



Metro de Madrid, S.A.

Estrategia de Migración

PEMS

CONTROL DOCUMENTAL

Elaborado por:	Metro de Madrid
Objeto:	Determinar la estrategia de Migración de SIAR
Versión	Descripción
1.0	Estrategia de Migración de SIAR
Lista de distribución:	

CONTROL DE CAMBIOS

Fecha Versión	Autor	Página	Cambio
31/03/2021	Metro		Primera versión
20/09/2021	Metro		Sustitución del nombre Solución de Migración por Bloque de Migración



Metro de Madrid, S.A.

INDICE DE CONTENIDO

1. Introducción	4
2. Objetivo del Documento.....	5
3. Estrategia de Migración	6
3.1. Infraestructura	8
<i>Metodología de estimación</i>	<i>10</i>
<i>Procedimiento de implantación</i>	<i>11</i>
3.2. Migración JBoss	15
<i>Alcance de la actividad</i>	<i>15</i>
<i>Procedimiento de implantación</i>	<i>15</i>
3.3. Actualización Frameworks	17
<i>Alcance de la actividad</i>	<i>17</i>
<i>Procedimiento de implantación</i>	<i>18</i>
3.4. Test de verificación	21
<i>Alcance de la actividad</i>	<i>21</i>
<i>Procedimiento de implantación</i>	<i>21</i>
3.5. Migración ODM & Migración CP/CPLEX.....	25
<i>Alcance de la actividad</i>	<i>25</i>
<i>Procedimiento de implantación</i>	<i>34</i>
3.6. Test Carga (20 casos prueba)	35
<i>Alcance de la actividad</i>	<i>35</i>
<i>Procedimiento de implantación</i>	<i>35</i>
3.7. Refactorización 15% SQL/ORM	36
<i>Alcance de la actividad</i>	<i>36</i>
<i>Procedimiento de implantación</i>	<i>37</i>
3.8. API Gateway	39
<i>Alcance de la actividad</i>	<i>39</i>
<i>Procedimiento de implantación</i>	<i>39</i>
3.9. Sustitución Frontales Web	40
<i>Alcance de la actividad</i>	<i>40</i>



Metro de Madrid, S.A.

<i>Análisis técnico</i>	<i>40</i>
<i>Framework.....</i>	<i>44</i>
<i>Arquitectura</i>	<i>46</i>
<i>Estrategia de Pruebas</i>	<i>50</i>
3.10. Integración y Entrega Continua	52
<i>Integración Continua</i>	<i>52</i>
<i>Entrega Continua</i>	<i>56</i>



Metro de Madrid, S.A.

1. Introducción

El principal objetivo del presente documento es el de realizar una descripción detallada sobre la Estrategia de la Migración del sistema SIAR (Sistema Integral de Asignación de Recursos) que Metro de Madrid (Metro) explota con la finalidad de realizar la gestión y seguimiento diario de la operativa de su personal en trenes y estaciones.

Este sistema es crítico para Metro porque ofrece, a través de sus interfaces de usuario y procesos tecnológicos, una visión lo más actualizada posible de la situación en estaciones y trenes permitiendo a los distintos usuarios del mismo, aplicar todas aquellas medidas necesarias para subsanar y corregir desviaciones en el servicio, evitando en buena medida que estas desviaciones se traduzcan en una disminución de la calidad del servicio ofertado a los usuarios de Metro de Madrid.



Metro de Madrid, S.A.

2. Objetivo del Documento

El objetivo del presente documento es el de detallar la estrategia de Migración de SIAR.



Metro de Madrid, S.A.

3. Estrategia de Migración

El alcance de la Ejecución de la Migración de SIAR consistirá en una serie de Bloques de Migración que proporcionarán a Metro una respuesta para los principales problemas actuales, de forma que sea posible dar continuidad en la explotación de la aplicación de forma controlada durante los próximos años, hasta que fuera necesario la realización de una nueva actualización.

Se conformará la infraestructura donde se deberá realizar la migración de SIAR. Será similar a la actual, optando por un escenario Activo-Pasivo sin redundancia para los entornos de Desarrollo y Pre Producción dejando única y exclusivamente el entorno de Producción redundante en ambos CPD's.

Bloques de Migración

Instalación/configuración de productos a nueva Infraestructura: Se realizará la instalación y/o configuración de los distintos productos software de modo que queden optimizados para el comienzo del resto de trabajos de Migración, incluyendo la instalación, copia con cambio de arquitectura y migración de 3 entornos de base datos Oracle, así como la instalación, customización y certificación de la alta disponibilidad con Veritas para la base de datos Oracle de producción. La revisión de versiones se realizará al inicio y durante la migración de tal forma que al final de la migración queden instaladas las versiones certificadas y compatibles más modernas posibles de todos los productos.

Versionado de Frameworks: Es necesaria la actualización de los frameworks que dan soporte técnico a SIAR ya que han sufrido una evolución natural a nuevas versiones, estando incluso algunos de ellos discontinuados.

Test de verificación: Pruebas de verificación de la corrección de las principales funcionalidades del sistema sobre la nueva infraestructura física y lógica. Se determinará una batería de pruebas para la medición del correcto funcionamiento del aplicativo tanto en el plano funcional como en su propio funcionamiento.

Migración de IBM ODM y CPLEX: Debido al fin de soporte de las versiones de IBM ODM y CPLEX utilizadas en la actualidad por SIAR es necesario realizar la migración de estos productos. El producto ODM 8.10 deja de tener compatibilidad con Solaris por lo que es necesario cambiar la infraestructura que pasará a ser sobre RedHat en lugar de Solaris.



Metro de Madrid, S.A.

Adaptación WebLogic a Jboss: Debido al fin de soporte de la versión de WebLogic utilizada en la actualidad por SIAR y al cambio de directriz en cuanto a los servidores de aplicación para las futuras aplicaciones de METRO, se realizará el cambio hacia Jboss.

Test de Carga: La ejecución de aproximadamente 20 casos de prueba que permitan verificar detalladamente el funcionamiento global del sistema. Se ejecutan bajo un enfoque de caja negra. Se deben realizar después de haber comprobado que cada módulo funciona correctamente, así como la integración entre ellos. Se verificará el rendimiento de procesos y concurrencia de usuarios.

Refactorización uso SQL y ORM: Revisión del rendimiento de las sentencias efectuadas en base de datos, su construcción en el ORM empleado por Metro (Hibernate) y el rendimiento de este ORM en la capa de acceso a datos de la aplicación. Se pretende realizar el análisis del 15% (lo más significativo) del acceso a datos del sistema.

Sustitución Frontales a HTML 5: Debido a la discontinuidad de JSF Rich Faces desde el año de 2016 y por problemas identificados dentro de las frecuentes actualizaciones de navegadores en Metro, se realizará la migración de los frontales de forma que estos dispongan de una tecnología avanzada (Angular, React u otras).

Implantación y uso API Gateway: Implantación y uso de API Gateway para la comunicación entre la capa Front y Back de SIAR Web. Se recomienda utilizar dentro del ámbito de integración con aplicaciones terceras.



Metro de Madrid, S.A.

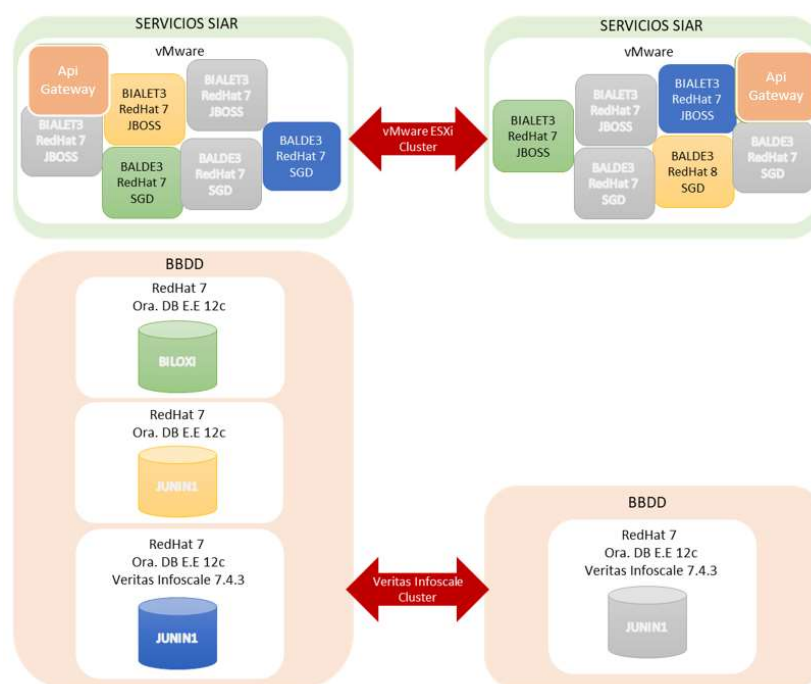
3.1. Infraestructura

Descripción de la Infraestructura

La infraestructura que se ha definido consiste en disponer de un Hardware desplegado en dos CPD's diferentes, de tal forma que esta infraestructura esté capacitada para soportar un entorno Pre Productivo (no existente en la infraestructura actual) y también la implementación de una pieza para funcionar como API Gateway.

Dentro de la nueva infraestructura se dispondrá de un cluster de FrontEnd con Hipervisor VMWARE aprovechando el stretch cluster del actual almacenamiento y las VLAN's extendidas entre ambos CPD's. Los servicios se desplegarán sobre máquinas virtuales con la versión de redhat más moderna que sea compatible con los productos a instalar en todos los entornos.

El servicio de base de datos (Oracle Database E.E.) para el entorno de Desarrollo y Preproducción se desplegará cada uno en un nodo basado en Red Hat y ubicados en el CPD A, pero sin contar con recursos de respaldo en el CPD B. En el caso del entorno de Producción se desplegará un cluster de Veritas Infoscale basados en Red Hat que permitirá disponer de la base de datos de producción en alta disponibilidad pudiendo ser balanceada en caso necesario al nodo del CPD B.



Escenario Infra AP-Parcial



Metro de Madrid, S.A.

El API Gateway que se utilice se desplegará en ambos servidores (CPD A y B), y dentro de cada uno dispondrá de una instancia Productiva y No Productiva. La utilización del API Gateway dentro de este proyecto se centrará en la comunicación entre la parte FrontEnd, que se desarrollará en Angular, con la parte BackEnd, no obstante, en un futuro esta herramienta podrá ser utilizada para otras aplicaciones o incluso para la integración con sistemas terceros.

La infraestructura prevista para SIAR se compondrá de una pareja de X86/Linux con comunicaciones de fibra y cobre redundadas en el caso del FrontEnd y estará compuesto por un servidor por CPD de doble procesador Intel Xeon Gold 6230 (20 Cores / 40Threads) o equivalente con un total de 1TB. de RAM por cada uno de ellos. Para el BackEnd de base de datos se dispondrá de tres servidores en el CPD A (Desarrollo, Preproducción y Producción) y un servidor en el Site B (Producción) cada uno de ellos contando con doble procesador Intel Xeon Platinum 8256 (4 Cores / 8Threads) o equivalente y 256Gb de RAM.



Escenario Infra AP Parcial HW



Metro de Madrid, S.A.

Versiones software a desplegar en la infraestructura

A continuación, a modo de referencia, se muestran las versiones iniciales de software previstas a desplegar en la nueva infraestructura.

Producto	Versión
vMware	7.0
Red Hat JBoss	7.1
Oracle Database E.E.	12c
Red Hat S O	7
Java SE	1.8
Veritas Infoscale	7.4.3
Oracle Secure Global Desktop (SGD)	5.6
IBM ODM	8.10.5
IBM CP-CPLEX	12.10.0.0

Tabla 1 - Licencias y Versiones

Las versiones deberán ser las más modernas disponibles y compatibles con los productos a instalar al comienzo del proyecto. Se deberán revisar durante la ejecución del proyecto de forma que en la puesta en producción de la aplicación las versiones sean las más modernas posibles compatibles con los productos.

Licenciamiento

VMware – Licenciamiento por sockets.

RedHat Jboss – Licenciamiento se podría realizar en base a un entorno físico o virtual, siendo que en el físico se cuenta el total de Cores del servidor y en el virtual los Cores virtuales (vCPU) asignados al producto

Oracle Database – Licenciamiento contabilizados por un factor 0,5 por Cores.

RedHat SO – Licenciamiento por Instancia.

Java SE – Licenciamiento por paquetes de 0,5 Cores (podría ser sustituido por Open JDK bajo elección de Metro de Madrid).

Veritas Infoscale – Licenciamiento por Core.

Oracle SGD – Licenciamiento por usuarios.

IBM ODM y CP-CPLEX – Licenciamiento por PVU.

Metodología de estimación

La estimación mencionada para la implantación de infraestructura se ha elaborado basándose en la experiencia en la realización de despliegues similares.



Metro de Madrid, S.A.

Procedimiento de implantación

Bases de datos Oracle. Se pretende instalar el software de Base de Datos Oracle 12c, 19c o superior en los 3 nuevos entornos de SIAR a la última versión estable y certificada con el resto de productos existente en el momento de la migración, siendo Metro el que decida finalmente, tras la propuesta del adjudicatario, la versión definitiva.

Además, se debe contemplar la copia de los entornos actuales en los nuevos con cambio de arquitectura de Oracle Solaris a RHEL, migración de 12.1 a 12c, 19c o superior y con mínimo tiempo de parada en 2 de los entornos (preproducción y producción). Se valorará el instalar en su momento CDB's y PDB's en los nuevos entornos. El entorno de Producción se instalará en una arquitectura de Alta Disponibilidad con Veritas.

Se deben contemplar al menos las siguientes fases para cada uno de los 3 entornos:

- Preparación inicial de los entornos de Desarrollo, Preproducción y Producción.
- Instalación y copia de 3 entornos de Base de Datos Oracle con cambio de arquitectura de Oracle Solaris a RHEL, migración de 12.1 a 12c, 19c o superior con sus últimas actualizaciones y con mínimo tiempo de parada en 2 de los entornos (preproducción y producción). Se valorará el instalar en su momento CDB's y PDB's en los nuevos entornos. El entorno de la Base de Datos de Producción estará en una arquitectura de Alta Disponibilidad con Veritas, hecho que debe ser tenido en cuenta para actuar de forma conjunta durante la instalación de ambos softwares. Los pasos que se realicen en la fase de las copias deben quedar documentados en detalle con las acciones realizadas.
- Actualización de parámetros acorde con la nueva versión de Oracle. Soporte de tuning durante el periodo y jornadas previstos para estos pasos.
- Documentación y soporte posterior. La documentación se facilitará en castellano y se aportará soporte digital modificable.

Alta disponibilidad de Veritas Infoscalle. Se pretende instalar el software de Veritas Infoscalle Enterprise para controlar la alta disponibilidad de una Base de Datos Oracle 12c, 19c o superior en el entorno de Producción que consta de 2 nodos que se encuentran en 2 ubicaciones físicas distintas con el sistema operativo RHEL, se dispone de LAN extendida y el almacenamiento entre sites será activo-



Metro de Madrid, S.A.

activo. En los nodos de base de datos de Desarrollo y Preproducción se instalará Veritas Infoscale Storage.

La versión del software de Veritas Infoscale será la última versión estable y certificada con el resto de productos existente en el momento de la migración, siendo Metro el que decida finalmente, tras la propuesta del adjudicatario, la versión definitiva.

Se deben contemplar al menos las siguientes fases:

- Preparación del entorno.
- Instalación, customización, realización de pruebas y certificación del entorno de Alta Disponibilidad de Producción de 2 nodos con Base de Datos Oracle.
- Documentación y soporte posterior. La documentación se facilitará en castellano y se aportará soporte digital modificable.

Servidor de aplicaciones. Se instalará el servidor de aplicaciones JBoss EAP en la versión más reciente compatible con el resto de productos necesarios, siendo Metro el que decida finalmente, tras la propuesta del adjudicatario, la versión definitiva. Se instalarán tres entornos, desarrollo, preproducción y producción con la configuración que se estime más adecuada para albergar los diferentes productos/aplicaciones necesarios para el proyecto.

Se deben contemplar al menos las siguientes fases:

- Decisión de la versión a instalar de acuerdo con las matrices de compatibilidad de los diferentes productos implicados y de la evolución del proyecto
- Preparación inicial de los entornos (creación de servidores, instalación de SO, instalación de JDK, etc.)
- Instalación de JBoss EAP
- Creación de los servers de JBoss necesarios con los perfiles adecuados según el tipo de aplicación/producto a desplegar en ellos
- Configuración como servicios para arranque/parada automática con el servidor
- Configuración necesaria (creación de datasources, logging, etc.) a realizar en el servidor de aplicaciones
- Creación/configuración de Jobs en Jenkins de Metro para el despliegue de los distintos componentes en los diferentes entornos o Instalación/despliegue de los productos necesarios en los diferentes entornos



Metro de Madrid, S.A.

- Pruebas
- Documentación de la instalación y configuración

Servidor Motor. Se instalará el servidor del motor en la versión más reciente compatible con el resto de productos necesarios, siendo Metro el que decida finalmente, tras la propuesta del adjudicatario, la versión definitiva. Se instalarán tres entornos, desarrollo, preproducción y producción con la configuración que se estime más adecuada para albergar los diferentes productos/aplicaciones necesarios para el proyecto.

Se deben contemplar al menos las siguientes fases:

- Decisión de las versiones de los productos a instalar de acuerdo con las matrices de compatibilidad de los diferentes productos implicados y de la evolución del proyecto
- Preparación inicial de los entornos (creación de servidores, instalación de SO, instalación de compiladores, etc.)
- Instalación de los productos de IBM necesarios
- Configuración como servicios para arranque/parada automática con el servidor o Configuración necesaria
- Creación/configuración de Jobs en Jenkins de Metro para el despliegue de los distintos componentes en los diferentes entornos
- Instalación/despliegue de los productos necesarios en los diferentes entornos
- Pruebas
- Documentación de la instalación y configuración

API Gateway. Se instalará el API Gateway en la versión elegida más reciente compatible con el resto de productos necesarios, siendo Metro el que decida finalmente, tras la propuesta del adjudicatario, la versión definitiva. Se instalará en los servidores y con la configuración que se estime más adecuada para el proyecto.

Se deben contemplar al menos las siguientes fases:

- Decisión de la versión a instalar de acuerdo con las matrices de compatibilidad y de la evolución del proyecto
- Preparación inicial de los entornos (creación de servidores, instalación de SO, etc.)
- Instalación del producto
- Configuración como servicio para arranque/parada automática con el servidor



Metro de Madrid, S.A.

- Configuración necesaria para adaptar el producto a las necesidades del proyecto o En caso necesario creación/configuración de Jobs en Jenkins de Metro para el despliegue del producto o componentes que deba albergar
- Despliegue de los componentes necesarios
- Pruebas
- Documentación de la instalación y configuración

Oracle Secure Global Desktop o equivalente. Se instalará el producto OSGD o equivalente en la versión más reciente compatible con el resto de productos necesarios, siendo Metro el que decida finalmente, tras la propuesta del adjudicatario, la versión definitiva. Se instalará en los servidores y con la configuración que se estime más adecuada para el proyecto.

Se deben contemplar al menos las siguientes fases:

- Decisión de la versión a instalar de acuerdo con las matrices de compatibilidad y de la evolución del proyecto
- Preparación inicial de los entornos (creación de servidores, instalación de SO, etc.)
- Instalación del producto
- Configuración como servicio para arranque/parada automática con el servidor
- Configuración necesaria para adaptar el producto a las necesidades del proyecto o En caso necesario creación/configuración de Jobs en Jenkins de Metro para el despliegue del producto o componentes que deba albergar
- Despliegue de los componentes necesarios
- Pruebas
- Documentación de la instalación y configuración



Metro de Madrid, S.A.

3.2. Migración JBoss

Alcance de la actividad

Adaptación de SIAR para eliminar de la aplicación la explotación de las librerías JMS nativas del servidor de aplicaciones WebLogic y utilización de nuevas librerías provistas por JBoss u otras de libre distribución.

Configuración del aplicativo para la explotación de los nuevos pools de conexiones creados en el servidor de aplicaciones JBoss.

Se han localizado las llamadas realizadas en el código de SIAR a las librerías del módulo Oracle WebLogic JMS (WLS JMS Client) que necesitarán recodificación para dejar de explotar la del software propietario del servidor.

La configuración relativa a explotación de los pools de conexión se encuentra ubicada en el fichero de aplicación web.xml, se incorporarán los nuevos direccionamientos jndi de estos datasources.

Procedimiento de implantación

Parte de aplicación: Se realizará el despliegue de la aplicación sobre un servidor Jboss EAP versión 7.1.

Se deberá crear dentro del proyecto siar-web, directorio WEB-INF, los ficheros jboss-web.xml y jboss-deployment-structure.xml, el segundo de estos solo si es necesario modificar dependencias o gestionar exclusiones sobre los paquetes incorporados por defecto en el servidor Jboss. Además, puede ser necesario realizar adaptaciones en los archivos web.xml, faces-config.xml, faces-config-maestros.xml del proyecto siar-web, en función de los frameworks que se estén utilizando.

Adaptaciones en el código fuente de SIAR para la explotación de las nuevas colas de mensajería. Se han localizado las intervenciones a realizar en la tabla adjunta:

Módulo	Proceso/Operación
Módulo de agentes	Proceso de carga del Anual Actualizado
Módulo de agentes	Proceso de carga de la Situación Ligera del Agente
Módulo de vacaciones	Proceso de cálculo de las vacaciones



Metro de Madrid, S.A.

Módulo de concesiones e incidencias	Carga de PAPs
Módulo de concesiones e incidencias	Proceso de Gestión de cambios
Módulo de concesiones e incidencias	Incidencias visa
Punteo	Preparación del punteo
Punteo	Gestión de punteos rápidos
GR	Carga de GR
GR	Deficit/superavit
GR	Equipos de apertura
GR	Apertura de servicio
GR	Cierre de servicio
GR	Previsiones
Asignación y planificación	Carga de las peticiones de preferencia
Asignación y planificación	Control presencia visa (Asistencia reconocimiento médico)
Asignación y planificación	Obtención de servicios libres
Integración con SAP	Envío HorasExtra
Integración con SAP	Envío Punteo
Integración con SAP	Envío Previsión de vacaciones

Tabla 4 – Puntos de invocación JMS

Jboss 7.x incorpora por defecto una versión de JSF (Java Server Faces) más actualizada de la que actualmente explota SIAR. Se plantean en este punto dos estrategias de actuación posibles:

- Excluir estas versiones JSF nativas en un primer momento hasta garantizar el correcto funcionamiento de SIAR con su versión actual y posteriormente acometer la actividad de elevación de versiones.
- Comenzar desde el inicio de la instalación con las actividades de subida de versiones de los frameworks, incluyéndose los relacionados con la capa de presentación (JSF, Facelets y RichFaces).

Creemos mejor la segunda de las opciones, dado que la subida de frameworks se considera una actividad susceptible de promover mejoras en el funcionamiento global de la aplicación al tener estos una depuración/calidad de mayor solidez obtenida a lo largo de años de evolución.



Metro de Madrid, S.A.

3.3. Actualización Frameworks

Alcance de la actividad

Corrección del número de errores localizados durante los procesos de análisis de la solución tras la sustitución de los frameworks a sus versiones recomendadas.

Para determinar el alcance se ha hecho lo siguiente:

Incremento de las versiones de los frameworks actualmente utilizados a las versiones identificadas en la tabla adjunta sobre los proyectos SIAR ubicados en una de las máquinas de Metro de Madrid con las siguientes finalidades:

- a) Localización de clases en las que el incremento de versiones ocasione errores
- b) Clasificación de errores para su valoración y estimación.

Steps	Nombre	Versión origen	Librería sustituyente	Versión destino
1	java	1.7	java	1.8
2	spring	3.2.3	spring	5.2.9
3	spring-security-core	3.2.9	spring-security-core	5.4.1
4	hibernate	3.2.6.ga	hibernate-agroal	5.4.23.Final
5	Apache POI	3.6	Apache POI	4.1.2
6	Xstream	1.3.1	Xstream	1.4.14
7	Freemarker	2.3.16	Freemarker	2.3.30
8	JXL	2.6.10	JXL	2.6.12
9	CXF	2.6.2	CXF	3.4.1
10	SwingX	1.0	SwingX	1.6.1
	SwingFX	1.0	SwingFX	
11	jsf-api	1.2_12_metro	javax.faces-api	2.3
	jsf-impl	1.2_12		
	jsf-facelets	1.1.14		
12	richfaces-ui	3.3.3.Final	richfaces-components-ui	4.3.7
	richfaces-impl	3.3.3.Final.everis	richfaces-core-impl	4.3.7
13	jsp-api	2.0	javax.servlet.jsp-api	2.3.3
14	servlet-api	2.0	javax.servlet-api	4.0.1
15	el-api	2.0	javax.el-api	3.0.0

Tabla 2 – Versionado de frameworks realizado



Metro de Madrid, S.A.

El análisis efectuado clasifica los errores encontrados durante las pruebas de elevación de versiones en cuatro categorías principales:

- a) Errores a nivel de paquete/librería - Independientemente del número de clases afectadas por el error, se estima un tiempo unitario para su corrección
- b) Errores a nivel de clase - Independientemente del número de clases afectadas por el error, se estima un tiempo unitario para su corrección
- c) Errores a nivel de método - Se tienen en cuenta el número de clases afectadas por cuanto sea necesaria la reescritura de los métodos en ellas incluidas
- d) Errores a nivel de variable - Se tienen en cuenta el número de clases afectadas por cuanto sea necesaria la reescritura de los métodos en ellas incluidas

Bloque	Nivel	Nº de errores
Java, Spring, Hibernate	Package	20
Java, Spring, Hibernate	Class	703
Java, Spring, Hibernate	Method	2293
Java, Spring, Hibernate	Variable	21
JSF	Package	24
JSF	Class	46
JSF	Method	15
JSF	Variable	5
Richfaces	Package	522
Richfaces	Class	3566
Richfaces	Method	902
Richfaces	Variable	55
SwingX	Package	57
SwingX	Class	327
SwingX	Method	10
SwingX	Variable	1

Tabla – Resumen errores localizados

Nota: No se han detectado errores de compilación en las páginas con extensión .xhtml, si bien esto no indica que los componentes empleados en las mismas se comporten adecuadamente con las actualizaciones planteadas. Puede que se precisen esfuerzos adicionales para su ajuste (no medibles a partir del ejercicio de compilación)

Procedimiento de implantación

Java



Metro de Madrid, S.A.

En primer lugar se deberá elevar la versión actual de Java a la versión Java 8. Para esto sería necesario actualizar el POM padre de la aplicación. En este se define en dos oportunidades la versión de Java, una bajo la etiqueta *java.version.source* y otra bajo la etiqueta *java.version.target*, ambas deben ser modificadas para utilizar la nueva versión.

Para la configuración del entorno de desarrolladores, es necesario sustituir en el IDE en la parte de preferencias de Java el JRE (*Java Runtime Environment*) instalado, cambiando la JDK (*Java Development Kit*) actual por la JDK correspondiente a Java 8, que debe haberse instalado previamente.

Además, debe cambiarse la configuración de los *build path* de cada proyecto de SIAR para que utilice la nueva JDK.

Al momento de pasar esta actualización a un entorno de desarrollo, pre-productivo o productivo, será necesario especificar en Jenkins la JDK que se ha utilizado.

Spring

La actualización de versión del framework de aplicación Spring debe realizarse también modificando el POM padre de la aplicación. En este caso se encuentra bajo la etiqueta *project.spring.version* se debe modificar la versión existente con la versión actual más adecuada. Posteriormente deben actualizarse los proyectos mediante Maven y forzar la actualización de versiones.

Spring Security

La actualización de versión del framework de seguridad Spring Security debe realizarse modificando el POM padre de la aplicación. En este caso se encuentra bajo la etiqueta *project.spring.security.version* se debe modificar la versión existente con la versión más actual adecuada. Posteriormente se debe actualizar los proyectos mediante Maven y forzar la actualización de versiones.

Hibernate Object Relational Mapping

Para actualizar la versión de Hibernate ORM es necesario modificar el POM del proyecto siar-core. En este debe sustituirse la dependencia actual de *org.hibernate* por la dependencia correspondiente a la versión actual más adecuada. Después, se deben actualizar los proyectos mediante Maven y forzar la actualización de versiones.

Java Server Faces

Actualmente en la aplicación se utilizan 3 archivos java (JAR) para Java Server Faces. Estos son jsf-api, jsf-impl y jsf-facelets. Desde que el framework Java Server Faces



Metro de Madrid, S.A.

evoluciono a su versión 2.0 todos los archivos Java mencionados previamente se encuentran dentro de uno solo denominado javax.faces-api.

Para actualizar estos frameworks es necesario modificar el POM del proyecto siar-web eliminando las 3 dependencias de los frameworks bajo la etiqueta *artifactId* siguiente: jsf-api, jsf-impl y jsf-facelets. Se debe añadir la nueva dependencia javax.faces-api en su versión 2.3 que es el último lanzamiento disponible para este proyecto. Posteriormente se deben actualizar los proyectos mediante Maven y forzar la actualización de versiones.

RichFaces

En primer lugar mencionar que el proyecto ha alcanzado el final de su vida útil en Junio de 2016.

Actualmente SIAR utiliza dos archivos java (JAR) para este framework, estos son richfaces-ui y richfaces-impl. Para actualizar estas versiones es necesario modificar el POM del proyecto siar-web y sustituir las dependencias bajo la etiqueta *artifactId* siguientes: richfaces-ui y richfaces-impl por las dependencias richfaces-components-ui y richfaces-core-impl respectivamente, en sus versiones 4.3.7. Deberán actualizarse los proyectos mediante Maven y forzar la actualización de versiones.

Procedimiento frente a posibles problemáticas

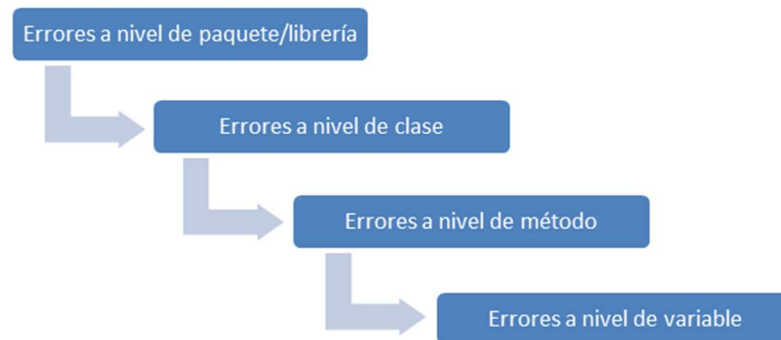
La estrategia recomendada para el proceso de migración en cuanto a la corrección de problemáticas debe sustentarse en iteraciones sucesivas, permitiendo detectar y corregir los problemas ocasionados por la actualización de cada framework. Este procedimiento permitiría adaptar eficazmente en caso de ser necesario el resto de librerías de soporte.

Orden de las correcciones

El análisis efectuado clasifica los errores encontrados durante las pruebas de elevación de versiones en cuatro categorías principales:



Metro de Madrid, S.A.



El orden de corrección de problemáticas en el software debe seguir el anteriormente citado, comenzando por errores catalogados como de paquete/librería y terminando por aquellos identificados como de variable.

El fin de este procedimiento de actuación parte de la consideración de que la corrección de errores en las capas superiores de aplicación (nivel paquete) podrá heredarse hacia las capas inferiores (niveles de clase y método), deshaciendo o corrigiendo problemas automáticamente.

3.4. Test de verificación

Alcance de la actividad

Para la realización de las pruebas para comprobación de la migración de SIAR en la nueva infraestructura con las versiones actualizadas de los frameworks, será necesario que el equipo de Metro de Madrid realice una identificación de los principales procesos que deberán ser probados.

Se incluye la verificación de la integración con otros sistemas externos.

Procedimiento de implantación

Test funcional global con el fin de verificar que la aplicación opera correctamente en su nuevo ecosistema de ejecución conformado por una nueva infraestructura física y lógica (caso de los servidores de aplicaciones) y con la actualización en el versionado de las librerías y frameworks que soportan el desarrollo.



Metro de Madrid, S.A.

Más allá de que otros procesos o funcionalidades puedan ser incorporados como prueba, a priori se determina en la siguiente tabla una batería de procesos sujetos a medición funcional.

Módulo	Prueba de verificación
Agentes	Ficha del Agente - Navegar por las diferentes pestañas de la ficha del agente
Condag	Lanzamiento de la condición de un agente para un año.
Condag	Lanzamiento de la condición de un agente para un mes.
Condag	Lanzamiento de la condición de un agente para un día.
Agentes	Consulta de la Situación del Agente
Agentes	Consulta de la Situación del Agente Ligera
Agentes	Consulta del Anual Actualizado
Agentes	Consulta del Periodo Jubilación Parcial
Incidencias	Consulta de Incidencias del agente
GR	Servicio Diario de Trenes - Consultar la ventana de reservas
GR	Servicio Diario de Trenes - Crear Incidencia
GR	Consulta del Estado de Estaciones
GR	Consulta del Estado de Trenes
GR	Mantenimiento Usuarios Áreas de Gestión - Alta y Consulta
GR	Carga del servicio diario
GR	Carga de las previsiones
GR	Reasignación por formulario Estaciones
GR	Reasignación por formulario MTE
ADC	Consulta Disponibilidad ADC
ADC	Carga Trenes/Rotaciones - Carga de Excel
ADC	Consulta de Cuadros de ADC
ADE	Hoja de Desplazamientos ADE - Alta y consulta
ADE	Consulta del informe de Disponibilidad Horaria
ADE	Carga Estructuras ADE - Consulta y carga de Excel
Punteo	Proceso de preparación del punteo
Punteo	Proceso de gestión de punteos rápidos
Punteo	Consulta de anomalías
Punteo	Consulta del punteo.
Punteo	Modificación manual del punteo
Asignaciones	Lanzamiento Anual MTE
Asignaciones	Lanzamiento Anual Jefe Sector
Asignaciones	Lanzamiento Vacante MTE
Asignaciones	Lanzamiento Vacante Jefe Sector



Metro de Madrid, S.A.

Asignaciones	Lanzamiento Avance Mensual Jefe Sector
Asignaciones	Lanzamiento Mensual Jefe Sector
Asignaciones	Lanzamiento Avance Diario MTE
Asignaciones	Lanzamiento Avance Diario Jefe Sector
Asignaciones	Lanzamiento diario MTE
Asignaciones	Lanzamiento diario Jefe Sector
Asignaciones	Lanzamiento Reconocimiento Médico
Asignaciones	Modificaciones sobre la asignación.
Asignaciones	Verificación de que la asignación se consolida y publica.
Asignaciones	Generación de fichero Excel con la publicación.
Mantenimientos	Alta de incidencias a agentes
Mantenimientos	Consulta de incidencias y cambios.
Mantenimientos	Lanzamiento del proceso de Gestión de Cambios.
Informes Birt	Consulta de ciertos informes Birt.
WebServices	Lanzamiento proceso de Situación Agente Ligera y anual actualizado
Cargas de sistemas externos	Carga de PAPs
Cargas de sistemas externos	Carga de Incidencias de Visa
Generación de ficheros	Generación del fichero de punteo para envío a RRHH
Concesiones	Gestión de cambios
Concesiones	Consulta de cambios
Concesiones	Validación inmediata de cambio de día libre entre agentes
Concesiones	Validación inmediata de cambio de día libre consigo mismo
Concesiones	Validación inmediata de cambio de servicio
Concesiones	Validación inmediata de cambio de FCs
Concesiones	Numero Cupo Agente - Consulta y Alta
Concesiones	Alta Solicitud Cambio Fiestas Compensadas por Descanso
Concesiones	Gestión Solicitud Reducción Jornada - Consulta y Alta
Maestros	Consulta calendarios
Vacaciones	Copia de Periodos de Vacaciones - Consulta y Copia
Vacaciones	Alta de Cambio de vacaciones Parcial
Vacaciones	Consulta Asignación Vacaciones RRHH
Vacaciones	Carga de Vacaciones de RRHH
Horas Extra	Gestión de Horas Extras - Consulta y Alta
Procesos Asíncronos	Consulta Procesos Asíncronos

Tabla 7 – Test de verificación funcional

Test de carga y rendimiento global para verificar que no existe degradación en el performance de la aplicación conforme a sus métricas habituales.



Metro de Madrid, S.A.

Más allá de que otros procesos o funcionalidades puedan ser incorporados como prueba, a priori se determina en la siguiente tabla una batería de procesos sujetos a medición relativa a rendimiento.

Módulo	Prueba
Condag	Lanzamiento de la condición de un agente para un año.
Condag	Lanzamiento de la condición de un agente para un mes.
Condag	Lanzamiento de la condición de un agente para un día.
Asignaciones	Lanzamiento Anual MTE
Asignaciones	Lanzamiento Anual Jefe Sector
Asignaciones	Lanzamiento Vacante MTE
Asignaciones	Lanzamiento Vacante Jefe Sector
Asignaciones	Lanzamiento Avance Mensual Jefe Sector
Asignaciones	Lanzamiento Mensual Jefe Sector
Asignaciones	Lanzamiento Avance Diario MTE
Asignaciones	Lanzamiento Avance Diario Jefe Sector
Asignaciones	Lanzamiento diario MTE
Asignaciones	Lanzamiento diario Jefe Sector
Asignaciones	Lanzamiento Diario + Avances MTE
Asignaciones	Lanzamiento Diario + Avances Jefe Sector
Asignaciones	Lanzamiento Reconocimiento Médico
Asignaciones	Publicación asignación Vacante
Asignaciones	Publicación asignación Avance Mensual
Asignaciones	Publicación asignación Mensual
Asignaciones	Publicación asignación Diario
GR	Carga del servicio diario
GR	Carga de las previsiones
GR	Reasignación por formulario Estaciones
GR	Reasignación por formulario MTE
Punteo	Proceso de preparación del punteo
Punteo	Proceso de gestión de punteos rápidos
Punteo	Modificación manual del punteo
Agentes	Proceso de carga de Situación del Agente Ligera
Agentes	Proceso de carga del Anual actualizado
Agentes	Consulta Situación del Agente
ADC	Consulta de la información de una línea.
ADE	Consulta de Hoja de día CAUPE para una línea en varias franjas
ADE	Consulta de Hoja de línea ADE



Metro de Madrid, S.A.

Concesiones	Gestión de cambios
Concesiones	Validación inmediata de cambio de día libre entre agentes
Concesiones	Validación inmediata de cambio de día libre consigo mismo
Concesiones	Validación inmediata de cambio de servicio
Concesiones	Validación inmediata de cambio de FCs
Cargas de sistemas externos	Carga de PAPs
Cargas de sistemas externos	Carga de Incidencias de Visa

Tabla 8 – Test de verificación rendimiento

3.5. Migración ODM & Migración CP/CPLEX

Alcance de la actividad

El objetivo respecto del motor de reglas y optimización es:

- 1) Implantación de los motores de reglas y optimización sobre la nueva infraestructura de forma que los resultados de los motores sean idénticos a los que había anteriormente.
- 2) Validar que la base de datos del Rule Execution Server se ha migrado de forma correcta a la nueva infraestructura y que sus nuevos conectores funcionan correctamente.
- 3) Instalación de los Rule Execution Server (RES) en los nuevos entornos de desarrollo, preproducción y producción.
- 4) Subir las versiones de las bibliotecas IBM ODM y CP/CPLEX que hay en SIAR y CLIENTE PESADO y aplicar el nuevo licenciamiento de IBM para que tenga soporte estableciendo una estrategia de licenciamiento durante el periodo de convivencia de anterior y nuevo SIAR de forma que se minimice el impacto en coste.
- 5) Adaptar los proyectos de reglas y optimización a las nuevas versiones de las bibliotecas de ODM y CP/CPLEX, teniendo en cuenta las características que se han eliminado en el salto de versión y desplegándolos como nuevos RuleApp sin sustituir a los anteriores.
- 6) Unificar las factorías de sesión en Java para los motores de reglas y reconfigurar los pool y carga del mismo al arranque de la aplicación.
- 7) Realizar un plan de automatización de pruebas unitarias contra los motores de reglas y optimización creando las aserciones necesarias para garantizar los mismos resultados en el antiguo RuleApp y en el nuevo, y que sirvan en el futuro como pruebas automatizadas para garantizar la regresión.



Metro de Madrid, S.A.

- 8) Realizar un plan de pruebas de rendimiento para comprobar que la calidad de respuesta en reglas y optimización sea óptima. Se tendrán en cuenta los procesos más críticos como la CONDAG y la coordinación de pruebas entre CONDAG y Motor de Asignación (Que también usa la CONDAG).
- 9) Validar inputs, ejecutar y verificar los outputs de las pruebas de integración y aceptación que diseñe Metro.

Tareas para la migración de ODM desde 8.8 a 8.10

Tras el estudio de los [puntos que van quedando obsoletos](#) en las características de IBM ODM, proyectos de reglas y bibliotecas ILOG se han identificado una serie de tareas proclives al cambio con el fin de adaptar el funcionamiento de los motores de reglas a la versión 8.10 de IBM ODM y corregir aquellas características que queden discontinuadas.

Para llevar a cabo la corrección es necesario comenzar por actualizar las bibliotecas de los motores de reglas en el SIAR actual y corregir las características obsoletas, adaptando así todos los proyectos de reglas para que funcionen como servicios de decisión con una versión moderna del motor de reglas.

Las tareas identificadas son las siguientes:

- Comenzar la migración sobre un entorno de desarrollo (se podrá utilizar inicialmente el nuevo entorno de preproducción) y realizar todas las tareas necesarias que se identifican una vez se disponga de SIAR y CLIENTE PESADO funcionando sobre la nueva infraestructura.
- Instalación de bibliotecas ODM 8.10 en SIAR actual y cliente pesado
- Corrección de características obsoletas en código marcadas como removed en la página de IBM (tanto en motor de asignaciones como en SIAR).
- Modificación de características obsoletas en los proyectos de reglas CA, DC, MA. Los proyectos de reglas modificados se desplegarán con una versión nueva pudiendo utilizar la versión nueva o antigua si fuera necesario.
- Corrección en SIAR de factorías de singleton de sesión y carga de pool al arranque
- Instalación de Rule Execution Server en la nueva infraestructura (JBoss) sobre una máquina Linux virtualizada con los recursos necesarios adaptados al licenciamiento.
- Validar la migración de la base de datos del RES que será actualizada o migrada y garantizar el funcionamiento del conector de la misma.



Metro de Madrid, S.A.

- Se traza un plan de automatización de pruebas contra el motor J2SE o motor del RES necesarias identificando mocks de entrada al motor y aserciones a la salida.
- Una vez validadas las pruebas se realizarán los despliegues en el siguiente entorno.
- Se valida un plan de pruebas funcionales y de aceptación realizado por Metro para comprobar el funcionamiento de todos los proyectos de reglas.

Respecto a las tareas propias de la instalación y configuración de los nuevos Rule Execution Server así como para la migración de los proyectos de reglas existen algunas referencias propias que se utilizarán como referencia para la migración:

- [Instalación ODM](#)
- [Migración del motor de reglas clásico al motor de decisiones](#)
- [Migración de proyectos de Eclipse](#)
- [Migración del entorno de ejecución](#)
- [Etiquetado SLM de ILMT para instalación en JBOSS](#)

Tareas propias para la migración de IBM CP/CPLEX a 12.10

- Comenzar la migración sobre un entorno de desarrollo (se podrá utilizar inicialmente el nuevo entorno de preproducción) y realizar todas las tareas necesarias que se identifican.
- Instalación de bibliotecas CP/CPLEX 12.10 en el motor de asignaciones.
- Corrección de características obsoletas en código marcadas como removed en la página de IBM (motor de asignaciones) y compilación del código C++ útil para el contexto JNI en un entorno Linux.
- Se traza un plan de automatización de pruebas necesarias intentando recuperar las que ya se hicieron que se basan en historias de usuario.
- Una vez validadas las pruebas se realizarán los despliegues en el siguiente entorno.
- Se valida un plan de pruebas funcionales y de aceptación realizado por Metro para comprobar el funcionamiento del motor de asignaciones desde el cliente pesado.

Versiones objetivo de bibliotecas de ODM y CPLEX

A continuación se muestran las versiones de las bibliotecas de IBM ODM y CP/CPLEX que se tienen en la actualidad y las que serán objetivo de la migración:



Metro de Madrid, S.A.

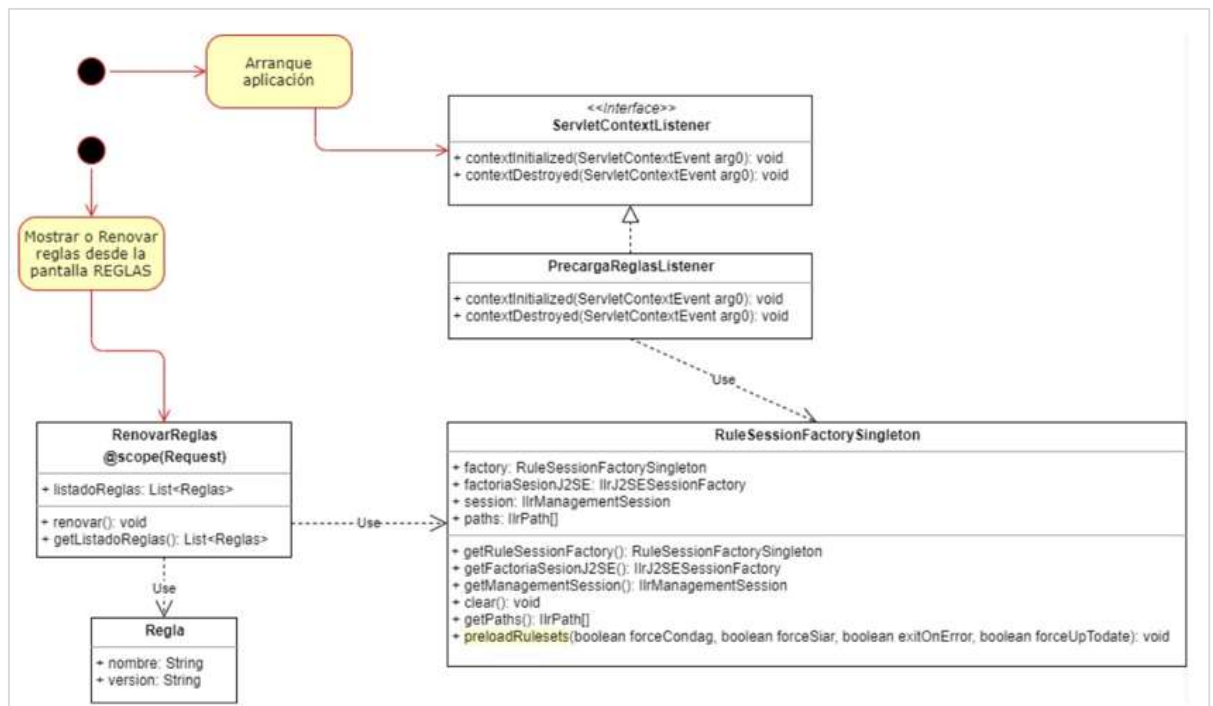
Producto	Versión actual	Versión tras la migración
IBM ODM	8.8	8.10.4
IBM CP/CPLEX	12.6	12.10.0.0

Tabla 9 – versiones de bibliotecas

Tunning J2SE en motor de reglas en SIAR

Existen dos puntos de mejora en el código Java de la aplicación de SIAR respecto de reglas y optimización:

- 1) Utilizar el mismo singleton que contiene la sesión de reglas tanto para los RuleApps que se precargan al arranque de SIAR como los que son ejecutados en el motor posteriormente (actualmente está duplicado).
- 2) Hacer un Tunning de la configuración del tamaño del pool para evitar la carga y liberación de RuleApps en memoria, siempre que no suponga un problema sustancial en la ocupación de recursos de memoria.



Tunning J2SE

Tras el análisis del código Java de SIAR se han encontrado dos instancias de factoría de sesiones del motor de reglas y debería de existir una única.

Una de ellas se llama RuleSessionFactorySingleton y carga al inicio al arranque de la aplicación (PrecargaReglasListener) y precarga en el motor todos los rulesets de CA y



Metro de Madrid, S.A.

DC, para que queden disponibles en el pool memoria para ejecutarse. Esta factoría está bien definida con el patrón singleton.

Por otro lado, existe otra instancia que es JRulesInvokerImpl que se carga como un bean en el contexto de Spring (applicationContext-negocioComun.xml) y tiene configurado el scope por defecto que es singleton. Esta instancia se obtiene del contexto cada vez que se va a invocar a reglas y también se compone de una sesión de reglas.

El problema es que existe una duplicidad de las sesiones de reglas en dos puntos diferentes de la aplicación. La precarga en diferentes sesiones hace que los rulesets se precargan en un pool y este pool no se vuelve a usar, puesto que el pool de sesión para ejecutarse es diferente. Por tanto, el pool utilizado para ejecución parte estando vacío y necesita ir precargando todas y cada una de las RuleApps.

Se mejorará el hecho de unificar los singleton de estas factorías en uno solo y que todos los rulesets queden bien precargados en el pool del motor de reglas para que tengan visibilidad y disponibilidad directa en memoria en cada petición a reglas. (Ampliación del tamaño del pool).

Por último, hay que verificar bien la clase RenovarReglas ya que también precarga las reglas en memoria dentro de un scope tipo Request. Si se desea se podrá reactivar la pantalla que gestiona la precarga del pool, que fue desactivada anteriormente. Esta precarga se hace dentro de la sesión de RuleSessionFactorySingleton.

Descripción del plan de automatización de pruebas

La automatización de pruebas comprende que cada uno de los proyectos de reglas contenga uno o varios casos de prueba para verificar el correcto funcionamiento del mismo mediante aserciones en la salida de su ejecución.

El objetivo de estas pruebas es doble:

- 1) Verificar que el resultado de la prueba es idéntico en el proyecto de reglas sin migrar y migrado.
- 2) Que sirva como automatización de pruebas para comprobar la regresión en la futura modificación en los proyectos de reglas antes de su despliegue, de forma que se valide que el resultado sigue siendo el correcto o es necesario que se modifique por un cambio de criterio.



Metro de Madrid, S.A.

Las pruebas se implementarán en Java, Junit directamente o haciendo uso de SoapUI y siendo lanzadas directamente sobre el decisión Service.

Durante el diseño de los casos de uso se ha de realizar una recopilación de datos de entrada para cada a cada proyecto de reglas. Para ello Metro necesita identificar los puntos de SIAR desde donde se realizan las llamadas a los diferentes proyectos de reglas. De esta forma se podrá ubicar qué acción en la web o motor de asignación desemboca una llamada en el motor y poder capturar así los datos de entrada al mismo para que sirvan como referencia para el caso de uso.

Descripción de las pruebas de rendimiento

Las pruebas de rendimiento sirven para garantizar el no deterioro del rendimiento de los motores de reglas y optimización migrados respecto a los anteriores, así como verificar el comportamiento frente a carga.

Se definirá un conjunto de pruebas junto con las personas adecuadas que conozcan en profundidad los procesos críticos respecto al funcionamiento 'end to end', como lo pueden ser la CONDAG, o ciertos procesos de ASIGNACIONES. Las pruebas de CONDAG y asignaciones deben estar sincronizadas ya que la CONDAG se utiliza en la parte web y en los procesos de ASIGNACION.

Se realizarán diferentes ejecuciones en el nuevo entorno de preproducción y se compararán los resultados con el entorno de producción actual.

Se ha de tener en cuenta que las ejecuciones en el entorno actual de SIAR se han de realizar antes de que dicho entorno deje de utilizarse o se desmonte. Estas ejecuciones serán utilizadas para tener un punto de referencia respecto de las pruebas de rendimiento que se hagan sobre la nueva infraestructura.

Como parte de las pruebas de rendimiento estará incluido el diseño del plan de pruebas, su ejecución y la comparación de los resultados.

Descripción de las pruebas de integración y aceptación



Metro de Madrid, S.A.

Para las pruebas de integración se volverá a lanzar la automatización de pruebas en los diferentes entornos donde se vaya desplegando y de esta forma validar unitariamente cada uno de los proyectos de reglas por separada.

Aparte Metro diseñará un conjunto de casos de uso que sirvan como acuerdo para el resultado de las pruebas de aceptación y dar así por válido el buen funcionamiento de SIAR y el MOTOR DE ASIGNACIONES una vez migrado sobre preproducción.

Morfología de los proyectos de reglas

- **CA:** Condición del agente. Lo usan tanto el motor de asignación como SIAR.
- **DC:** Diseño común y dentro de este están los de **GR:** Gestión de recursos.
- **MA:** Motor asignación (cliente pesado).

Proyectos de reglas	
DC	32
CA	47
MA	13
Total	92

Proyecto	Flujos	Reglas	Tablas	Árboles	Reglas Tec	Funciones
DC	59	453	4	0	0	0
CA	150	628	35	118	0	0
MA	182	565	0	3	91	53
Total	391	1646	39	121	91	53

Proyecto	% por num proyectos	W num proyectos	% por reglas y tablas	W núm. reglas	Media	Relación
DC	35%	10	27%	1	35%	2,2
CA	49%	10	39%	1	48%	2,8
MA	15%	10	34%	1	17%	1

Tabla 10 – Proyectos IBM



Metro de Madrid, S.A.

Características eliminadas en la versión de ODM 8.10

En las nuevas versiones de IBM ODM 8.10 el fabricante ha definido una serie de características que quedan obsoletas pero que aún tienen continuidad (deprecated) y una serie de características que directamente han sido eliminadas (removed).

El objetivo de la migración del motor de reglas es adaptar aquellas características (removed) que dejan de tener soporte y que quedan definidas en la siguiente tabla:

Deprecated feature	Module	Recommended migration	Removed
IBM Cloud Private deployment	IBM Operational Decision Manager on IBM Cloud Private	Move to one of the following options: <ul style="list-style-type: none">- Operational Decision Manager on Certified Kubernetes Learn more...- Operational Decision Manager in the context of the IBM Cloud Pak® for Automation Learn more...	Removed from V8.10.4. Still available in V8.10.3.
IBM WebSphere MQ V7.5	Decision Server Insights	Use IBM WebSphere MQ V8.0 or V9.0. Learn more...	Operational Decision Manager V8.10.0
Operational Decision Manager Pattern		Move to one of the following options: <ul style="list-style-type: none">- Operational Decision Manager on Certified Kubernetes Learn more...- Operational Decision Manager in the context of the IBM Cloud Pak for Automation Learn more...	Operational Decision Manager V8.10.0
Support for Java 7	All modules	Use Java 8. Learn more...	Operational Decision Manager V8.10.0
Enterprise JavaBeans (EJB) API for Rule Execution Server	Decision Server Rules	Use POJO rule sessions. Learn more...	Operational Decision Manager V8.10.0
Message-driven rule beans (MDB) API for Rule Execution Server	Decision Server Rules	Use POJO rule sessions. Learn more...	Operational Decision Manager V8.10.0
Insight Map Viewer	Decision Server Insights	As an alternative to using the Insight Map Viewer to represent geospatial attributes on a map, you can follow the steps in the following article on ODMDev Center: Viewing maps in Decision Server Insights	Operational Decision Manager V8.9.1.0



Metro de Madrid, S.A.

Deprecated feature	Module	Recommended migration	Removed
Import wizard	Decision Server Rules and Decision Center	Use the Samples Console in Rule Designer to import samples and tutorials. Learn more...	Operational Decision Manager V8.9.0
Decision Server Events	Decision Server Events	The capabilities provided by Decision Server Events have been replaced by corresponding capabilities in Decision Server Insights. Decision Server Insights is a scalable transactional event processing system with rule-based temporal reasoning and analytics capabilities. Learn more...	V8.9.0
Rule Solutions for Office	Rule Solutions for Office	The Decision Center Business console provides a powerful web-based decision table editor. Since V8.8.1, this includes exporting a decision table to Excel. Learn more...	V8.9.0
Scorecard Modeler	Scorecard Modeler	Implement a scoring system based on regular decision tables, or invoke an external scoring service. Learn more...	V8.9.0
Business Rules Embedded	Business Rules Embedded	Adopt Decision Center and its web components for a custom UI, and Decision Server Rules for execution with the decision engine. Learn more...	V8.9.0
Managed Transparent Decision Services (MTDS)	Decision Server Rules	Use HTDS or create your own web service around RES session with the Java API. Learn more...	Operational Decision Manager V8.10.0

Tabla 11 – Deprecated features

Características eliminadas en la versión de CP/CPLEX 12.10

The parameter [time spent polishing a solution \(deprecated\)](#), CPX_PARAM_POLISHTIME, has been deprecated and will be removed in a future release.

The value -1 (meaning that the user specified to CPLEX to perform no crossover at the end of barrier optimization) of the parameter [barrier crossover algorithm](#), CPXPARAM_Barrier_Crossover, was deprecated in a previous release and has now been **removed**. Other values of this parameter remain valid.

CPLEX Benders solver properly deals with the case of a problem with an unbounded master problem, which is tackled by iteratively separating Benders cuts that cut off unbounded rays of the continuous relaxation of the master problem, until either the problem becomes bounded, or the complete MILP is proven to be infeasible or unbounded. Therefore, the solution status values CPXMIP_BENDERS_MASTER_UNBOUNDED and CPX_STAT_BENDERS_MASTER_UNBOUNDED cannot occur anymore.



Metro de Madrid, S.A.

Error code

The error code CPXERR_PRM_HEADER was deprecated in a previous release and has been **removed**.

The header in CPLEX parameter (PRM) files is now optional and, therefore, this error cannot occur in the future. See PRM file format: parameter settings.

Procedimiento de implantación

Requisitos necesarios previos a la migración del motor de reglas y optimización

Como primer paso es necesario mover SIAR a la nueva infraestructura instalada con entorno virtualizados Linux para desarrollo, preproducción y producción, con los siguientes requisitos previos:

Para cada entorno de SIAR:

- Nueva VM, con Java 8 y servidor de aplicaciones JBOSS con su licencia aplicada.
- SIAR actual funcionando y aceptado en ese servidor de aplicaciones JBOSS.
- Licencias ODM adquiridas.

Para cada entorno de cliente pesado:

- Nueva VM con Java 8 y Oracle SGD instalado y configurado tal y como está ahora con su licencia aplicada.
- Cliente pesado/motor asignaciones instalado tal y como está ahora.
- Licencias ODM y CPLEX adquiridas.

Para cada entorno de Rule Execution Server (si está separado de SIAR):

- Nueva VM con servidor de aplicaciones JBOSS con su licencia aplicada.

Para cada entorno de base de datos:

- Base de datos del RES migrada a nueva versión de Oracle.
- Conectores actualizados que apunten a las bases de datos migradas para cada entorno.

Para las pruebas automatizadas y de aceptación:

- Se requiere que personas diferentes a las que realicen la migración y preferiblemente que conozcan funcionalmente la aplicación trabajen en diseñar un plan de pruebas para todos los proyectos de reglas DC, GR, CA, MA.



Metro de Madrid, S.A.

Las pruebas de aceptación se realizarán validando funcionalmente que para los mismos datos de entrada se obtengan datos de salida idénticos tanto en la antigua versión de SIAR como a actual ². Esto define un hito o punto crítico necesario previo a la migración, que es la definición de las pruebas funcionales entre el proveedor y Metro. Estas pruebas deben contemplar el funcionamiento de todos los proyectos de reglas para poder validarlos de forma completa. Para ello se hace necesario capturar los datos de entrada y salida en el motor de reglas para el SIAR actual en cada uno de esos casos de uso definidos. Una vez se tengan estos datos de entrada e identificado el lugar desde el cual se lanzan en la aplicación de forma manual entonces se podrá trazar un plan de automatización de pruebas de forma que se lancen las aserciones unitarias correspondientes a cada llamada de reglas. Esta automatización de pruebas servirá de ahora en adelante como punto de comprobación para las pruebas de regresión para los futuros cambios que se hagan en reglas y futuras migraciones. Las pruebas se realizarán en preproducción.

[2] Antes de desmontar SIAR anterior se harán las pruebas de rendimiento para poder tener una referencia en la comparativa con la infraestructura.

3.6. Test Carga (20 casos prueba)

Alcance de la actividad

Diseño, ejecución y análisis de 20 casos de prueba relativas a carga del sistema y rendimiento.

Se proponen 20 casos de prueba en sincronía con la actividad de refactorización del 15% de los accesos a base de datos a través del ORM de aplicación (Hibernate) descrito en el apartado 3.7

Procedimiento de implantación

La ejecución de estas pruebas permite verificar detalladamente el funcionamiento global del sistema. Se ejecutan bajo un enfoque de caja negra. Se deben realizar después de haber comprobado que cada módulo funciona correctamente, así como la integración entre ellos.

Se realizan en un entorno que permita la certificación del sistema y tiene ya características técnicas similares a lo que será el entorno real.



Metro de Madrid, S.A.

- a) **Tareas previas.** Previo a la realización de las pruebas es fundamental la estrategia a seguir. Definición y conocimiento de las operativas a probar, componentes técnicos que intervienen, y por lo tanto la funcionalidad crítica que debe ser probada a nivel de rendimiento de la aplicación.

Del mismo se realizará la instalación de herramientas necesarias para la ejecución de las tandas de test sobre un entorno de similar dimensionamiento al productivamente existente y en completo aislamiento para evitar la existencia de interferencias que desvirtúen los resultados obtenidos.

- b) **Grabación de scripts.** El equipo debe disponer de operativas estandarizadas y buenas prácticas en las tareas de grabación, correlación y parametrización de derivan en un incremento significativo a corto plazo en la efectividad y calidad de las pruebas realizadas.
- c) **Modelización de escenarios.** La modelización de los escenarios no es sino la implementación de la estrategia de pruebas, definiendo el número de usuarios concurrentes, la rampa de entrada, tiempo de meseta, thinktime, pacing, etc. Definiendo un tipo de prueba u otra se procura reproducir lo más fielmente posible la actividad de la aplicación en el entorno productivo. Se implantarán unos estándares respecto a la creación de los escenarios de prueba es imprescindible para unificar criterios y mejorar los estudios.
- d) **Definición de la monitorización.** El estudio de las métricas de las distintas capas que intervienen en la aplicación es fundamental para determinar el rendimiento de las aplicaciones. Se definirán los umbrales de riesgo pertinentes para unificar criterios respecto a la evaluación del rendimiento de la aplicación.
- e) **Ejecución e informe de resultados.**

3.7. Refactorización 15% SQL/ORM

Alcance de la actividad



Metro de Madrid, S.A.

Refactorización en el uso de Hibernate ORM por parte de la capa de aplicación encargada de comunicar con la base de datos y consultas SQL empleadas en un porcentaje estimado del 15% de los accesos a datos de SIAR al SGBD Oracle.

Se localizan todos los puntos de la aplicación en los cuales la aplicación genera consultas a la base de datos, distinguiendo como las más críticas aquellas encapsuladas en objetos de acceso a datos que emplean sentencias de tipo lectura (SELECTS) y las cuáles retornan colecciones de datos. A este respecto se han detectado 577 métodos ubicados en los DAOs del paquete siar-core que cumplen este fin.

Se ha distribuido el número total de accesos a datos (lecturas que retornan colecciones) según la siguiente estructura de dificultad:

- a) Grado alto – en torno al 15% de las sentencias
- b) Grado medio – en torno al 42,5% de las sentencias
- c) Grado bajo – en torno al 42,5% de las sentencias

Procedimiento de implantación

Con la información obtenida en la anterior etapa del proyecto (test de carga), tanto en los puntos críticos en base de datos, es decir, en las sentencias y procesos más pesados, como en las zonas más críticas de la aplicación en las cuales se determine que la capa de acceso a datos presenta peores rendimientos, se procederán a ejecutar los trabajos de:

Recodificación

Tunning

Reconstrucción de sentencias SQL

Estas acciones se realizaran con las siguientes finalidades:

- 1) En primer lugar, mejorar el rendimiento de aquellas consultas cuyo coste de ejecución suponga mermas en el rendimiento de la aplicación.
- 2) Derivada de la anterior, impedir un alto grado de procesamiento en la base de datos, liberando recursos de la misma.



Metro de Madrid, S.A.

Los trabajos a realizar en este punto quedan determinados por las siguientes actividades:

Con las métricas obtenidas en la fase de pruebas de carga y rendimiento para los 20 casos de prueba diseñados, se procede a:

Realizar una analítica de sentencias ejecutadas en base de datos, tanto en el plano de rendimiento ofrecido por la sentencia, bien en volumen de información recuperada bien en tiempo de localización de resultados, como por la estructura de la sentencia para diseñar un nuevo plan de ejecución de la misma o customizar el ya existente, atendiendo a estrategias de indizado, hinteo, yuncionado o recodificación como fórmulas para obtener rendimientos óptimos en las actividades propias de base de datos.

A partir de este punto, y dado que estas consultas son construidas en la capa ORM de la aplicación, se procederá a la recodificación del software Java encargado de la citada construcción y bajo las directivas de buen funcionamiento especificadas desde la capa de base de datos.

Esta recodificación afectará fundamentalmente a las estructuras Hibernate de construcción HQL (examples y criterias), no descartándose la eliminación, en caso de aplicar, de esta estrategia de consulta para pasar a una construcción estática de las órdenes (Plain SQL).

El último de los puntos susceptibles de medición en este ámbito se trata del proceso de fetch o transporte de resultados a la capa de aplicación desde la base de datos, bien por deterioros de performance derivados de un elevado número de resultados bien por la estrategia de lectura de los cursores empleada.

Respecto a este último punto comentado, indicar que se ha detectado que las capas de presentación de la aplicación realizan una paginación durante la exposición de tablas de resultados obtenidos de base de datos propia del componente de presentación (JSF) y no derivada de una paginación real en base de datos, entendiéndose que este punto provoca sobreesfuerzos en base de datos y capa de acceso a datos para cargar unos resultados que en muchas ocasiones no son accedidos por el usuario de aplicación.

La corrección de este punto, fetch innecesario, puede derivar en mejoras de rendimiento sobre todas las capas y componentes de la aplicación.

El índice por tanto de trabajos incorporados en esta actividad se resume en la imagen adjunta:



Metro de Madrid, S.A.

Capa de aplicación

- ❑ Construcción de la sentencia
- ❑ Recuperación y volumen de información

Base de datos

- ❑ Comportamiento de la sentencia

3.8. API Gateway

Alcance de la actividad

Se considera dentro del alcance de este proyecto la implementación del API Gateway dentro de la arquitectura de SIAR, con la respectiva instalación, configuración y posterior uso dentro de la comunicación entre las capas de Back y Front de SIAR Web, de forma que se securice y controle las llamadas entre la capa Front y la capa Back, además de localizar de una manera más eficiente posibles errores. Esto se debe porque tras la implantación de la nueva tecnología de frontales Angular, se estima que las funcionalidades y pantallas de SIAR estarán divididas por módulos funcionales, con lo que a nivel Front existirá un desacoplamiento de la arquitectura.

Procedimiento de implantación

Los pasos para el proceso de implantación serían los siguientes:

- Revisar entorno de infraestructura para la implantación.
- Lanzar la instalación de montaje de todos los productos necesarios en las máquinas.
- Probar la instalación.
- Testear que has dado de alta correctamente la organización.
- Crear el primer proxy de tus sistemas backend para comprobar que todo funciona correctamente.

Una vez realizados estos pasos, se deberán probar los distintos servicios a exponer según se vaya realizando su desarrollo.



Metro de Madrid, S.A.

3.9. Sustitución Frontales Web

Alcance de la actividad

El principal objetivo de este apartado es el de especificar una solución correcta y eficaz para un óptimo funcionamiento de los frontales de SIAR de Metro de Madrid. Se mencionarán los puntos débiles actuales y como se solucionarán tras la evolución a las nuevas tecnologías.

Dentro del alcance de este apartado también se contempla que SIAR WEB podrá autenticarse a través de los métodos existentes en Metro de Madrid, específicamente ADFS.

Análisis técnico

Diferencias y recomendaciones en la sustitución de frontales RichFaces a Angular:

- SPA vs MPA
- Server Side Rendering vs Client Side Rendering
- Monolito vs Web Components

SPA vs MPA: En líneas generales, una **MPA** (Multi Page Application), es un aplicativo web que requiere recargar la página para navegar sobre él. Por el contrario, una **SPA** (Single Page Application), no requiere recargar la página para navegar ya que su contenido se modifica dinámicamente.

El concepto de SPA surge mucho más recientemente que el de MPA y fue adoptado por empresas modernas como Google, Facebook, Twitter, etc. Sin embargo, ambas ofrecen pros y contras

Velocidad. Una SPA ofrece una velocidad de carga y navegación mucho mayor que una MPA ya que no necesita recargarse completamente cada vez que se



Metro de Madrid, S.A.

efectúa un cambio de vista. Los recursos compartidos del aplicativo web se mantienen cargados todo el tiempo y solo es preciso cargar en cada caso la información extra necesaria para renderizar una u otra vista. Además, la información puede cargarse en pequeños incrementos en lugar de precisar una carga completa para empezar a renderizar.

Acoplamiento. Una SPA está fuertemente desacoplada ya que el front y el back se encuentran separados. Las SPAs consumen APIs que les proporciona el servidor para leer y representar información. En las MPAs, front y back son interdependientes.

Search Engine Optimization (en adelante como SEO): Una MPA ofrece mejor posicionamiento web ya que cada página puede optimizarse por separado para aumentar su ranking de posicionamiento. Si el posicionamiento web es muy importante, como un E-Commerce o un Marketplace, una MPA podría ser la opción más interesante.

Experiencia de Usuario. Una SPA ofrece mucha mayor compatibilidad entre dispositivos y, al renderizarse mucho más rápido, una experiencia más fluida. En una MPA tenemos las vistas más diferenciadas entre ellas y la navegación es clara, aunque nada dinámica.

Seguridad: En una MPA necesitamos securizar cada vista por separado, mientras que en una SPA solo necesitamos securizar los endpoints que consuma.

Desarrollo. Una de las mayores ventajas de una SPA es que podemos reutilizar todo el código backend y podemos codificar front y back en paralelo, lo que reduce muy considerablemente el tiempo total necesario.

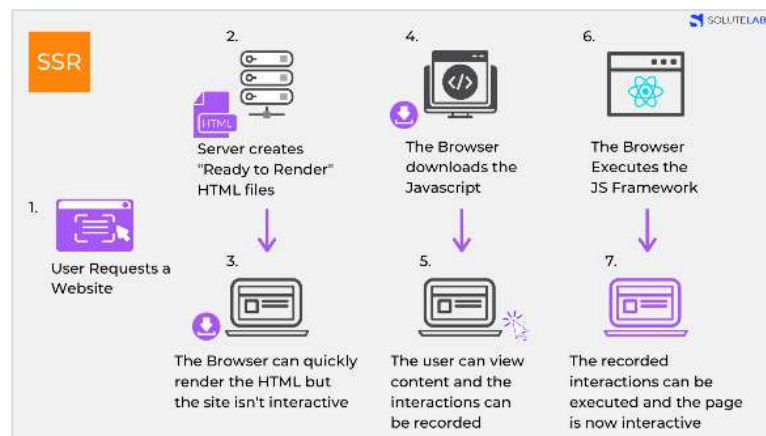
En conclusión, ya que las necesidades de Metro de Madrid no implican un fuerte posicionamiento SEO y sí una buena velocidad de funcionamiento, **una SPA es la mejor opción, aunque en la actualidad se está utilizando una MPA.** Además, esta opción implica **menores tiempos de desarrollo, mayor facilidad de mantenimiento y una mejor experiencia de usuario.**



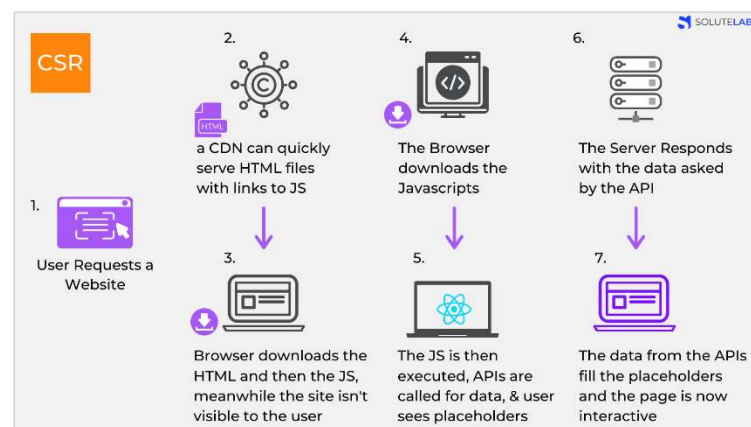
Metro de Madrid, S.A.

Server Side Rendering vs Client Side Rendering

Server Side Rendering (en adelante SSR) significa que el renderizado de la vista web se lleva a cabo por parte del servidor y se envía directamente al frontal para ser renderizado tal cual lo recibe.



Client Side Rendering (en adelante CSR) consiste en delegar en el cliente la responsabilidad de renderizar las vistas, obteniendo solo la información necesaria por parte del servidor.



Velocidad. Con CSR obtenemos una velocidad de carga y navegación mucho mayor que con SSR, ya que no es necesario esperar a que el servidor renderice y envíe toda la información. Hoy en día prácticamente cualquier dispositivo moderno es capaz de realizar esta tarea sin esfuerzo, liberando al servidor de una gran carga. Además, la información puede cargarse en pequeños incrementos en lugar de precisar una carga completa para empezar a renderizar.



Metro de Madrid, S.A.

Acoplamiento. Con CSR podemos desacoplar front y back, mientras que con SSR irían interconectados.

SEO. El SSR permite mejor posicionamiento web ya que cada página puede optimizarse por separado para aumentar su ranking de posicionamiento. Si el posicionamiento web es muy importante, como un e-commerce o un Marketplace, SSR podría ser la opción más interesante.

Experiencia de Usuario. Permitir que el cliente tome las riendas del renderizado permite utilizar un amplio abanico de efectos y transiciones que conformen una mejor experiencia de usuario, además de la ganancia de velocidad y fluidez que supone.

Seguridad. Mientras que en SSR necesitamos securizar cada vista por separado, con CSR solo necesitamos securizar los endpoints que consuma la aplicación.

Coste. CSR nos permite liberar al servidor de carga de trabajo, lo que implica un menor coste, ya que cada cliente que consuma la aplicación está responsabilizándose de una parte importante del trabajo. Además podemos reutilizar todo el código backend y codificar front y back en paralelo, lo que reduce muy considerablemente el tiempo total necesario.

En conclusión, debido que las necesidades de Metro de Madrid principalmente necesita de una buena velocidad de funcionamiento, **utilizar CSR sería la mejor opción**, aunque en su lugar se está utilizando SSR en la actualidad. Además, esta opción implica un **menor gasto en servidores, menores tiempos de desarrollo, mayor facilidad de mantenimiento y una mejor experiencia de usuario**.

Monolito vs MPA

Se denomina monolito al front-end que consta de un único bloque indivisible de código, mientras que la arquitectura basada en Web Components consiste en dividir la aplicación en diversos bloques de código independientes que se agrupan para formar estructuras más complejas.

Las principales diferencias son:



Metro de Madrid, S.A.

- **Reusabilidad.** Una arquitectura basada en Web Components permite reutilizar un Web Component tanto en múltiples puntos de una misma aplicación, como en aplicaciones externas, maximizando la rentabilidad del esfuerzo invertido en su desarrollo.
- **Legibilidad.** Una aplicación monolito puede resultar extremadamente complicada de entender y mantener, sobre todo si es de gran tamaño. Dividirla en Web Components facilita muchísimo esta tarea, ya que tenemos bloques diferenciados a los que acceder para centrarnos únicamente en lo necesario.
- **Testing.** Testear una aplicación basada en Web Components resulta mucho más sencillo que testear una aplicación monolito, ya que tenemos pequeños trozos de código independientes para testear unitariamente. Esto resulta crucial para asegurar la calidad y fiabilidad del proyecto.
- **Mantenimiento.** Resulta mucho más sencillo solucionar un problema, añadir funcionalidad o realizar mejoras de cualquier tipo si tenemos un pequeño fragmento de la aplicación aislada del resto y podemos centrarnos exclusivamente en él. Además, de este modo es mucho más complicado causar daños colaterales.

En conclusión, **al tratarse de frontales de gran tamaño, resulta muy beneficioso hacer una división en Web Components para ganar reusabilidad de código, aumentar la legibilidad y mantenimiento del mismo y facilitar su testeo de cara a asegurar su calidad y fiabilidad a largo plazo.** Sin embargo, en la actualidad, no se está llevando a cabo dicha composición.

Framework

ANGULAR 10+



Angular es un Framework de diseño de aplicaciones web y una plataforma de desarrollo para crear SPAs eficientes y sofisticadas. Desde sus orígenes en 2014, ha sido un rotundo éxito y no ha dejado de evolucionar, contando ya con más de 10 versiones oficiales de trayectoria. Desarrollado por Google y de uso completamente gratuito, resulta una de las mejores alternativas para cualquier tipo de aplicación web, especialmente para SPAs complejas.



Metro de Madrid, S.A.

Sus principales características son:

Velocidad y rendimiento

- Angular optimiza el código JavaScript para obtener el mejor rendimiento en las máquinas virtuales actuales.
- Gracias a su componente Router, el código se divide de modo que los usuarios solo cargan el necesario para renderizar lo que necesiten en cada momento.
- Al ser universal, renderiza casi instantáneamente independientemente del servidor y la tecnología que éste utilice.

Arquitectura basada en componentes

- Angular ofrece de forma nativa una arquitectura basada en componentes y proporciona todo lo necesario para establecer una comunicación óptima entre ellos. Por lo tanto, presenta una sinergia natural de cara a utilizar Web Components.
- Cada "pieza" de la UI puede ser un componente con su propia lógica y estilos que es totalmente reutilizable, legible y fácilmente mantenible.
- Por defecto, cada componente lleva asociado su propio conjunto de tests unitarios que ayudan a garantizar la calidad y fiabilidad del código.

Soporte

- Angular pertenece a Google, lo cual supone una garantía en lo que a soporte se refiere.
- Al tratarse de una de las herramientas más utilizadas en el mundo, tiene una gran comunidad de desarrolladores, lo cual facilita mucho la contratación de personal cualificado.
- El proyecto está en constante evolución, por lo que existe la posibilidad de realizar sencillas actualizaciones para seguir a la última.

Multiplataforma

- Angular ofrece compatibilidad multiplataforma de forma nativa, por lo que el código puede ser reutilizado para escritorio, móvil, Tablet, etc.
- También es posible crear aplicaciones de escritorio compatibles con los principales sistemas operativos (Windows, Linux y Mac).

TypeScript

- Angular utiliza TypeScript de forma nativa, es decir, utiliza código tipado para mayor consistencia y prevención de errores.
- A diferencia de JavaScript, TypeScript se parece mucho a Java y a muchos lenguajes de programación orientados a objetos (los más populares), lo que facilita la curva de aprendizaje y perfeccionamiento a muchos más profesionales.



Metro de Madrid, S.A.

RxJs

- Angular ofrece compatibilidad total con RxJs, una poderosa herramienta que facilita la programación reactiva.
- Gracias a RxJs, es posible establecer una comunicación sencilla entre componentes basada en observables.
- Al funcionar de forma asíncrona, permite optimizar tiempos de carga y rendimiento general de la aplicación.

Ivy

- Desde la versión 9, Angular utiliza Ivy como su compilador estándar, el cual se encarga de convertir el código de la aplicación en JavaScript y HTML que los navegadores puedan entender y renderizar.
- Su principal ventaja es que optimiza el tiempo de compilación en más de un 50%, minimiza el tamaño total de la aplicación y reduce mucho el tamaño del bundle.

Ecosistema

- Existe un enorme ecosistema formado por librerías, plugins, paquetes, add-ons, herramientas de desarrollo, etc. Que hace realmente sencillo encontrar el complemento que necesitemos en cada situación.
- Angular desarrolla y mantiene Angular Material, una librería gráfica exclusiva con todo tipo de componentes plug and play totalmente customizable.

En conclusión, **Angular** es una gran elección para el caso de uso de Metro de Madrid, ya que satisface las necesidades básicas de ofrecer un **buen rendimiento, facilitar el desarrollo, mantenimiento y testeo de un gran aplicativo como SIAR** y además aporta muchos otros beneficios bajo la seguridad que ofrece Google de cara a recibir soporte y actualizaciones a largo plazo.

Arquitectura

En este apartado definiremos la arquitectura front básica de la aplicación utilizando Angular como framework y proponiendo el uso de un API RESTful.

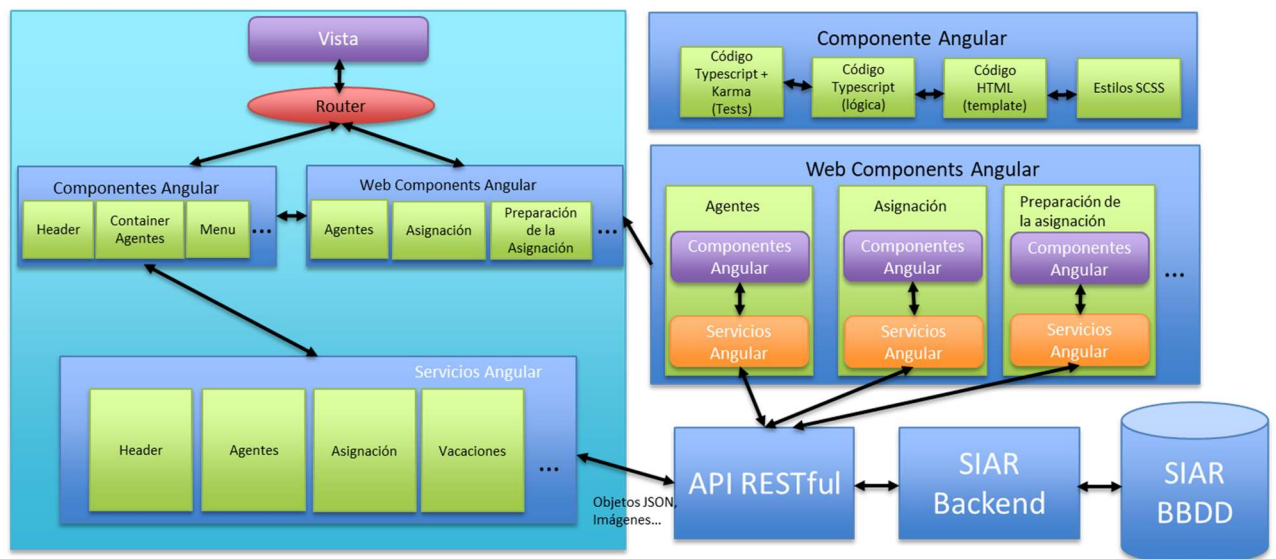
Capas

- **Componentes.** Conforman el núcleo del frontal, conteniendo la lógica para renderizar cada vista, sus correspondientes templates HTML y hoja de estilos SCSS y los Tests unitarios que aseguran la calidad y fiabilidad de la lógica del componente. Los componentes pueden pertenecer al proyecto principal o importarse en forma de Web Components.



Metro de Madrid, S.A.

- **Servicios.** Son los encargados de comunicarse con el servidor a través de endpoints expuestos por un API RESTful. Son invocados por los componentes para obtener la información necesaria en cada caso mediante objetos JSON.
- **Router.** Es el encargado de orquestar los componentes para que se renderice correctamente según el estado de la navegación e interacciones por parte del usuario.
- **Vista.** Al tratarse de una SPA, tenemos una única vista que va mostrando dinámicamente unos u otros componentes en función de la ruta y estado de la aplicación.
- **API RESTful.** Capa externa al frontal que se comunica con el backend para traer la información solicitada en cada caso por parte de los servicios.



Tareas principales a abordar (Arquitectos Front-end)

- Definir esqueleto base de la aplicación Angular con Angular-cli
- Identificar e instalar principales dependencias del proyecto
- Crear blueprint para Web Components
- Definir guía de estilos
- Crear librería para UI
- Definir estructura API RESTful y endpoints necesarios
- Desarrollar app piloto con el esqueleto principal que importe y use Web Components generados con el blueprint, emplee la librería de UI y utilice mocks simulando el API.

Proceso adaptativo



Metro de Madrid, S.A.

Veamos algunos consejos prácticos para adaptar la arquitectura actual a la nueva propuesta:

- Crear un componente por cada elemento del menú. Puede ser una buena división inicial, aunque debe estar sujeta a cambios. Para cada uno de ellos pueden surgir múltiples "sub-componentes", siendo recomendable llegar al nivel más pequeño que resulte indivisible, siguiendo un modelo atómico. Una vez identificados los componentes principales de la aplicación, es el momento de convertirlos en Web Components, extrayendo su código de la aplicación para luego importarlos a voluntad.
- Crear un servicio por cada vista actual, que en lugar de requerir al backend una vista completa, pedirá únicamente aquellos datos que el Frontend desconozca. Como en el caso anterior, se trataría de una aproximación inicial sujeta a cambios. Al principio es muy recomendable el uso de mocks para entender bien qué se necesita y no esperar a tener el API terminado. Ambos deberían desarrollarse en paralelo y siguiendo una filosofía ágil.
- Definir los estilos base de la aplicación, que serán utilizados por múltiples componentes, aunque cada uno de ellos tenga su propia hoja de estilos para su caso específico. Muy recomendable tener en cuenta Angular Material.
- Escribir tantos tests unitarios como sea posible lo antes posible. Esto ayudará a entender qué se quiere conseguir y, una vez conseguido, a asegurar su correcto funcionamiento a medida que se avance en el desarrollo.



Conclusiones

SIAR es una aplicación pesada que necesita ofrecer un alto rendimiento y una gran fiabilidad para garantizar el éxito en sus funciones. Actualizar los frontales de SIAR utilizando las tecnologías, arquitectura y procesos detallados en este documento es la mejor opción para conseguir los objetivos de la migración deseados, con las condiciones dadas por parte de Metro de Madrid. Además existen otros beneficios importantes que se lograrán tras la migración:

- El código será mucho más mantenible y escalable.



Metro de Madrid, S.A.

- Al utilizar tecnologías extendidas y contrastadas, resultará sencillo encontrar personal cualificado cuando sea necesario.
- También ganaremos un plus de seguridad y fiabilidad a raíz de componetizar la aplicación y automatizar procesos.
- Hemos elegido minuciosamente todas las herramientas para aseguraremos tener soporte a largo plazo.
- Conseguiremos desacoplar el front y el back, por lo que se puede paralelizar el trabajo y optimizar y securizar cada parte por separado.
- Construiremos un flujo de trabajo ágil que garantice la calidad del trabajo.



Metro de Madrid, S.A.

Estrategia de Pruebas

Tipos de baterías de pruebas estimadas para verificación de la migración de los frontales web de SIAR

Tipología	Descripción, alcance y objetivos					
Test Unitario	Verificación de la pieza codificada					
Test Integrado	Verificación de la correcta integración de las piezas codificadas entre sí y con otros módulos o componentes del sistema					
Test funcional	Fase de Definición	Donde a partir de un input documental y de los requisitos iniciales llegaríamos a obtener los siguientes resultados aplicando las siguientes herramientas, activos y knowhow	Estrategia de pruebas	Perfiles funcionales garantizan la calidad y cobertura de pruebas		
			Planes de prueba validación funcional	Generación de los planes de pruebas con cobertura óptima y catalogación del caso a probar en función de la importancia del proceso que cubre.	Clasificación de los casos de prueba	
					CORE	Contienen los flujos claves de negocio
					AMPLIADOS	Aquellos que contengan pruebas de prioridad alta, ampliando el alcance de los casos de prueba CORE.
					COMPLEMENTARIOS	Casos de prueba de funcionalidades secundarias o de una importancia menor.
		Informes de pruebas	Definición de cómo serán los informes de resultados e indicadores			

Tabla 13 estrategia de pruebas



Metro de Madrid, S.A.

Tabla 13 estrategia de pruebas (continuación)

Test funcional (continuación)	Fase de Ejecución	Estrategia de ejecución basada en 3 ciclos. Ejecutada tras los trabajos de desarrollo	Ciclo 1	Ejecución de todos los casos de prueba definidos	
			Fase de rework en caso de detección de incidencias	Resolución de defectos encontrados en el primer ciclo	Recodificación para la resolución de defectos encontrados
			Ciclo 2	Asegurar la corrección de todas las incidencias detectadas y asegurar que no se afecta al resto del sistema	Se prueban el 100% de los casos erróneos del primer ciclo y un porcentaje representativo de los casos que resultaron en OK para certificar la inexistencia de regresiones
			Fase de rework en caso de detección de incidencias	Resolución de defectos encontrados en el primer ciclo	Recodificación para la resolución de defectos encontrados
			Ciclo 3	Certificar la calidad del desarrollo y la correcta resolución de todas las incidencias detectadas	Se prueban el 100% de los casos erróneos del segundo ciclo y un porcentaje representativo de los casos que resultaron en OK para certificar la inexistencia de regresiones
	Fase de Revisión	Análisis de resultados obtenidos en las métricas	Revisión y garantía que toda la documentación generada y requerida por metodología es correcta y ha sido almacenada en los repositorios destinados a tales efectos		



Metro de Madrid, S.A.

3.10. Integración y Entrega Continua

Integración Continua

Control de versiones

Para el proyecto de SIAR el sistema de control de versiones es GIT. GIT, al ser un sistema de control de versiones distribuido, tiene ventajas respecto a los sistemas de control de versiones centralizados (p.e. SVN):

- Permite a los usuarios trabajar de forma autónoma cuando no están conectados a la red.
- Los cambios desarrollados no tienen la obligación de ser compartidos inmediatamente con otros distintos nodos.
- Espacio de trabajo único.
- Integraciones y movimientos entre ramas fáciles, rápidos y no penalizan recursos.

Para trabajar con el mismo se utilizará **Gitflow**.

El trabajo se organiza en dos ramas principales (protegidas):

Rama Master: cualquier commit que pongamos en esta rama debe estar preparado para subir a producción

Rama Develop: rama en la que está el código que conformará la siguiente versión planificada del proyecto.

Además de estas dos ramas, Se proponen las siguientes ramas auxiliares:

Feature: Ramas nacida de develop que debe contener únicamente cambios atómicos. Sobre estas ramas se realizan los desarrollos. Una vez finalizado el desarrollo se integra sobre develop.

Release: Rama nascida de develop. Estas ramas se utilizan para preparar el siguiente código en producción. En estas ramas se hacen los últimos ajustes y se corrigen los últimos bugs antes de pasar el código a producción incorporándolo a la rama master.

Hotfix: Esas ramas se utilizan para corregir errores y bugs en el código en producción. Funcionan de forma parecida a las Releases Branches, siendo la principal diferencia que los hotfixes no se planifican.



Metro de Madrid, S.A.

Versionado

Las versiones se etiquetan con tres números MAJOR.MINOR.PATCH (ejemplo: 2.15.5) donde:

PATCH: Toda clase de arreglos y parches compatibles con la versión anterior.

MINOR: Cualquier nueva funcionalidad compatible con la versión anterior (o deprecación de funcionalidades antiguas).

MAJOR: Cambios no compatibles con la versión anterior.

Añadido a ello, en las ramas Feature y develop se añadirá el label "snapshot", en las ramas release "rc" y en las ramas Hotfix "Hotfix".

**A tener en cuenta: Cuando se incrementa el contador de MINOR, el contador de PATCH se reinicia a 0. Cuando se incrementa el contador de MAJOR, el contador de MINOR y el de PATCH se reinician a 0.*

Gestión de pipelines (Jenkins)

Jenkins es la herramienta que orquesta todo el flujo de integración continua, concatenando las diferentes tareas que se le indiquen en los diversos pipelines creados.

¿Qué ventajas nos va a proporcionar Jenkins?

Gran automatización, lo que conlleva una mayor rapidez de realizar flujos que se pueden repetir varias veces al día. Por ejemplo la compilación, ejecución de pruebas y publicación de artefactos generados. Esto conlleva a que el desarrollador no malgaste su tiempo en procesos repetitivos.

Su **integración con K8S**, lo que le da una gran escalabilidad y la posibilidad de trabajar con contenedores e imágenes docker para levantar los diferentes esclavos.

Fácilmente configurable. Jenkins se puede modificar y extender fácilmente con Plugins. Implementa código de forma instantánea, genera informes de prueba.

Da un nivel **más de seguridad a todo el flujo de la integración continua**, mediante gestión de credenciales y roles.

Por todo ello se utilizará el Jenkins corporativo.

Producto	Versión
Jenkins	2.230

Tabla 14 – Versión Jenkins



Metro de Madrid, S.A.

Para su configuración se utilizarán ficheros Jenkinsfile. Los mismos seguirán el mismo ciclo de vida que las aplicaciones utilizando Git y Gitflow y las pruebas se realizarán apuntando a una rama previa.

Existirán por tanto varios Jobs por repositorio:

Job de FeatureBuild: Este job está destinado a garantizar la calidad y estabilidad de la rama develop en git. Es decir, este job tiene como misión hacer un primer Check de código en las ramas Feature de git, para no llevar nada a develop que no funcione o no cuente con la calidad mínima de código. El job compilará el código situado en las ramas Feature.

Job de Build: El job de Build está destinado a generar versiones snapshot de los artefactos generados tras la compilación y empaquetado del código fuente. Este job también tiene la posibilidad de desplegar dicho artefacto en un entorno de desarrollo. El job compilará el código situado en la rama develop.

Job de ReleaseCandidate: mediante este job se pueden generar versiones RC desde el código almacenado en las ramas release.

Job de Release: el job de release generará una versión final y un tag del artefacto del código fuente llevado a la rama master.

Job de Hotfix: En el caso de que haya que realizar un cambio urgente de algunas de las versiones finales tageadas, este job se ejecutará a consecuencia de la modificación de este cambio urgente. Este job generará una versión Hotfix y un tag. Por otro lado, no tendrá en cuenta el empeoramiento de la calidad de código ya que se entiende la urgencia del cambio y se intenta no bloquear la publicación del artefacto.

Todos estos Jobs de ejecutaran de manera automática haciendo uso de la funcionalidad hooks de GIT. Los mismos se ejecutarán al hacer push en las ramas anteriormente mencionadas.

Análisis de calidad de código (SONAR)

Para realizar análisis de código se utilizará el SonarQube corporativo. Esta herramienta posee varias ventajas:

La plataforma soporta actualmente más de 20 lenguajes incluyendo Java, JavaScript, Cobol, PL, C#...

La herramienta puede extenderse a través de plugins, tanto para soportar más lenguajes como para soportar nuevos lenguajes.

Hay plugins para Eclipse, para Jenkins, Maven, Reportes, Métricas adicionales, etc.



Metro de Madrid, S.A.

Documentación oficial de SonarQube: <https://docs.sonarqube.org/latest/>

Para una aplicación correcta del análisis de código se revisaran los siguientes valores de Sonar, creando un **Quality Gate** para ello:

Blocker Issues → Incidencias bloqueantes

Cognitive Complexity → Mide cómo de complicado es entender el código de la aplicación

Coverage → % de cobertura de código fuente que cubren los test unitarios

Critical Issues → Incidencias críticas

Cyclomatic Complexity → es el número de casos de test que hay que tener para disponer de un 100% de cobertura

Duplicated Lines (%) → % de líneas duplicadas de código de la aplicación

Technical Debt Ratio → % de código que contiene deuda técnica, es decir, incumplimientos de calidad en términos de facilidad de mantenimiento vs tiempo que se estima en solucionar

Unit Test Success (%) → % de test unitarios que se ejecutan y finalizan con éxito

Unit Tests → Número de test unitarios implementados

Vulnerabilities → Son incidencias que tienen que ver con la seguridad de la aplicación

Repositorio de artefactos (NEXUS)

La herramienta corporativa Nexus es el repositorio de artefactos elegido para mantener en un lugar seguro todos los artefactos generados bajo la integración continua.

Además de contenerlos en un lugar seguro, también servirá para que los equipos puedan acceder a artefactos comunes de otros equipos así como de artefacto públicos oficiales.

Por lo tanto, esta aplicación va a contener por una parte los artefactos de aplicación, artefactos públicos oficiales, como por ejemplo Maven central o npmjs.

Se generaran los repositorios:

Local: Repositorio SNAPSHOTS, RELEASES y THIRDPARTY (Dependencias)

Remote: Mirrors a repositorios públicos externos.

Virtual: Agrupaciones de repositorios

Producto	Versión
Nexus	3.19.1-01

Tabla 15 – Versión Nexus



Metro de Madrid, S.A.

Entrega Continua

Entornos SIAR: Para cada una de las aplicaciones de SIAR existirán los siguientes entornos:

Desarrollo: Entorno para probar los cambios realizados en la rama develop. Sobre el mismo se podrán desplegar versiones "snapshot", versiones "RC", versiones "hotfix" y versiones finales. El despliegue en este entorno se realizará de manera automática en el flujo de integración continua.

Preproducción: Entorno donde se realizarán pruebas e2e. Sobre el mismo se validarán todas las funcionalidades que subirán a en el siguiente pase. En este entorno se realizarán despliegues de versiones "RC", versiones "hotfix" y versiones finales.

Producción: Entorno al que accede el cliente final. En este entorno se realizarán despliegues de versiones "hotfix" y versiones finales. Todo lo que suba a este entorno debe haber sido probado en Desarrollo y Preproducción.

Flujo de pruebas

Si en un pase se detecta un problema en las pruebas funcionales se realizarán de nuevo todos los cambios necesarios en Git, generar una nueva versión (con su consiguiente modificación de versión según corresponda) y debe ser probada de nuevo. Una vez se haya validado el correcto funcionamiento de la aplicación la misma podrá ir a producción.

La gestión de los pases se realizará del mismo modo que se realizan hasta el momento.

Automatización de despliegues

Los despliegues se realizarán desde la herramienta Jenkins. La configuración de los Jobs (jenkinsfile), del mismo modo que el código, se encontrarán dentro de GIT y seguirán su propio flujo de pruebas.

Existirán 2 jobs:

Despliegue Preproducción: Este job realizará las tareas necesarias para realizar el despliegue del artefacto generado en la integración continua (Nexus) en el entorno de preproducción. Permitirá despliegues de versiones "RC", "hotfix" y finales.

Despliegue Producción: Este job realizará las tareas necesarias para realizar el despliegue del artefacto generado en la integración continua (Nexus) en el entorno de Producción. Permitirá despliegues de versiones "hotfix" y finales.



Metro de Madrid, S.A.

Los permisos para la ejecución de estos Jobs están limitado a las personas con responsabilidad sobre los mismos dentro de la organización.