
	DOCUMENTO ANALISIS DE APLICACIONES		
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS		X-XX-X-XX
		XX-XX-202X	V-X

INFORME IDENTIFICACION Y ANALISIS APLICACIONES MONOLITICAS


MINISTERIO DE MINAS Y ENERGÍA

REPÚBLICA DE COLOMBIA

ESTRATEGIA DE TRANSFORMACIÓN DIGITAL

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

	Pág.
INTRODUCCIÓN.....	4.
1. CONTEXTO INSTITUCIONAL Y METODOLÓGICO.....	5
1.1 MARCO DE ARQUITECTURA EMPRESARIAL (MAE).....	5
1.2 EL CATÁLOGO SIS-INF COMO INSTRUMENTO DE GESTIÓN.....	5
2. CRITERIOS DE IDENTIFICACIÓN DE SISTEMAS MONOLÍTICOS.....	5
2.1 VARIABLES DE INFRAESTRUCTURA Y DESPLIEGUE.....	5
2.2 VISIÓN HOLÍSTICA Y PRUEBAS TÉCNICAS.....	5
3. FUNDAMENTACIÓN TÉCNICA DE LA ARQUITECTURA MONOLÍTICA.....	6
3.1 CARACTERÍSTICAS ESTRUCTURALES Y ACOPLAMIENTO.....	6
3.2 DESAFÍOS DE ESCALABILIDAD Y RIESGOS OPERATIVOS.....	6
4. ANÁLISIS DE FRAMEWORKS Y TECNOLOGÍAS HEREDADAS.....	7
4.1 STRUTS 1.X (EL ESTÁNDAR DE JAVA DE INICIOS DE LOS 2000)	7
4.2 WEB FORMS (LA ERA CLÁSICA DE MICROSOFT.NET).	7
4.3 JAVA 6 Y 7 (LOS "MOTORES" DE ESA ÉPOCA).....	8
4.4. LA INTEROPERABILIDAD COMO EVIDENCIA DE AISLAMIENTO.....	8
5. SISTEMAS DE INFORMACION IDENTIFICADOS	9
5.1 SISTEMA DE PLANEACIÓN ESTRATÉGICA - SIGAME.....	9
5.2 GESTIÓN DE SUBSIDIOS DE GLP EN CILINDROS.....	10
5.3 ADMINISTRACIÓN DE SUBSIDIOS Y TRANSPORTE: SISEG Y SITH.....	11
5.4 EL ECOSISTEMA DE PORTALES CMS.....	12
5.5 NEON (EL SISTEMA DE GESTIÓN DE RECURSOS FÍSICOS Y CONTRATACIÓN)....	12

	DOCUMENTO ANALISIS DE APLICACIONES IDENTIFICACION DE ESTRUCTURAS MONOLITICAS		
			X-XX-X-XX
			XX-XX-202X V-X

5.6 AVANZAME..... 12

6. CARACTERIZACIÓN DE OTROS ACTIVOS MONOLÍTICOS..... 13

7. DEBILIDADES COMUNES IDENTIFICADAS..... 14

7.1. EL DESAFIO DE LA INTEROPERABILIDAD (MAE.LI.AI.03).....14


7.2. LA GOBERNABILIDAD DE LOS ACTIVOS DE INFORMACION (MAE.LI.ASI.03).....15

CONCLUSIONES..... 15

ÍNDICE DE TABLAS

Tabla 1: Control de cambios 9

Tabla 2: Ruta de aprobación 9

	DOCUMENTO ANALISIS DE APLICACIONES		
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS		X-XX-X-XX
			XX-XX-202X

INTRODUCCION

La arquitectura de los sistemas de información en las entidades de la administración pública colombiana se fundamenta en los lineamientos establecidos por el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC), específicamente a través del Marco de Arquitectura Empresarial (MAE). Este marco exige que las instituciones modelen, describan y mantengan actualizada su arquitectura de información para habilitar la generación de valor en el desarrollo de su misionalidad. En este contexto, el Catálogo de Sistemas de Información (SIS-INF) permite identificar la estructura fundamental y los principios organizativos de los activos digitales de la entidad.


Dentro de los patrones arquitectónicos identificados en el Ministerio de Minas y Energía, la arquitectura monolítica emerge como una constante histórica en los sistemas legados y de misión crítica. Este modelo se caracteriza por integrar todas las funcionalidades de una aplicación en una única unidad de ejecución, compartiendo recursos, estados y dependencias internas en un solo proceso.

La identificación de sistemas monolíticos a partir del catálogo SIS-INF requiere un análisis multidimensional. Es necesario examinar variables como el sistema operativo, el lenguaje de programación, el tipo de despliegue, el motor de base de datos y las debilidades reportadas por los equipos técnicos.

Los sistemas monolíticos suelen estar atados a versiones antiguas (como Windows Server 2008 o SQL Server 2008). Estas tecnologías no fueron diseñadas para la nube o contenedores, lo que refuerza que el sistema es una pieza única instalada en un servidor específico.

Lenguaje y Despliegue: Si el catálogo menciona un empaquetado "JAVA WAR", significa que todo el código (frontend, backend y reportes) se comprime en un solo archivo. Si una parte falla, todo el sistema se cae, lo cual es la definición técnica de un monolito.

Bajo un enfoque "holístico" (visión integradora) significa que, si un sistema usa Java 7, se despliega como WAR, corre en una versión de Windows antigua y los ingenieros desarrolladores determinan que es difícil de actualizar, tenemos todas las "pruebas técnicas" para clasificarlo como un monolito, incluso si el catálogo no lo dijera explícitamente.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

1. CONTEXTO INSTITUCIONAL Y METODOLOGICO

1.1. MARCO DE ARQUITECTURA EMRESARIAL (MAE)

La gestión tecnológica del Ministerio se rige por el MAE de MinTIC, el cual obliga a las entidades a mantener un inventario detallado de su arquitectura para asegurar que la tecnología sea un habilitador misional. Este marco establece que la arquitectura debe ser flexible para evitar la obsolescencia, un requisito que choca directamente con la naturaleza rígida de los sistemas monolíticos identificados.

1.2. EL CATALOGO DE SIS-INF COMO INSTRUMENTO DE GESTION

El Catálogo de Sistemas de Información (SIS-INF) es el artefacto que permite identificar la estructura fundamental y los principios organizativos de los sistemas de información de la entidad. A través de este, se identifican variables críticas como el sistema operativo, el motor de base de datos y el tipo de despliegue, facilitando un diagnóstico basado en datos técnicos reales.

2. CRITERIOS DE IDENTIFICACION DE SISTEMAS MONOLITOS.


2.1. VARIABLES DE INFRAESTRUCTURA Y DESPLIEGUE

La identificación de un sistema como monolito se basa en evidencias concretas registradas en el SIS-INF:

- **Infraestructura:** Uso de sistemas operativos obsoletos como Windows Server 2008 o motores de base de datos SQL Server 2008, los cuales no fueron diseñados para entornos de nube o contenedores.
- **Empaquetado:** La mención de archivos "JAVA WAR" (Web Application Archive) indica que el frontend, el backend y los reportes están comprimidos en una sola pieza.

2.2. VISIÓN HOLÍSTICA Y PRUEBAS TÉCNICAS.

Un enfoque "holístico" permite clasificar sistemas incluso cuando no están declarados explícitamente como monolitos. Si un sistema combina Java 7, despliegue WAR y una infraestructura antigua, se considera técnicamente un monolito debido a la dificultad inherente para realizar actualizaciones modulares sin comprometer todo el ecosistema.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

3. FUNDAMENTACION TECNICA DE LA ARQUITECTURA MONOLITICA.


3.1. CARACTERÍSTICAS ESTRUCTURALES Y ACOPLAMIENTO

La arquitectura monolítica representa un modelo tradicional de desarrollo de software donde toda la aplicación se construye como una unidad indivisible. Los componentes de interfaz de usuario, lógica de negocio y acceso a datos están estrechamente acoplados y se despliegan conjuntamente en una sola instancia o servidor. En el entorno de las entidades gubernamentales, este patrón ha sido predominante debido a su simplicidad inicial de desarrollo y a la facilidad que ofrece para realizar pruebas integrales y depuraciones centralizadas en etapas tempranas de los proyectos.

A nivel estructural, un monolito se identifica por poseer una base de código única que engloba todas las funcionalidades, una comunicación interna directa mediante llamadas a funciones (en lugar de APIs externas) y, generalmente, una base de datos centralizada compartida por todos los módulos. En el catálogo SIS-INF, la presencia de tecnologías como Java con empaquetado WAR (Web Application Archive), frameworks como Struts 1.x o 2.0, y entornos.NET con Web Forms, constituye evidencia técnica primaria de esta arquitectura.

3.2. DESAFIOS DE ESCALABILIDAD Y RIESGOS OPERATIVOS.

A pesar de sus ventajas en proyectos pequeños o medianos, el modelo monolítico presenta desafíos significativos a medida que los sistemas crecen. La escalabilidad suele estar limitada al crecimiento vertical (aumentar recursos del servidor único), el acoplamiento fuerte dificulta la adopción de nuevas tecnologías y cualquier fallo en un módulo secundario puede comprometer la disponibilidad de toda la aplicación. Estos riesgos son explícitamente reconocidos en las fichas técnicas del Ministerio, donde la transición hacia microservicios se plantea frecuentemente como una oportunidad de mejoramiento evolutivo para evitar la obsolescencia estructural.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

4. ANALISIS DE FRAMEWORKS Y TECNOLOGIA HEREDADAS

El análisis exhaustivo de todas las columnas del catálogo SIS-INF permite identificar patrones transversales que justifican la clasificación de estos sistemas como monolíticos, más allá de haberse identificado explícitamente en el levantamiento de la información.

- **La dependencia de infraestructura heredada (el anclaje de la infraestructura legacy).**

Existe una correlación directa entre la arquitectura monolítica y el uso de sistemas operativos y motores de base de datos antiguos. En su mayoría los sistemas identificados como monolíticos operan sobre versiones de Windows Server 2008 o 2012 y SQL Server 2008 o 2012. Esta dependencia no es casual; los frameworks de la época (Struts 1.x, Web Forms, Java 6/7) fueron diseñados para ejecutarse como servicios de larga duración en servidores dedicados.

Las herramientas de desarrollo que fueron el estándar de la industria aproximadamente entre los años 2000 y 2012. En ese periodo, la prioridad era la rapidez de desarrollo y la centralización, lo que dio origen a la mayoría de los sistemas que hoy consideramos "legados" o monolíticos.

4.1. STRUTS 1.X (EL ESTÁNDAR DE JAVA DE INICIOS DE LOS 2000)


Es uno de los primeros marcos de trabajo (frameworks) para crear aplicaciones web en Java.

- Por qué es monolítico: Obliga a que toda la aplicación se empaquete en un único archivo llamado WAR (Web Application Archive).
- Su limitación: Utiliza un controlador centralizado (ActionServlet) y archivos de configuración XML muy pesados que, al crecer el sistema, se vuelven difíciles de mantener y escalar de forma independiente.

4.2. WEB FORMS (LA ERA CLÁSICA DE MICROSOFT.NET)

Lanzado por Microsoft en 2002, permitía crear páginas web arrastrando controles (botones, tablas) como si fuera una aplicación de escritorio.

- Por qué es monolítico: Existe un acoplamiento total entre el diseño de la página (.aspx) y la lógica de programación (.aspx.cs). No permite separar fácilmente las funciones del sistema en piezas pequeñas.
- Su limitación: Depende de una tecnología llamada ViewState que hace que las páginas sean pesadas y lentas, dificultando su modernización hacia arquitecturas de microservicios.

	DOCUMENTO ANALISIS DE APLICACIONES		
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS		X-XX-X-XX
	XX-XX-202X	V-X	

4.3. JAVA 6 Y 7 (LOS "MOTORES" DE ESA ÉPOCA)

Son versiones de la plataforma Java que hoy se consideran obsoletas (Java 6 salió en 2006 y Java 7 en 2011).

- Contexto de soporte: Oracle finalizó el soporte comercial para estas versiones (2018 para Java 6 y 2022 para Java 7), lo que significa que ya no reciben parches de seguridad oficiales.
- Relación con el Monolito: Estos lenguajes fueron diseñados para ejecutarse en servidores grandes y estables, no en "contenedores" ligeros como Docker. Los sistemas contruidos sobre ellos suelen ser difíciles de migrar porque muchas librerías modernas ya no son compatibles con estas versiones antiguas.

Se identifican como "frameworks de la época" porque comparten tres características críticas:

1. Despliegue Único: Todo el sistema (contabilidad, usuarios, reportes) se sube al servidor como una sola pieza.
2. Escalabilidad Vertical: Si el sistema falla o se pone lento, la única solución es comprar un servidor más grande, no se puede "repartir" la carga en varios servidores pequeños.
3. Dependencia de Infraestructura: Están atados a sistemas operativos antiguos (como Windows Server 2008) que también están llegando al fin de su vida útil.


Esta "herencia de infraestructura" se traduce en una escalabilidad limitada. Como se observa en sistemas como SARA y AUTOGESTIÓN, el despliegue consiste en una "APP empaquetada en WAR, y desplegada en Wildfly, donde la BD se encuentra en el mismo servidor".

Este patrón de "todo en una caja" es la definición física de un monolito, donde la latencia de red es nula pero el riesgo de fallo total es máximo.

4.4. LA INTEROPERABILIDAD COMO EVIDENCIA DE AISLAMIENTO.

La pestaña de 'Catálogo Integraciones' proporciona evidencia indirecta pero contundente sobre la naturaleza monolítica de los sistemas. Los monolitos, al ser unidades cerradas, suelen interoperar mediante tres métodos predominantes identificados en la entidad:

- ETL (Extract, Transform, Load): Utilizado masivamente por el sistema INTÉGRAME para extraer datos de sistemas como SICOM, ANM y ANH. El uso de ETL indica que los sistemas de origen no poseen APIs granulares para compartir información en tiempo real, por lo que se debe "volcar" la base de datos centralizada periódicamente.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X


- Web Services SOAP: Sistemas como ARGO y AVANZAME utilizan SOAP para la radicación de documentos. SOAP es el estándar de comunicación preferido por las arquitecturas monolíticas de la era SOA (Service Oriented Architecture), caracterizado por contratos rígidos y pesados.
- Conexiones Punto a Punto: La integración entre CARGA ME y SIVEEIC mediante correos electrónicos o consultas manuales para autorizar operadores, revela la falta de una capa de servicios desacoplada que permita la comunicación automática entre dominios.

5. SISTEMAS DE INFORMACION IDENTIFICADOS

5.1. ANÁLISIS DEL SISTEMA SIGAME

El sistema de Proceso de Planeación Estratégica - SIGAME, es una herramienta fundamental para la definición de indicadores de gestión, objetivos y planes de acción institucional. Al analizar sus atributos técnicos, se encuentran múltiples evidencias de una arquitectura monolítica:

Columna Analizada	Hallazgo Técnico	Implicación Arquitectónica
ARQUITECTURA	Frontend Web Forms C#, Backend C#,.Net Framework 4.8, IIS.	El uso de Web Forms y.Net Framework 4.8 indica un acoplamiento rígido entre la lógica de presentación y de servidor, típico de los monolitos de Microsoft de la década pasada.
SISTEMA OPERATIVO	Microsoft Windows Server 2012 (64-bit).	Sistema operativo que soporta despliegues de IIS para aplicaciones monolíticas tradicionales.
PLATAFORMA BD	SQLServer 2019.	Base de datos centralizada para todos los módulos (Planeación, Riesgos, Documental, Auditoría).

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

FUNCIONALIDADES	Múltiples módulos integrados (Riesgos, Documental, Auditoría, Plan de mejoras).	La coexistencia de módulos tan diversos en un solo artefacto bajo.NET Framework confir
------------------------	---	--

5.2. SISTEMA SUBSIDIOS GLP EN CILINDROS


Este sistema, permite el registro de subsidios al consumo de GLP en diversas zonas rurales del país. La evidencia de su naturaleza monolítica es explícita y se ve reforzada por sus componentes tecnológicos:

- 5.2.1. En la columna de "arquitectura" se identifica como "Monolito, Python 3.6, Unicorn, Android/Kotlin".
- 5.2.2. El uso de Unicorn como servidor de entrada para una aplicación Python 3.6 indica que el backend funciona como un proceso único que gestiona todas las peticiones de empresas, usuarios y vendedores.
- 5.2.3. Base de Datos unificada, Utiliza Postgres 9.4 como repositorio centralizado para beneficiarios, ventas y registros de empresas.

Aunque el sistema tiene una aplicación móvil (que es lo que ven los vendedores), el "cerebro" que procesa todo en el servidor es un solo bloque de código (escrito en Python 3.6).

- Lo que significa que no hay independencia. Si se quiere actualizar una pequeña función en el registro de beneficiarios, se debe detener, volver a compilar y reiniciar todo el servidor de subsidios.
- Tener una App móvil no significa que el sistema sea moderno; si el backend es indivisible, sigue siendo un monolito "disfrazado" con una cara nueva.

El catálogo menciona que este sistema necesita migrar a servidores nuevos como una debilidad.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

5.3. SISEG Y SITH BASADOS EN JAVA.


- **Identificación Técnica del SISEG (Sistema de Información para la administración del Fondo de Solidaridad para Subsidios y Redistribución de Ingresos de los servicios públicos domiciliarios de energía eléctrica y de gas):** Se define como "Monolito, Java 8, postgres, struts 2.0, jasperreports 8.2, wildfly 15, JAVA WAR". El framework Struts 2.0 es un modelo clásico de arquitectura MVC (Modelo-Vista-Controlador) que organiza toda la aplicación alrededor de un solo controlador que despacha peticiones. El empaquetado en un archivo "JAVA WAR" significa que todos los componentes (librerías de JasperReports, lógica de subsidios del FOES y administración de usuarios) se comprimen en un solo archivo desplegado en el servidor Wildfly.
- **Identificación Técnica del SITH (Sistema de Información para el Transporte de Hidrocarburos) :** Similar al SISEG, utiliza "Java 7, SqlServer, Struts 2.0... JAVA WAR". El uso de Java 7 refuerza la naturaleza legacy del monolito, incrementando los riesgos de seguridad y limitando la capacidad de utilizar librerías modernas. La debilidad de "Falta plan de mantenimiento" y su fecha de construcción en 2016 sugieren un sistema que ha permanecido estático en su arquitectura original.

5.4. EL ECOSISTEMA DE PORTALES CMS

El Ministerio cuenta con varios portales basados en el CMS Wagtail (Python/Django), incluyendo el Portal Web institucional, la Intranet, EITI Colombia y el Buzón de Integridad. Aunque utilizan tecnologías relativamente modernas, el catálogo los identifica sistemáticamente como monolitos.

SISTEMA	SIGLA	EVIDENCIA DE MONOLITO EN EL CATÁLOGO	OPORTUNIDAD DE MEJORA IDENTIFICADA
Portal Web	SEDE ELECTRÓNICA	Esta en arquitectura Monolito.	Cambiar a una arquitectura de Microservicios para evitar el Monolito.
Intranet	INTRANET	Esta en arquitectura Monolito.	Generación de documentación técnica y funcional para mitigar el aislamiento del bloque de código.
EITI Colombia	EITI	Esta en arquitectura Monolito.	Cambiar a una arquitectura de Microservicios para evitar el Monolito.
Buzón de Integridad	LINEA ETICA	Esta en arquitectura Monolito.	Cambiar a una arquitectura de Microservicios para evitar el Monolito.

La clasificación de estos portales como monolitos se sustenta en que el CMS Wagtail/Django agrupa

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

en un solo repositorio y proceso la gestión de contenidos, la lógica de las PQRSD, el Chatbot Indy y la administración de usuarios. Esta estructura centralizada facilita la gestión de contenidos pero crea una dependencia total de la disponibilidad del servidor Nginx 1.21 y la base de datos Postgres 13.7 para todas las funciones del portal. La recurrente recomendación técnica de migrar a microservicios en estos casos es una admisión clara de que la arquitectura actual ha llegado a su límite de escalabilidad y resiliencia.

5.5. NEON (El sistema de Gestión de Recursos Físicos y Contratación).


- Se identifica en el catálogo como "Monolito".
- Opera sobre "Microsoft Windows Server 2008 R2" y "SQL Server 2008". Estas versiones de software base carecen de soporte, lo que hace que el monolito sea vulnerable y difícil de migrar sin una reconstrucción completa de su lógica de negocio, la cual está escrita en "Java antiguo".
- Centraliza la totalidad del ciclo de vida contractual (CDP, Cuentas de cobro, Activos fijos, Almacén, Viáticos). Al ser un monolito, la falla en el módulo de viáticos puede paralizar la gestión de cuentas de cobro de toda la entidad.

La recomendación de "Migración prioritaria de base de datos y sistema operativo" y la necesidad de "Desarrollar/Exponer capa de servicios (API REST)" evidencian que el sistema carece actualmente de interfaces modernas para interoperar, funcionando como un silo de información cerrado.

5.6. AVANZAME

Es el sistema de gestión integral para el seguimiento de proyectos del sector minero-energético. Aunque su ficha técnica muestra el uso de tecnologías modernas como Spring Boot 3.1, mantiene componentes monolíticos heredados que condicionan su operación.


- El sistema integra "Java 8 EE + JSF" para componentes heredados bajo arquitectura MVC junto con servicios RESTful. Esta mezcla indica un "monolito modular" en transición, donde gran parte de la lógica (Contratos, Proyectos, Financiero, Becas, LMS) sigue residiendo en un empaquetado central gestionado por Nginx.
- Se reporta que "El sistema tiene caídas, se debe validar la alta disponibilidad". En una arquitectura distribuida, la caída de un servicio no afectaría a los demás; en Avanzame, la inestabilidad de un componente parece comprometer la sesión de los 720 usuarios a perpetuidad.
- Utiliza una única base de datos Postgres 12 para más de 15 módulos diferentes, lo que genera cuellos de botella en la concurrencia y dificulta el escalado independiente de los tableros de BI frente a la carga de datos de los operadores de red.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

6. OTROS SISTEMAS IDENTIFICADOS COMO MONOLÍTICOS

Siguiendo la revisión del catálogo SIS-INF, se identifican los siguientes sistemas adicionales bajo el patrón monolítico:

ID	Nombre del Sistema	Evidencia Técnica de Monolito
SI_MME_2025_019	Mina del Conocimiento	Definido como "Monolito en la columna de arquitectura. Es un desarrollo interno indivisible para la gestión de manuales y libros institucionales.
SI_MME_2025_020	Aula Virtual (Moodle)	"Monolito Proyecto PHP". Moodle es estructuralmente una aplicación monolítica donde el core, los plugins y la base de datos Postgres 13 operan en un mismo entorno de ejecución.
SI_MME_2025_021	SISEG Energía	"Monolito, Java 8 JSP, Spring... Artefacto War". Comparte componentes comunes con el SISEG Hidrocarburos, reforzando el patrón de monolitos Java de la entidad.
SI_MME_2025_022	Aplicativo de asistencias digitales	"Proyecto Monolito" basado en PHP 7 y Laravel. Aunque usa un framework moderno, el despliegue es una unidad única sin conexión al directorio activo.
SI_MME_2025_023	Fondo de Becas Viejo	"Arquitectura Monolito (Sitio Web, Capa de acceso y Capa de desarrollo)". Utiliza Visual Basic.NET sobre infraestructura obsoleta de 2005.
SI_MME_2025_026	Encuesta de cultura	"Arquitectura Monolito cada componente esta en un docker". Es un ejemplo de "monolito contenedorizado", donde a pesar de usar Docker, el backend Python/Django sigue siendo una pieza única.
SI_MME_2025_030	Normativame	Se infiere monolito por el uso de PHP y base de datos MySQL centralizada, además de tener el código encriptado, lo que impide cualquier desacoplamiento.

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

SI_MME_2025_032	RAIS	"Monolito todo en el mismo servidor". Utiliza React para el frontend pero un backend acoplado en C# ASP.NET Core API que reside físicamente en la misma máquina.
SI_MME_2025_034	SIMINERO	"Monolito" basado en Java j2sdk y Oracle 11G. Es una plataforma histórica para la automatización de trámites mineros.
SI_MME_2025_035	SIGASA	"Monolito" basado en Symfony 3.3 y MySQL. Gestiona conflictos socioambientales como una unidad de software cerrada.
SI_MME_2025_042	ARGO SGDA	"Monolito PHP 8". Basado en el software libre Orfeo, centraliza la gestión documental y correspondencia en un solo bloque ejecutable sobre Apache Tomcat.

7. DEBILIDADES COMUNES IDENTIFICADAS


Al revisar la columna "DEBILIDADES" de forma comparativa, se encuentran términos recurrentes que actúan como indicadores de una arquitectura monolítica:

- **"Falta de plan de mantenimiento"**: Común en SISEG, SITH, GLP y SIGAME. En un monolito, el mantenimiento es una tarea compleja porque un cambio menor requiere testear toda la regresión del sistema.
- **"No se cuenta con el código fuente"**: Identificado en SUNA, SDG y Normativame. Esto confirma que el sistema es un bloque binario inalterable, la forma más extrema de monolito legacy.
- **"Obsolescencia tecnológica"**: Mencionado explícitamente en NEON y sugerido en todos los sistemas que usan Java 7 o Windows 2008.
- **"Arquitectura Monolito"**: En los desarrollos Python se identifica una visión técnica donde el sistema se percibe como una pieza única que debe ser "fortalecida" o "cambiada a microservicios".

La identificación de esta mayoría de sistemas bajo arquitectura monolítica tiene implicaciones directas para el cumplimiento de los lineamientos de MinTIC y el desarrollo institucional.

7.1. EL DESAFIO DE LA INTEROPERABILIDAD (MAE.LI.AI.03).

El Marco de Referencia exige que las entidades diseñen arquitecturas de información que permitan intercambios mediante el uso de lenguajes comunes. Los monolitos identificados, al carecer de una capa de servicios moderna (REST APIs), obligan a la entidad a depender de "integraciones inteligentes" desarrolladas a medida, como la que existe entre **Avanzame y NEON**.¹ Estas integraciones son frágiles;

	DOCUMENTO ANALISIS DE APLICACIONES	
	IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	
	X-XX-X-XX	
	XX-XX-202X	V-X

cualquier cambio en el "monolito financiero" (NEON) puede romper la sincronización de contratos en el "monolito de proyectos" (Avanzame), generando inconsistencias de datos que la entidad debe validar manualmente.¹¹

7.2. LA GOBERNABILIDAD DE LOS ACTIVOS DE INFORMACION (MAE.LI.ASI.03)

La caracterización detallada en el SIS-INF permite al Ministerio tomar decisiones de "Ciclo de Vida". El catálogo muestra una estrategia clara derivada de la arquitectura:

- **Marchitar:** Para monolitos sin código fuente o excesivamente obsoletos (SUNA, SDG, Fondo de Becas Viejo, Normativame).
- **Mejorar/Fortalecer:** Para monolitos críticos que requieren una transición hacia arquitecturas modernas (SISEG, SITH, Portal Web, NEON).
- **Mantener:** Para monolitos estables que cumplen su función específica con bajo riesgo (SARA, KOHA, Autogestión).

CONCLUSIONES

El análisis exhaustivo del "Catálogo SIS-INF" permite concluir que el Ministerio de Minas y Energía posee una arquitectura monolítica que es el estándar dominante, representando más del 80% de los sistemas analizados. Esta situación no es el resultado de una mala planificación, sino de la evolución histórica de sistemas que fueron contruidos para resolver problemas específicos bajo los paradigmas tecnológicos vigentes en su momento.

Las evidencias técnicas encontradas en todas las columnas del catálogo —desde el empaquetado WAR y el uso de frameworks legacy hasta la dependencia de infraestructura obsoleta y la falta de APIs granulares— sustentan de manera irrefutable la clasificación de estos sistemas como monolitos.

La transición hacia microservicios, ya iniciada con sistemas como el VUT, es la respuesta técnica natural para superar las limitaciones de los monolitos identificados. Sin embargo, este proceso debe ser incremental, priorizando aquellos sistemas con alta criticidad misional y mayor deuda técnica, utilizando estrategias de modernización que permitan extraer valor de los monolitos actuales mientras se construye la arquitectura del futuro.


	DOCUMENTO ANALISIS DE APLICACIONES IDENTIFICACION DE ESTRUCTURAS MONOLITICAS		
			X-XX-X-XX
			XX-XX-202X V-X

Tabla 1: Control de cambios

CONTROL DE CAMBIOS		
Versión	Descripción	Fecha
1.0	DOCUMENTO ANALISIS DE APLICACIONES IDENTIFICACION DE ESTRUCTURAS MONOLITICAS	23/02/2026

Tabla 2: Ruta de Aprobación

RUTA DE APROBACIÓN		
ELABORÓ O ACTUALIZÓ: HARVEY GORDILLO SAAVEDRA JEISSON FABIAN PEREZ RODRIGUEZ	REVISIÓN METODOLOGICA:	APROBÓ: