Proyecto SIARE Paraguay

Guía de ciclo de vida de la solución

Tabla de contenidos

1 Objetivo

El objetivo del presente documento es presentar una guía para la gestión del ciclo de vida de la solución.

2 Marco metodológico

La presente guía se enmarca en la cultura DEVOPS representada por los valores CAMP.

- Cultura (Culture): Una cultura de objetivos compartidos entre los equipos, colaboración, visión sistémica (tanto Devs como Ops forman parte de la "gran figura" y la entienden) y empatía mutua. Se podría resumir en una cultura ágil.

- Automatización (Automation): La automatización como herramienta para cumplir con los objetivos. Minimiza los tiempos de despliegue y de recovery. Minimiza el riesgo de tareas manuales aumentando la calidad. Permite obtener feedback rápido y por lo tanto un aprendizaje rápido.

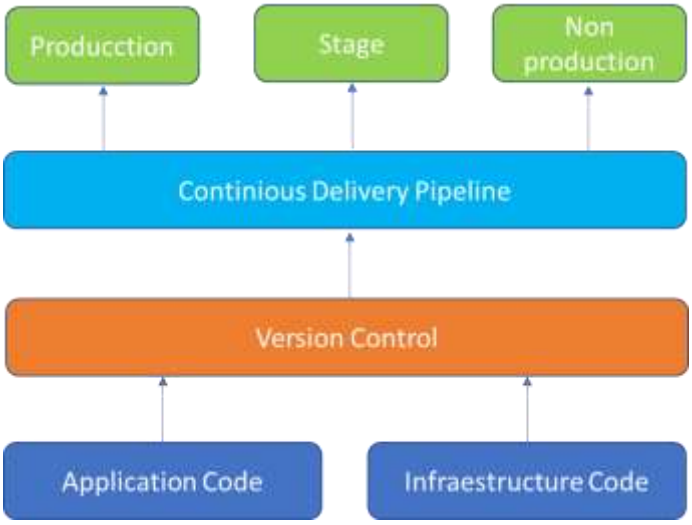
- Measuring: Medir para poder obtener feedback del producto y del proceso que permita "ajustar el rumbo" sabiendo que aquello que se mide cambia el comportamiento de los equipos. Tomar y comunicar los indicadores adecuados para los diferentes stakeholders.

- Sharing: Compartir los objetivos. Compartir la responsabilidad. Compartir las lecciones aprendidas entre los equipos, en particular entre los equipos de desarrollo y operaciones. Compartir tiempo

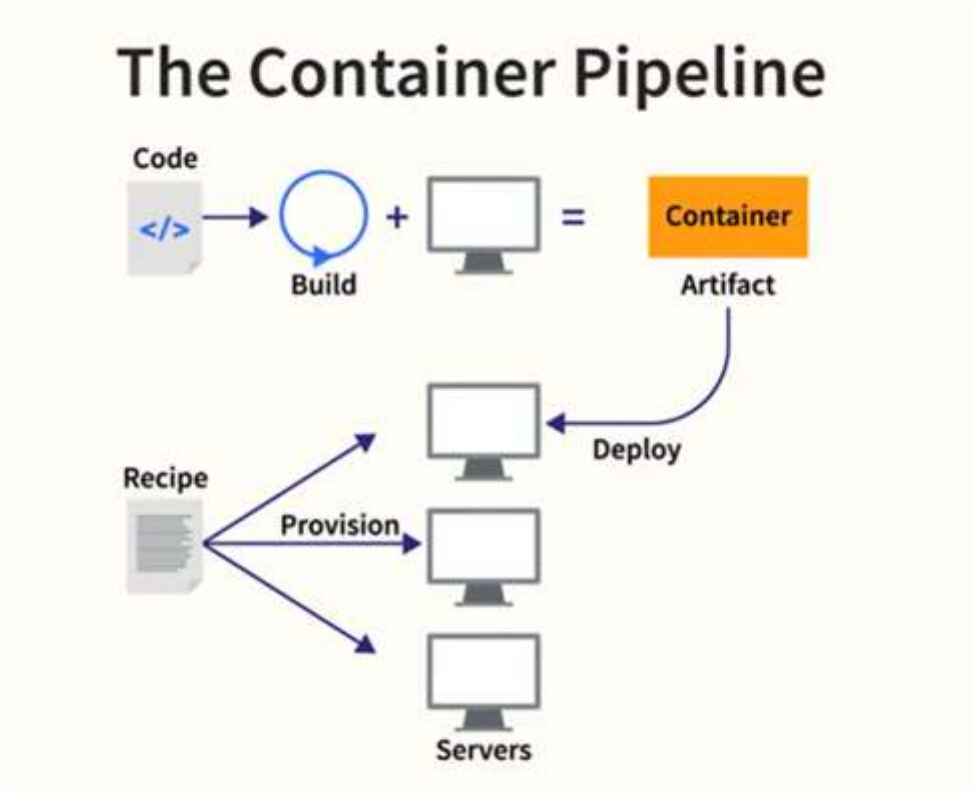
En particular se basa en los siguientes principios metodológicos:

- *Primero Personas / Luego Procesos / y finalmente Herramientas*

- *Infraestructura como código*



- *Inmutable deployment*

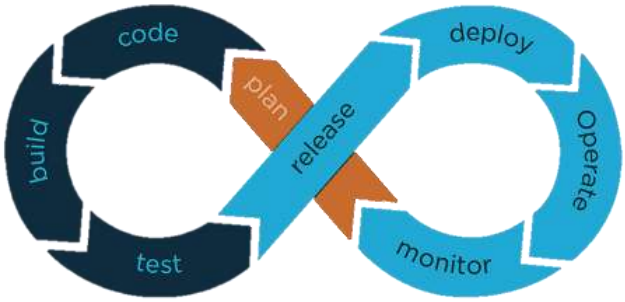


3 Roles claves (People)

- Arquitecto de la solución
- Release Manager
- Desarrolladores solución
- Desarrolladores automatizaciones
- Especialistas Infraestructura

4 Procesos (Process)

Como ya fue mencionado los procesos que se detallan a continuación fueron concebidos teniendo en cuenta la cultura devops.



4.1 Ambientes

Los ambientes definidos en la instanciación del proyecto son:

- **Ambiente Local de Desarrollo**: Ambiente de desarrollo local en el equipo de cada desarrollador con una copia actualizada de la última versión del repositorio de código.
- **Ambiente de Desarrollo**: Ambiente de desarrollo que se utiliza para ejecutar productos o componentes de manera centralizada a nivel de desarrollo. Por ejemplo, bases de datos o componentes que no puedan ser ejecutados en los ambientes locales de cada desarrollador.
- **Ambiente de Integración**: Ambiente en el cual se integran los diferentes módulos de la solución y se realizan las pruebas de integración.
- **Ambiente de Test**: Ambiente en el cual el equipo de desarrollo realiza las pruebas funcionales y no funcionales.
- **Ambiente de UAT**: Ambiente en el cual se realizan las pruebas de aceptación, funcionales y no funcionales por parte del usuario.
- **Ambiente de Staging o Preproducción**: Ambiente previo a producción, que conserva características similares y donde se hacen pruebas de carga y pruebas piloto con datos reales
- **Producción**: Ambiente en el cual se opera el sistema productivo.

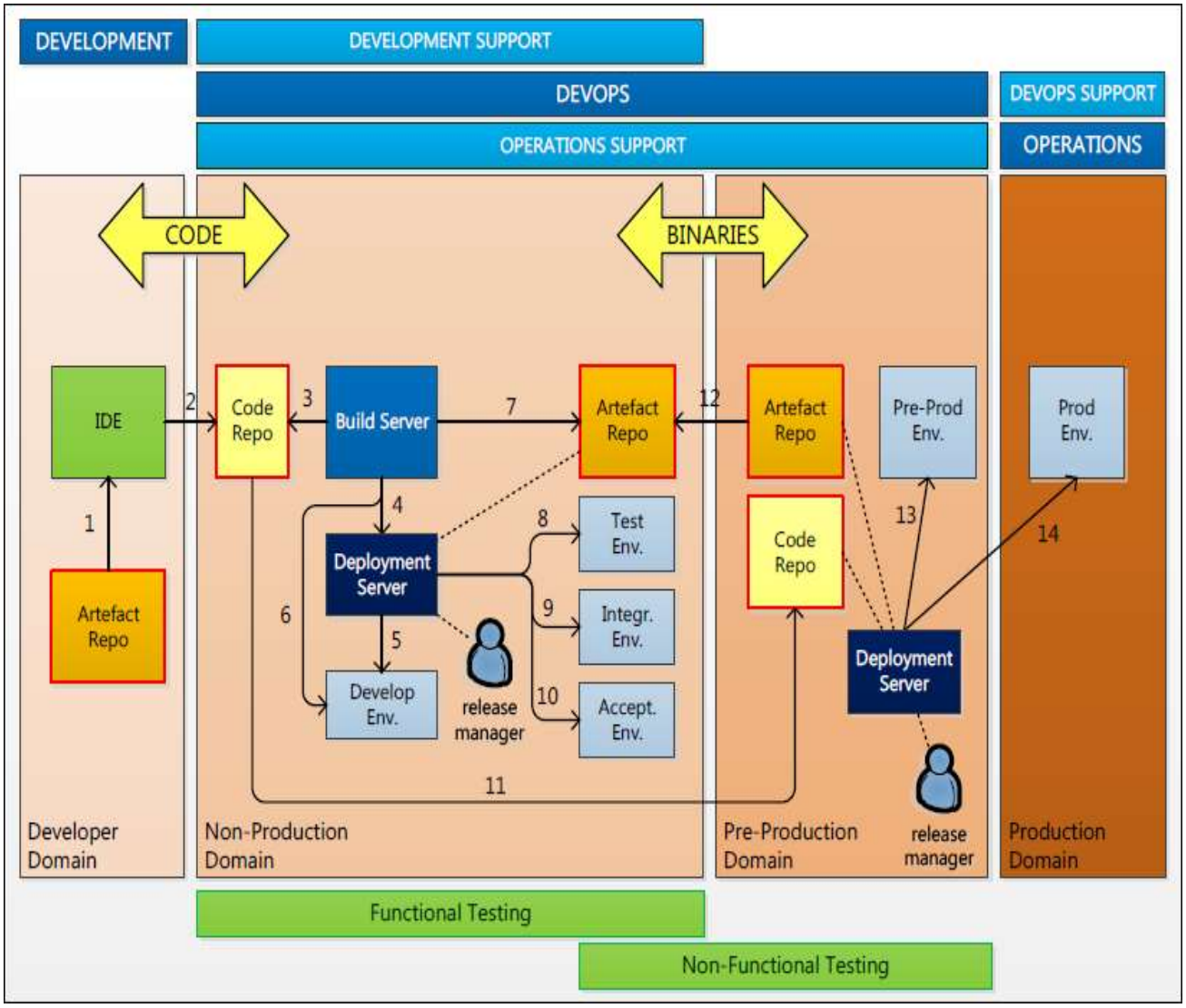
Cuando una pieza de código está lista, esta se pasa al ambiente de integración, utilizando el ciclo devops definido más adelante en este documento. En este ambiente se realizan pruebas integradas a nivel de desarrollo, para luego pasar al ambiente de Test

Una vez que se consolida una versión para reléase, esta se promueve al ambiente de Test, de forma que el equipo de QA realice las pruebas pertinentes.

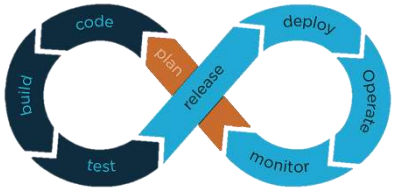
Finalizada la validación, se genera una versión de reléase candidata, que se promueve al ambiente de UAT para la validación por parte del cliente.

Una vez pasado el ciclo de pruebas funcionales esta se promueve al ambiente de Staging para la realización de pruebas no funcionales y para su posterior pasaje a producción.

A continuación, se presenta un diagrama de alto nivel en donde se detalle el ciclo completo de la gestión de los ambientes mencionados:



4.2 Planificación



Considera todas las actividades correspondientes a la gestión y ejecución del proyecto, las cuales estarán gestionadas con apoyo en la herramienta **Redmine**.

4.3 Codificación

Comprende todas las actividades necesarias para el desarrollo colaborativo de aplicaciones.

Cada desarrollador trabaja de forma local en su máquina utilizando una copia del repositorio y al finalizar su trabajo lo integra con el repositorio Git y consumiendo los servicios que requiera del ambiente de desarrollo centralizado.

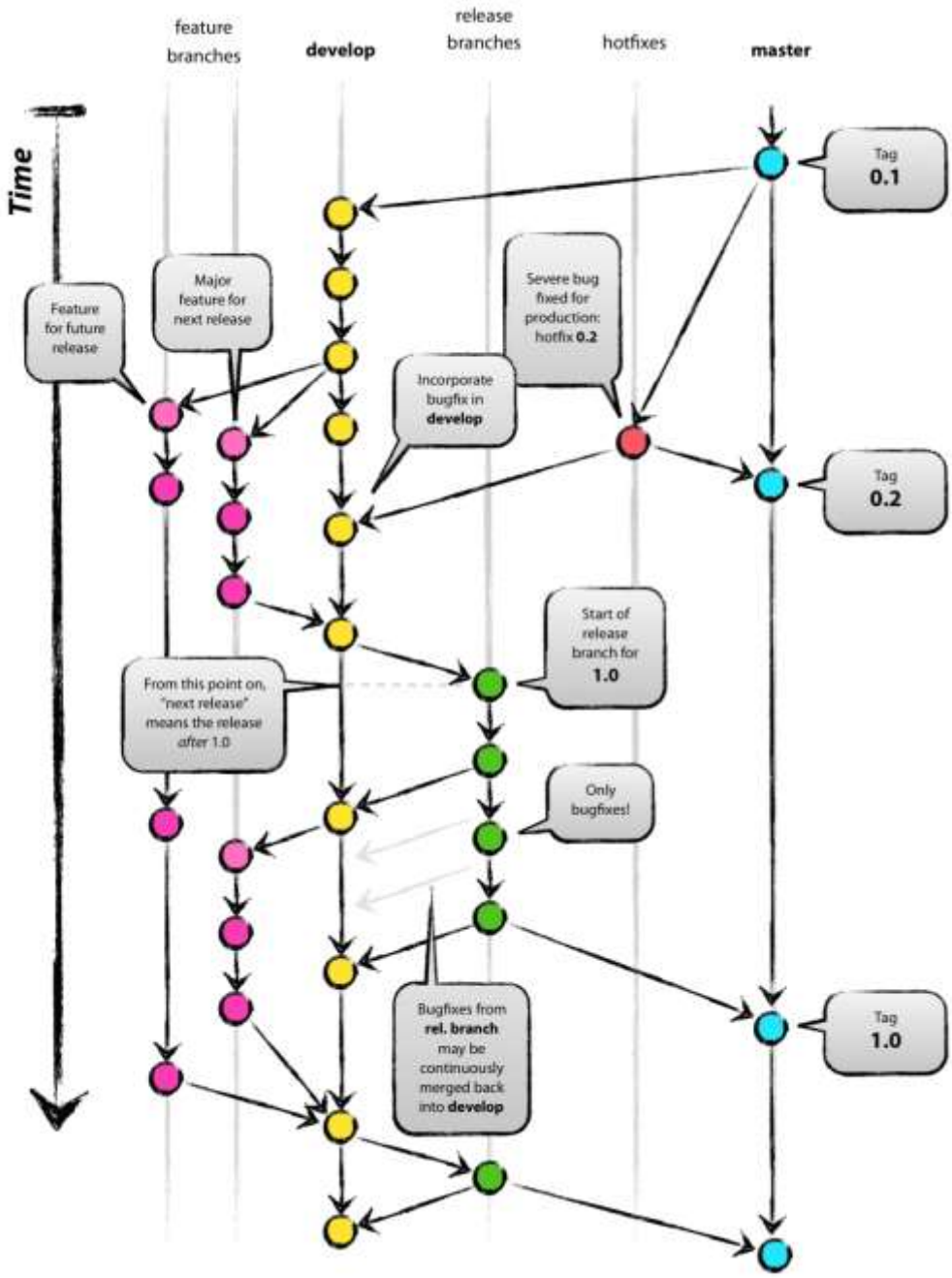
Considera las siguientes actividades:

Obtención de código fuente. creación del feature branch sobre el cual se trabajará.
Codificación de la solución. Codificación de la solución de software.
Codificación de las pruebas unitarias. Codificación de las pruebas unitarias para los componentes principales de la solución. Estas pruebas unitarias pueden cubrir las funciones básicas de set-get y funciones de integración cuando esto sea requerido.
Ejecución de pruebas unitarias y de servicios en entorno local. Considera la ejecución de pruebas lanzadas desde las herramientas de desarrollo y en el entorno local de cada desarrollador.
Análisis estático de código. Se implementa a través del plugin que integra con Sonarqube, el cual permite realizar validación de código estático con el uso de reglas estándar y ampliar con la definición de reglas específicas. Adicionalmente se agrega plugin de seguridad para validación OWASP de seguridad.
Revisión de código fuente (code-review). Actividades de revisión de código fuente en la etapa final del sprint, la revisión por pares peer-review es realizada entre desarrolladores, el code-review es realizado mediante la revisión del líder técnico del equipo de trabajo.
Merge a rama develop y creación de rama release. Es la última actividad del desarrollador para dar fin a la actividad de codificación, se realiza según la definición de gestión de versiones especificada en el apartado dedicado a tal fin

4.3.1 Gestión del código fuente

Para la gestión colaborativa de versionado de código fuente, se utilizará **GitLab** y el workflow estándar de mercado **gitflow**. A continuación, se describe el diagrama general de uso y algunos escenarios de trabajo.

4.3.1.1 Diagrama de versionado de código fuente GitFlow



4.3.2 Ejecución de pruebas de calidad (Ambiente de Test)

Se corresponde con el primer nivel de pruebas funcionales, la ejecución de estas es manual, por el equipo del proveedor. Se utilizarán como herramientas de apoyo:

- **TestLink**: para la definición de casos, planes de prueba y registro de la ejecución de estas.
- **Redmine**: para el reporte de defectos o tareas al equipo de desarrollo.

Se itera en este ciclo tantas veces como sea necesario, una vez finalizada la ejecución del plan de pruebas y con la aprobación del plan completo se ejecuta los PIPES JENKINS para realizar el despliegue en el siguiente entorno.

4.3.3 Ejecución de pruebas de aceptación de usuario (Ambiente UAT)

Corresponde a la ejecución de pruebas de aceptación del usuario, conceptualmente debería cubrir casuísticas de negocio end-to-end que certifiquen el correcto funcionamiento del negocio tanto para el incremento de producto entregado como los casos de regresión definidos. Al igual que la etapa anterior, su ejecución es manual, realizada por usuarios de negocio del cliente con apoyo de equipo IT. Utilizarán como herramientas de apoyo:

- **TestLink**: para la definición de casos, planes de prueba y registro de la ejecución de estas.
- **Redmine**: para el reporte de defectos o tareas al equipo de desarrollo.

Se itera en este ciclo tantas veces como sea necesario, una vez finalizada la ejecución del plan de pruebas y con la aprobación del plan completo se ejecutan PIPES JENKINS para realizar el despliegue en el siguiente entorno.

4.4 Liberación y Despliegue (Release and Deploy)

Esta etapa del proceso se repite para el despliegue del producto ejecutable en cada uno de los entornos de trabajo.

5 FRAMEWORKS Y TECNOLOGÍAS DE REFERENCIA

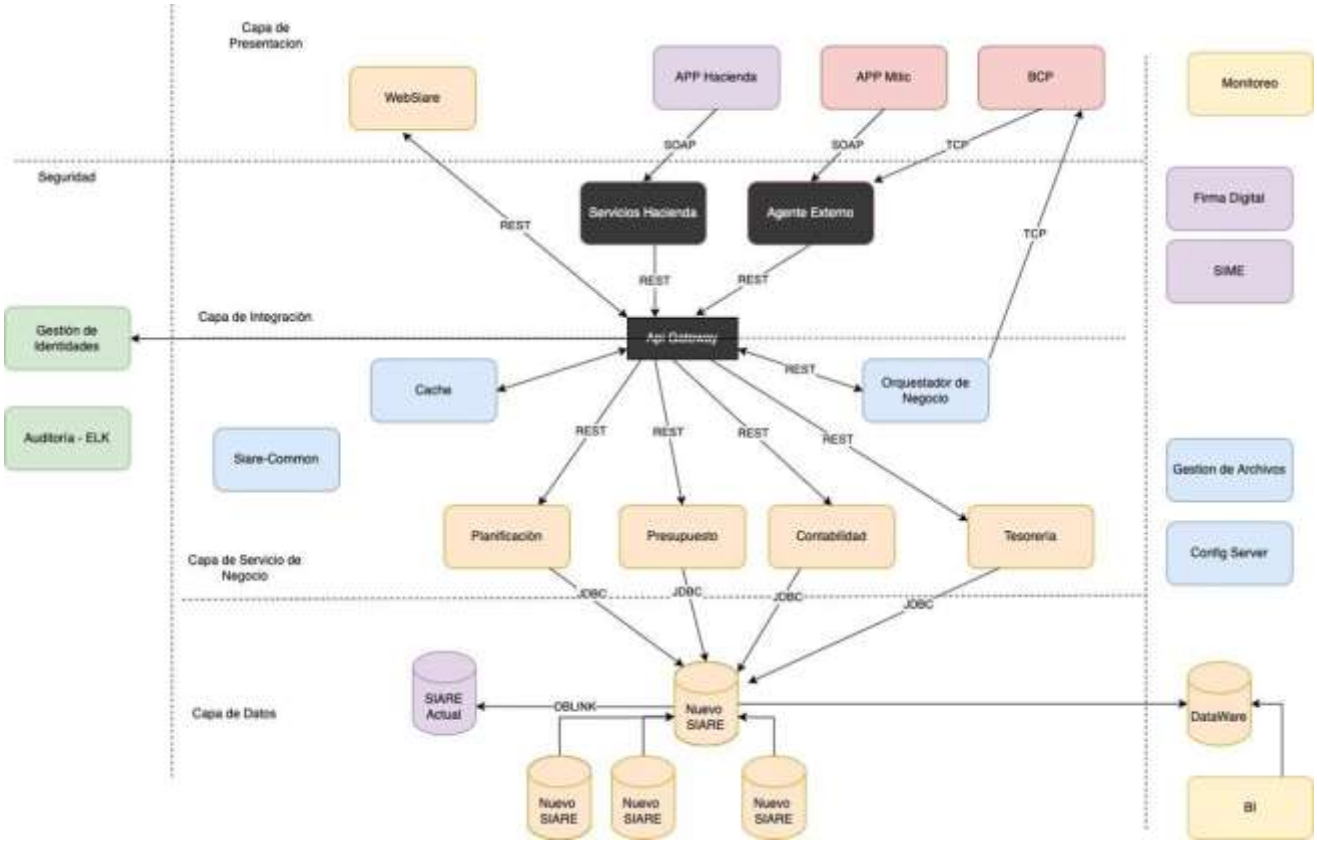
Frameworks / Tecnología	Versión	Upgrade Corto Plazo	Uso
Java	OpenJDK 17		Backend Reglas de negocio
Angular	~16.0.0	~17.0.0	Front end
node.js	18.17.1		Entorno de trabajo para JavaScript.

Frameworks / Tecnología	Versión	Upgrade Corto Plazo	Uso
			Librerías y frameworks para el front end
Bootstrap	~5.0.0		Front end
PrimeNG	~15.0.0	~16.0.0	Plantilla de componentes de interfaz web para angular
Hibernate	5.5.x		Servicios de persistencia
Log4J LogBack	2.14.1 1.2.5		Servicios de Logging y Auditoría
Spring Framework	5.3.9	6.x.x	
Springboot (Wildfly +) Spring config (config server)	2.7.4	3.x.x	Empaquetado
Dockers	20.10		Contenedores
drawIO	14.9.1		Modelado de la arquitectura
GitLab	14.1		Repositorio de código fuente y configuraciones.
GitLab plug in	x		Plug in de GitLab para Ansible / Groovy Para administrar el código fuente generado.
GitLab plug in	1.5.20		Plug in de GitLab para Jenkins que permite disparar orquestaciones de Jenkins ante eventos de GitLab.
Git Bash, Git Kraken, Sourcetree	X		Clientes git
Maven	3.8.1		Empaquetado
Jenkins	2.361.4		Orquestación de automatizaciones
Nexus	3.x		Repositorio de artefactos.
SonarQube	9.7.1		Análisis estático de código.

Frameworks / Tecnología	Versión	Upgrade Corto Plazo	Uso
Ansible Groovy	2.9 4.0		Construcción de Pipes de automatización.
Docker Compose	1.29.2		Generar y administrar el entorno de contenedores en el puesto de trabajo.
Eclipse STS, IntelliJ			IDE de desarrollo
Visual Studio Code	1.81.1		IDE de desarrollo
JasperReports & TIBCO JasperStudio	6.19		Ide para Reportería
jsPDF	2.5.1		Librería para reportería de front end
Spring Tools	4		Plugin del framework Spring para eclipse
Junit	5		Creación y ejecución de pruebas unitarias
Plugin de SonarQube para Eclipse	x		Plugin de SonarQube para eclipse
WSO2AM	4.0.0		Api Gateway
WSO2IS	5.11.1		Identity Server
Keycloak	16.1.0		Identity Server
Grevitee	2		Api Gateway
TyK	3.2.1		Api Gateway
ELK (Elasticsearch, Logstash, Kibana)	7.14		Gestión de logs y trazas de auditoría.
Filebeat	8		Gestión centralizada de log
Postgresql	13+		Base de datos
Postgis	3.1		Base de datos georeferenciada
Oracle	19		Base de datos
Pentaho Data Integration Tool	PDI-CE 9		ETL
Apache HOP	1.1.0		ETL

Frameworks / Tecnología	Versión	Upgrade Corto Plazo	Uso
Kubernetes	V1.24+		Gestión de contenedores
Grafana Prometheus	8.0.6 2.29		Monitoreo de la solución y telemetría
VAULT Hashicorp	1.8		Gestión de secrets
Redmine	4.2.2		Gestión de actividades, defectos y requerimientos
Testlink	1.9.20		Gestión del proceso de testing
Tableau	2021.03		Explorador y visualizador de datos
Redis	7.x.x		Gestión de cache
Kafka (kraft como deseable)	7.1.1	Actualización a KRAFT	Gestión de Streams
Apache Flink (deseable)	1.17.1		Gestion de estado en mensajes

6 ARQUITECTURA PROPUESTA


Capa de Presentacion
WebSiare
APP Hacienda
APP Mitic
BCP
Monitoreo
SOAP
SOAP
TCP
Seguridad
Servicios Hacienda
Agente Externo
Firma Digital
SIME
REST
REST
REST
TCP
Capa de Integración
Gestión de Identidades
Cache
Orquestador de Negocio
REST
Auditoria - ELK
Siare-Common
REST
REST
REST
REST
Gestion de Archivos
Planificación
Presupuesto
Contabilidad
Tesoreria
Config Server
Capa de Servicio de Negocio
JDBC
JDBC
JDBC
JDBC
SIARE Actual
DBLINK
Nuevo SIARE
DataWare
Capa de Datos
Nuevo SIARE
Nuevo SIARE
Nuevo SIARE
BI