Manual Técnico Integral del Proyecto ASSIP‑ERP

Empresa: ERP ASSIP SOLIDARIA Y FINANCIERA

Autor: Carlos González Pérez — Dirección de Tecnología

Fecha: 21/10/2025

Versión: 1.0

# Índice

1. Resumen Ejecutivo

2. Arquitectura General

3. Esquemas de Base de Datos

4. Reglas de Seguridad y Control de Acceso

5. Gestión Multi‑Agencia

6. Auditoría y Trazabilidad

7. Reporting e Informes Dinámicos

8. Filosofía y Reglas de Desarrollo

9. Estructura de Módulos y Responsabilidades

10. Protocolo de Comunicación y Fases de Trabajo

11. Escalabilidad y Futuras Integraciones

# 1. Resumen Ejecutivo

ASSIP‑ERP es un sistema integral de gestión empresarial diseñado para el sector solidario y financiero. Su objetivo es unificar procesos operativos, administrativos y contables de cooperativas, fondos de empleados y mutuales bajo una arquitectura modular, escalable y auditable.

# 2. Arquitectura General

El sistema sigue una arquitectura cliente‑servidor de tres capas:  
• Backend: desarrollado en Java 17 con Spring Boot.  
• Frontend: Angular Standalone (versión 20+), con servicios REST.  
• Base de Datos: PostgreSQL, organizada por esquemas independientes por dominio.  
  
Comunicación a través de API REST JSON con autenticación JWT.  
Los entornos de trabajo se separan en:  
• DEV: desarrollo local (spring.profiles.active=dev)  
• PROD: producción con configuración segura y logs controlados.

# 3. Esquemas de Base de Datos

La base de datos PostgreSQL está organizada en esquemas funcionales:  
• seguridad — Usuarios, roles, permisos, sesiones, auditorías.  
• general — Agencias, zonas, subzonas, parámetros y empresa.  
• hoja\_vida — Datos personales, familiares, laborales, SARLAFT, referencias.  
• contabilidad — Catálogo de cuentas, comprobantes, tipos y auxiliares.  
• depositos — Aportes, ahorros, movimientos y extractos.  
• cdats — Certificados de depósito a término fijo y tasas.  
• cartera — Créditos, abonos, pagos, garantías.  
• activos\_fijos — Bienes, depreciación, ubicación.  
• nomina — Empleados, pagos, deducciones.  
• tesoreria — Bancos, egresos, cuentas por pagar/cobrar.  
• tarjetas — Tarjetas débito/crédito con switch transaccional.  
• reporting — Metadatos, vistas y control de informes.

# 4. Reglas de Seguridad y Control de Acceso

La autenticación utiliza tokens JWT con información de usuario, rol y agencia.  
Cada usuario pertenece a una agencia y su ámbito de acción depende del módulo:  
• En contabilidad, solo puede registrar transacciones de su propia agencia.  
• En depósitos o cartera, puede recibir transacciones de asociados de otras agencias.  
• En hoja\_vida, puede consultar o actualizar datos desde cualquier agencia, registrando siempre quién realizó la acción.  
  
Las contraseñas se almacenan con hashing BCrypt y cada sesión es sin estado (stateless). Los controladores están protegidos por filtros de autenticación y roles específicos.

# 5. Gestión Multi‑Agencia

Cada entidad registra el código de agencia. El sistema maneja jerarquía:  
• Agencia Central (principal)  
• Agencias sede A, B, C, etc.  
  
Esto permite consolidar información o filtrar por agencia.  
  
Ejemplo: generar un balance contable unificado o listado de afiliados de todas las agencias.

# 6. Auditoría y Trazabilidad

Todas las operaciones generan registros de auditoría:  
• usuario\_creacion, usuario\_actualizacion  
• fecha\_creacion, fecha\_actualizacion  
• agencia y origen (IP, User‑Agent)  
  
Los eventos se almacenan en la tabla seguridad.log\_evento y pueden ser consultados desde el módulo de Reporting.

# 7. Reporting e Informes Dinámicos

El módulo de reporting permite listar datos de cualquier esquema. Usa metadatos registrados automáticamente (tablas y campos). Los usuarios pueden construir listados o informes seleccionando campos disponibles sin requerir cambios en el backend. La seguridad controla el acceso por rol o permiso.

# 8. Filosofía y Reglas de Desarrollo

ASSIP‑ERP prioriza claridad, estabilidad y mantenibilidad. Las fases de trabajo son:  
• [ANÁLISIS] – comprensión y documentación.  
• [DISEÑO] – estructura, DTOs, relaciones, sin código final.  
• [IMPLEMENTAR] – codificación completa y pruebas.  
  
Antes de aplicar cambios:  
1. Crear copia de seguridad.  
2. Sincronizar BD con Flyway.  
3. Verificar rutas, nombres y versiones.

# 9. Estructura de Módulos y Responsabilidades

|  |  |  |  |
| --- | --- | --- | --- |
| Módulo | Backend | Frontend | BD / Esquema |
| Seguridad | Spring Boot | Angular Auth | seguridad |
| General | Spring Boot | Angular General | general |
| Hoja de Vida | Spring Boot | Angular HojaVida | hoja\_vida |
| Contabilidad | Spring Boot | Angular Contabilidad | contabilidad |
| Depósitos | Spring Boot | Angular Depósitos | depositos |
| CDATs | Spring Boot | Angular CDATs | cdats |
| Cartera | Spring Boot | Angular Cartera | cartera |
| Activos Fijos | Spring Boot | Angular Activos | activos\_fijos |
| Nómina | Spring Boot | Angular Nómina | nomina |
| Tesorería | Spring Boot | Angular Tesorería | tesoreria |
| Tarjetas | Spring Boot | Angular Switch | tarjetas |
| Reporting | Spring Boot | Angular Reportes | reporting |

# 10. Protocolo de Comunicación y Fases de Trabajo

El equipo y las IAs colaboradoras deben seguir estas reglas:  
• Usar encabezados [ANÁLISIS], [DISEÑO], [IMPLEMENTAR].  
• Documentar toda decisión antes de escribir código.  
• Un solo tema por hilo.  
• Revisar compatibilidad entre esquemas antes de aplicar migraciones.  
• Mantener registro de versiones de backend, frontend y BD.

# 11. Escalabilidad y Futuras Integraciones

El diseño modular permite agregar nuevos esquemas y microservicios:  
• API pública para integraciones externas.  
• Soporte para switch transaccional de tarjetas.  
• Migración desde sistemas VB6 con JSON y procedimientos almacenados.  
• Posible adopción de contenedores (Docker) en fases futuras.