave y Tecnologías

1. Núcleo On-Premise (Backend + DB + IA)

Componente	Tecnología	Función
Backend	**FastAPI (Python)**	API REST/WebSocket para control, monitoreo, IA. Asíncrono y rápido.
Base de Datos	**PostgreSQL + TimescaleDB**	Almacena todo: usuarios, órdenes, sensores (time-series), modelos IA.
Cache / Cola	**Redis**	Cache de KPIs, colas de comandos industriales (pub/sub).
IA / ML	**PyTorch + Scikit-learn + ONNX**	Modelos predictivos empaquetados, sin código fuente expuesto.
Mensajería	**RabbitMQ**	Para eventos asíncronos (alertas, triggers, logs).
Seguridad	**JWT + OAuth2 + Vault**	Autenticación, autorización granular, gestión de secretos.

> ✅ **¿Por qué no MongoDB?** → PostgreSQL + TimescaleDB cubre estructurado, time-series y JSONB. Más consistente, seguro y maduro para entornos industriales.

2. Comunicación Industrial (Edge Layer)

Protocolo	Tecnología (Python)	Uso
OPC UA	`opcua-asyncio`	Conexión segura con PLCs modernos.
Modbus TCP	`pymodbus`	PLCs y dispositivos legacy.
MQTT	`paho-mqtt`	Sensores IoT de bajo consumo.
HTTP/WebSocket	FastAPI nativo	Comunicación con HMIs web y móviles.

> 🔌 **Gateways Industriales**: Raspberry Pi 5 con Docker, actuando como pasarela segura entre LAN industrial y núcleo.

3. Interfaces de Usuario

Interfaz	Tecnología	Características
Web (Dashboard)	**React + Vite + Tailwind**	Responsive, dark mode, gráficos en tiempo real.
Móvil (Operarios)	**React Native**	Offline-first, escaneo QR, notificaciones push.
Pública (Cloud)	**React (ligero)**	Solo lectura, sin control, datos anonimizados.

4. Infraestructura y DevOps

Capa	Tecnología	Observaciones
Contenedores	**Docker**	Aislamiento, portabilidad, fácil deploy.
Orquestación	**Docker Compose (MVP)**	Luego Kubernetes si escala a múltiples plantas.
CI/CD	**GitLab CI/CD**	Automatización de tests y despliegues.
Infra como Código	**Terraform + Ansible**	Provisionamiento de servidores y configuración.
Monitoreo	**Prometheus + Grafana**	Métricas del sistema, no de la planta (por ahora).

🔐 Flujo de Seguridad (Zero Trust Industrial)

1. **Autenticación**:
   - MFA obligatorio para acciones críticas (Google Authenticator o llave física).
   - JWT con expiración corta (15 min) + refresh token.
2. **Autorización**:
   - RBAC por rol, máquina, y acción (ej: “técnico puede ver OTs, no detener líneas”).
   - Políticas definidas en base de datos.
3. **Encriptación**:
   - En tránsito: TLS 1.3 entre todas las capas.
   - En reposo: AES-256 (discos LUKS, backups cifrados).
4. **Auditoría**:
   - Todo comando, login, y cambio de estado se loggea en PostgreSQL.
   - Logs inmutables (WAL archiving + backup fuera de sitio).
5. **Aislamiento**:
   - VLAN separada para OT (Operational Technology).
   - Firewall industrial (OPC UA con certificados X.509).

🔄 Flujo de Datos (Ejemplo: Lectura de Sensor → Dashboard)

1. Sensor → envía dato vía Modbus TCP → Gateway (Raspberry Pi).

2. Gateway → traduce a JSON → envía por HTTP/WebSocket → Backend (FastAPI).
3. Backend → valida token → guarda en `sensor_data` (TimescaleDB) → publica en Redis.
4. Redis → notifica a WebSocket → Dashboard Web actualiza en tiempo real.
5. IA → lee datos cada 5 min → predice fallo → guarda alerta en DB → notifica por push a app móvil.

📈 Escalabilidad y Alta Disponibilidad

- **Vertical**: Servidores on-premise con redundancia (HAProxy + PostgreSQL streaming replication).
- **Horizontal**: Si se expande a múltiples plantas, cada una con su núcleo local + sincronización cloud.
- **Edge Computing**: Procesamiento de IA ligera en gateway (ONNX Runtime) para baja latencia.
- **Cloud Bursting**: Solo para entrenamiento de IA pesado (opcional, con datos anonimizados).

🧪 Entorno de Desarrollo vs Producción

Característica	Desarrollo (Local)	Producción (On-Premise)
DB	PostgreSQL + TimescaleDB en Docker	PostgreSQL cluster + réplica síncrona
Backend	FastAPI con reload automático	FastAPI en contenedores, detrás de Nginx
IA	Modelos en .pkl o .onnx locales	Modelos ONNX cifrados en Vault
Hardware	Laptop + emulador de PLC	Servidores industriales + gateways reales
Seguridad	JWT simple, sin MFA	MFA, RBAC, encriptación total

🧭 ¿Cómo empezar a desarrollar?

```
# 1. Clona el repo de documentación
git clone https://github.com/tu-empresa/orquestador-industrial-5.0-docs.git

# 2. Instala PostgreSQL + TimescaleDB (ver 12-deploy-config.md)
# 3. Ejecuta el script inicial de DB (en /sql-scripts/init.sql)

# 4. Levanta backend local
cd backend
pip install -r requirements.txt
uvicorn main:app --reload

# 5. Abre http://localhost:8000/docs → ¡Swagger listo!

# 6. Conecta un PLC simulado (ver /simulators/plc-simulator.py)
```

📌 Decisiones Clave (Ver también 11-decisiones-tecnicas.md)

- **PostgreSQL + TimescaleDB > MongoDB**: Consistencia, time-series, madurez industrial.
- **FastAPI > Django/Flask**: Asíncrono, ideal para IIoT, autodocumentación (Swagger).
- **React Native > Flutter**: Reutilización de lógica con React Web, ecosistema maduro.
- **On-Premise Core > Cloud**: Datos industriales nunca salen de la planta. Cloud solo para público.
- **IA embebida > IA en cloud**: Sin fugas de secretos industriales, sin latencia.

> “La arquitectura no es un dibujo bonito. Es la promesa de que el sistema no se caerá, no se colgará, y no traicionará la confianza de quien lo usa.”