ENTREGABLE 2-B

**MATRIZ DE RIESGO,**

**REGLAS, INDICADORES Y PERFILES DE RIESGOS DE LA DGA**



**Documento de Arquitectura de Gestor FileNet**

**Versión 1.0**

**03 de julio de 2024**

#### Historial de versiones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Elaborado por** | **Revisado por** |
| **1.0** | 03/07/2024 | Creación y revisión del documento | * Percy Silva Paredes | Diego Alexi Gonzales Vasquez |
| **1.1** | 19/11/2024 | Modificación del documento | Javier Rosado |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Tabla de contenido

[1 Introducción 5](#_Toc183081821)

[1.2 Propósito del documento 5](#_Toc183081822)

[1.3 API de gestión de documentos 5](#_Toc183081823)

[2 Gestor FileNet 5](#_Toc183081824)

[2.1 Normativa técnica y arquitectónica 6](#_Toc183081825)

[3 Arquitectura 11](#_Toc183081826)

[3.1 Vista de Arquitectura General 12](#_Toc183081827)

[3.2 Componentes 12](#_Toc183081828)

[3.3 Características Generales 14](#_Toc183081829)

[3.4 Vista de Infraestructura 14](#_Toc183081830)

[3.5 Vista de Implementación 18](#_Toc183081831)

[3.5.1 Microservicios 19](#_Toc183081832)

[3.5.2 Arquitectura de Datos 19](#_Toc183081833)

[3.5.3 Integraciones 20](#_Toc183081834)

[3.5.4 Auditoría 21](#_Toc183081835)

[3.5.4.1 Capa de Datos 21](#_Toc183081836)

[3.5.4.2 Capa de Servicios 21](#_Toc183081837)

[3.5.5 Trazabilidad 21](#_Toc183081838)

[3.5.6 Monitoreo de plataformas 22](#_Toc183081839)

[3.5.7 Gestión de Logs 22](#_Toc183081840)

[3.6 Vista de Despliegue 27](#_Toc183081841)

[4 Patrones usados 28](#_Toc183081842)

[5 Stack Tecnológico de la Arquitectura 29](#_Toc183081843)

[6 Referencias 30](#_Toc183081844)

Índice de tablas

Índice de figuras

[Figura 1 – Diagrama General del Componente 6](#_Toc170375874)

[Figura 2 – Diagrama General de Gestor FileNet 15](#_Toc170375875)

[Figura 3 – Diagrama Clasificación de los Microservicios 24](#_Toc170375876)

[Figura 4 – Diagrama de Auditoria de basado en stack de ELK 26](#_Toc170375877)

[Figura 5 – Diagrama de Implementación de Monitoreo 27](#_Toc170375878)

[Figura 6 – Diagrama de Niveles de Log 29](#_Toc170375879)

[Figura 7 – Diagrama de Implementación de Log Centralizado 31](#_Toc170375880)

[Figura 8 – Diagrama de Despliegue de Gestor FileNet 33](#_Toc170375881)

[Figura 9 – Diagrama de Despliegue Físico de Gestor FileNet 34](#_Toc170375882)

[Figura 10 – Stack Tecnológico de la Arquitectura 36](#_Toc170375883)

# Introducción

Este documento detalla la especificación de la arquitectura del proyecto de Gestor FileNet, usando diferentes vistas para apreciar los diferentes aspectos del sistema.

## Propósito del documento

El presente documento tiene como objetivo mostrar mediante diagramas, modelos y artefactos, las consideraciones técnicas y tecnológicas (plataforma) en la que será implementada la solución. Esbozar los aspectos funcionales de la aplicación, definir los mecanismos de despliegue y distribución del sistema.

## API de gestión de documentos

La VUCE no cuenta con un componente centralizado y reutilizable para la gestión de documentos en la plataforma ECM FileNet.

# Gestor FileNet

El Gestor Filenet es un componente transversal de la Ventanilla Única de Comercio Exterior (VUCE), orientado a facilitar las operaciones relacionadas con la gestión de archivos almacenados en el repositorio de FileNet.

Se adjunta una vista general de lo que abarca la solución.

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

Figura – Diagrama General del Componente

## Normativa técnica y arquitectónica

En este capítulo, se establece la normativa técnica y arquitectónica para el desarrollo y operación de una propuesta de arquitectura en el contexto de la plataforma VUCE 2.0. El propósito es definir los lineamientos y estándares que regirán los aspectos técnicos, de seguridad, integración y rendimiento del sistema, asegurando que cumpla con los requisitos funcionales y no funcionales necesarios para una operación eficiente, segura y escalable. Esta normativa abarca desde la gestión de datos y persistencia hasta la integración de servicios, diseño de interfaz de usuario, eficiencia en el desempeño, disponibilidad y mantenimiento, promoviendo un entorno que facilite la interoperabilidad y la sostenibilidad del componente a largo plazo:

1. **Empleará repositorios de datos definida en la arquitectura patrón para VUCE 2.0.**

* El módulo de Gestor FileNet tendrá un mecanismo de persistencia basado en el estándar JPA para las bases de datos:
  + Transaccional EDB PostgreSQL Enterprise Advanced, en este repositorio de datos se almacenará los atributos y datos de configuración de parámetros.
* La lógica de negocio se implementará en los componentes de backend con microservicios y de capa media, por lo que las reglas de negocio, validaciones y algoritmos como tal serán implementados en diferentes componentes de estas capas, no se contempla el uso de procedimientos almacenados, funciones o programas que ejecuten lógica de negocio en la base de datos.
* Cada microservicio que acceda a la base de datos transaccional tendrá su propio usuario de aplicación y configurado su pool de conexiones. Esto se implementa a través del uso de Spring Boot, que utiliza HikaryCP para la gestión del pool de conexiones; donde, se ha establecido el valor máximo de 10 conexiones por cada microservicio (valor por defecto).

1. **Administración y Monitoreo.**

* El Componente de Gestor FileNet registrará todas las trazas de las operaciones integrándose a la herramienta de logging centralizado en el stack de ELK para que posteriormente se explotará los logs a través del módulo de Kibana para hacer las consultas. Para mayor detalle, revisar la sección 3.5.7 Gestión de Logs del presente documento.
* El Componente de Gestor FileNet implementará el monitoreo de todas las plataformas que reutiliza como Apache Kafka que estará monitoreado utilizando las herramientas de Prometheus y Grafana.
* Se configurará alertas y acciones en Kibana cuando no cumplan las reglas definidas.
* Se configurará dataviews y dashboards en Kibana orientado a equipo de Soporte.
* Se deben monitorear las siguientes métricas para los microservicios de Gestor FileNet:

|  |  |  |
| --- | --- | --- |
| METRICAS | TRAZABILIDAD TECNICA | TRAZABILIDAD DISTRIBUIDA |
| * Memoria: heap usage, non-heap usage, tiempo usado en garbage collection, cantidad de clases cargadas. * Threads: conteo * Thread pools de app server: active, idle. | Log: fecha, tipo, mensaje, traceId (el mismo traceId de la trazabilidad distribuída) | Trazas: id, trace.id, timestamp, duration.ms, name, parent.id, service.name  La versión instrumentada de las aplicaciones tiene: métodos, consultas sql, etc  La propagación del trace.id es automática. |

1. **Portabilidad**

El Componente de Gestor FileNet tendrá la capacidad de ser desplegado en servicios de nube privada que soportes contenedores (Dockers).

Las versiones utilizadas se encuentran en el documento de Arquitectura Patrón, sección 6.7.4 SOFTWARE BASE, ítem e. Listado de Software usado en la arquitectura.

Para el Gestor FileNet la portabilidad de aplicación en nube será lograda a través de uso contenedores empaquetados en Docker file. Los contenedores creados con Docker son altamente portables, lo que permite a los desarrolladores mover aplicaciones entre diferentes entornos y plataformas fácilmente.

1. **Seguridad.**

El Componente de Gestor FileNet deberá tener en cuenta los siguientes requerimientos de seguridad:

* Cumplirá con los 10 riesgos de seguridad más importantes en aplicaciones web según la organización OWASP (Open Web Application Security Project). Se tomará en cuenta en el desarrollo de los componentes los siguientes:

1. A1-Inyección
2. A2-Pérdida de autenticación
3. A3-Exposición de datos sensibles
4. A4-Entidades Externas XML (XXE)
5. A5-Pérdida de Control de Acceso
6. A6-Configuración de Seguridad Incorrecta
7. A7-Secuencia de Comandos en Sitios Cruzados (XSS)
8. A8-Deserialización Insegura
9. A9-Componentes con vulnerabilidades conocidas
10. A10-Registro y Monitoreo Insuficientes.

* Para cumplir este requerimiento se usará el análisis de código estático con SonarQube a todo código fuente desarrollado por El Componente de Gestor FileNet. Se usará el nivel A que es 0 Vulnerabilidades de Seguridad. A continuación, los top 10 de OWASP cubiertos con Sonarqube Developer Edition para las tecnologías usadas Java y JS.
* El Componente de Gestor FileNet asegurará que gestiona contraseñas de forma robusta, no deberán colocar contraseñas dentro del código fuente, en caso de uso de contraseñas en archivos de configuración deberá estar encriptado. En ambiente OCP / Kubernetes se utilizan los Secrets y variables de ambiente.
* El Componente de Gestor FileNet operará bajo los protocolos SSLv3/TLSv1.2 en adelante. Su configuración se debe realizar en el Route de OCP, así como en los componentes de infraestructura como WS02 que exponen los servicios hacia los usuarios finales externos, para lo cual se debe utilizar un certificado SSL de una CA de confianza.
* El Componente de Gestor FileNet deberá tener la capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente. Para ello se establecerá políticas de control de acceso a los datos usando listas de control de acceso, configuración de políticas en el Firewall, segmentación de redes, empleo de matriz de responsabilidades (RACI) para limitar el acceso a recursos sensibles solo a roles previamente autorizados.
* El Componente de Gestor FileNet deberá tener la capacidad de prevenir accesos o modificaciones no autorizados a datos o programas de ordenador. Para ello se implementará control de accesos a los recursos a través de políticas de seguridad en el API Manager y se usará listas de control de acceso en cada componente de la arquitectura.
* Se realizarán pruebas de penetración automatizadas en tiempo de ejecución usando la herramienta OWASP ZAP.

1. **Eficiencia y Desempeño**

El Componente de Gestor FileNet tendrá en cuenta los siguientes requerimientos de eficiencia y desempeño:

* El uso de CPU, RAM y Storage requerido para el funcionamiento será sustentado con pruebas no funcionales. Se revisará los resultados y demostrará que hace un uso eficiente de los recursos. Este componente deberá como mínimo tener la siguiente configuración en la asignación de recursos:
  + ReplicaSet mínimo:3, ReplicaSet máximo: 10
  + CPU:500m RAM: 3GB mínimo (por la gestión documental y su de Buffer)
* Para mejorar el desempeño de la solución se van a implementar las siguientes tácticas:
  + Comunicación asíncrona: Para los escenarios más transaccionales la comunicación y donde los documentos tiene mayor peso en MB serán utilizando el protocolo TUS y el uso de Apache Kafka. Al usar este estilo de comunicación entre diferentes componentes, lo que se busca es evitar los bloqueos al realizar peticiones, mejorando la experiencia del usuario.
  + Escalabilidad: se configurará los pods de los microservicios con escalabilidad automática horizontal HPA que es componente del gestor de contenedores a nivel de alarmas de RAM y CPU. Estos valores serán monitoreados, revisados y ajustados con los resultados de las pruebas no funcionales (rendimiento y estrés).
  + Eficiencia: se configurará pools de conexiones a la base de datos por cada microservicio de alta concurrencia con valores máximos de hasta 100 conexiones por cada pod para soportar un escenario de 600 transacciones por segundo. El framework usado para microservicios Springboot usa por defecto el Pool de Conexiones Hikari porque es el que mejor rendimiento tiene comparado con Tomcat JDBC y otras implementaciones.
* Para asegurar que se cumpla con estos requerimientos se realizaran pruebas no funcionales de rendimiento, estrés y carga y se entregará los SLA.
* Se realizará un análisis de código estático de cada componente con la herramienta Sonarqube, que asegurará que el código fuente cumpla las métricas de calidad definidas en el documento “Devops - Metricas e Indicadores de Calidad SonarQube.docx”.

1. **Disponibilidad y Confiabilidad**

El Componente de Gestor FileNet tendrá en cuenta los siguientes requerimientos de disponibilidad:

* Tendrá una alta disponibilidad, la misma que deberá ser 24\*7 los 365 días del año. Este tiempo es válido con excepción de los periodos de mantenimiento que serán comunicados oportunamente a los usuarios vía notificación. La alta disponibilidad deberá ser validada con la actual infraestructura tecnológica proporcionada. El Componente de Gestor FileNet deberá operar según lo previsto aún en presencia de fallos de hardware o software, todos los elementos de la aplicación deberán estar en alta disponibilidad y balanceo a nivel del software.
* Para el cumplimiento de este requerimiento no funcional serán implementados:
  + Para Disponibilidad: Se utilizará el despliegue de los componentes de la aplicación en contenedores para tener elasticidad (escalabilidad horizontal) y de esta forma las diferentes funcionalidades puedan ser resilientes durante un instante o a lo largo de un periodo determinado. También se desplegará en 2 clusters de contenedores (1 por cada Data Center) en modo activo – pasivo y capa de datos también usará replicación de datos. Se monitoreará la disponibilidad de los componentes de la plataforma y se configurará alertas que notifiquen a los responsables cuando se supere el umbral de disponibilidad (99% para componentes de VUCE).
  + Para confiabilidad: se utilizará el patrón SAGA mediante el cual por medio de eventos se asegura que el sistema realice las transacciones correctamente ya sea mediante coreografía u orquestación, garantizando la coherencia de los datos. También se implementará el Patrón Circuit Breaker con reintentos en los microservicios. Se monitoreará el porcentaje de transacciones exitosas y con error en un intervalo de tiempo, y cuando se supere el umbral de errores se notificará a los responsables.
  + Se coordinará con los proveedores de las plataformas la ejecución de pruebas de alta disponibilidad de los componentes implementados y se revisará los resultados de las pruebas.

1. **Mantenibilidad**

El Componente de Gestor FileNet cumplirá con los siguientes requerimientos de Mantenibilidad:

* Será modular, una actualización en uno de los módulos no deberá tener impacto en los otros. Del mismo modo una caída de un modelo no deberá afectar a los otros. Esto se logra, debido a que la comunicación entre microservicios se realiza a través de interfaces definidas, utilizando protocolos ligeros como HTTP o REST.
* Se implementará las funcionalidades con arquitectura de microservicios, que estarán diseñados como servicios independientes, autónomos y ligeros, por lo que cada uno de ellos puede ser desplegado y modificado sin afectar otros aspectos esenciales de la aplicación. De hecho, cada módulo cuenta con su propia base de datos (es decir, no acuden todos a la misma), lo cual reduce mucho el margen de sobrecarga y la caída de la aplicación.
* Todos los componentes back end contarán con 80% de cobertura de las pruebas unitarias.
* Los componentes generarán un LOG con diferentes niveles de visualización de modo que se pueda efectuar un seguimiento al flujo de cualquier módulo e identificar algún punto de error o inconsistencia. Todo error será registrado en el repositorio centralizado de logs basado en el componente stack ELK.
* Para asegurar la mantenibilidad, el código fuente de Gestor FileNet será revisado con la herramienta Sonarqube, que cumplirá las siguientes métricas de calidad definidas en el documento “Devops - Metricas e Indicadores de Calidad SonarQube.docx”.

|  |  |
| --- | --- |
| **INDICADOR** | **VALOR** |
| Código Repetido | < 1% |
| Documentación del Código | > 20% |
| Cobertura | > 80% |
| Cumplimiento de Reglas | > 90% |
| Mantenibilidad | A |
| Deuda Técnica | 4 h |
| Seguridad | A |

# Arquitectura

En esta sección se detalla los componentes que forman parte de la solución de Gestor FileNet.

## Vista de Arquitectura General

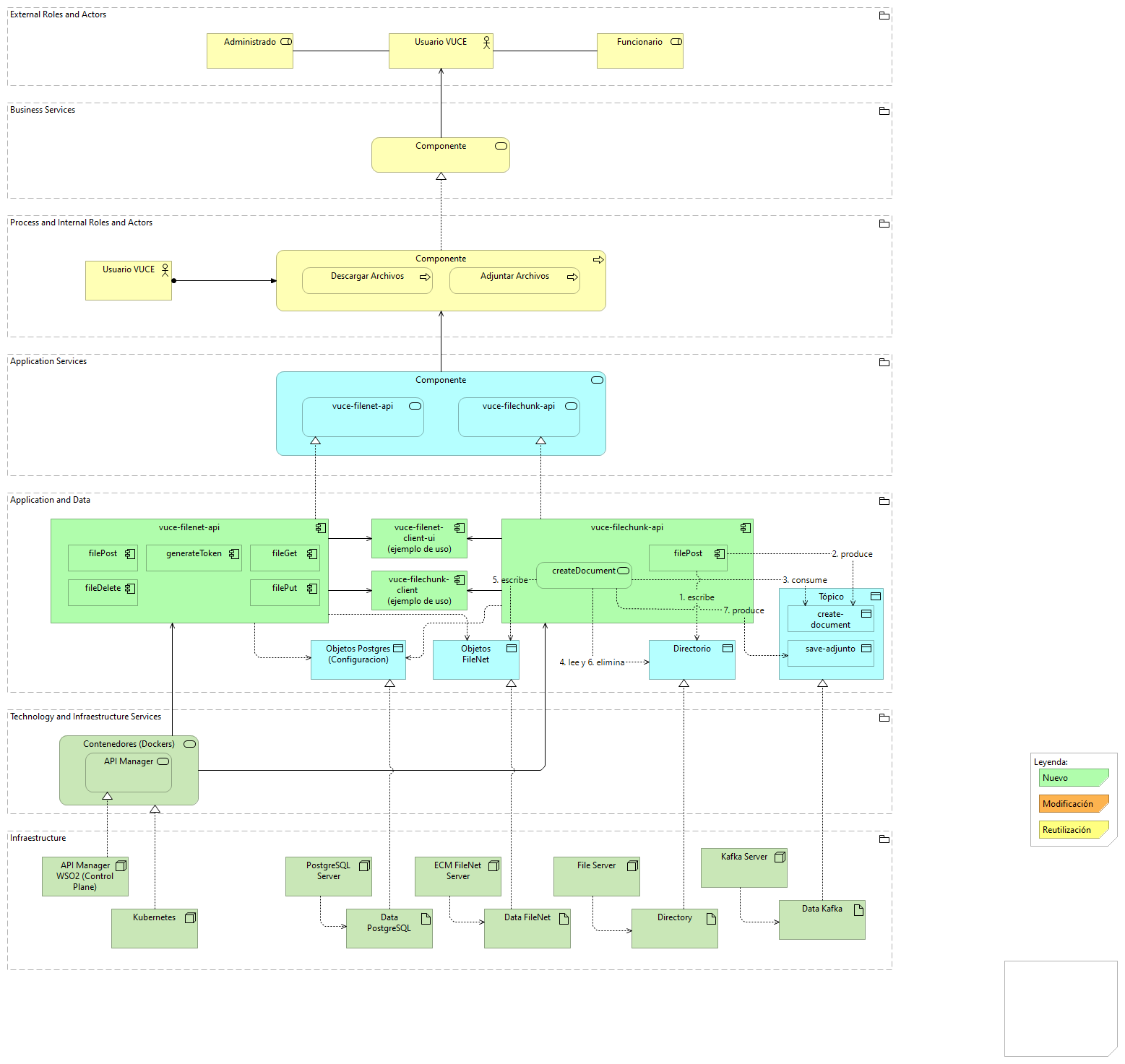


Figura – Diagrama General de Gestor FileNet

## Componentes

Se describe los componentes de la solución agrupados por área, como se detalla a continuación:

1. **Base de datos PostgreSQL**, en este repositorio de datos se almacenará las bases de datos, tablas, atributos del modelo físico de datos para los documentos.
2. **ECM FileNet de IBM,** es un producto que permite la gestión de contenidos, para este caso utilizado para poder grabar, leer y eliminar los archivos que requiera los diferentes procesos del nuevo componente.
3. **Contenedores**, los componentes estarán orientado a la arquitectura de microservicios y contenerizados, dado que estos artefactos serán desplegados en el gestor de contenedores de OpenShift o Kubernetes.
4. **Herramientas,** a continuación, se lista una relación de herramientas que serán usados por el nuevo componente Buzón Electrónico:
   1. **Apache Kafka**, usado para el servicio de gestión de colas, productores y subscriptores, cualquier evento que se requiere desacoplar deberá de usar este componente para convertir un proceso síncrono en asíncrono.

El listado de Software y sus versiones se encuentran establecidos en el Documento de Arquitectura Patrón, sección 6.7.4 SOFTWARE BASE, ítem e. Listado de Software usado en la arquitectura.

1. **Microservicios de BackEnd,** la implementación del backend al estar basado en estilo arquitectónico de microservicios, esto facilita las tareas y disminuye los tiempos de implementación, pruebas y despliegue, en comparación a una solución monolítica. Los microservicios implementarán una persistencia poliglota, es decir la información puede persistir en diferentes mecanismos (bases de datos relacionales, no relacionales, en memoria). Deberán de basarse en la documentación de los lineamientos de desarrollo y Stack Tecnológico que tiene la VUCE 2.0.

A continuación, se lista los dominios de microservicios identificados:

* vuce-filenet-api: microservicio que se encargará de exponer métodos el registro, actualización, eliminación y obtención de documentos de FileNet de IBM, e internamente para obtener datos relacionados a configuración en la BD Postgresql.
* vuce-filechunk-api: microservicio que se encargará de exponer métodos para el registro de documentos de gran tamaño, a través de su recepción en un conjunto de partes de menor tamaño, tales que, utilizando un algoritmo le permitirá consolidar el documento y almacenarlo en FileNet de IBM; e internamente para obtener datos relacionados a configuración en la BD Postgresql. Este microservicio tiene la particularidad de utilizar el protocolo TUS (<https://tus.io/>)

## Características Generales

Se describe algunas características generales de la implementación del Componente de Gestor FileNet:

* Mejorar la disponibilidad de la plataforma: Al tener componentes independientes, en caso de fallo o mantenimiento de un componente, los demás elementos pueden seguir trabajando sin inconvenientes.
* Facilidad en la evolución de la plataforma: Se pueden incorporar nuevas tecnologías tanto en el back como en la persistencia, minimizando el impacto sobre los componentes ya desarrollados.
* Facilitar el mantenimiento: Al tener más componentes independientes que cumplen con responsabilidades específicas, los ajustes se hacen sobre elementos puntuales, disminuyendo la interdependencia y el acoplamiento de componentes.
* Mejora la escalabilidad: La plataforma está desplegada sobre componentes que pueden escalar en tiempo de ejecución, acorde con las capacidades de infraestructura con las que cuenta VUCE.
* Integraciones desacopladas: Las integraciones a servicios provistos por terceros se van a desacoplar con el fin de evitar cambios en la plataforma VUCE cuando un tercero modifique algún aspecto de su servicio, para ello se hará uso de adaptadores que ocultaran la complejidad del acceso a los servicios externos.
* Alargar el ciclo de vida de la aplicación al utilizar tecnologías de punta, que permite la evolución de la solución en el tiempo.
* La aplicación es distribuida, conformada por muchos componentes interdependientes que se despliegan de forma independiente.
* Está basada en microservicios que exponen interfaces que usan de estándares de comunicación como HTTP, REST, y basada en mensajería como Apache Kafka, los componentes pueden ser consumidos y reutilizados por otros microservicios y aplicaciones sin acoplarse a través de lenguajes de programación o librerías compartidas.
* La elasticidad es un punto clave en la solución, dado que al estar basada en microservicios y al estar desplegada en la plataforma de contenedores, aprovecha las ventajas de esta para crecer o decrecer en cantidad de nodos en tiempo real.
* Soporte con DevOps reducirá los tiempos de despliegue al automatizar tareas de ensamblaje, configuración y despliegue de la solución.

## Vista de Infraestructura

A continuación, se describe los ambientes de infraestructura donde estará ejecutándose los componentes de la solución:

**Ambiente de Desarrollo**

|  |  |
| --- | --- |
| Componentes | |
| |  |  | | --- | --- | | Plataforma de Contenedores en Openshift | | | Consola Web | <https://console-openshift-console.apps.desarrollo.vuce.gob.pe> | | API Clúster | <https://api.desarrollo.vuce.gob.pe:6443> | | Servidor de Imágenes | <https://default-route-openshift-image-registry.apps.desarrollo.vuce.gob.pe> | | URL Ingress | <http://router-default.apps.desarrollo.vuce.gob.pe> | |  |  | | |
| |  |  | | --- | --- | | WSO2 API Manager | | | Portal Publicador | <https://control-plane-apim-desa.vuce.gob.pe/publisher/apis> | | Portal Developer | <https://control-plane-apim-desa.vuce.gob.pe/devportal/apis> | | Dominio gateway | gateway-apim-desa.vuce.gob.pe | |  |  | | |
| |  |  | | --- | --- | | Base de datos en Postgres EDB | | | Dirección IP | 192.168.80.20 | | Puerto | 5444 | |  |  | | |
| |  |  | | --- | --- | | Base de datos en Cache REDIS | | | Dirección IP | 10.19.34.183 | | Puerto | 6379, 26379 | |  |  | | |
| |  |  | | --- | --- | | Apache Kafka | | | Dirección IP | 172.23.12.27 | | Puerto | 9200 | | Consola Monitoreo | <http://172.23.12.27:9000/> | |  |  | | |
| |  |  | | --- | --- | | FileNet IBM | | | Consola Web | <http://192.168.120.66:9080/acce/> | | Nombre del Dominio | VUCEDEV | | Nombre del Almacén de Datos | STRGDEV | |  |  | | |
| Jenkins | Consola Web: <http://172.23.12.29:8080/login> |
| GitLab | Consola Web: <http://10.19.34.184/users/sign_in> |
| JFrog Artifactory | Consola Web: <http://10.19.34.189:8082/ui> |
| Sonarqube | Consola Web: <http://172.23.12.34:9000/projects> |

**Ambiente de Certificación**

|  |
| --- |
| Componentes |
| |  |  | | --- | --- | | Plataforma de Contenedores en Openshift | | | Consola Web | <https://console-openshift-console.apps.certificacion.vuce.gob.pe> | | API Clúster | <https://api.certificacion.vuce.gob.pe:6443> | | Servidor de Imágenes | <https://default-route-openshift-image-registry.apps.certificacion.vuce.gob.pe> | | URL Ingress | <http://router-default.apps.certificacion.vuce.gob.pe> | |  |  | |
| |  |  | | --- | --- | | WSO2 API Manager | | | Portal Publicador | <https://control-plane-apim-test.vuce.gob.pe/publisher/apis> | | Portal Developer | <https://control-plane-apim-test.vuce.gob.pe/devportal/apis> | | Dominio gateway | gateway-apim-test.vuce.gob.pe | |  |  | |
| |  |  | | --- | --- | | Base de datos en Postgres EDB | | | Dirección IP | 192.168.140.23 | | Puerto | 5444 | |  |  | |
| |  |  | | --- | --- | | Base de datos en Cache REDIS | | | Dirección IP | 10.19.34.201 , 10.19.34.202 , 10.19.34.203 | | Puerto | 6379, 26379 | |  |  | |
| |  |  | | --- | --- | | Apache Kafka | | | Dirección IP | 10.19.34.207 | | Puerto | 9200 | | Consola Monitoreo | <http://10.19.34.207:9000/> | |  |  | |
| |  |  | | --- | --- | | FileNet IBM | | | Consola Web | <http://192.168.120.67:9080/acce/> | | Nombre del Dominio | VUCEQA | | Nombre del Almacén de Datos | STRGQA | |  |  | |

**Ambiente de Producción**

|  |
| --- |
| Componentes |
| |  |  | | --- | --- | | Plataforma de Contenedores en Openshift | | | Consola Web | <https://console-openshift-console.apps.produccion.vuce.gob.pe> | | API Clúster | <https://api.produccion.vuce.gob.pe:6443> | | Servidor de Imágenes | <https://default-route-openshift-image-registry.apps.produccion.vuce.gob.pe> | | URL Ingress | <http://router-default.apps.produccion.vuce.gob.pe> | |  |  | |
| |  |  | | --- | --- | | WSO2 API Manager | | | Portal Publicador | <https://control-plane-apim.vuce.gob.pe/publisher/apis> | | Portal Developer | <https://control-plane-apim.vuce.gob.pe/devportal/apis> | | Dominio gateway | gateway-apim.vuce.gob.pe | |  |  | |
| |  |  | | --- | --- | | Base de datos en Postgres EDB | | | Dirección IP | 192.168.110.36 | | Puerto | 5444 | |  |  | |
| |  |  | | --- | --- | | Base de datos en Cache REDIS | | | Dirección IP | 10.19.34.216, 10.19.34.217, 10.19.34.218 | | Puerto | 6379, 26379 | |  |  | |
| |  |  | | --- | --- | | Apache Kafka | | | Dirección IP | 10.19.34.140 alias host: atlegacy-kafka-zookeeper1.vuce.gob.pe | |  | 10.19.34.141 alias host: atlegacy-kafka-zookeeper2.vuce.gob.pe | | Puerto | 9200 | | Consola Monitoreo | <http://10.19.34.140:9000/> | |  |  | |
| |  |  | | --- | --- | | FileNet IBM | | | Consola Web | <http://192.168.90.16:9080/acce/> | | Nombre del Dominio | VUCEPROD | | Nombre del Almacén de Datos | STRGPROD | |  |  | |

## Vista de Implementación

Los microservicios serán implementados siguiendo una clasificación lógica, como se puede apreciar en el diagrama de uso general indicado a continuación:

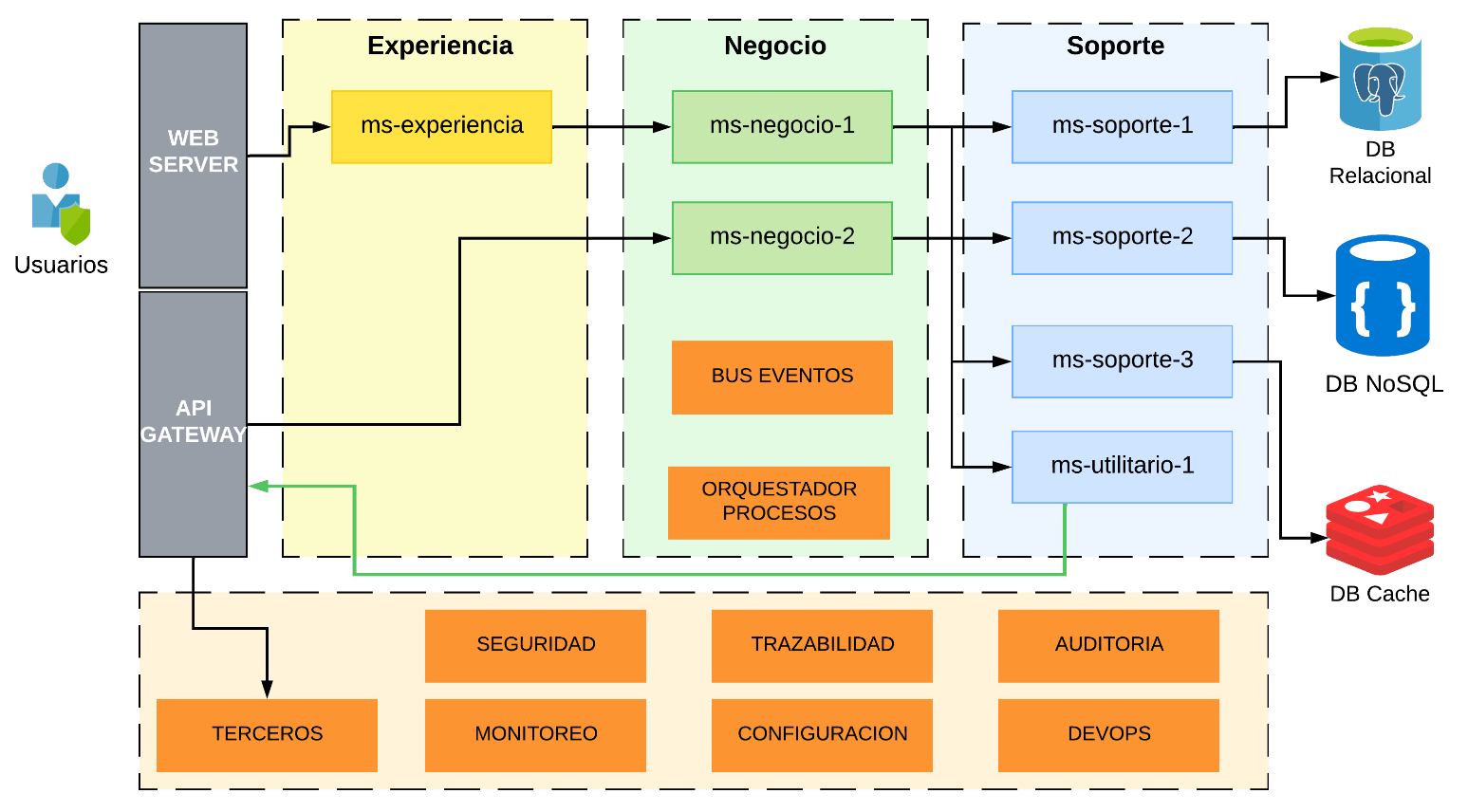


Figura – Diagrama Clasificación de los Microservicios

## Microservicios

Los microservicios serán implementados según su clasificación lógica estándar según aplique:

* **Microservicio de Experiencia:** Maneja la lógica de presentación y validaciones específicas de los datos que son propia a cada consumidor que utilizará algún recurso. Por ejemplo, si tenemos un api del recurso usuarios que sirve a un canal web, posiblemente este canal necesite información distinta a un canal móvil que consuma el mismo recurso y su lógica funcional o técnica sea distinta.
* **Microservicio de Negocio:** Implementan una funcionalidad transversal del negocio. Los microservicios de negocio tienen la responsabilidad de exponer información relacionada con las entidades de negocio e implementar la lógica de negocio que necesite uno o más canales de negocio. Tienen la capacidad de realizar orquestación sobre otros microservicios de negocio o de sistema. Encapsulan la funcionalidad reutilizable usando el lenguaje ubicuo (DDD) por cada contexto limitado. Soportar los eventos que ocurran en cada contexto.
* **Microservicio de Sistema o Soporte:** No son estratégicos para el negocio. Tiene implementaciones técnicas del sistema principal como acceso a datos y integraciones internas o externas. Para la presente arquitectura los microservicios vuce-filenet-api y vuce-filechunk-api son de tipo soporte.

## Arquitectura de Datos

A continuación, se presentan lineamientos y buenas prácticas que deben ser utilizados para la arquitectura de datos:

**1. Definir objetivos claros:**

Antes de comenzar a diseñar la arquitectura de datos, es fundamental definir claramente los objetivos que se pretenden alcanzar. ¿Qué problemas se buscan resolver con los datos? ¿Qué información se necesita para respaldar las decisiones estratégicas del negocio? ¿Cómo se utilizarán los datos para generar valor?

**2. Comprender los datos:**

Es esencial comprender la naturaleza, el volumen, la variedad y la velocidad de los datos que se gestionarán. Se debe realizar un análisis exhaustivo de los datos existentes, identificando sus fuentes, formatos, estructuras y calidad.

**3. Modelar los datos:**

Se debe crear un modelo de datos que represente la estructura lógica de los datos, incluyendo entidades, atributos y relaciones entre ellas. El modelo de datos debe ser preciso, consistente y alineado con los objetivos de la organización. Utilizar herramientas de modelado de datos como ERWin, PowerDesigner, o similar. En el documento técnico de Buzón Electrónico2.0, se presentará las especificaciones técnicas del modelo de datos.

**4. Seleccionar las tecnologías adecuadas:**

La elección de las tecnologías de almacenamiento, procesamiento y análisis de datos debe basarse en las necesidades específicas de la organización, considerando factores como el volumen de datos, el rendimiento requerido, la escalabilidad y la seguridad. En el documento de Arquitectura Patrón, en la sección 6.3.1. CAPAS DE ARQUITECTURA DE APLICACIONES, ítem f. Capa de Datos, se presenta las herramientas recomendadas y sus características técnicas.

**5. Implementar un enfoque modular:**

La arquitectura de datos debe diseñarse de manera modular, dividiéndola en componentes independientes y reutilizables. Esto facilita la implementación, el mantenimiento y la evolución de la arquitectura a medida que cambian las necesidades del negocio.

**6. Garantizar la seguridad y la calidad de los datos:**

La seguridad de los datos es un aspecto fundamental de la arquitectura de datos. Se deben implementar medidas de seguridad adecuadas para proteger los datos contra accesos no autorizados, alteraciones y pérdidas. La calidad de los datos también debe ser monitoreada y gestionada para garantizar su confiabilidad y precisión.

**7. Documentar la arquitectura:**

Es importante documentar la arquitectura de datos de manera clara y completa, incluyendo diagramas, descripciones detalladas de los componentes y tecnologías utilizadas, y procesos de gestión de datos. La documentación facilita la comprensión de la arquitectura por parte de diferentes interesados y permite realizar cambios y actualizaciones de manera eficiente. En el documento técnico de Buzón Electrónico2.0, se presentará las especificaciones técnicas del modelo de datos.

## Integraciones

Se presenta dos escenarios de integración:

* **Integración con fuentes internas:** Cuando un microservicio se comunica con un servicio interno expuesto en el API Manager, primero obtendrá un JWT Token a través del API Manager y posteriormente se consumirá el servicio interno. Para el presente caso, los microservicios son altamente transaccional y por el Payload que gestiona en cada petición (por el peso de los documentos), se evitará que sea expuesto en APIM para su consumo.

## Auditoría

## Capa de Datos

* Se agregarán campos a las tablas para registrar el usuario y fecha de creación, usuario y fecha de modificación en la Base de Datos PostgreSQL.

## Capa de Servicios

* A nivel de auditoría de aplicaciones, se implementará usando el esquema actual de Logging centralizado basado en el stack ELK, donde se almacenará todos los registros de auditoría y se explotarán usando un Dashboards de Auditoria en Kibana.

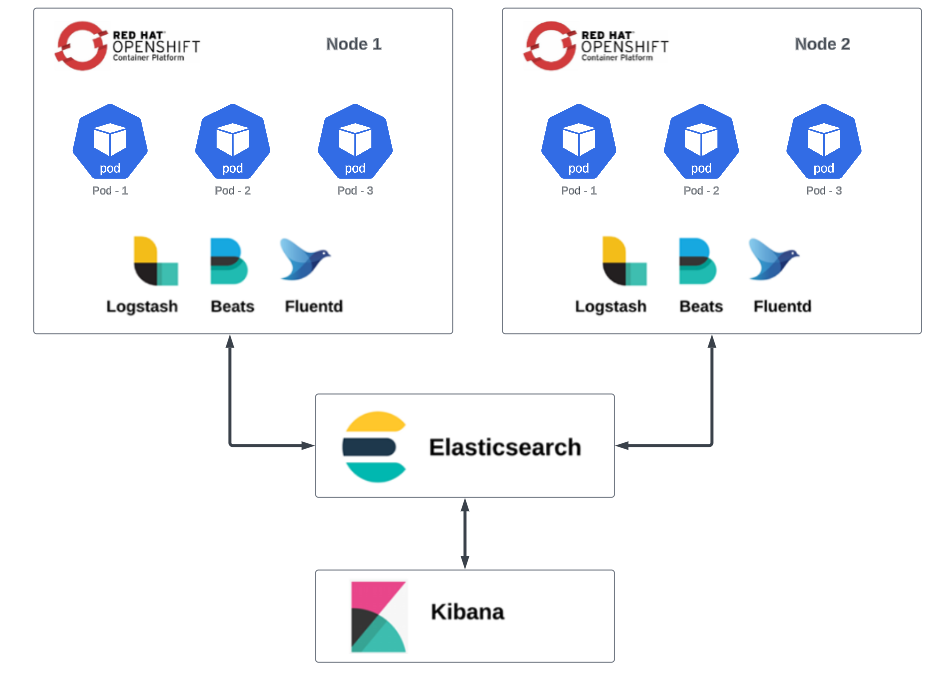


Figura – Diagrama de Auditoria de basado en stack de ELK

## Trazabilidad

La trazabilidad distribuida permite una manera de seguir y entender el comportamiento de las peticiones de manera individual al tiempo que atraviesan nuestras conexiones de microservicios. Estos seguimientos nos permiten comprender las dependencias de los servicios, localizar errores, detectar posibles latencias durante las peticiones y mejorar el rendimiento. En definitiva, las trazas distribuidas nos permiten estudiar una petición desde su punto origen hasta su destino final al tiempo que es observada por cada uno de los servicios con los que interactúa. Esta información se explotará en la herramienta ELK

## Monitoreo de plataformas

Para el monitoreo se usará los servicios de infraestructura basado en Prometheus y Grafana,que son herramientas para monitorizar y supervisar clúster de contenedores, microservicios, servicios y fuentes de datos. Se monitoreará y configurará alertas para los siguientes componentes:

* Apache Kafka

## Gestión de Logs

En una arquitectura de Microservicios, tener los registros de aplicaciones o Logs en archivos representan una tarea muy tediosa, ineficiente y trae consigo problemas de control, de búsqueda y limpieza, además de saber que en los contenedores los archivos guardados son destruidos con la eliminación del contenedor. Aun teniendo los archivos de Logs externalizados a los contenedores, seguirían existiendo problemas para realizar búsquedas y realizar métricas de los datos almacenados.

Para solventar el problema de tener muchos archivos de Logs, se optó por realizar una solución centralizada a través de herramientas con la suite de Elastic, que contiene tres productos claves en la solución: ElasticSearch, Logstash y Kibana, comúnmente llamado a éstos en conjunto como ELK, por sus iniciales.

Para los eventos del Componente de Gestor FileNet se tiene en cuenta la siguiente clasificación de la información a registrar:

* Eventos de Seguridad: Son aquellos registros que se enfocan en eventos de detección de accesos no autorizados e inicios de sesión, tenemos los eventos registrados en el Servicio de Autenticación.
* Eventos de Componentes: Son aquellos registros que se producen para proveer información útil en la ejecución de los componentes de la Arquitectura tales como Servidor de Base de Datos, Servidor de Gestor de Colas, Servidor de Cache.
* Eventos de Transacciones e Información Funcional: Son aquellos registros que se producen para proveer información de las operaciones de la capa de negocio.

Para saber exactamente qué es lo que se requiere monitorear es necesario clasificar los logs en cuanto a su severidad e importancia se pueden observar los detalles que se exponen en el cuadro.

|  |  |  |
| --- | --- | --- |
| **Código** | **Severidad** | **Descripción** |
| **ERROR** | Errores | Excepciones del Sistema, que afectan su rendimiento o la disponibilidad de funcionalidades críticas. |
| **WARN** | Advertencias | Comportamientos atípicos del Sistema, cambios en la configuración, etc. |
| **DEBUG** | Depuración | Información de seguimiento de nuevas funcionalidades, o para evaluar el comportamiento de operaciones críticas del sistema. |
| **INFO** | Información | Información relacionada a requerimientos no funcionales, como eventos de reinicio, inventario de componentes disponibles. |

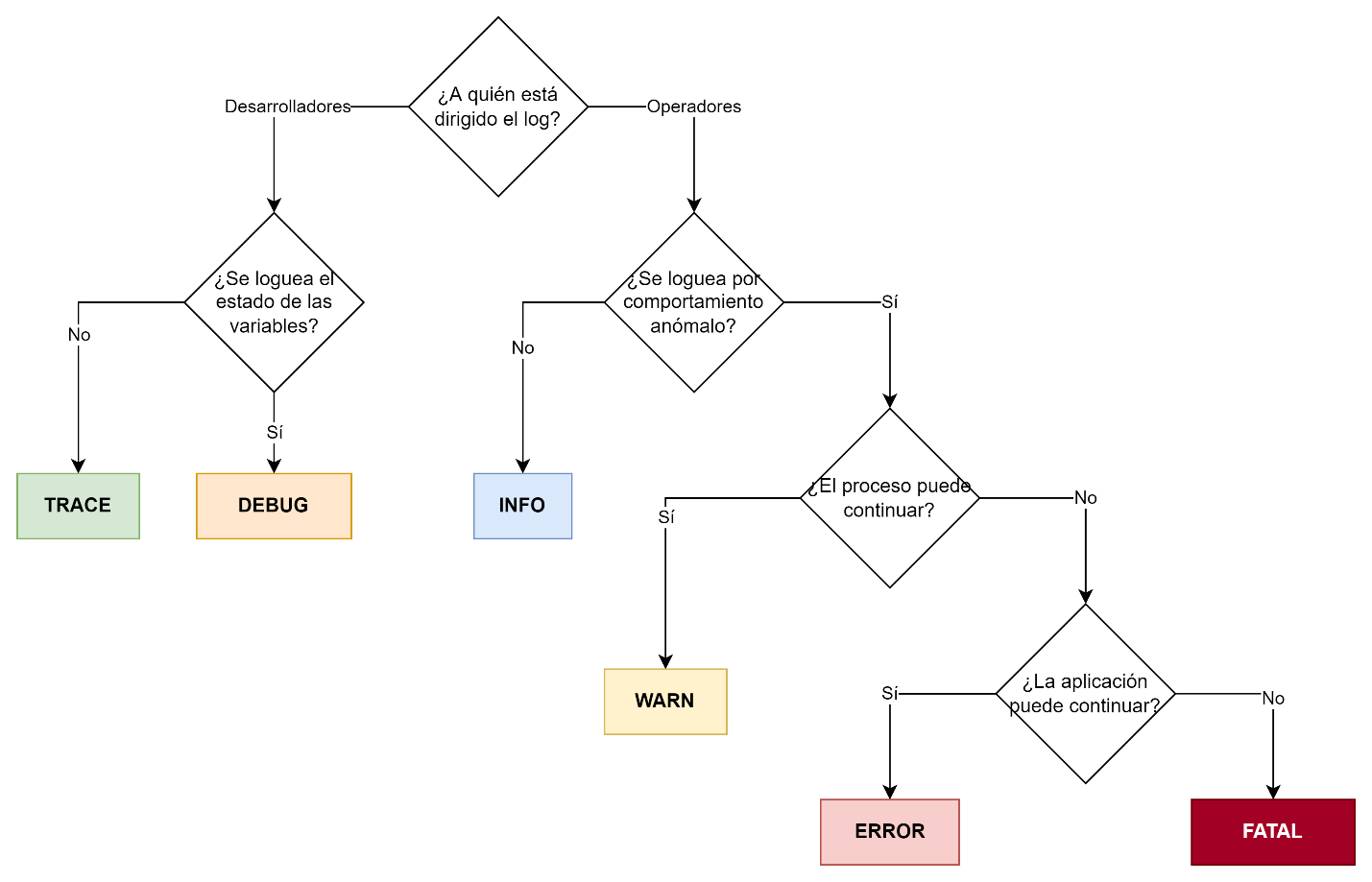


Figura – Diagrama de Niveles de Log

El Instituto Nacional de Estándares y Tecnología de los Estados Unidos (NIST), define, en su publicación SP-800-92, la gestión de logs como “el proceso para generar, transmitir, almacenar, analizar y desechar los datos de eventos del sistema”. Esta definición se puede generalizar a cualquier log generado por un sistema o aplicación. A continuación, se presenta la estructura propuesta para el registro de mensajes:

| **ATRIBUTO** | **TIPO** | **DESCRIPCION DEL ATRIBUTO** |
| --- | --- | --- |
| transaccion\_id | Carácter | Identificador único de registro de log. Si no se proporciona por la aplicación origen, se genera uno en Formato UUID. |
| evento\_id | Carácter | Identificar único del evento generado por el usuario. Si no se proporciona por la aplicación origen, se genera uno en Formato UUID. |
| origen | Carácter | Cuando el mensaje a registrar se genera a partir del intercambio de información entre dos entidades, se requiere identificar cuál es la entidad consumidora del servicio. |
| timestamp | Fecha | Fecha de registro de Log Formato Fecha Hora y milisegundos. YYYY-MM-DD HH24: MI: SS. |
| usuario\_id | Carácter | Identificador del usuario que realizo el evento. |
| usuario\_ip | Carácter | Dirección IP del host en donde se origina el evento |
| tipo | Carácter | Tipo de mensaje de log. Código que permite identificar los tipos de mensajes. |
| metodo | Carácter | Si es una petición los métodos GET, POST, PUT, DELETE. |
| nivel | Carácter | Nivel de Severidad de Log. Ej. INFO, ERROR. |
| componente | Carácter | Nombre del Componente que registro el evento. En el caso de microservicio sería el nombre de la clase que implementa la lógica. |
| version | Numérico | Versión del Componente que registra el evento. |
| operacion | Carácter | Nombre de la operación que registro el evento. En el caso de microservicio sería el nombre del método. |
| host | Carácter | Nombre del Hostname o IP del servidor del componente que registro el evento. |
| usuario\_agente | Carácter | Nombre del agente navegadores usados por el usuario. |
| session\_id | Carácter | Identificador de la sesión generada para el usuario. |
| app\_name | Carácter | Nombre de la aplicación donde se registra el evento. Puede ser un microservicio, keyloack, bus de eventos. |
| mensaje | Carácter | Contenido de los datos del Log. En formato JSON |
| estado | Numérico | Estado de la respuesta. Ejemplo 200 respuesta exitosa, <> 200 respuesta fallida. |

Para implementar el esquema de log centralizado usando la suite de Elastic, se usarán agentes filebeat que se instalaran en los componentes de la Arquitectura de Gestor FileNet, estos agentes recopilan información de los logs generados y lo envían de forma asíncrona al Logstash. El componente de Logstash parseara los formatos necesarios y lo almacenara en el repositorio de ElasticSearch. Finalmente, los datos serán explotados y analizados a través de los Dashboards creados en Kibana por los usuarios.

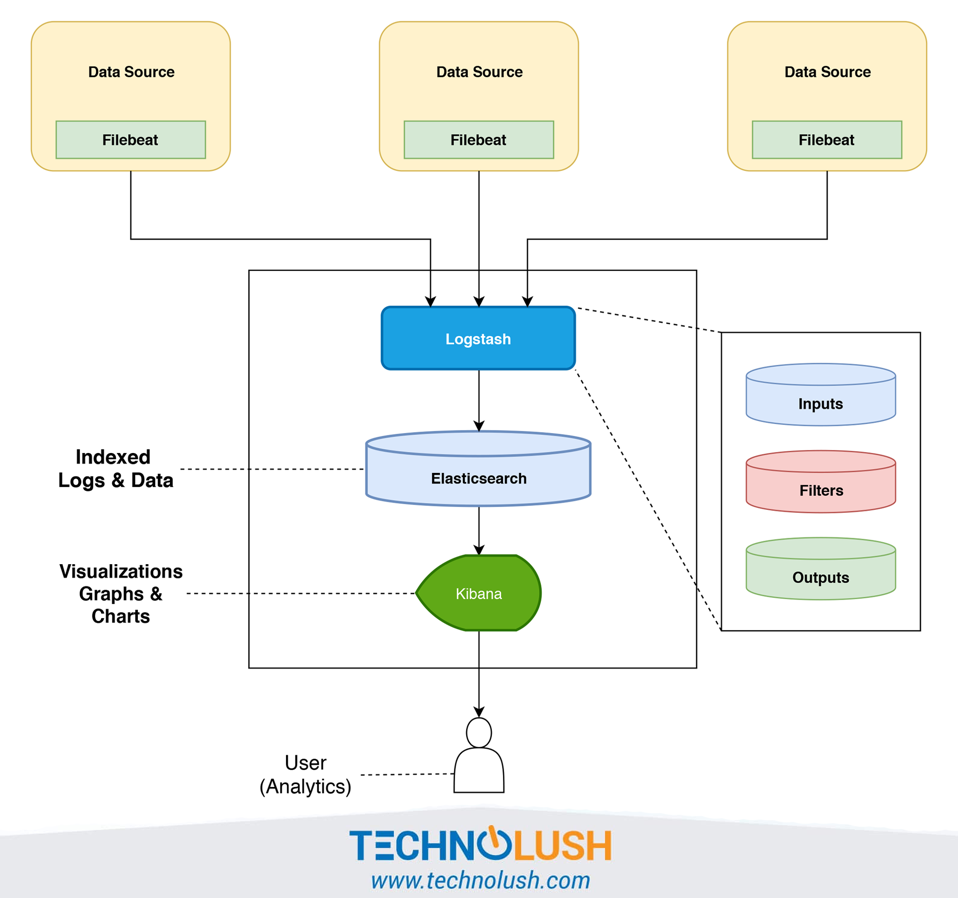


Figura – Diagrama de Implementación de Log Centralizado

A continuación, los componentes usados para la implementación:

* **FileBeat:** Esta herramienta hace parte del grupo Beats de Elastic y su función radica en ser transportador de logs para archivos locales. Para el caso del backend se despliega como pod dentro del cluster de Kubernetes, de tal manera que monitorean los archivos de logs de los Pods específicos. Esta información se reenviar a Elasticsearch para su almacenamiento.
* **Logstash:** Es un pipeline de procesamiento de datos gratuito y abierto del lado del servidor que ingesta datos auditoría los transforma y luego los envía a Elasticsearch.
* **Elasticsearch:** Es una base de datos no relacional de almacenamiento con funciones incorporadas de búsqueda de texto y análisis de datos. Componente usado como repositorio para almacenar datos no estructurados de los logs.
* **Kibana:** Es una herramienta open-source que pertenece a la compañía Elastic la cual permite visualizar y explorar datos que se encuentran indexados en Elasticsearch. Componente usado para la explotación de los datos almacenados de logs y para poder filtrar y ordenar los registros.

Para facilitar el análisis de logs, se deben crear los siguientes paneles en el Kibana:

* Panel de Log Técnico: Usado para los logs de cada componente de arquitectura BD, Bus de Eventos, Cache, etc.
* Panel de Log Funcional: Uso para análisis de log de tipo funcional que contiene registros de log transaccional.
* Panel de Disponibilidad: Usado para mostrar la disponibilidad de los servicios de negocio como por ejemplo los tramites de las entidades.

## Vista de Despliegue

A continuación, se describe muestra el diagrama de despliegue de los componentes:

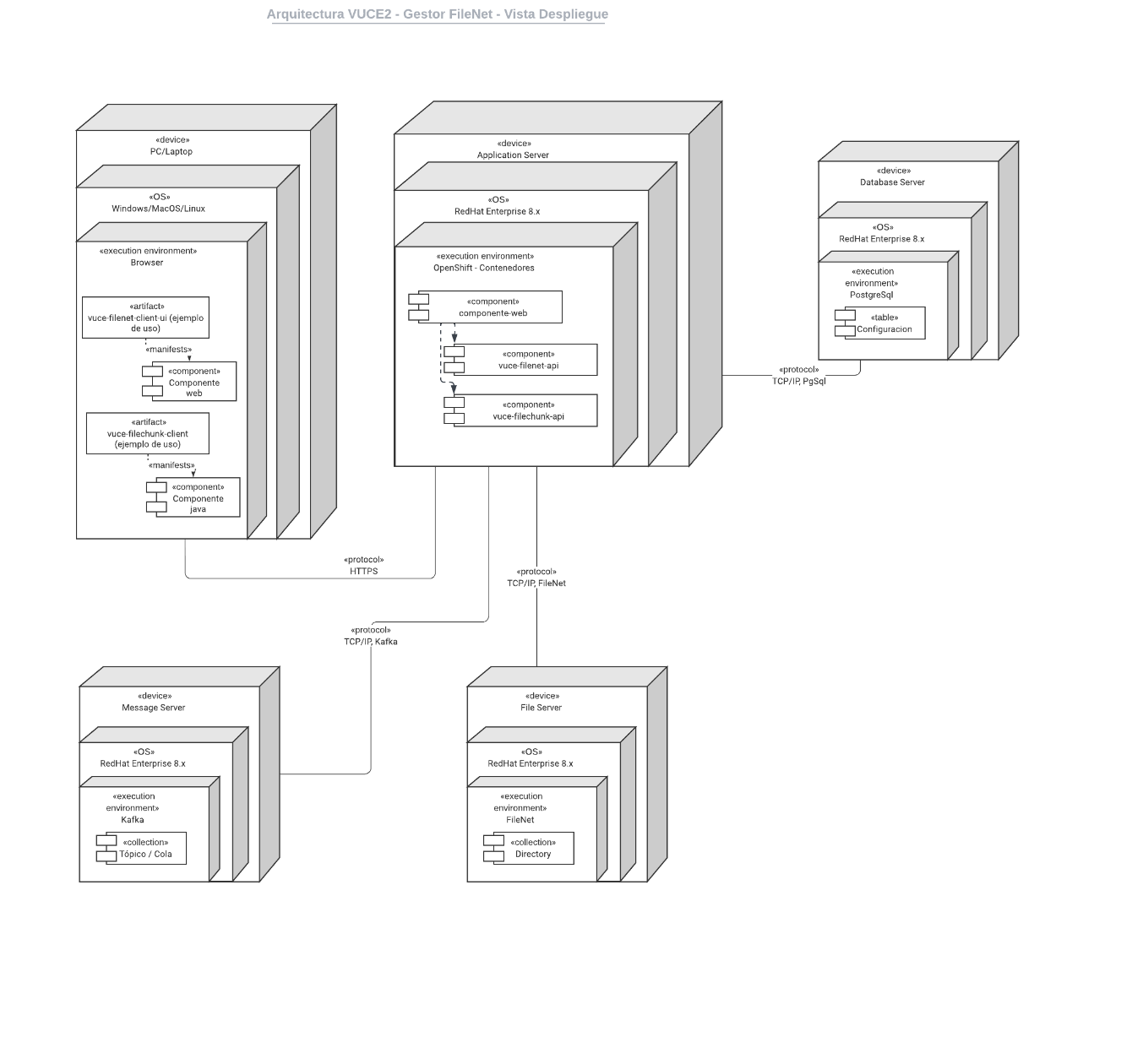


Figura – Diagrama de Despliegue de Gestor FileNet

Diagrama, Word

Descripción generada automáticamente

Figura – Diagrama de Despliegue Físico de Gestor FileNet

# Patrones usados

Los Patrones para Microservicios corresponden a los detallados en el documento Documento-de-Arquitectura-Patron-version-3.7.docx, sección 7. PATRONES PARA MICROSERVICIOS. A continuación, los Patrones para los microservicios usados en Gestor FileNet:

| **PATRÓN** | **UTILIZACIÓN** |
| --- | --- |
| OBSERVABILIDAD | **API Health Check:** Referencia para su implementación: https://www.baeldung.com/spring-boot-actuators  **Log Aggregation:** Su implementación se da con el uso de Loggers a nivel de código fuente y su explotación a través de ELK.  **Trazabilidad Distribuida:** Referencia para su implementación: https://spring.io/projects/spring-cloud-sleuth#overview  **Seguimiento de excepciones:** Su implementación se da con el uso de Loggers a nivel de código fuente y su explotación a través de ELK.  **Métricas de aplicaciones:** Actualmente se encuentra en proceso de evaluación la ampliación de licencias de APM Dynatrace requeridas para monitorear los componentes, cuando se concrete, se reemplazará las herramientas Grafana y Prometheus.  **Registro de auditoría:** Su implementación se da a través del módulo de Auditoría 2.0 y campos de auditoría a nivel de tablas de base de datos. |
| GESTIÓN DE FALLOS | **Circuit Braker, Bulk Head, Retry Pattern**  Todos estos patrones serán implementados usando la librería Resilience4j |

# Stack Tecnológico de la Arquitectura

A continuación, las tecnologías a usar por cada capa de arquitectura con sus respectivas versiones.

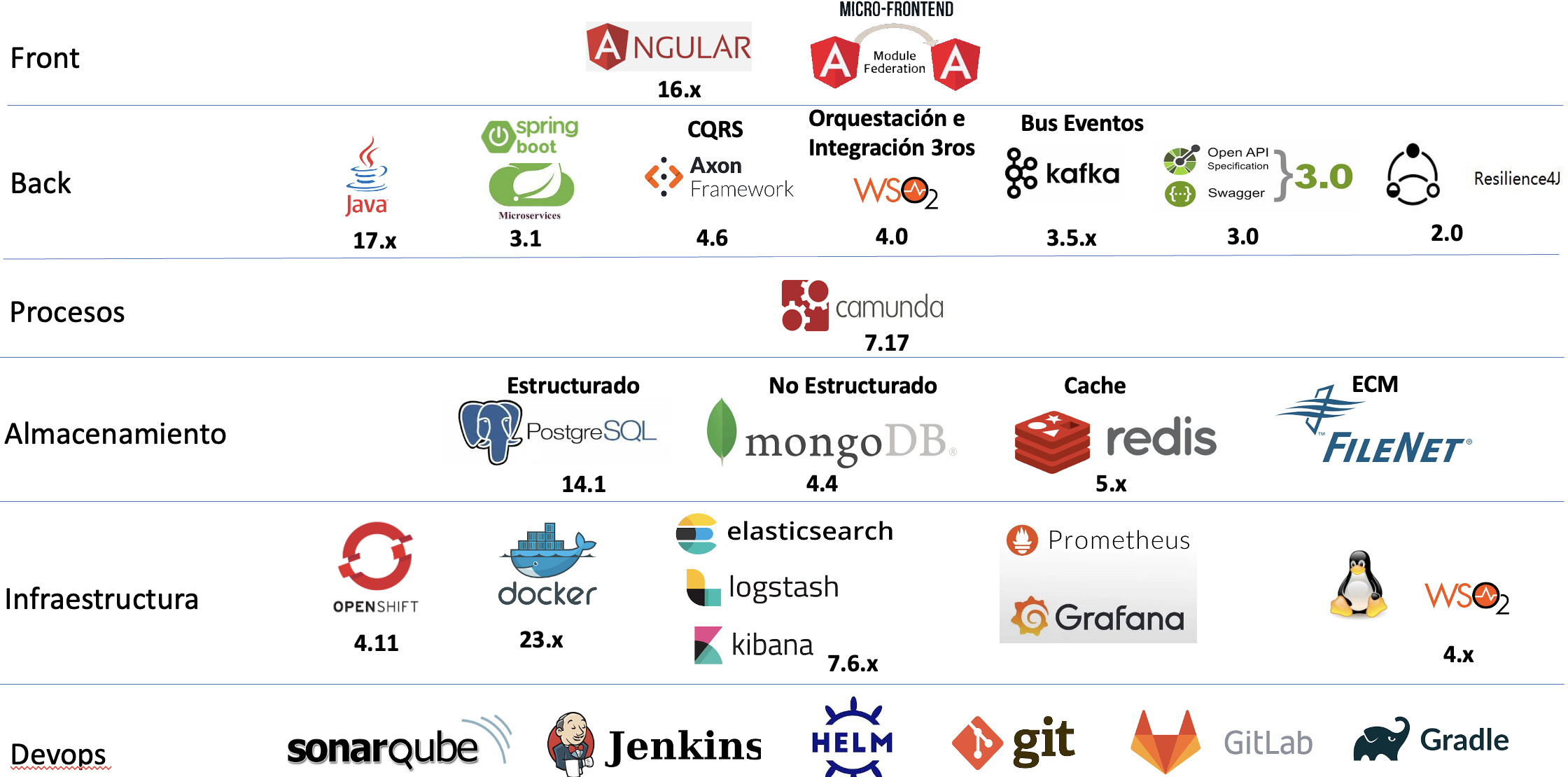


Figura – Stack Tecnológico de la Arquitectura

# Referencias

Sun Microsystems. (1997). *Java Code Convention*s

<https://www.oracle.com/technetwork/java/codeconventions-150003.pdf>