# Verificación de Módulos vs. Documentos Oficiales – Proyecto Ofitec.ai

En este informe se comparan los módulos desarrollados (QHSE, Analítica Avanzada, Portal/UI Roles, Infra/CI-CD & Backups, etc.) con las especificaciones **funcionales, visuales, estructurales y de base de datos** definidas en los **documentos oficiales** de Ofitec.ai. Se identifica para cada módulo si cumple completamente con los lineamientos oficiales, en qué puntos requiere ajustes, y se brindan **sugerencias específicas de mejora** con ejemplos cuando corresponde.

## Módulo QHSE (Calidad, Seguridad, Salud y Medio Ambiente)

**Cumplimiento Funcional:** El módulo QHSE aborda la gestión de seguridad y calidad en obra. Según la **Estrategia de Creación de Páginas** oficial, la sección *Operaciones de Obra* requiere funcionalidades de **HSE inteligente**, incluyendo detección automática de EPP (equipos de protección personal) mediante IA, listas de chequeo digital para inspecciones, registro inmediato de incidentes con geolocalización y análisis de tendencias en seguridad. Si el módulo QHSE desarrollado contempla estas características (por ejemplo, integración de visión artificial para detectar uso de cascos/chalecos, formularios de inspección y un sistema de registro de incidentes en terreno), entonces cumple con lo funcional. **De lo contrario, sería necesario ampliarlo** para incluir: detección visual de cumplimiento de EPP, checklists obligatorios de seguridad, y registro de incidentes en tiempo real con ubicación GPS. Además, la estrategia exige analíticas de seguridad (*“Analytics: Tendencias seguridad, causas raíz”*), por lo que el módulo debería generar reportes de tendencia de accidentes e identificar causas raíz automáticamente. Si el módulo QHSE no está explotando datos históricos de incidentes para extraer patrones, se recomienda incorporar un **submódulo de analítica HSE** alineado con la sección *6.2 HSE Analytics* (tendencias estadísticas de incidentes, detección de patrones y predicción de probabilidad de futuros incidentes).

**Cumplimiento Visual:** En cuanto a la interfaz, el módulo QHSE debe respetar la **Estrategia Visual** corporativa. Esto implica usar la paleta de colores definida (fondos oscuros, acento verde lima #84CC16, etc.) y componentes comunes como tarjetas, tablas y badges. Por ejemplo, una página de incidentes HSE debería mostrar indicadores de estado (“Abierto”, “En investigación”, “Cerrado”) con **badges de color semántico** (verde para cumplido, ámbar para pendiente, rojo para crítico) conforme a la definición oficial. Si el módulo QHSE ha seguido estos lineamientos (uso de Card con bordes de 1px y radio de 12px para paneles, ProgressBar para porcentajes de cumplimiento de inspecciones, etc.), entonces es visualmente conforme. Si no, **se sugiere ajustar el estilo**: por ejemplo, asegurar que los formularios de incidentes usen los Input estandarizados (sin sombras, con estilo consistente) y que los dashboards de seguridad utilicen componentes KPICard para métricas de accidentabilidad, siguiendo la identidad visual unificada.

**Cumplimiento Estructural:** Estructuralmente, el módulo QHSE debe integrarse con la arquitectura modular de Ofitec.ai sin duplicar funcionalidad existente. Los documentos oficiales enumeran los módulos principales y sus relaciones; QHSE no se menciona explícitamente allí, pero es probable que extienda a **site\_management** (reportes diarios) y a **project\_risk** (riesgos) para aportar la perspectiva de seguridad. Si el módulo QHSE está aislado, **se recomienda** enlazarlo con estos módulos: por ejemplo, que las *incidencias HSE registradas en terreno* alimenten automáticamente el módulo de **Riesgos del proyecto**, creando riesgos de tipo “Seguridad” o ajustando la *Matriz IA de Riesgos* (como indica la estrategia, los riesgos deben categorizarse incluyendo HSE). Adicionalmente, en la arquitectura de directorios organizada, los módulos personalizados de Odoo residen en custom\_addons/. QHSE debe ubicarse allí y respetar la separación backend/frontend definida por la **Ley de Puertos** (no debe exponer una interfaz fuera del frontend Next.js oficial en puerto 3001). Si el módulo QHSE pretendiera tener una mini-web o dashboard aparte, eso violaría la Regla #1 y #2 de puertos – *la UI debe canalizarse por el portal principal (3001) y los datos vía API en 5555*. Hasta ahora no hay indicios de infracción en QHSE, pero es un punto a vigilar.

**Cumplimiento Base de Datos:** La documentación *Mapeo Base de Datos – Páginas* especifica las tablas necesarias para funcionalidades HSE. En *Operaciones de Obra*, sección **4.2 HSE Inteligente**, se requieren tablas como incidentes\_hse, inspecciones\_seguridad y epp\_detecciones para almacenar incidentes, resultados de inspecciones y detecciones de EPP por IA. Es crucial verificar que el módulo QHSE efectivamente **cree o utilice estas tablas** en la base de datos. Si el módulo desarrollado no contempló la creación de dichos modelos de datos (por ejemplo, si solo manejó incidencias de forma superficial), estaría incompleto respecto a la especificación. **Sugería**: implementar el modelo de datos conforme a la “Ley de Base de Datos” y el mapeo oficial. Por ejemplo, ofitec.incident con campos obligatorios (fecha, tipo, gravedad, coordenadas, fotos adjuntas), ofitec.safety\_check para checklist de inspecciones (con quién, resultado, acciones correctivas) y una estructura para almacenar resultados de detección de EPP (quizá integrando con un servicio IA externo). Además, cualquier tabla nueva debe seguir los **estándares NASA de integridad** indicados en la Ley de Base de Datos, incluyendo validaciones (p.ej., si se almacena RUT del responsable del incidente, validar el dígito verificador como exige el Art. II).

**Conclusión QHSE:** *Parcialmente Cumplido.* El módulo QHSE cubre la necesidad general de gestionar Seguridad y Salud en obra, pero **requiere ajustes para alinearse totalmente** con los lineamientos oficiales. En particular, debe incorporar las funciones de **IA para detección de EPP y analítica de incidentes**, asegurarse de **alimentar la matriz de riesgos corporativa**, y crear las **tablas HSE definidas oficialmente**. Visualmente, solo pequeños refinamientos serían necesarios si aún no se aplicó el design system (ej: uso de colores y componentes estándar). Estas mejoras asegurarán que QHSE esté 100% en línea con las especificaciones funcionales y de datos de Ofitec.ai.

## Módulo Analítica Avanzada

**Cumplimiento Funcional:** El módulo de *Analítica Avanzada* (probablemente relacionado con IA y BI del sistema) debe cubrir las capacidades de predicción y análisis automátizados delineadas en los documentos oficiales. La **Estrategia de Creación de Páginas** especifica un módulo transversal de **IA & Analytics** con varias funciones clave: *“Predicciones Avanzadas”* en costos, plazos, riesgos y performance, *“Copilots por Módulo”* (asistentes de IA conversacionales para Finanzas, Proyectos, HSE, Documentos), y *“Automatización de Decisiones”* (alertas inteligentes, aprobaciones automáticas basadas en criterios, recomendaciones). Si el módulo Analítica Avanzada desarrollado incluye, por ejemplo, **modelos de ML** que proyectan sobrecostos o retrasos de cronograma, dashboards de pronóstico y un componente de *chatbot* ejecutivo (Copilot) capaz de responder preguntas con datos reales, entonces se alinea bien con lo requerido. Si no, **debería ampliarse**. Por ejemplo, se esperaría que este módulo integre el motor de IA (quizás basado en LangChain/Transformers como se menciona en la arquitectura) para ofrecer: simulaciones de escenarios financieros (optimista/base/pesimista), cálculo automático de probabilidades de riesgo (e.g. usando datos de desviaciones de presupuesto en la *Matriz IA de Riesgos*), y generación de alertas proactivas. Además, la analítica avanzada debe aprovechar que **el sistema ya cuenta con 34.428 registros reales** en la base (órdenes, proyectos, proveedores); un cumplimiento ejemplar sería demostrar que el módulo usa esa riqueza de datos para entrenar modelos predictivos. Si actualmente el módulo solo hace reporting tradicional, es **necesario incorporar técnicas de ML/IA** conforme a la visión del proyecto.

**Cumplimiento Visual:** Muchas funcionalidades de analítica avanzada se plasman en dashboards y herramientas interactivas, cuya apariencia debe seguir la Estrategia Visual. Esto implica gráficos y tarjetas con estilo consistente. Por ejemplo, el *Dashboard de Predicciones* debería usar componentes KPICard para mostrar métricas previstas (p.ej., *“Costo proyectado próximo trimestre”*, *“Probabilidad de retraso en Proyecto X”*) de manera destacada. Si el módulo incluye visualizaciones (gráficas de líneas, barras, etc.), deben respetar la paleta (fondos oscuros en modo principal, líneas acento verde para resaltar tendencias positivas, rojo para alertas) y tipografías definidas. Un detalle a verificar es el **modo Dark/Light**: la estrategia visual prevé ambos modos. Asegurarse de que los dashboards de analítica funcionen en modo oscuro (por defecto) con los colores correctos y que puedan alternar a modo claro manteniendo legibilidad. Si no se consideró, **se sugiere implementar tokens de color** para gráficos adaptables a ambos temas. En resumen, en lo visual el módulo debería estar **bien integrado con el sistema de diseño**, usando los componentes de tablas para listados de pronósticos, badges para señalar estados de predicción (e.g., proyección confiable o incierta), y siguiendo la usabilidad consistente del portal.

**Cumplimiento Estructural:** El módulo de Analítica Avanzada es transversal; estructuralmente debe funcionar como **servicio desacoplado pero integrado**. Los documentos oficiales enfatizan que los consumidores de datos **deben usar vistas canónicas** y no acceder directamente a tablas crudas. Por ejemplo, si Analítica calcula previsiones de costos, debería leer de la vista vista\_proyectos\_resumen o de purchase\_orders\_unified a través de v\_facturas\_compra, en lugar de consultas ad-hoc fuera del esquema unificado. **Revisar** si el módulo respeta esto: cualquier extracción de datos financieros debe pasar por las vistas v\_\* obligatorias definidas en la Ley de Base de Datos, garantizando una “única fuente de verdad”. Si no es el caso, es un punto de mejora crítico (p. ej., usar v\_facturas\_compra para obtener montos por proveedor en vez de tablas temporales). Otro aspecto estructural es la **integración con otros módulos**: la analítica avanzada debería tomar insumos de *project\_financials, project\_risk,* etc., y alimentar resultados de vuelta, posiblemente vía nuevas tablas de predicciones. La hoja de ruta de datos sugiere crear tabla predicciones\_ml para almacenar pronósticos de IA asociados a la Matriz de Riesgos. **Verificar** si el módulo ya está creando una estructura así; de no ser así, sería recomendable añadir un modelo ofitec.prediction donde se guarden, por ejemplo, el ID del proyecto, el tipo de predicción (costo, plazo, riesgo) y el resultado/score calculado por IA. Esto permitiría que otros módulos (ej: Riesgos) consuman esas predicciones, logrando la transversalidad pretendida. Finalmente, a nivel arquitectura, confirmar que este módulo no rompe la organización: debe residir donde corresponde (posiblemente parte de ai\_bridge o un submódulo nuevo), con su lógica IA en backend (Odoo/Flask) y solo exposición de resultados vía el frontend oficial (no servicios aislados).

**Cumplimiento Base de Datos:** Como se mencionó, es fundamental que Analítica Avanzada utilice el esquema de datos oficial. La **Ley Oficial de Base de Datos** impone estándares rigurosos: integridad referencial, no duplicación y monitoreo. Si el módulo realiza consultas complejas, debe adherirse a las consultas autorizadas. Por ejemplo, para analíticas financieras, la Ley incluye una query estándar para *Proveedores Top* y *Proyectos Reales*. Asegurarse de no re-inventar métricas que ya están definidas oficialmente, sino reutilizar la lógica (por ejemplo, contar órdenes y sumar montos por proyecto tal como en la consulta aprobada). En cuanto a nuevos datos generados: cualquier nueva tabla, como la sugerida de predicciones, debe seguir las normas de calidad NASA: campos con tipos correctos, timestamps, claves foráneas si aplica, índices para evitar duplicados, etc. Si el módulo Analítica genera datasets intermedios (p. ej., una tabla con series temporales para pronóstico de cashflow), **debe implementar backups y validaciones** equivalentes a las de las tablas críticas. Dado que la analítica se basa en datos críticos, conviene aplicar el *Sistema anti-duplicados multicapa* si se importan datos externos para IA, y monitorear la calidad de datos de entrada diariamente (Art. VI: monitoreo de métricas críticas diarias, aunque en la ley se refiera más a DB, el espíritu aplica) – por ejemplo, validar que los registros utilizados para predicción se mantengan actualizados y sin anomalías. Por último, si el módulo utiliza la base SQLite integrada (ofitec\_dev.db), asegurar que cualquier carga intensiva (p. ej., entrenar un modelo) no comprometa la integridad; podría requerir usar la estrategia de backup 3-2-1 antes de operaciones pesadas.

**Conclusión Analítica Avanzada:** *Parcialmente Cumplido.* El módulo de Analítica Avanzada probablemente cubre algunas áreas (quizá dashboards básicos o informes), **pero para cumplir plenamente** debe incorporar todas las aristas de IA planteadas oficialmente: copilot conversacional multi-módulo, predicciones ML para costos/plazos/riesgos, y automatización de respuestas (ej., notificar cuando una métrica excede umbral). Se recomienda **apoyarse fuertemente en las vistas canónicas y datos ya disponibles** (más de 34k registros reales) para dar potencia al módulo. Implementar un almacenamiento formal de resultados de IA (p. ej., tabla de predicciones) y exponerlos al usuario en el dashboard ejecutivo. Con estos ajustes, el módulo no solo cumpliría con las especificaciones, sino que destacaría como un factor diferenciador (tal como lo resalta el documento de mapeo: la existencia de un sistema de conciliación e IA listo desde el día 1 es ventaja competitiva, y lo mismo aplica a predicciones avanzadas).

## Módulo Portal / UI Roles

**Cumplimiento Funcional:** El módulo identificado como *Portal UI Roles* parece referirse a dos aspectos: (1) la implementación del **Portal Cliente** (un portal web con vistas específicas para clientes externos) y (2) la gestión de **Roles de usuario y accesos** en la interfaz. En los documentos oficiales, ambos puntos están claramente definidos. Por un lado, la **sección 7. Portal Cliente** de la estrategia define páginas como *Vista Proyecto para el Cliente*, *Reportes Ejecutivos* y *Interacción Digital*, enfocadas en transparencia y comunicación. Por otro lado, en la sección 9.1 de Configuración/Admin se describe la **Gestión de Usuarios Empresarial** con funcionalidades avanzadas de seguridad: SSO con Google, invitaciones controladas, gestión de roles (Administrador, PM, Supervisor, etc. con permisos por proyecto), auditoría de actividades y controles como MFA, restricción por IP, etc..

Para evaluar el módulo Portal/UI Roles: - Si su alcance incluye el **Portal Cliente**, debe verificarse que ofrezca a los usuarios externos las vistas indicadas: por ejemplo, que un cliente pueda ingresar y ver el avance de su proyecto con fotos y hitos (avance visual), consultar documentos entregables de ese proyecto (con acceso restringido), ver estados de pago/facturación relacionados, e incluso interactuar vía mensajería con el equipo de proyecto. **Si el módulo desarrollado omite alguna de estas páginas o funcionalidades clave, requiere expansión.** Un indicio: en la estructura de datos se menciona que para Portal Cliente realmente no se requieren datos nuevos sino filtrar los existentes (proyectos, órdenes) por cliente específico, lo cual simplifica la implementación – si el módulo no lo hace aún, se sugiere implementar un filtro por Customer\_Name en las consultas de proyectos/órdenes. - Respecto a **UI Roles** (gestión de roles), el módulo debería proveer interfaces para que administradores asignen roles a usuarios, definan permisos por proyecto, etc. Ofitec.ai ya tiene un módulo de seguridad (ofitec\_security), por lo que este componente de UI debe integrarse con él. Es decir, aprovechar los modelos de roles/permisos existentes en backend. Si el módulo UI Roles simplemente enumeró los roles (Administrador, Gerente, PM, Supervisor, etc.) pero **no implementó las características avanzadas** (SSO, invitaciones, auditoría), entonces está incompleto frente a la especificación oficial. **Se recomienda** incorporar dichas características gradualmente: por ejemplo, integrar autenticación Google OAuth2 tal cual sugiere la estrategia, habilitar la invitación de nuevos usuarios mediante tokens de invitación enviados por email (y solo accesible a administradores), y generar registros de auditoría cada vez que se cambian roles o permisos (posiblemente usando un modelo de log de seguridad). Asimismo, en términos de datos, la hoja de mapeo indica que *no existen aún* tablas para usuarios/roles en la base (se marcan como “a crear completamente”). Si el módulo Portal/UI Roles no las ha creado, deberá definir modelos para usuarios\_sistema, roles\_permisos y sesiones\_activas conforme a esa guía.

**Cumplimiento Visual:** Por ser un módulo centrado en la interfaz, es crucial que Portal/UI Roles cumpla plenamente la Estrategia Visual. Para el **Portal Cliente**, la UI debe ser coherente con el resto del frontend (Next.js 14 + Tailwind, según se indicó). Esto implica usar la **paleta corporativa** y componentes existentes. Un dashboard para clientes podría reutilizar la estructura de Dashboard ejecutivo pero mostrando solo KPIs relevantes para el cliente (por ejemplo, % de avance físico, facturación del proyecto, próximos hitos). Debe asegurarse que los **componentes visuales** sean los estándar: p. ej., usar gráficas integradas en cards, listados de documentos con la tabla estilizada (cabeceras grises claras en modo light, o gris oscuro en dark), y botones de acción (descargar documento, enviar mensaje) con los estilos primarios/secundarios definidos. Para la parte de **Administración de Roles**, probablemente haya páginas de configuración donde un administrador ve la lista de usuarios y asigna roles/proyectos. Esas pantallas deben seguir la línea de componentes de formularios (inputs, selects, toggles) consistentes. Un detalle importante es la **identidad visual corporativa**: la estrategia visual habla de *“UI/UX profesional implementada”* en el frontend. Esto se traduce en tener un portal cliente pulido, con branding adecuado y sin inconsistencias. Si el módulo UI Roles fue generado adhiriendo a un diseño genérico, conviene revisar contra el manual visual; por ejemplo, asegurarse de usar la fuente “Inter” en todo el texto, usar íconos y colores para estados (activo/inactivo, autorizado/no autorizado) tal como los tokens de estado definen.

**Cumplimiento Estructural:** Estructuralmente, este módulo toca dos aspectos: **navegación/estructura de páginas** y **arquitectura de permisos**. En la navegación, todas las páginas nuevas (cliente o admin) deben insertarse correctamente en el portal Next.js principal (menú, rutas, etc.) siguiendo la **metodología de creación de páginas**. La Ley de Puertos enfatiza “un solo frontend” para todas las páginas de usuario, por lo que el Portal Cliente no debe ser una aplicación separada: debe estar alojado en el mismo Next.js en nuevas rutas (/cliente o similares). Si el módulo intentaba un portal aparte, sería una desviación – debe corregirse unificándolo. En cuanto a permisos, es vital que las restricciones de acceso estén implementadas: los documentos oficiales indican que cada rol solo debe ver lo que le corresponde (ej.: un cliente solo ve su proyecto, un PM solo sus proyectos asignados). Sería un **incumplimiento grave** si, por ejemplo, un usuario sin permiso pudiera acceder vía URL directa a datos de otro proyecto. Se espera que el módulo UI Roles use los **permisos dinámicos por proyecto** definidos en ofitec\_security – posiblemente aprovechando un *middleware* o *hook* en el frontend que consulte a la API qué proyectos tiene autorizados el usuario logueado. Si esto no está implementado, es una prioridad corregirlo para cumplir la separación de roles. Finalmente, estructural es también mantener la única fuente de verdad documental: cualquier referencia en la interfaz a qué páginas existen o qué roles están disponibles debería provenir de los **documentos oficiales centralizados** (por ejemplo, si la interfaz lista las páginas disponibles para un rol, basarse en lo definido en *ESTRATEGIA\_CREACION\_PAGINAS.md* para ese rol). El *Estado del Sistema Organizado* recalca que **todos los aspectos del desarrollo deben consultar los docs oficiales como única fuente de verdad**. Esto se traduce en que el desarrollo de este módulo haya seguido esas guías en lugar de requerimientos ad-hoc.

**Cumplimiento Base de Datos:** Actualmente, Odoo maneja usuarios y accesos internamente, pero dado que la intención es independizar la plataforma, los documentos oficiales contemplan nuevas tablas canónicas para usuarios y roles. Si el módulo Portal/UI Roles aún se apoya solo en el modelo Odoo default, eventualmente deberá migrar a las tablas unificadas. Concretamente, se deberían crear: - Tabla usuarios\_sistema con campos como ID, email, nombre, hash de contraseña o ID federado (si SSO), rol principal, estado activo, etc. - Tabla roles\_permisos para definir qué rol tiene qué permisos (posiblemente almacenando pares rol-recurso/proyecto). - Tabla sesiones\_activas para controlar tokens o sesiones abiertas, útil para auditoría y seguridad adicional.

Estas tablas aún no existen (“A CREAR COMPLETAMENTE” indica el mapeo), así que su ausencia en el módulo no es un fallo si todavía se usa la autenticación de Odoo, pero es algo que **debe planificarse**. Además, *Ley de Base de Datos* no menciona explícitamente estos modelos, pero las prácticas de integridad aplican: por ejemplo, si se crea usuarios\_sistema, debería haber una restricción para que un mismo email no se duplique (UNIQUE), y un registro de usuario no se elimine sin antes re-asignar sus proyectos (integridad referencial con, digamos, tabla de proyectos asignados). También, **backup de seguridad**: las credenciales y roles son datos sensibles; habría que asegurar backups frecuentes y encriptación de ciertos campos (el documento de estrategia 9.1 menciona certificados SII cifrados, similarmente las contraseñas deben almacenarse con hash seguro, y posiblemente tokens 2FA también cifrados). Otro aspecto es la consistencia con **ofitec\_security**: si Ofitec.ai seguirá usando Odoo para autenticación base, conviene sincronizar la base de datos Odoo (res.users, etc.) con estos nuevos esquemas, o por lo menos documentar claramente la transición.

**Conclusión Portal/UI Roles:** *Requiere Ajustes Notables.* Aunque la definición de roles y páginas está planteada, el módulo debe **reforzarse para cubrir a cabalidad lo indicado oficialmente**. En Portal Cliente, incluir todas las vistas clave (avance con fotos, estados de pago, comunicación) y filtrarlas por cliente. En gestión de usuarios, implementar las mejoras de seguridad (SSO, control de invitaciones, auditoría) para alinearse con estándares empresariales. Asimismo, crear las bases de datos de usuarios/roles para independizar la gestión de Odoo. Visualmente, mantener la coherencia total con el diseño corporativo. Con estas mejoras, el portal ofrecerá una experiencia completa y segura, cumpliendo la promesa de transparencia a clientes y control granular a administradores.

## Módulo Infraestructura, CI/CD & Backups

**Cumplimiento Funcional:** Este módulo (o documentación asociada) cubre los aspectos de despliegue, integración continua, entrega continua y respaldos del sistema. Funcionalmente, debe asegurar que el entorno sigue las convenciones oficiales: único backend en Flask/Odoo y único frontend Next.js, integrados en una pipeline CI/CD robusta. El **Diagnóstico de Problemas** oficial identificó previamente inconvenientes severos en la infraestructura, como uso de un puerto de frontend incorrecto (3002 en vez de 3001) y múltiples servidores corriendo simultáneamente. Gracias a esas observaciones, sabemos que cumplir la **Ley de Puertos** y consolidar servidores es mandatorio. Por lo tanto, el módulo Infra/CI-CD debe reflejar que: - El **frontend** se ejecuta en puerto **3001** en todos los entornos oficiales (dev/prod) y no se despliegan frontends paralelos en 3000 o 3002. El informe de diagnóstico mostró que se corrigió el package.json para forzar 3001. Asegurarse de que en los scripts de CI/CD se use siempre ese puerto (por ejemplo, si hay un npm run start, que use -p 3001). Cualquier instrucción en contrario requeriría ajuste. - El **backend** se concentra en puerto **5555** y solo uno (usando server\_organizado.py exclusivamente). En la configuración Docker o de entornos, debe levantarse solo un contenedor/servicio para la API, evitando duplicados. Si el módulo Infra describe un docker-compose, confirmar que mapea 5555 para Flask y 3001 para Next (y no otros puertos). - No haya **servicios huérfanos**: la Ley de Puertos reserva ciertos rangos (3000/3002 no usar, 8000-8999 para Odoo, etc.). El pipeline CI/CD debe validar eso, quizás corriendo pruebas que confirmen puertos activos autorizados. Si no está contemplado, sería buena práctica añadir un paso de verificación (como sugiere la ley, usar netstat o curl a /health en los puertos esperados).

En términos de **CI/CD**, la documentación técnica ya provee ejemplos de workflows de GitHub Actions para ejecutar pruebas automáticamente en cada push. El módulo Infra debe incorporar algo similar o referenciarlo. De no haberse hecho, **se recomienda configurar un pipeline CI** con al menos: - **Jobs de testing:** correr pytest con cobertura en cada commit (como el ejemplo YAML dado). - **Jobs de build/deploy:** por ejemplo, construir la imagen Docker y desplegar en un ambiente staging, asegurando persistencia de la base de datos entre despliegues. - **Integración continua de IA**: Si hay componentes de IA (entrenamiento de modelos, etc.), quizás no en cada commit pero planificar su actualización periódica.

**Cumplimiento Visual:** Este módulo es más de backend/operativo, por lo que “visual” aplica menos. Sin embargo, parte de infraestructura es la **monitorización**, que sí tiene una visualización (Grafana, etc.). Los documentos oficiales (Estrategia Técnica) sugieren paneles de Grafana recomendados y un diagrama de monitoreo con Prometheus/ELK. Si el módulo Infra incluye la configuración de monitoreo, debería **cumplir con esa arquitectura**: desplegar un *Prometheus Exporter* para Odoo, un stack ELK para logs, y Grafana con paneles para rendimiento, estado de proyectos, integraciones y calidad de código. Visualmente, esos paneles deben seguir las convenciones (aunque Grafana tiene su propio estilo, la consistencia vendría en usar los métricas definidas: e.g., mostrar “Jobs en cola (queue\_job)” como se indicó). Esto es un plus; si no está implementado aún, podría ser una mejora futura. No hay mucho más en lo visual, salvo quizás documentar bien en la UI (por ejemplo, una página de “Estado del Sistema” que consuma /api/status ya existe, y muestra que todo esté verde).

**Cumplimiento Estructural:** Aquí entra fuertemente la **Ley de Puertos** y la organización de carpetas. La **estructura organizada** final se delineó en *ESTADO\_SISTEMA\_ORGANIZADO.md*: todos los documentos oficiales en docs\_oficiales/, scripts en scripts\_utilidades/, etc.. El módulo Infra debe respetar esa estructura. Por ejemplo, asegurarse de que los **scripts de backup** y CI estén ubicados lógicamente (quizá en scripts\_utilidades/ o en un directorio ci/). Además, la Ley de Base de Datos Artículo VII exige una política de backups muy clara (regla 3-2-1 y frecuencias). **Verificar** si en la configuración actual: - Se realiza **backup automático diario** de la base de datos (punto de la 02:00 AM). Si no, implementar un job (puede ser vía GitHub Actions, orquestrador, o incluso un cron en el servidor) que dumpee la base cada madrugada y lo suba a almacenamiento externo. - Se guardan múltiples copias en diferentes medios: la regla 3-2-1 implica por lo menos una copia local, una en la nube y una offsite. Si el módulo no menciona almacenamiento offsite (por ejemplo, AWS S3, Google Drive), se sugiere integrarlo. *Dato:* Ya existe integración con Google Drive para documentos; potencialmente el backup podría aprovechar esa conexión para subir respaldos, aunque lo óptimo serían servicios especializados. - Se prueba la recuperación: un backup no vale si no hay plan de restore. ¿El módulo Infra detalla procedimientos de recuperación? Debería, siguiendo Art. VIII de la Ley BD (protocolo de emergencia: stop inmediato, restaurar último backup verificado, validación completa). Recomendar definir un **playbook de recuperación** y quizás un script automatizado para restaurar la DB desde un backup file, con verificaciones post-restore (conteo de registros, integridad referencial, etc.).

Otro aspecto estructural es la eliminación de componentes legacy. El diagnóstico mostró que había archivos duplicados (server.py vs server\_organizado.py) y múltiples procesos conflictivos. Tras la reorganización, el pipeline CI debe garantizar que **solo el código organizado se use**. Por ejemplo, asegurarse que ningún desarrollador vuelva a ejecutar server.py antiguo, quizá marcándolo como obsoleto en la documentación. También, la **checklist de cumplimiento** en Ley de Puertos exige que **no haya servidores en puertos no autorizados** – se puede incorporar una verificación en CI que intente arrancar el sistema y compruebe que solo están escuchando 3001 y 5555. Dado que en el diagnóstico se hizo *taskkill* de procesos manualmente, sería útil automatizar la detección de puertos ocupados en el entorno CI para evitar colisiones en los runners.

**Cumplimiento Base de Datos:** La infraestructura debe soportar la base de datos NASA-level del sistema. Esto significa no solo backups (ya cubiertos) sino también herramientas de mantenimiento. En la estructura final, se enumeran scripts como nasa\_database\_restructurer.py y verificar\_estado\_db.py en utilidades. **Comprobar** si el módulo Infra/CI ha incluido la ejecución de esos scripts en los momentos adecuados. Por ejemplo, ejecutar verificar\_estado\_db.py regularmente (podría ser parte de CI nightly) para garantizar que la integridad 6/6 constraints siga ok. La estrategia de CI podría ser: tras despliegue, correr un comando que invoque ese verificador y si falla (encuentra inconsistencia), alertar o revertir. Asimismo, mantener actualizadas las **vistas financieras canónicas** definidas en la Ley BD Art. IX. Existe un script tools/create\_finance\_views.py mencionado; sería ideal que en cada migración/actualización de BD, la pipeline corra dicho script para (re)crear las vistas idempotentemente. Si el módulo Infra no lo contempló, se sugiere añadirlo para cumplimiento pleno, ya que así se garantiza que las vistas v\_facturas\_compra etc. estén siempre presentes y correctas tras cambios en tablas base.

Finalmente, considerar la **seguridad de datos** en CI/CD: credenciales, tokens, etc., no deben estar expuestos. Usar los mecanismos de secreto de GitHub Actions para claves (por ejemplo, la contraseña de la BD, tokens de API). Y relacionándolo con la base de datos, si hay migraciones de esquema (Odoo modules updates), integrarlas en la pipeline para no romper la compatibilidad de datos.

**Conclusión Infra/CI-CD & Backups:** *Mayormente Cumplido, con Mejoras Recomendadas.* Tras la reorganización, la infraestructura cumple con los lineamientos críticos: el backend corre en 5555 y el frontend en 3001 (corrigiendo el fallo previo), y se eliminó la duplicidad de servidores. Las tareas CI/CD básicas (tests automatizados) parecen estar encaminadas con GitHub Actions. No obstante, para alcanzar un cumplimiento completo de las especificaciones oficiales, sugerimos: - **Formalizar la estrategia de backups 3-2-1**: implementar backups horarios/diarios y almacenamiento externo, con verificación de integridad post-backup. - **Automatizar controles de puertos y procesos**: integrar en CI algún script que valide que solo estén activos los puertos oficiales (3001, 5555) durante los despliegues, previniendo regresiones. - **Incluir scripts de mantenimiento de BD** en el pipeline: verificación de integridad y recreación de vistas canónicas en cada despliegue para asegurar la base de datos se mantiene “Nivel NASA”. - **Expandir la monitorización**: desplegar/promover la pila de monitoreo (Prometheus + Grafana + ELK) para vigilancia continua, como valor agregado importante para DevOps.

Con estos refuerzos, la infraestructura de Ofitec.ai no solo estaría cumpliendo al 100% con las *“leyes” oficiales del sistema*, sino que ganaría robustez y confiabilidad dignas de un entorno empresarial crítico.

## Resumen de Cumplimiento por Módulo

* **Módulos que cumplen casi completamente:** *Infraestructura/CI-CD* (tras correcciones recientes cumple las reglas de puertos y unificación de servidor, solo restan optimizaciones en backups y monitoreo), *en parte Portal/UI Roles* (estructura de roles definida y portal integrado en frontend único, pero falta profundidad en seguridad avanzada). Estos módulos se adhieren en gran medida a las especificaciones oficiales, con alineación clara a las leyes de puertos y estándares visuales, requiriendo solo mejoras incrementales para alcanzar el 100% de conformidad.
* **Módulos que requieren ajustes importantes:** *QHSE* y *Analítica Avanzada* necesitan ampliarse para cubrir todas las funcionalidades pretendidas en los documentos oficiales. QHSE debe incorporar IA en seguridad (detección EPP) y conectar sus datos con riesgos, además de implementar las tablas nuevas para incidentes y checklists. Analítica Avanzada debe integrar las capacidades de predicción y copilot en todos los dominios (finanzas, proyectos, etc.), utilizar plenamente los datos existentes y formalizar el almacenamiento de resultados de ML.
* **Sugerencias específicas:** Para **QHSE**, añadir módulo de visión computacional para reconocer elementos de seguridad en fotos (ej. usar un modelo pre-entrenado de detección de cascos), y generar reportes automáticos de *leading indicators* de seguridad (índices de frecuencia de incidentes, etc.), aprovechando la nueva tabla de incidentes. Para **Analítica**, implementar un *dashboard IA* donde el usuario pueda hacer preguntas en lenguaje natural sobre los datos (basado en los Copilots por módulo), y desplegar algoritmos de forecasting (p. ej., ARIMA o regresión) alimentados por las vistas financieras canónicas. En **Portal/UI Roles**, priorizar la incorporación de SSO Google y registro de auditoría de acciones de usuario para cumplir las políticas empresariales; además, crear en base de datos las entidades de usuarios y roles, migrando gradualmente desde Odoo a esta nueva gestión unificada. En **Infra/CI-CD**, robustecer el pipeline con jobs de backup y restore test (validando que un backup restaurado produce un sistema funcional), y configurar alertas (por ejemplo, si una prueba de integridad de datos falla, que notifique al equipo inmediatamente).

En conclusión, tras esta comparación detallada, **ninguno de los módulos está completamente fuera de rumbo**, pero aquellos más ligados a IA y a nuevas funcionalidades (QHSE, Analítica) requieren un **empuje adicional** para alcanzar el nivel de visión planteado en los documentos oficiales. Con los ajustes propuestos – todos ellos alineados con lineamientos ya definidos – Ofitec.ai fortalecerá la coherencia entre su implementación actual y los estándares funcionales, visuales, estructurales y de datos que se ha fijado como “única fuente de verdad”. Cada sugerencia aquí dada busca precisamente reforzar ese alineamiento, de modo que el sistema final sea consistente, completo y esté preparado para su misión en entornos reales con la máxima calidad.

**Referencias:**

* Documentos oficiales del proyecto (Ley de Puertos, Ley de Base de Datos, Estrategias de Páginas y Visual, Mapeo DB-Páginas, Estado del Sistema, etc.), Ofitec.ai, 2024-2025, etc. (ver citas).