Instalación Inicial del Software Dozzier ERP

Documento Técnico de Instalación y Configuración de Componentes de Software

Preparado para: Dozzier S.A.S.

Preparado por: Dozzier

Documento: PEN-DOZ-DOC-004

Versión: 1.0

Clasificación: Confidencial

Fecha: Febrero 2026

Control de Versiones

Versión	Fecha	Autor	Descripción
1.0	2026-02-19	Dozzier	Versión inicial del documento

1. Introducción

El presente documento describe el procedimiento técnico de instalación inicial del software **Dozzier ERP**, plataforma de gestión empresarial integral desarrollada por Dozzier sobre la base tecnológica del producto Dozzier ERP, configurada con la marca `APP_BRAND=dozzier`.

Dozzier ERP es un sistema de planificación de recursos empresariales (ERP) basado en web, diseñado para operar en entornos contenerizados mediante Docker Compose. Este documento cubre la instalación del software aplicativo, sus dependencias, módulos funcionales, capas de procesamiento asíncrono, tareas programadas, mecanismos de seguridad y procedimientos de validación.

El documento está dirigido al equipo técnico encargado del despliegue y operación del sistema en los ambientes de producción, preproducción y desarrollo.

2. Alcance Técnico

El alcance de este documento comprende:

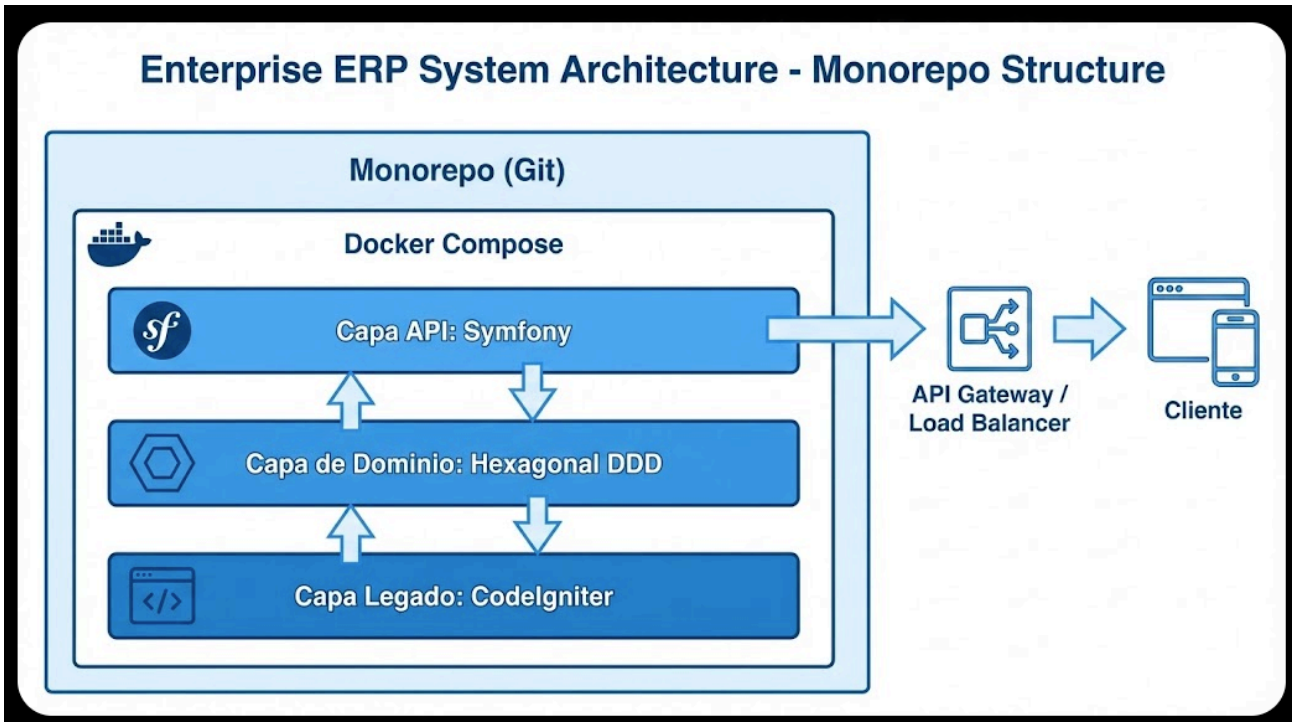
- Descripción de la arquitectura de software y sus componentes.
- Procedimiento paso a paso de instalación del software Dozzier ERP.
- Configuración de los módulos funcionales del sistema.
- Instalación y configuración de la capa de procesamiento asíncrono (RabbitMQ + Supervisor).
- Configuración de tareas programadas (Crunz Scheduler).
- Mecanismos de seguridad y autenticación integrados.
- Procedimientos de validación y pruebas automatizadas.
- Descripción de los ambientes de despliegue implementados.

Fuera de alcance: La instalación del sistema operativo, del motor de base de datos y de los servidores de aplicación se documentan en los documentos PEN-DOZ-DOC-001, PEN-DOZ-DOC-002 y PEN-DOZ-DOC-003, respectivamente.

3. Arquitectura del Sistema

3.1 Visión General

Dozzier ERP se estructura como un **monorepo** que integra múltiples capas tecnológicas bajo un único repositorio de código fuente. La arquitectura adopta un modelo **híbrido** que combina el framework CodeIgniter (capa legado) con una capa moderna basada en **Domain-Driven Design (DDD)** con arquitectura hexagonal y una capa API construida sobre **Symfony**.



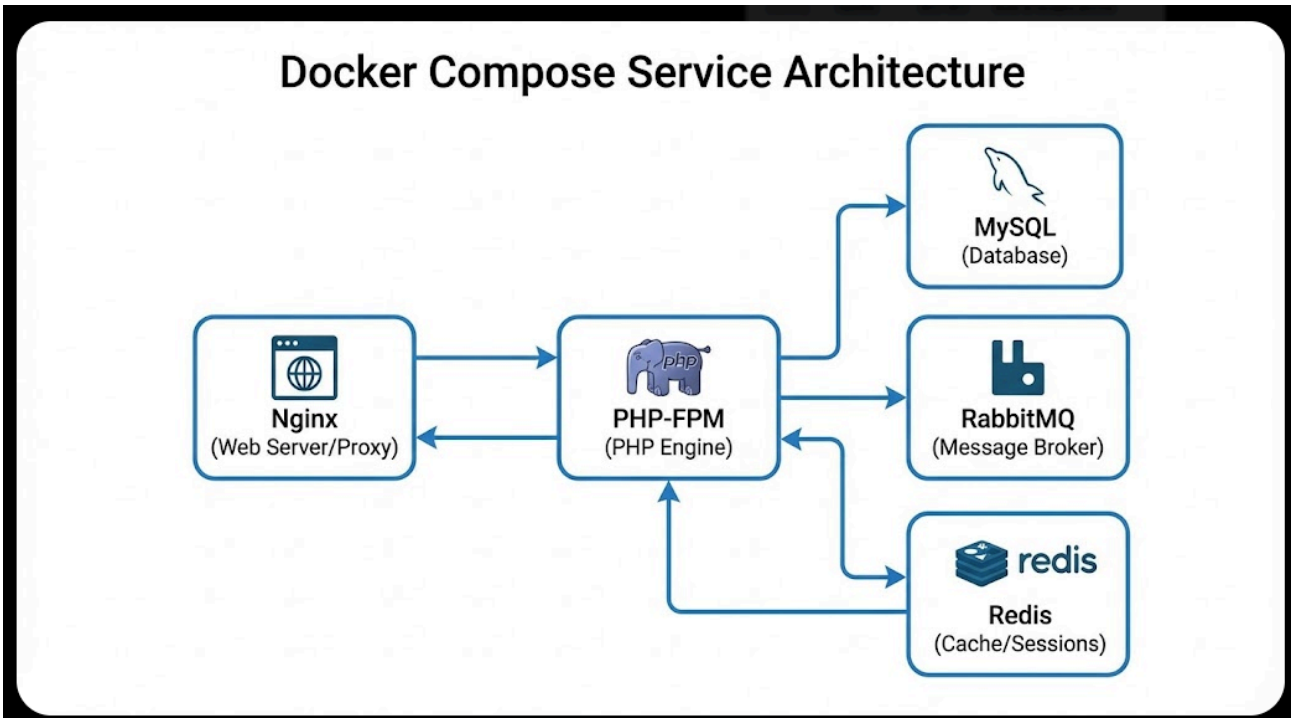
3.2 Componentes Arquitectónicos

Capa	Tecnología	Ubicación en Repositorio	Descripción
Legado	CodeIgniter	`application/`	Módulos heredados, controladores y vistas
Dominio (DDD)	PHP 8.2	`src/Dozzier/`	Contextos delimitados con arquitectura hexagonal
API Moderna	Symfony	`apps/dozzier/backend/`	Endpoints REST, servicios desacoplados
Frontend Principal	Vue.js	`apps/dozzier-vue1/` , `apps/dozzier-vue2-part1/` a `part4/`	Interfaces de usuario SPA
Frontend Complementario	React	`apps/dozzier-react/`	Componentes especializados
Orquestador Frontend	TurboRepo	Raíz del monorepo	Gestión de builds y workspaces
Contenedores	Docker Compose	`docker-compose.yml`	Orquestación de servicios

3.3 Modelo de Contenedores

El sistema opera mediante Docker Compose, con los siguientes servicios principales:

Servicio	Imagen Base	Puerto	Función
`php`	PHP 8.2-FPM	—	Motor de ejecución PHP
`nginx`	Nginx	80, 443	Servidor web y proxy inverso
`mysql`	MySQL 8.0	3306	Motor de base de datos
`rabbitmq`	RabbitMQ	5672, 15672	Broker de mensajería asíncrona
`redis`	Redis	6379	Caché y sesiones



4. Componentes del Stack Tecnológico

4.1 Backend

Componente	Versión	Propósito
PHP	8.2+	Lenguaje principal del servidor
Composer	2.x	Gestor de dependencias PHP
CodeIgniter	3.x	Framework web (capa legado)
Symfony	6.x	Framework API (capa moderna)
Phinx	Última estable	Motor de migraciones de base de datos
PHPStan	Última estable	Análisis estático de código

4.2 Frontend

Componente	Versión	Propósito
Node.js	18.x LTS	Entorno de ejecución JavaScript
npm	9.x+	Gestor de paquetes con workspaces
Vue.js	2.x / 3.x	Framework de interfaz de usuario principal
React	18.x	Framework de interfaz de usuario complementario
TurboRepo	Última estable	Orquestador de builds del monorepo

4.3 Infraestructura de Soporte

Componente	Propósito
RabbitMQ	Cola de mensajes para procesamiento asíncrono
Supervisor	Gestión de procesos consumidores (workers)
Crunz	Programador de tareas (scheduler)
Redis	Almacenamiento en caché y gestión de sesiones
Sentry	Monitorización de errores en tiempo real

5. Proceso de Instalación

El proceso de instalación de Dozzier ERP se ejecuta en fases secuenciales. Cada fase depende de la finalización exitosa de la fase anterior.

5.1 Requisitos Previos

Antes de iniciar la instalación, se debe verificar la disponibilidad de:

- **Sistema operativo** configurado según el documento PEN-DOZ-DOC-003.
- **Servidores de aplicación** (Nginx, PHP-FPM) instalados conforme al documento PEN-DOZ-DOC-001.
- **Motor de base de datos** MySQL 8.0 operativo según el documento PEN-DOZ-DOC-002.
- **Docker** y **Docker Compose** instalados y funcionales.
- **Git** instalado para clonación del repositorio.
- Acceso de red al repositorio privado en GitHub.
- Credenciales de acceso configuradas (SSH key o token de acceso personal).

5.2 Fase 1 — Clonación del Repositorio

La primera fase consiste en obtener el código fuente desde el repositorio privado de GitHub.

Clonar el repositorio
git clone git@github.com:Dozzier/dozzier.git /var/www/html # Verificar la clonación
cd /var/www/html git status

El directorio raíz del proyecto (`/var/www/html`) se convierte en el document root del servidor web Nginx. Es importante destacar que el document root apunta directamente a `/var/www/html` , **no** a un subdirectorio `/var/www/html/public` .

5.3 Fase 2 — Configuración de Archivos de Entorno

Se deben copiar y configurar los archivos de entorno a partir de las plantillas proporcionadas:

```
# Archivo de entorno principal cp
.env.example .env

# Configuración de CodeIgniter
cp application/config/config.php.example application/config/config.php
cp application/config/database.php.example application/config/database.php

# Configuración de frontend
cp public/assets/js/default/config.js.example public/assets/js/default/config.js
```

5.3.1 Variables de Entorno Críticas

El archivo `.env` debe configurarse con los siguientes parámetros esenciales:

Variable	Valor	Descripción
APP_BRAND	dozzier	Identifica la marca del ERP
APP_ENV	production	Ambiente de ejecución
DB_HOST	Según infraestructura	Host del servidor MySQL
DB_DATABASE	dozzier	Nombre de la base de datos
DB_USERNAME	Según infraestructura	Usuario de base de datos
DB_PASSWORD	Según infraestructura	Contraseña de base de datos
RABBITMQ_HOST	Según infraestructura	Host del broker RabbitMQ
REDIS_HOST	Según infraestructura	Host del servidor Redis
SENTRY_DSN	Proporcionado por Dozzier	DSN para monitoreo de errores

Nota: Las credenciales de bases de datos y servicios externos se proporcionan por canal seguro durante el proceso de implementación. Nunca se almacenan en el repositorio de código fuente.

5.3.2 Configuración de Base de Datos (CodeIgniter)

El archivo `application/config/database.php` debe configurarse con los parámetros de conexión al motor MySQL:

```
$db['default'] = array(
        'hostname' => 'mysql',                  // Nombre del servicio Docker 'username'
        => '***',                                 // Credenciales proporcionadas
        'password' => '***',
        'database' => 'dozzier', 'dbdriver' =>
        'mysqli', 'char_set' => 'utf8mb4',
        'dbcollat' => 'utf8mb4_unicode_ci',
);
```

5.3.3 Configuración del Frontend

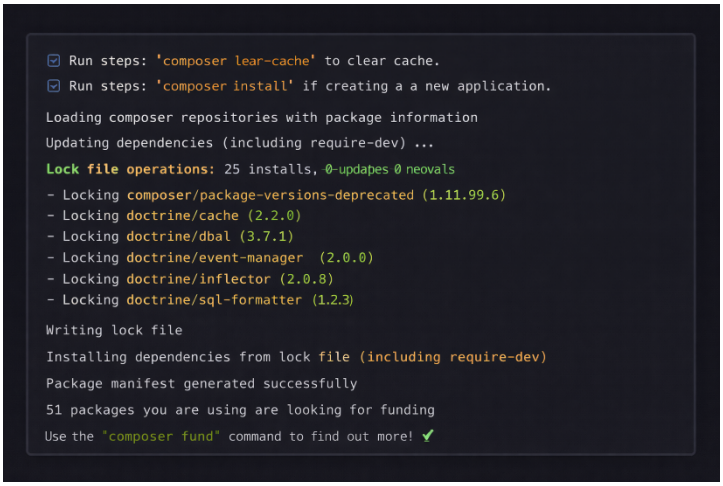
El archivo `public/assets/js/default/config.js` se ajusta con la URL base del sistema y los parámetros de la marca Dozzier.

5.4 Fase 3 — Instalación de Dependencias PHP

Se ejecuta Composer para resolver e instalar todas las dependencias del backend:

```
docker compose exec php composer install --no-dev --optimize-autoloader
```

La bandera `--no-dev` excluye dependencias de desarrollo en ambientes de producción. La bandera `--optimize-autoloader` genera un mapa de clases optimizado para mejorar el rendimiento.



5.5 Fase 4 — Instalación de Dependencias JavaScript

Se instalan las dependencias de todas las aplicaciones frontend mediante npm workspaces:

```
npm install
```

npm resuelve automáticamente las dependencias de todos los workspaces definidos en el monorepo: aplicaciones Vue.js, React y paquetes compartidos.

5.6 Fase 5 — Importación de Base de Datos

Se carga el esquema base de la base de datos desde el dump proporcionado:

```
docker compose exec -T mysql mysql -u root --password="***" dozzier < etc/databas
```

Este dump contiene:

- Estructura completa de tablas (esquema DDL).
- Datos maestros iniciales (catálogos, roles, permisos).
- Configuración de la marca Dozzier.
- Plan de cuentas contable base.

5.7 Fase 6 — Ejecución de Migraciones

Las migraciones pendientes se aplican mediante Phinx para llevar la base de datos al estado más reciente:

```
docker compose exec php vendor/bin/phinx migrate -e production
```

Phinx mantiene un registro de migraciones ejecutadas en la tabla `phinxlog`, garantizando la idempotencia del proceso. Cada migración se ejecuta una única vez.

5.8 Fase 7 — Compilación del Frontend

Se ejecuta el proceso de compilación de todas las aplicaciones frontend:

```
npm run build
```

TurboRepo orquesta la compilación en el orden correcto, respetando las dependencias entre workspaces. El proceso genera los artefactos estáticos (JavaScript, CSS, assets) optimizados para producción con minificación y tree-shaking.

Las aplicaciones compiladas incluyen:

Aplicación	Directorio	Tecnología
dozzier-vue1	`apps/dozzier-vue1/dist/`	Vue.js
dozzier-vue2-part1	`apps/dozzier-vue2-part1/dist/`	Vue.js
dozzier-vue2-part2	`apps/dozzier-vue2-part2/dist/`	Vue.js
dozzier-vue2-part3	`apps/dozzier-vue2-part3/dist/`	Vue.js
dozzier-vue2-part4	`apps/dozzier-vue2-part4/dist/`	Vue.js
dozzier-react	`apps/dozzier-react/dist/`	React

5.9 Fase 8 — Inicio de Servicios Docker

Se inician todos los contenedores de la plataforma:

```
docker compose up -d
```

Se verifica el estado de los servicios:

```
docker compose ps
```

Todos los servicios deben reportar estado `running` (o `healthy` si tienen health checks configurados).

5.10 Fase 9 — Limpieza de Cachés

Se ejecuta la limpieza de cachés para garantizar que todas las configuraciones se apliquen correctamente:

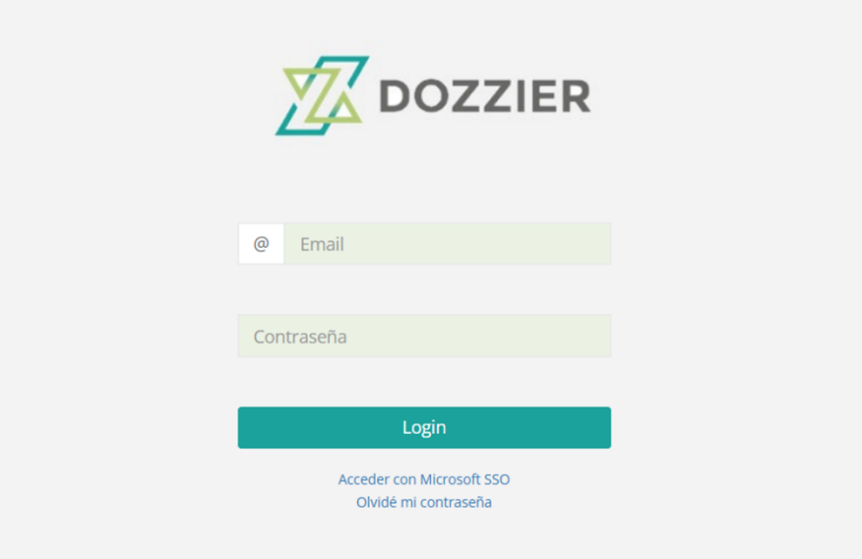
```
# Caché de Symfony API
docker compose exec php rm -rf apps/dozzier/backend/var/cache/dev

# Caché general make
clean-cache
```

5.11 Verificación Post-Instalación

Al finalizar la instalación, se realizan las siguientes verificaciones:

Verificación	Comando / Acción	Resultado Esperado
Servicios Docker	`docker compose ps`	Todos los servicios en estado `running`
Conectividad MySQL	`docker compose exec mysql mysql -u root -p -e "SELECT 1"`	Conexión exitosa
Conectividad RabbitMQ	Acceso a panel de gestión (puerto 15672)	Panel accesible
Acceso web	Navegación a URL del sistema	Página de login de Dozzier ERP
Conectividad Redis	`docker compose exec redis redis-cli ping`	Respuesta `PONG`



6. Módulos Funcionales

Dozzier ERP integra un conjunto completo de módulos funcionales que cubren las necesidades operativas de la organización. Todos los módulos comparten la misma base de datos y mecanismos de autenticación.

6.1 Mapa de Módulos

Módulo	Descripción	Funcionalidades Principales
Ventas	Gestión del ciclo de ventas	Facturación electrónica, gestión de clientes, registro de cobros, notas de crédito/débito
Compras	Gestión del ciclo de compras	Órdenes de compra, gestión de proveedores, recepción de mercancía
Inventario	Control de inventario	Catálogo de productos, gestión de bodegas, control de stock, traslados entre bodegas
Contabilidad	Gestión contable y financiera	Asientos de diario, plan de cuentas, centros de costo, reportes financieros
Contratos	Gestión contractual	Creación y seguimiento de contratos, vigencias, renovaciones
Alquileres	Gestión de arrendamientos	Contratos de alquiler, facturación periódica, control de activos arrendados
Servicios	Gestión de servicios	Registro de servicios prestados, facturación por servicios
Proyectos	Gestión de proyectos	Seguimiento de proyectos, asignación de recursos, control presupuestal

Recursos Humanos	Gestión del talento humano	Expedientes de empleados, estructura organizacional
Nómina	Gestión de nómina	Planillas de pago, cálculo de deducciones, liquidaciones, ciclos de pago
POS	Punto de venta	Facturación rápida, gestión de caja, transacciones en punto de venta
Legal	Gestión jurídica	Documentación legal, seguimiento de procesos
Portal de Solicitudes	Portal de autogestión	Catálogo de ítems, solicitudes con flujos de aprobación, carrito de compras

6.2 Navegación del Sistema

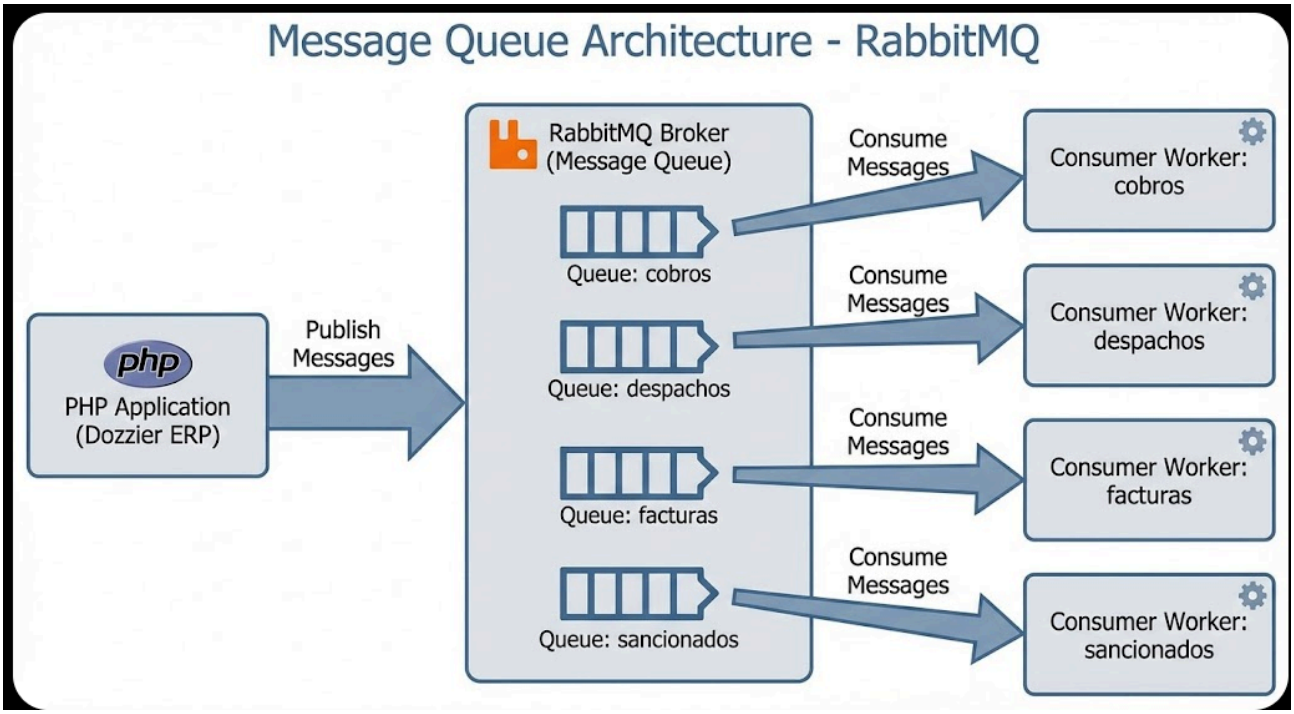
Los módulos se organizan en dos niveles de acceso en la interfaz de usuario:

Barra superior (Topbar):

- Ventas | Compras | Inventario | Contabilidad | Contratos

Menú desplegable (≡▼):

Alquileres | Servicios | Proyectos | Recursos Humanos | Nómina | POS | Legal | Portal de Solicitudes



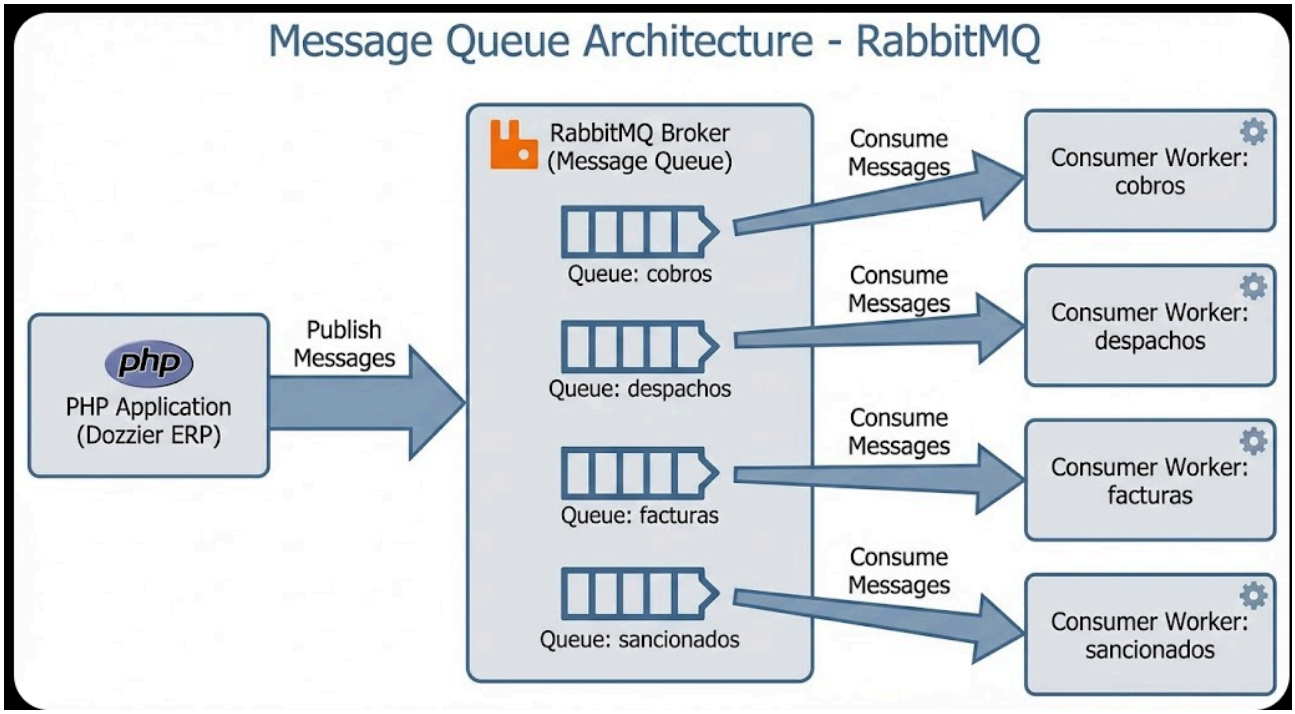
6.3 Dependencias entre Módulos

Módulo Origen	Módulo Destino	Relación
Ventas	Contabilidad	Facturación genera asientos contables automáticos
Compras	Inventario	Recepción de órdenes actualiza stock
Compras	Contabilidad	Órdenes de compra generan registros contables
Nómina	Contabilidad	Planillas generan asientos de nómina
Inventario	Contabilidad	Movimientos de inventario afectan cuentas contables
Portal de Solicitudes	Compras	Solicitudes aprobadas pueden generar órdenes de compra

7. Capa Asíncrona (RabbitMQ + Supervisor)

7.1 Arquitectura de Mensajería

Dozzier ERP utiliza **RabbitMQ** como broker de mensajería para el procesamiento asíncrono de operaciones que no requieren respuesta inmediata al usuario. Los mensajes se publican desde la aplicación principal y son consumidos por procesos worker dedicados.



7.2 Consumidores (Workers)

Consumidor	Cola	Función
consumer_cobros	Cobros	Procesamiento de transacciones de cobros y conciliaciones
consumer_despachos	Despachos	Procesamiento de despachos y logística de inventario
consumer_facturas	Facturas	Generación y procesamiento de facturas electrónicas
consumer_sancionados_reverse_scan	Sancionados	Escaneo inverso de listas de sancionados para cumplimiento regulatorio

7.3 Configuración de Supervisor

Los consumidores se gestionan mediante **Supervisor**, que garantiza su ejecución continua y reinicio automático en caso de fallo. La configuración típica de cada worker en Supervisor es:

```
[program:consumer_cobros]
command=php /var/www/html/artisan queue:consume cobros autostart=true
autorestart=true
startsecs=5
stopwaitsecs=30
numprocs=1
stdout_logfile=/var/log/supervisor/consumer_cobros.log
stderr_logfile=/var/log/supervisor/consumer_cobros_error.log
```

Los parámetros relevantes incluyen:

Parámetro	Valor	Descripción
autostart	true	Inicia automáticamente con Supervisor
autorestart	true	Reinicia en caso de terminación inesperada
startsecs	5	Segundos antes de considerar el proceso iniciado
stopwaitsecs	30	Tiempo de espera para terminación limpia
numprocs	1	Número de instancias del proceso

7.4 Monitorización de Colas

El panel de gestión de RabbitMQ (puerto 15672) permite monitorear en tiempo real:

- Número de mensajes en cola (ready, unacked, total). •

Tasa de publicación y consumo de mensajes.

- Estado de los consumidores conectados.
- Métricas de rendimiento por cola.

8. Tareas Programadas (Crunz Scheduler)

8.1 Configuración del Scheduler

Dozzier ERP utiliza **Crunz** como motor de tareas programadas, invocado cada minuto mediante cron del sistema operativo:

```
* * * * * cd /var/www/html && vendor/bin/crunz schedule:run >> /dev/null 2>&1
```

Crunz evalúa las tareas registradas y ejecuta aquellas cuya programación coincide con el minuto actual.

8.2 Tareas Registradas

Las tareas programadas incluyen, entre otras:

Tarea	Frecuencia	Descripción
Notificaciones por correo	Configurable	Envío de notificaciones pendientes por email
Cierre de períodos contables	Diaria	Verificación y procesamiento de períodos contables
Procesamiento de eventos de dominio	Cada minuto	Despacho de eventos pendientes del bus de dominio
Limpieza de archivos temporales	Diaria	Eliminación de archivos temporales expirados
Sincronización de datos	Configurable	Sincronización con sistemas externos según configuración

8.3 Verificación del Scheduler

Para verificar que el scheduler está operativo:

```
# Listar tareas registradas
docker compose exec php vendor/bin/crunz schedule:list

# Verificar el cron del sistema crontab -l |
grep crunz
```

9. Seguridad y Autenticación

9.1 Mecanismos de Seguridad Implementados

Capa	Mecanismo	Descripción
Transporte	HTTPS / TLS	Cifrado de todas las comunicaciones cliente-servidor
Autenticación	RBAC	Control de acceso basado en roles y permisos granulares
SSO	Azure Active Directory	Autenticación única corporativa mediante Azure AD
Monitoreo	Sentry	Captura y notificación de errores en tiempo real
Sesiones	Redis	Almacenamiento seguro de sesiones con expiración configurable

9.2 Control de Acceso Basado en Roles (RBAC)

El sistema implementa un modelo de permisos granular:

- **Roles**: Agrupaciones de permisos asignables a usuarios (ej. Administrador, Contador, Vendedor).
- **Permisos**: Acciones específicas sobre módulos y recursos (ej. `facturas.crear`, `inventario.consultar`).
- **Multitenancy**: Cada empresa (tenant) gestiona sus propios roles y permisos de manera independiente.

9.3 Integración con Azure Active Directory

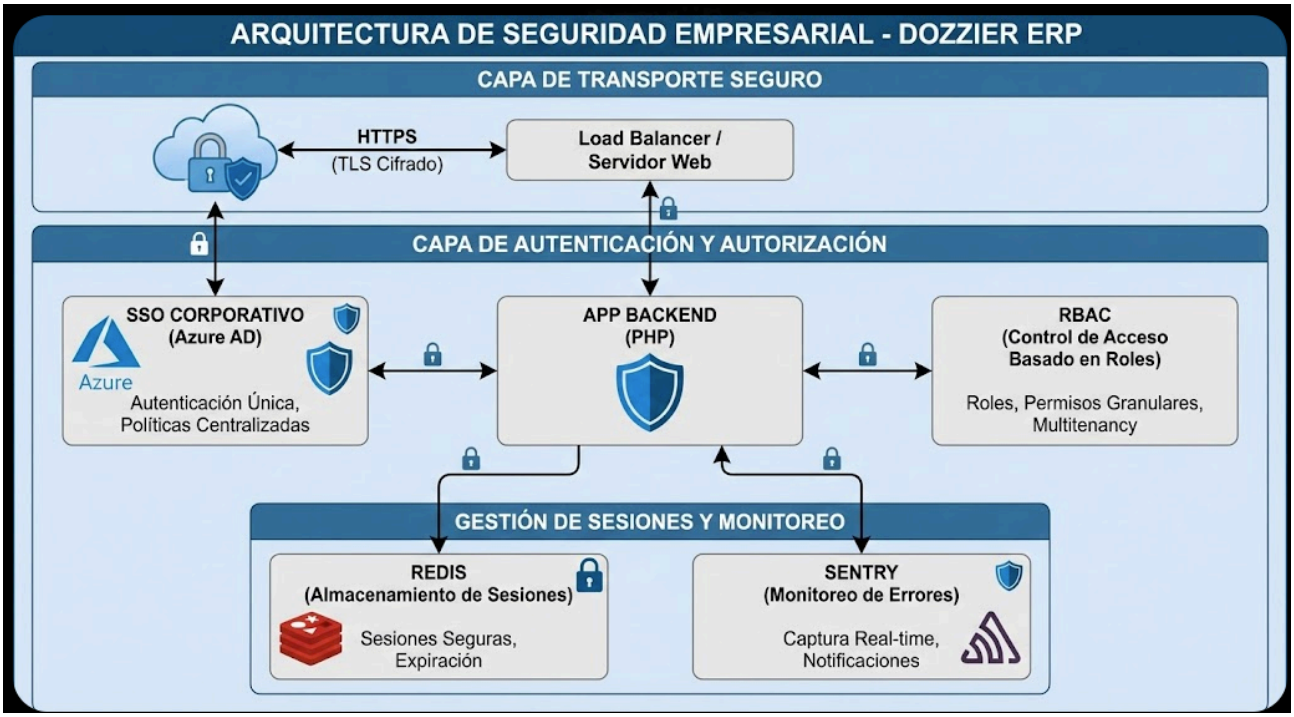
La autenticación mediante Azure AD SSO permite:

- Inicio de sesión único con credenciales corporativas.
- Sincronización de usuarios desde el directorio activo.
- Aplicación de políticas de seguridad centralizadas (MFA, bloqueo por intentos fallidos).
- Revocación centralizada de acceso.

9.4 Monitoreo de Errores (Sentry)

Sentry se integra tanto en el backend PHP como en las aplicaciones frontend para:

- Captura automática de excepciones no controladas.
- Agrupación inteligente de errores recurrentes.
- Notificaciones al equipo de desarrollo en tiempo real.
- Trazas de contexto (usuario, request, environment) para diagnóstico.



10. Validación y Pruebas

10.1 Suite de Pruebas Automatizadas

Dozzier ERP mantiene una suite integral de pruebas automatizadas ejecutadas mediante el pipeline de CI/CD:

Tipo de Prueba	Herramienta	Comando	Cobertura
Análisis estático	PHPStan	`make static- analysis`	Detección de errores de tipado y lógica
Pruebas unitarias	PHPUnit (3 suites)	`make test`	Lógica de dominio y servicios
Pruebas de aceptación	Behat	`make behat`	Flujos de negocio completos
Pruebas E2E	Playwright	`npx playwright test`	Interfaz de usuario en navegador

10.2 Análisis Estático (PHPStan)

PHPStan realiza análisis estático del código PHP sin necesidad de ejecución:

- Verificación de tipos en parámetros, retornos y asignaciones.
- Detección de llamadas a métodos inexistentes.
- Validación de acceso a propiedades de objetos.
- Cumplimiento de tipos estrictos (`declare(strict_types=1)`).

10.3 Pruebas Unitarias (PHPUnit)

La suite de PHPUnit se organiza en tres suites de pruebas que cubren:

- **Suite de dominio**: Lógica de negocio en los contextos delimitados (`src/Dozzier/`).
- **Suite de aplicación**: Servicios de aplicación y casos de uso.
- **Suite de infraestructura**: Repositorios, adaptadores y servicios externos.

10.4 Pruebas de Aceptación (Behat)

Las pruebas Behat validan escenarios de negocio completos descritos en lenguaje Gherkin, verificando que los flujos funcionales operan conforme a los requisitos.

10.5 Pruebas End-to-End (Playwright)

Playwright ejecuta pruebas automatizadas en navegador, simulando interacciones reales del usuario con la interfaz gráfica de Dozzier ERP.

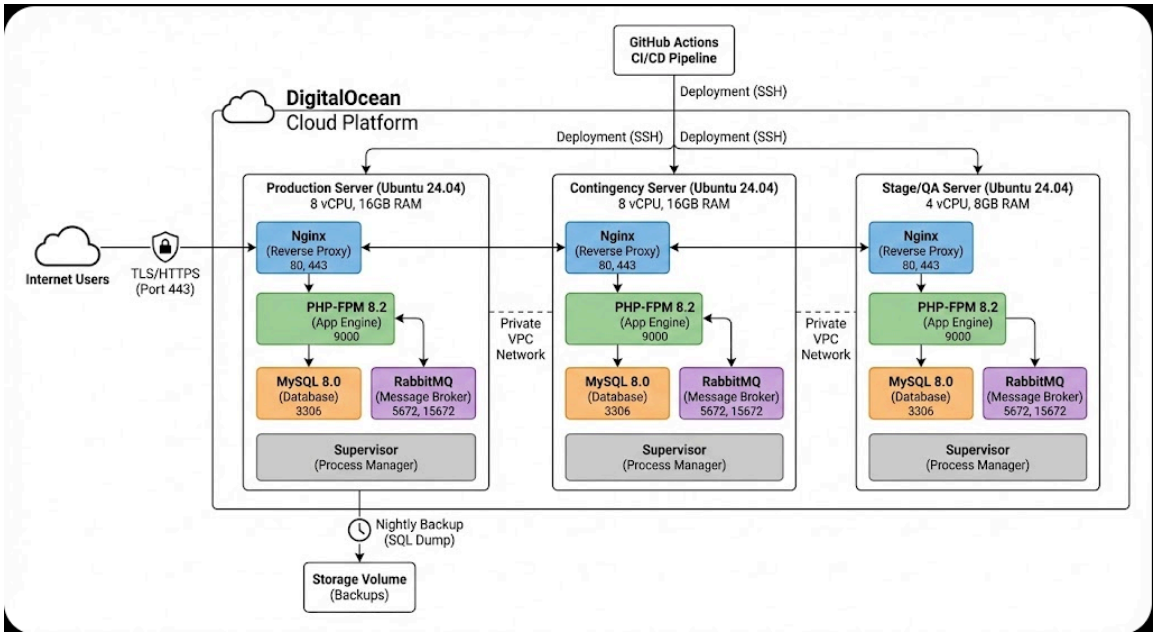
10.6 Validación Post-Instalación

Tras la instalación, se recomienda ejecutar la suite de validación completa:

```
# Análisis estático make
static-analysis

# Pruebas unitarias make test

# Pruebas de aceptación make
behat
```



11. Ambientes Implementados

11.1 Matriz de Ambientes

Ambiente	Propósito	Rama Git	Despliegue
Desarrollo	Desarrollo activo y pruebas locales	Feature branches	Local (Docker Compose)
QA	Pruebas de calidad y validación	`QA`	Automático (GitHub Actions)
Preproducción	Validación final antes de producción	`staging`	Automático (GitHub Actions)
Producción	Ambiente operativo del cliente	`master`	Automático (GitHub Actions)

11.2 Pipeline de CI/CD (GitHub Actions)

El proceso de integración y despliegue continuo se ejecuta mediante GitHub Actions con las siguientes etapas:

1. **Análisis estático**: PHPStan verifica la integridad del código.
2. **Pruebas unitarias**: PHPUnit ejecuta las tres suites de pruebas.
3. **Pruebas de aceptación**: Behat valida los flujos de negocio.
4. **Pruebas E2E**: Playwright verifica la interfaz de usuario.
5. **Despliegue**: Conexión SSH al servidor de destino, pull del código, ejecución de migraciones, compilación de frontend y limpieza de cachés.

Push → PHPStan → PHPUnit (3 suites) → Behat → Playwright → Deploy (SSH)

11.3 Proceso de Despliegue Automático

El despliegue a producción se ejecuta automáticamente al fusionar cambios en la rama `master`:

1. GitHub Actions establece conexión SSH con el servidor de producción.
2. Se ejecuta `git pull` para obtener la última versión del código.
3. Se ejecutan las migraciones de base de datos pendientes (`phinx migrate`).
4. Se instalan dependencias actualizadas (`composer install` , `npm install`).
5. Se compilan los assets del frontend (`npm run build`).
6. Se limpian las cachés del sistema.
7. Se reinician los consumidores de RabbitMQ mediante Supervisor.

11.4 Balanceo de Carga

Para atender una base de aproximadamente **300 usuarios concurrentes**, la infraestructura soporta la implementación de un DigitalOcean Load Balancer que distribuye el tráfico entre múltiples instancias del servidor Nginx. Esta configuración permite:

- Distribución equitativa de la carga de trabajo.
- Alta disponibilidad mediante redundancia de instancias.
- Escalamiento horizontal agregando instancias adicionales según demanda.

12. Conclusiones

El presente documento ha detallado el procedimiento completo de instalación inicial del software Dozzier ERP, cubriendo desde la clonación del repositorio hasta la verificación de operatividad del sistema en producción.

Los aspectos clave del despliegue incluyen:

- **Arquitectura contenerizada**: Docker Compose facilita la reproducibilidad y portabilidad del ambiente.
- **Proceso automatizado**: El pipeline de CI/CD garantiza que cada despliegue pase por las etapas de análisis estático, pruebas unitarias, de aceptación y end-to-end antes de llegar a producción.
- **Procesamiento asíncrono**: RabbitMQ con Supervisor asegura el procesamiento confiable de operaciones de larga duración.
- **Seguridad integral**: HTTPS, RBAC, Azure AD SSO y Sentry proporcionan múltiples capas de protección y monitoreo.
- **Módulos funcionales completos**: Los trece módulos integrados cubren el espectro completo de operaciones empresariales requeridas.

Dozzier mantiene la responsabilidad del soporte técnico, actualizaciones y evolución del software conforme a los términos contractuales establecidos.

Este documento contiene información confidencial y propietaria de Dozzier
Su distribución está restringida exclusivamente a las partes autorizadas bajo el contrato vigente.
Prohibida su reproducción total o parcial sin autorización expresa.