eso-pic

SANDRA

ECOSISTEMA SOBERANO DE INFRAESTRUCTURA

WHITE PAPER TÉCNICO

Versión 1.0.0 | Abril 2026

Proyecto

Sandra

Código Abierto / Sovereign Tech

CARACAS, VENEZUELA

1 Resumen Ejecutivo

Sandra es un ecosistema distribuido determinista diseñado en Go y Rust que trasciende el middleware tradicional para convertirse en un **Sistema Operativo de Capa 7** (Layer-7 OS). Este documento técnico presenta la arquitectura, componentes fundamentales y capacidades del ecosistema Sandra, fundamentado en los principios de soberanía tecnológica, seguridad Zero-Trust y rendimiento extremo.

El ecosistema se compone de cuatro pilares fundamentales: **Valquiria** (Sandra Server - núcleo de orquestación en Go), **Loki** (Sandra Sentinel - motor de cálculo en Rust), **Freya** (Sandra Desktop Container - entorno seguro de escritorio) y **Ela** (Sandra UI Console - panel de administración). Juntos, estos componentes forman una plataforma empresarial completa para la gestión de nómina, documentación y operaciones críticas.

Palabras clave: Middleware, Capa 7, Soberanía Tecnológica, Go, Rust, Zero-Trust, gRPC

2 La Relevancia de la Capa 7

2.1 Del Hardware a la Intención

Los sistemas operativos tradicionales gestionan hardware y controladores a nivel de Capa 1-4 del modelo OSI. Sandra, en cambio, opera en la **Capa 7 (Aplicación)** donde reside la inteligencia del negocio. Esta abstracción permite gestionar no solo datos, sino **intenciones y relaciones lógicas** entre entidades.

2.2 Abstracción Lógica

Sandra oculta la heterogeneidad de redes, sistemas operativos y lenguajes de programación (PHP, Go, Rust, TypeScript) mediante una **Capa Común** denominada Interlogical. Esta abstracción permite que componentes heterogéneos se comuniquen de manera fluida sin conocer los detalles de implementación de sus pares.

2.3 Orquestación de Intención

A diferencia de los middleware tradicionales que simplemente mueven datos, Sandra los interpreta. El sistema decide flujos de trabajo, valida contextos de dispositivos, y sincroniza estados en tiempo real mediante protocolos seguros.

2.4 Zero-Trust L7

La arquitectura Zero-Trust de Sandra garantiza que cada mensaje sea autenticado y autorizado individualmente. La validación no ocurre solo en los bordes del sistema, sino en la estructura profunda de cada mensaje, inspeccionando contenido semántico más allá deHeaders tradicionales.

Principios Fundamentales de Sandra

- **Soberanía Tecnológica**: Código abierto con licenciamiento que permite derivados cerrados para sectores regulados.
- **Seguridad por Diseño**: Cifrado AES-256, autenticación JWT, validación profunda de mensajes.
- **Rendimiento Extremo**: Pipeline asíncrono, procesamiento in-memory, parsing SIMD.
- **Determinismo**: Comportamiento predecible bajo cualquier carga, sin condiciones de carrera.

3 Arquitectura del Ecosistema

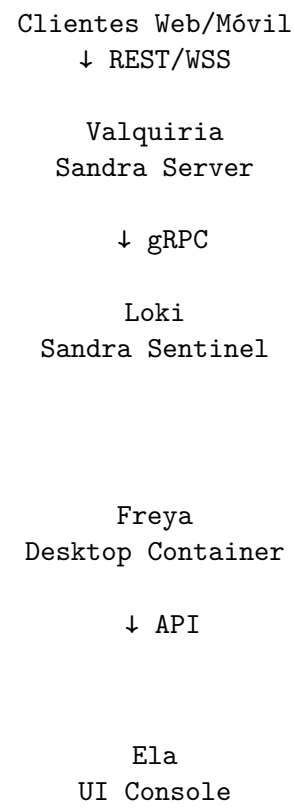
El ecosistema Sandra implementa una arquitectura de microservicios con comunicación síncrona (gRPC) y asíncrona (WebSockets), optimizada para cargas de trabajo de alta disponibilidad.

3.1 Visión General de Componentes

Cuadro 1: Componentes del Ecosistema Sandra

Componente	Nombre	Tecnología	Función Principal
Valquiria	Sandra Server	Go 1.24	Orquestación REST/gRPC
Loki	Sandra Sentinel	Rust	Cálculo de nómina
Freya	Desktop Container	Tauri/Rust	Aplicación de escritorio segura
Ela	UI Console	Angular	Panel administrativo SRE

3.2 Diagrama de Arquitectura



4 Valquiria: Sandra Server (Go)

Valquiria constituye el núcleo de orquestación del ecosistema Sandra. Desarrollado en Go 1.24+, proporciona una plataforma escalable para la gestión de APIs REST, WebSockets y comunicación gRPC con el motor de cálculo.

4.1 Arquitectura Zero-Trust

El servidor implementa una arquitectura Zero-Trust donde cada solicitud es autenticada, autorizada y validada independientemente, sin confiar en ningún componente externo.

```go
package main

type Hub struct {
    Shards    map[uint8]*Shard
    Register  chan *Client
    Broadcast chan []byte
}

func (h *Hub) HandleSharding(id string) *Shard {
    shardID := calculateHash(id) % 256
    return h.Shards[shardID]
}
```

4.2 WSHub: Sistema de Sharding

El componente WSHub implementa un sistema de particionamiento (sharding) con 256 fragmentos independientes, eliminando los bloqueos de CPU al distribuir las conexiones WebSocket en shards separados.

Características Técnicas de WSHub

- **256 Shards**: Particionamiento de conexiones para paralelización real
- **Sincronización en Tiempo Real**: Estado auditado continuamente
- **Conexiones Persistentes**: WebSockets con reconexión automática
- **Balanceo de Carga**: Distribución uniforme de clientes

4.3 ModulosAPP: Gestión Documental

El sistema de gestión documental integra un motor de PDF y CDN interno con las siguientes capacidades:

- **Encriptación al Vuelo**: Cifrado AES-256 de documentos en tránsito
- **Firmas Digitales**: Inmutabilidad garantizada mediante firmas criptográficas
- **Generación de PDFs**: Motor de renderizado para nóminas, reportes y constancias

- **Caché Inteligente**: CDN distribuido para optimización de recursos estáticos

4.4 Endpoints Principales

Endpoint	Descripción
`POST /v1/api/auth/login`	Autenticación de usuarios
`POST /v1/api/auth/logout`	Cierre de sesión
`GET /v1/api/nomina/directiva`	Consulta de directivas de nómina
`POST /v1/api/plotes`	Generación de plots/gráficos
`WS /ws/hub`	WebSocket para actualizaciones en tiempo real

4.5 Configuración del Servidor

```
[server]
port=80
ssl_port=443
grpc_port=8443

[database]
host=localhost
port=5432
name=ipsfa
user=sandra
password=${SANDRA_DB_PASS}
```

5 Loki: Sandra Sentinel (Rust)

Sandra Sentinel (Loki) es el motor de cálculo del ecosistema, desarrollado en Rust para garantizar rendimiento extremo y seguridad de memoria. Su función principal es el procesamiento de nómina a gran escala mediante técnicas de computación in-memory.

5.1 Arquitectura de Procesamiento

Loki implementa una arquitectura de tubería asíncrona (async pipeline) que maximiza el throughput mediante el procesamiento simultáneo de lotes de datos.

Capacidades de Procesamiento

- **In-Memory Hash Join**: Algoritmos Build & Probe para unión de 500k+ registros
- **Pipeline Asíncrono**: Procesamiento de lotes sin tiempos muertos
- **Parsing SIMD**: Instrucciones vectoriales para JSON Zero-Copy
- **gRPC Streaming**: Comunicación bidireccional con el servidor

5.2 Ejecución vía Manifiestos

Loki opera mediante manifiestos JSON que definen los parámetros de procesamiento:

```
{
  "nombre": "Nomina Enero 2026",
  "ciclo": "2026-01",
  "directiva": "DIR-2026-01",
  "componente": "EJ",
  "partida": "201",
  "filtros": {
    "status": 201,
    "grado": "Oficial"
  },
  "modulos": ["Base", "Directiva"],
  "calcular": true,
  "emitir": true
}
```

5.3 Tipos de Nómina Soportados

Código	Descripción
C	Nómina de Conceptos
P	Prima
A	Anticipo
D	Descuento
Npat	Patria (Pago de patria)

5.4 Métricas de Rendimiento

Benchmarks de Loki

- **Fusión Completa**: 1.2 segundos para 500k+ registros
- **Throughput**: ∼500,000 registros/segundo
- **Consumo de Memoria**: Optimizado para datasets grandes
- **Cero Fugas de Memoria**: Garantizado por el ownership de Rust

6 Freya: Sandra Desktop Container

Sandra Desktop Container (Freya) es un entorno de ejecución seguro que aísla micro-aplicaciones del sistema host. Utiliza Tauri con Rust como backend y Angular como frontend, eliminando las vulnerabilidades asociadas a los runtime de Node.js.

6.1 Arquitectura de Seguridad

Freya implementa múltiples capas de protección:

Características de Seguridad

- **Memory-Safe**: Sin runtime de Node.js, solo Rust compilado
- **Aislamiento de Procesos**: Entornos de ejecución completamente aislados
- **Protocolo Proprietario**: `sandra-app://` para comunicación segura
- **Hot-Patching**: Actualizaciones de seguridad sin reinicio

6.2 Cifrado y Almacenamiento

Componente	Tecnología
Base de Datos	SQLite Cipher (AES-256)
Hashing de Credenciales	Argon2id (resistente a GPU)
Cifrado Local	AES-256-GCM
Aislamiento	Sandboxing a nivel de proceso

6.3 Casos de Uso

Freya está diseñado para escenarios que requieren:

- **Entornos de Alta Seguridad**: Instalaciones en redes aisladas
- **Aplicaciones Financieras**: Nómina y gestión de recursos humanos
- **Operaciones Críticas**: Sistemas que no pueden depender de conectividad
- **Cumplimiento Regulatorio**: Datos sensibles que requieren cifrado local

7 Ela: Sandra UI Console

Sandra UI Console es el centro de comando del ecosistema, desarrollado en Angular con arquitectura SRE (Site Reliability Engineering).

7.1 Características Principales

Funcionalidades de la Consola

- **Heartbeat en Tiempo Real**: Monitoreo de latencia y salud de nodos vía gRPC
- **Bitácora Inmutable**: Trazabilidad completa con firmas digitales
- **RBAC Granular**: Gestión de perfiles y roles JWT
- **Debugging en Caliente**: Inspección sin detener producción

7.2 Arquitectura Observabilidad

La consola implementa un sistema completo de observabilidad:

- **Métricas**: Recolección de indicadores de rendimiento (CPU, memoria, latencia)
- **Logs**: Agregación centralizada de logs estructurados
- **Traces**: Trazabilidad distribuida de peticiones
- **Alertas**: Notificaciones configurables basadas en umbrales

8 Seguridad y Cumplimiento

El ecosistema Sandra está diseñado desde su concepción con seguridad como requisito fundamental.

8.1 Modelo de Seguridad

1. **Autenticación**: JWT con expiración configurable y refresh tokens
2. **Autorización**: RBAC con permisos granulares a nivel de recurso
3. **Cifrado**: AES-256 para datos en reposo y en tránsito (TLS 1.3)
4. **Validación**: Esquemas JSON rigurosos en todas las APIs
5. **Rate Limiting**: Protección contra ataques de fuerza bruta

8.2 Cumplimiento Normativo

Sandra está diseñado para cumplir con:

- **ISO/IEC 27001**: Sistema de Gestión de Seguridad de la Información
- **ISO 22301**: Gestión de la Continuidad del Negocio
- **Ley de Infogobierno (Venezuela)**: Requisitos de soberanía tecnológica
- **LGPD/GDPR**: Protección de datos personales

9 Licenciamiento

La estrategia de licenciamiento de Sandra es fundamentalmente **estratégica**, priorizando la autonomía tecnológica.

9.1 Modelos de Licencia

Licencia	Tipo	Uso Recomendado
Apache 2.0	Permisiva	Módulos principales, innovación abierta
MIT	Permisiva	Utilidades, herramientas auxiliares
Propietaria	Cerrada	Derivados para sectores regulados

9.2 Ventajas de la Estrategia

Licencias Permisivas

- Permite innovación sin restricciones
- Facilita la privatización de derivados
- Ideal para sectores defensa y gobierno
- Cumple normativas de clasificación de seguridad

La licencia Apache 2.0 permite que el software final sea cerrado y clasificado, una característica fundamental para proyectos gubernamentales y de defensa.

10 Conclusiones

Sandra representa una nueva generación de middleware empresarial, fundamentado en principios de soberanía tecnológica y seguridad Zero-Trust. La combinación de Go para orquestación y Rust para cálculo de alto rendimiento ofrece lo mejor de ambos mundos: productividad del desarrollo y seguridad de memoria.

10.1 Resumen de Beneficios

- **Escalabilidad Horizontal**: Arquitectura preparada para crecer con la demanda
- **Determinismo**: Comportamiento predecible bajo cualquier condición
- **Seguridad Militar**: Cifrado, autenticación y autorización de nivel empresarial
- **Soberanía Tecnológica**: Código abierto con control total sobre el software
- **Rendimiento Extremo**: Procesamiento de millones de registros en segundos

10.2 Contacto y Contribuciones

Para más información sobre el Proyecto Sandra:

- **Repositorio**: github.com/proyecto-sandra
- **Documentación**: docs.sandra.io
- **Comunidad**: comunidad@sandra.io

"La convergencia Go-Rust en Sandra es una respuesta ética a la fragilidad del software moderno."

Fecha de publicación: Abril 2026
Versión del documento: 1.0.0
Proyecto Sandra - Código Abierto / Sovereign Tech