
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

# DESARROLLO DE UTILIDADES Y OPTIMIZACIÓN DE ACCESO AL REPOSITORIO DE DATOS FIJOS DE BBVA CIB MEDIANTE UN SISTEMA CACHÉ

Autor: Alfonso Villarino Arias

Director: Atilano Fernández-Pacheco Sánchez-Migallón

**Madrid**

Julio 2017

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

DESARROLLO DE UTILIDADES Y OPTIMIZACIÓN DE ACCESO AL REPOSITORIO DE DATOS FIJOS DE BBVA CIB MEDIANTE UN SISTEMA CACHÉ

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico (2016/2017) es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Alfonso Villarino Arias Fecha: 07/julio/2017

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón Fecha: 07/julio/2017

Vº Bº del Coordinador de Proyectos

Fdo.: David Contreras Bárcena Fecha: ……/ ……/ ……

**AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO**

***1º. Declaración de la autoría y acreditación de la misma.***

El autor D._____

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: _____, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

***2º. Objeto y fines de la cesión.***

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

***3º. Condiciones de la cesión y acceso***

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

***4º. Derechos del autor.***

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

***5º. Deberes del autor.***

- El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

***6º. Fines y funcionamiento del Repositorio Institucional.***

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a …….. de ……... de …….

**ACEPTA**

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

| |
|---|
| |


ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

# DESARROLLO DE UTILIDADES Y OPTIMIZACIÓN DE ACCESO AL REPOSITORIO DE DATOS FIJOS DE BBVA CIB MEDIANTE UN SISTEMA CACHÉ

Autor: Alfonso Villarino Arias

Director: Atilano Fernández-Pacheco Sánchez-Migallón

**Madrid**

Julio 2017

# Agradecimientos

En primer lugar, me gustaría agradecerle mi TFG y toda mi carrera a toda mi familia, que han sido las personas que han hecho posible mis estudios en una escuela como es ICAI, además de ayudarme en los momentos más duros en la carrera.

Así mismo, me gustaría agradecerle a toda la gente que he conocido a lo largo de la carrera, y que me han soportado, por animarme a finalizar mis estudios y a que crea que cualquier cosa que quiera hacer es posible si le pongo ganas.

Para finalizar, me gustaría agradecerle, a todos los trabajadores del departamento de RDR del BBVA, la oportunidad que me han brindado para trabajar como uno más desde el primer día que llegue al banco. Solo tengo palabras de agradecimiento para ellos.

Muchas gracias a todos los que habéis hecho posible esto.

# DESARROLLO DE UTILIDADES Y OPTIMIZACIÓN DE ACCESO AL REPOSITORIO DE DATOS FIJOS DE BBVA CIB MEDIANTE UN SISTEMA CACHÉ

**Autor: Villarino Arias, Alfonso.**
Director: Fernández-Pacheco Sánchez-Migallón, Atilano.
Entidad Colaboradora: BANCO BILBAO VIZCAYA ARGENTARIA, S.A.

## RESUMEN DEL PROYECTO

En este proyecto se tratará de llevar a cabo una optimización de la obtención de información de la base de datos mediante el uso de la cache de Oracle Coherence. Para ello se desarrollarán diversas utilidades para facilitar tanto la gestión como la administración de la aplicación gestora de las caches. Así mismo, se implementarán diversos procesos para mejorar la conciliación y asegurar la correcta comunicación entre Golden Source y Oracle Coherence.

**Palabras clave**: Caché Oracle Coherence, Base de datos, Optimización, Middleware

### 1. Introducción

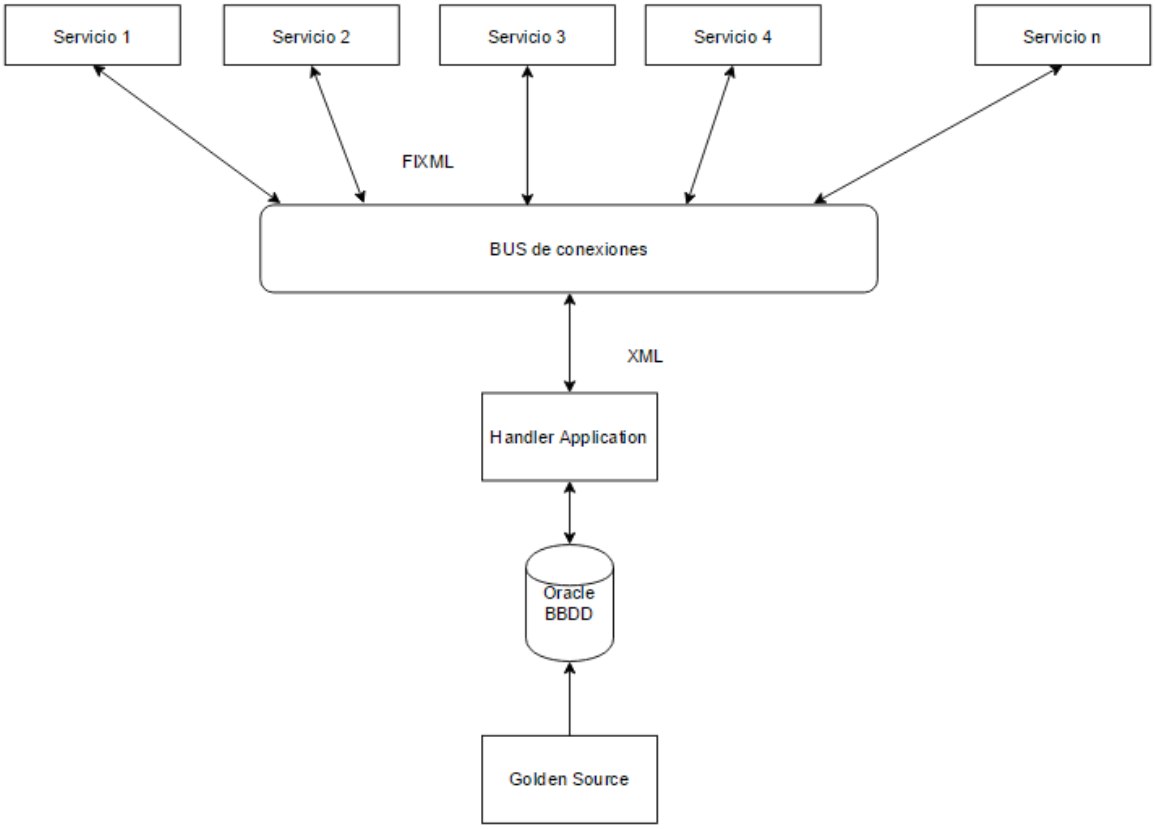
La información es una de las piezas claves de la sociedad actual. El problema de la información se da en el momento del acceso a su almacenamiento en las bases de datos, ya que se suelen producir cuellos de botella. Por ello es de vital importancia de una buena planificación en el acceso a la memoria, para poder optimizar la obtención de dicha información y reducir los tiempos de latencia al mínimo posible.

### 2. Definición del proyecto

Para la optimización del funcionamiento de la base de datos, que existe actualmente en el repositorio global de BBVA CIB (Corporative Investment Bank) se ha desarrollado un nuevo modelo del funcionamiento de la plataforma de gestión de peticiones y respuestas existentes.

Este nuevo modelo se denominará Fase I, por la posibilidad de la existencia de un nuevo modelo en un futuro si el actual Proyecto funciona de la manera esperada.

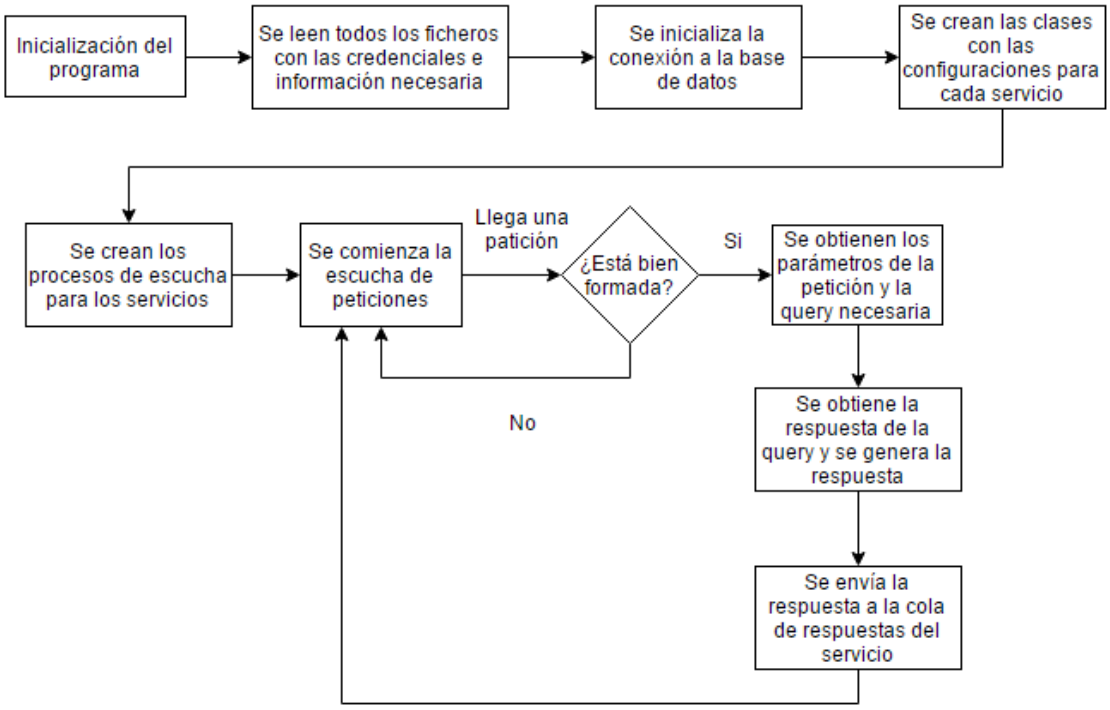
La Fase I está representada por el diagrama que se muestra a continuación.



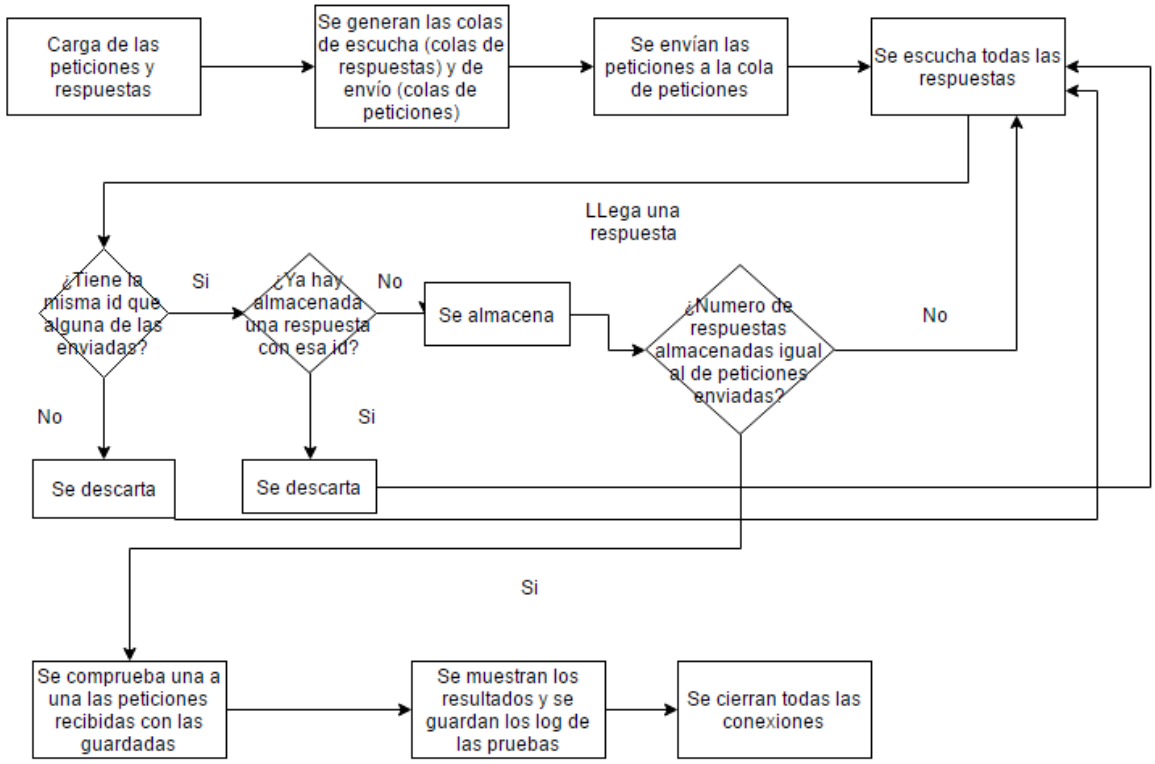

## 3. Descripción del sistema

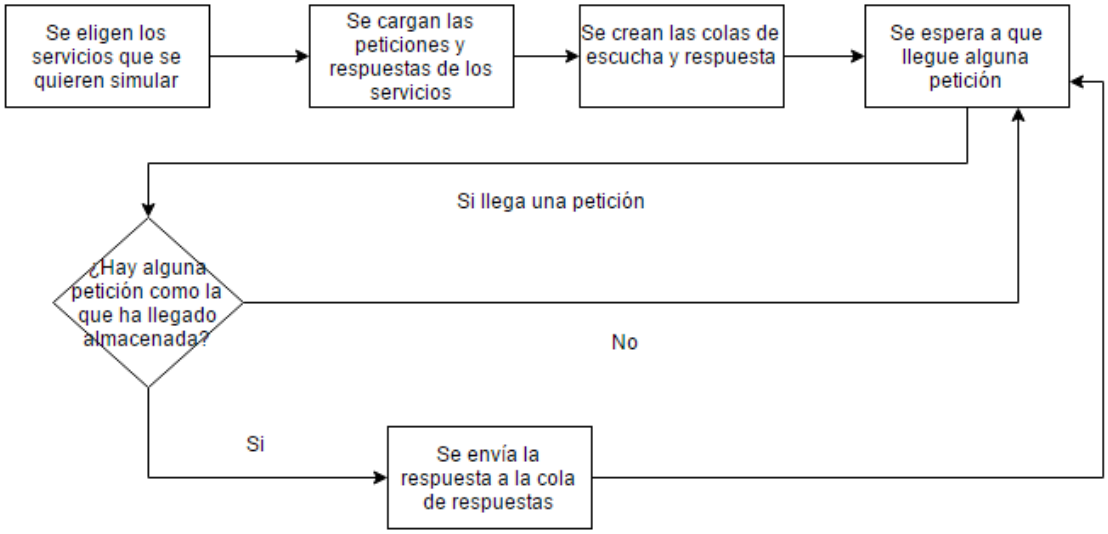
Para la realización del modelo, mostrado en el epígrafe anterior, se han desarrollado dos aplicaciones.

- En primer lugar se trata de la aplicación de gestión de las peticiones y las respuestas del BBVA, cuyo funcionamiento dado es el siguiente.




- Una herramienta de pruebas regresivas que permita comprobar el funcionamiento de la plataforma anterior, facilitando el proceso a los encargados del mantenimiento, así mismo como de la simulación de nuevos entornos de trabajo, dentro de la propia empresa. El funcionamiento de la aplicación se ilustra en los siguientes diagramas.






## 4. Resultados

En el ámbito de los resultados cabe diferenciar entre los obtenidos en la herramienta de pruebas regresivas y los obtenidos en la plataforma de gestión:

- **Pruebas regresivas**: El funcionamiento de dicha herramienta se ha producido como todo un éxito en el ámbito de la empresa. Este éxito se debe al resultado de agilizar el proceso de comprobación del funcionamiento de la plataforma de gestión de peticiones, hasta ahora implantada (Golden Source), y ha permitido comprobar el funcionamiento de la nueva e incluso ayudar a la corrección de errores en el tratamiento de las peticiones e incorporarlas dentro del ciclo de vida del software.

- **Plataforma de gestión**: La nueva plataforma de gestión de peticiones y respuestas se ha presentado como una novedad muy bien recibida dentro del área Corporate Investment Banking, de entidad empresarial. Esto se ha debido gracias al alto rendimiento que está ofreciendo, frente a la plataforma anterior, y la eficiencia con la que está respondiendo a las peticiones que le llegan.

## 5. Conclusiones

El proyecto desarrollado ha cumplido con creces cada uno de los objetivos que se tenían previsto realizar, con éxito. Así, en esta Fase I del proyecto, se ha conseguido que los tiempos de respuesta de la base de datos hayan disminuido del orden de entre 300ms y varios segundos la respuesta de las peticiones de los servicios de BBVA.

Esto hizo posible que uno de los retos más importantes fuese solventado, la falta de eficiencia en la velocidad de respuesta que estaba llevando a cabo la plataforma de gestión, Golden Source, de las peticiones y respuestas.

## 6. Referencias


[1] Página Oficial de Oracle: https://www.oracle.com

[2] Página Oficial Oracle Coherence:
http://www.oracle.com/technetwork/middleware/coherence/overview/index.html

[3] "Software Engineering 9", Ian Sommerville, Pearson, 2010:
https://www.pearson.com/us/higher-education/product/Sommerville-Software-Engineering-9th-Edition/9780137035151.html

# DEVELOPMENT OF UTILITIES AND ACCESS OPTIMIZATION TO THE FIXED DATA REPOSITORY OF BBVA CIB THROUGH A CACHE SYSTEM

**Author: Villarino Arias, Alfonso.**
Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano.
Collaborating Entity: BANCO BILBAO VIZCAYA ARGENTARIA, S.A.

## ABSTRACT

This Project consists in the optimization of the information obtaining from the database with the Oracle Coherence´s cache. For this purpose, some utilities will be developed to ease the management and the administration of the management application. Moreover some process will be implemented to improve the conciliation between Golden Source and Oracle coherence and to ensure a good communication between this two tools.

**Keywords**: Oracle Coherence´s Cache, Database, Optimization, Middleware.

### 1. Introducción

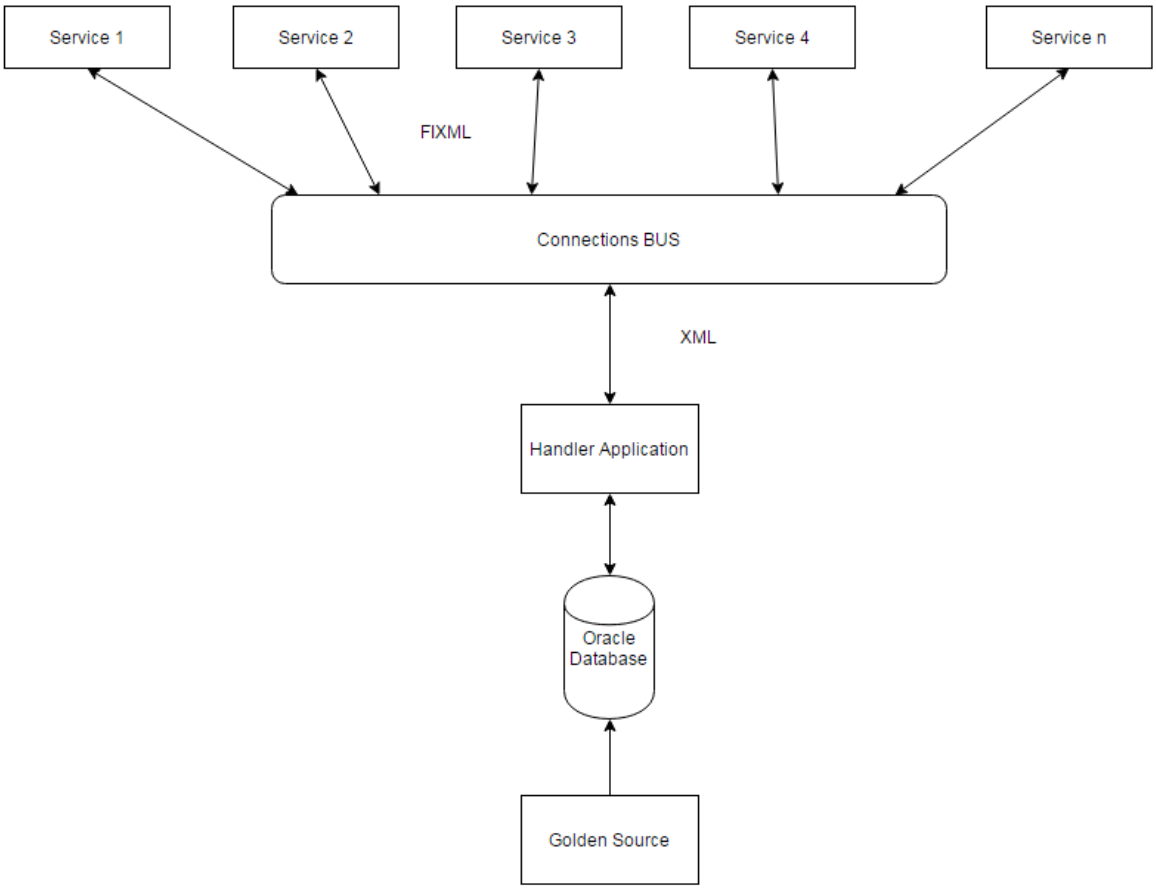
Information is one of the main pieces in our society. The most important problem that information has is produced when the access of this information kept in databases is taking place, due to bottleneck use to be suffered. This is the reason why a good planning in the memory access to be able to optimize the obtaining of the information and to reduce the latency times as possible is necessary.

### 2. Project definition

For the database working optimization, which nowadays exists in the BBVA CIB global repository, a new operating model for the program that is responsible for the requests and responses management has been developed.

This new model will be called Phase I, due to the possibility of a new model in the future if the project works correctly.

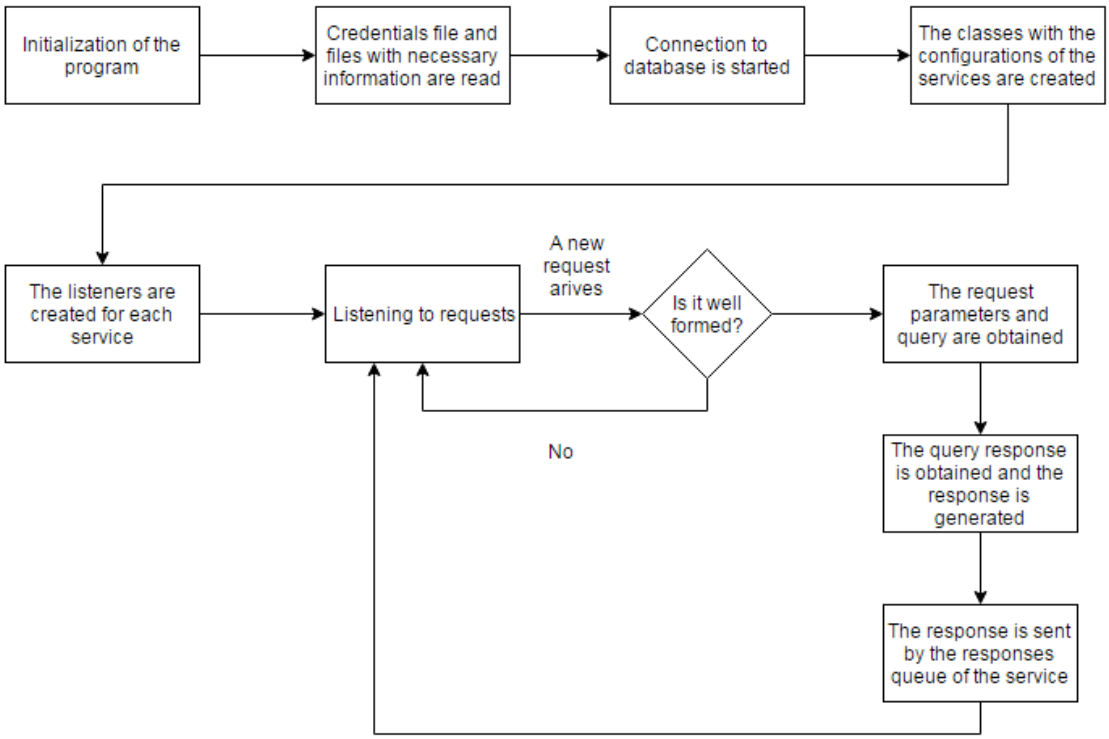
Phase I is represented by the next diagram.



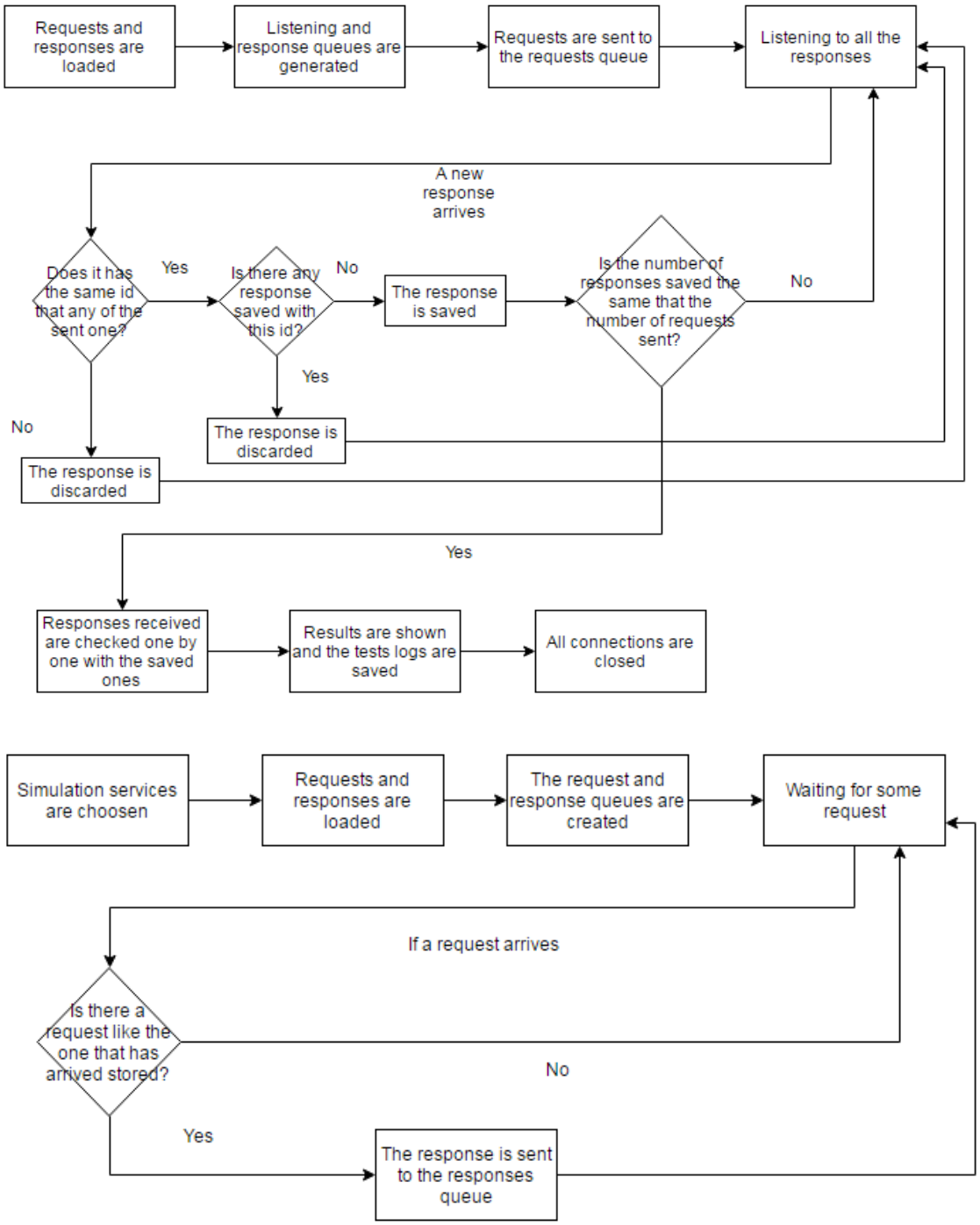

## 3. Descripción del sistema

For develop the model presented in the last chapter, two applications have been developed.

- First application is the responsible of manage the requests and responses from BBVA, which work is the next.




- Secondly a testings tool that permits check the correct work of the platform presented before. This application will make easy the work of the maintenance people in charge and the simulation of new work environments too. The functioning of the application is shown next.




## 4. Resultados

In the field of results the differentiate between obtained in the testings tool program and those obtained in the Handler application has to be shown:

- **Testings tool**: The performance of this application has been successfully in the field of the company. This success is debt to the agility of the performance checking in the requests and responses management application that was working until now, and it has permitted check the performance of the new

application too. Moreover tis application has corrected some errors in the management of the requests and has helped to applicate them.

- **Handler application**: The new application has been a welcome innovation in the area if Corporate Investment. This is due to the high performance the application is offering and to the efficiency which it is answering to the requests it receives.

## 5. Conclusiones

The project developed has more than fulfilled the targets that were fixed. In the phase I a decrease of the answering time between 300ms and a few seconds of the requests from BBVA services was got.

This decrease makes possible that one of the most important challenges was solved, which is the lack of efficiency in the answering speed in Golden Source, the management platform of requests and responses.

## 6. Referencias


[1] Official Oracle's web page: http://www.oracle.com

[2] Official Oracle Coherence's web page: http://www.oracle.com/technetwork/middleware/coherence/overview/index.html

[3] "Software Engineering 9", Ian Sommerville, Pearson, 2010: https://www.pearson.com/us/higher-education/product/Sommerville-Software-Engineering-9th-Edition/9780137035151.html

# *Índice de la memoria*

# *Índice de figuras*

# *Índice de tablas*

# Capítulo 1. INTRODUCCIÓN

La sociedad actual se encuentra tan acostumbrada al uso y tratamiento de la información, que incluso se ha acuñado con el nombre de Sociedad de la información. Actualmente la información es uno de los pilares de la economía moderna, tanto es así que las mayores empresas del mundo dedican su negocio, exclusivamente, a compartir la información de sus clientes, estas empresas son las Redes Sociales.

La información juega un papel clave en cualquier tipo de transacción que se lleva a cabo, ya que cualquier persona se informa previamente de la otra parte de la transacción antes de llevarla a cabo. Así mismo, ninguna empresa contrata a ningún empleado sin llevar a cabo una búsqueda previa de su información personal, para garantizar que cumpla el perfil deseado por la empresa.

El dominio de la información en la actualidad es tal, que abundan los casos de políticos que tienen que retractarse, o incluso dimitir, por comentarios que quedan guardados desde hace años en bases de datos de información a la espera para salir a la luz y atormentar a los que pronunciaron dichas palabras.

Toda esta información tiene que estar almacenada en algún lugar, las bases de datos, que se encargará de distribuirlo y de responder a las peticiones llegadas por los usuarios que quieran saciar sus necesidades de conocimiento. Esta información sale de la base de datos por un canal que será el encargado de llevarlo al destinatario. El problema radica en este canal de salida que pasa a convertirse en un único punto de salida de información pudiendo llegar a producirse un cuello de botella si llegan demasiadas peticiones.

Un cuello de botella se produce al tratar de enviar cualquier tipo de comunicación por un único punto que es capaz de procesar una menor tasa de bits de la que estás enviando, lo cual producirá un retraso en las comunicaciones.

Estos retrasos no serán importantes si se trata de información como emails, en la cual los tiempos de retraso no generan ningún tipo de malestar entre los interlocutores. El problema aparece cuando la comunicación que se está dando una comunicación a tiempo real, videoconferencias, telefonía IP; o el tiempo de respuesta es de vital importancia y cuyo retraso no será tolerable por los usuarios, tanto los de la aplicación como los de los sistemas.

Por ello, es de vital importancia poseer una base de datos que tenga un tiempo de respuesta ínfimo, una infraestructura de conexión de base de datos con un alto ancho de banda y con alta redundancia y un software que se encargue de su gestión de manera efectiva y lo más rápida posible.

Este es el caso de la empresa bancaria BBVA (Banco Bilbao Vizcaya Argentaria, S.A.), el cual quiere optimizar el funcionamiento de su base de datos, mediante una caché de Oracle Coherence y la actualización del software de gestión del balanceo de información entre la Base de Datos y la Caché, y de gestión de las respuestas dadas por la Base de datos. Además, se pretende actualizar el funcionamiento de la respuesta del sistema gestor de información en su conjunto, Base de datos y Caché, para optimizar el tiempo de respuesta y el tamaño de la misma, para no congestionar la red.

Así mismo se tratará de realizar diferentes herramientas de administración y de gestión del sistema de bases de datos de la empresa, las cuales permitan comprobar el correcto funcionamiento del mismo.

# Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Un proyecto software, como el realizado en este proyecto, se lleva a cabo no solamente gracias a las líneas de código escritas durante el proceso de desarrollo, sino que también son de vital importancia todas las herramientas que se han empleado para poder finalizar el proyecto con la mejor calidad posible.

Como el tema principal de este proyecto es el desarrollo software, las herramientas empleadas en la consecución de sus objetivos, serán enfocadas al mundo del desarrollo software.

Así se distinguirá entre programas para escribir código, programas enfocados a la documentación del código escrito, se verán tecnologías de programación, programas para la gestión de las bases de datos, entornos de gestión de ficheros, así como otras tecnologías que están establecidas en BBVA y a las cuales uno tiene que amoldarse y aprender a utilizar.

A continuación se procederá a describir las herramientas que se han empleado para la consecución del proyecto, así como una explicación de por qué se seleccionaron esas tecnologías y no otra de las que existen en el mercado.

## *2.1 NETBEANS Y ECLIPSE NEON*


NetBeans y Eclipse son IDES de programación, esto es, plataformas en las cuales se puede escribir el código para el desarrollo de una aplicación facilitándote el proceso de escritura, además de permitir compilar el código a medida que se va escribiendo y su ejecución con pulsar un solo botón.

Además de estas ventajas, cada uno de estos IDEs tienen sus particularidades, las cuales han hecho que sean los elegidos para realizar tareas dentro del proyecto software realizado.

Por un lado NetBeans, versión 8, ha sido, por su uso intuitivo y potente en la programación con Java swing, la herramienta elegida para el desarrollo de las herramientas de administración, debido a la gran cantidad de interfaces gráficos que debían de ser desarrollados.

Por el otro lado se encuentra Eclipse, versión Neon, el cual se ha empleado para llevar a cabo todo el desarrollo de optimización de la base de datos de BBVA. Una de las mayores ventajas que ha hecho que se imponga este entorno de desarrollo, frente a otros, es la accesibilidad a un gran mercado de plugins, siempre accesibles de manera gratuita, para facilitar muchas de las tareas de programación, como por ejemplo compartir el código en un Git de trabajo entre varios compañeros de manera instantánea.

Además de las ventajas de facilidades de desarrollo que ofrecen, tanto NetBeans como Eclipse, también cabe destacar dos puntos positivos a favor: son gratuito y de código abierto, por lo que siempre hay actualizaciones de bugs y resulta una gran elección por su

precio nulo; cuenta con una gran comunidad de desarrolladores, por lo que ante cualquier problema que surge con el entorno de desarrollo, siempre se puede encontrar respuestas y soluciones buscando en internet.

## *2.2 JAVA*


Java es un lenguaje de programación multiplataforma y orientado a objetos elaborado por Sun Microsystems, y actualmente perteneciente a Oracle. El lenguaje de programación Java tiene una característica importante, que es un lenguaje plataforma, esto quiere decir que, simplemente instalando la JVM, o Java Virtual Machine, cualquier máquina será capaz de ejecutar un programa Java.

Este lenguaje de programación fue el elegido para el desarrollo del proyecto debido a su gran uso en todo el mundo, facilitando encontrar soluciones a problemas comunes en internet, además de su versatilidad y su fácil implementación en cualquier plataforma, lo que hace permite ejecutar las aplicaciones que se crean en este desarrollo en las máquinas con Windows, Linux o Mac OS.

## *2.3 SQLDEVELOPER*


SQL Developer es un gestor de bases de datos, perteneciente a la compañía Oracle, el cual permite tener toda la información almacenada, en la base de datos de BBVA, de manera sencilla, desde un interfaz gráfico el cual te permite hacer cualquier tipo de consulta sobre la base de datos, además de permitir modificar cualquier tipo de dato de la propia base de datos.

No existe una gran diferencia entre los gestores de base de datos del mercado y este ha sido seleccionado particularmente por política de empresa, debido a que toda la tecnología de almacenamiento de información de la que dispone BBVA pertenece a la compañía Oracle.

## *2.4 PUTTY*

Putty es un cliente de conexiones SSH, Telnet y TCP, el cual permite gestionar una máquina de manera remota, esto es, desde otra máquina con el simple conocimiento de la dirección del host al que se quiere acceder y de un usuario y contraseña.

Esto permite que se puedan ejecutar programas en servidores situados en otros edificios o instalaciones, sin necesidad de estar situado físicamente en el lugar de dichos servidores, al mismo tiempo que otros usuarios ejecutan otros procesos en dichas máquinas.

Este cliente es Open Source, con actualizaciones de bugs de manera periódica, además de ser sencillo de instalar y de usar, mediante una consola de comandos convencional. Por todas estas ventajas ha sido el seleccionado para ejecutar los desarrollos llevados, a cabo en este proyecto, en servidores del banco, para ser accesibles por cualquier trabajador del mismo.

## *2.5 FILEZILLA*


Filezilla es un cliente de conexión FTP, el cual permite la transmisión de ficheros entre dos hosts conociendo la dirección IP del host al que se quiere conectar, el usuario y la clave. Esto ha permitido subir los programas realizados, en el proyecto, al servidor de desarrollo del banco, permitiendo así ser empleado por el resto de usuarios del banco, y de manera remota mediante PUTTY.

Filezilla es el cliente FTP más empleado en la actualidad y funciona en código abierto, lo que permite que tenga actualizaciones constantes y que su uso sea seguro y estable. Por ello ha sido el cliente FTP elegido para las transferencias de archivos a los servidores.

## *2.6 GOLDENSOURCE*


GoldenSource es un IDE de programación del manejo de la información para entidades financieras, el cual permite gestionar las respuestas que da la base de datos a las peticiones provenientes de cualquiera de los servicios de los que dispone el banco, se verá en el siguiente capítulo la estructura actual con más detalle.

Este IDE de programación permite desarrollar los flujos que describe la información, mediante sencillos diagramas de flujo mediante programación Drag and Drop, es decir, cogiendo un elemento predefinido y soltándolo donde corresponde, describiendo así el camino que debe seguir la información.

La elección de este IDE ha sido por política de empresa, ya que tiene un contrato con la compañía GoldenSource Corporation, por los servicios de la misma.

## *2.7 SONARQUBE*

SonarQube es una plataforma online que permite la evaluación del código fuente para ser observado por diferentes usuarios. Esta permite subir el código que se desarrolla en los IDEs de programación, mediante un plugin de los mismos, para que lo analice automáticamente y muestre las características más relevantes del código: Bugs, Código duplicado, número de líneas de código de testing, porcentaje del código que ha sido comprobado, entre otros.

Además permite que sean varios los usuarios que tengan acceso a este DashBoard para realizar el seguimiento del código escrito, permitiendo así la programación colaborativa y tener un feedback muy útil para mejorar el código.

SonarQube ha sido elegido debido a que es una plataforma de comprobación de código muy extendida y su uso es libre, esto es, es una plataforma Open Source, con una gran comunidad que la respalda y actualizaciones periódicas.

## *2.8 TRELLO*

Trello es una plataforma para la planificación de proyectos de metodologías ágil, permitiendo crear tableros con cada uno de los sprints que se realizan para la consecución del proyecto.

Trello fue la plataforma elegida para el desarrollo de la metodología ágil que se realizaba, durante el transcurso del proyecto, debido a su interfaz intuitivo, su facilidad de uso y su coste nulo.

## *2.9 ORACLE COHERENCE*

Oracle Coherence es la memoria de acceso rápido, o caché, que ofrece la empresa Oracle. Dicha memoria está basada en un sistema "data grid" de datos, el cual está optimizado para procesar información a grandes distancias.

Así mismo cabe destacar que está diseñada para ofrecer una mayor confiabilidad, escalabilidad y actuación que los sistemas tradicionales de almacenamiento de información.

Además, una de las grandes ventajas, que supone Oracle Coherence, es la duplicidad que produce, de la información que le llega a un servidor, a todos los servidores de la caché. Esto permite que no se produzca pérdida de información, teniendo siempre servidores de respaldo, y que el tiempo de latencia sea nulo, debido a que la descarga siempre se produce en local.

## *2.10 FIXML*


FIXML, o más conocido como Financial XML, es un lenguaje de programación el cual implementa el protocolo FIX para realizar el intercambio de información financiera mediante el lenguaje de programación XML.

Este se presenta como un lenguaje de programación importante en el sector bancario por su necesidad a la hora de realizar todas las transacciones de información financiera dentro del propio banco, o con otras entidades financieras.

# Capítulo 3. ESTADO DE LA CUESTIÓN

BBVA es una empresa, del sector bancario, que se encarga del manejo de la información financiera de millones de personas, tanto físicas como jurídicas, alrededor del mundo, por lo que el manejo de esta cantidad de información es de vital importancia para desarrollar su actividad con normalidad.

Para llevar a cabo la gestión de las peticiones, dentro del ámbito CIB o Corporate Investment Banking, se hace uso de la tecnología de Golden Source, la cual permite la programación de aplicaciones a través de diagramas de flujo "semibásicos" y la inclusión de pequeñas líneas de código, simplificando así el trabajo de la programación.

En el ámbito del almacenamiento, BBVA, cuenta con una base de datos de Oracle, encargada de guardar toda la información que procesa la empresa para su posterior uso. Así mismo cuenta con una memoria de acceso rápido para agilizar la devolución de la información que más frecuenta su uso, como es la Caché de Oracle Coherence.

Estos sistemas de almacenamiento se encuentran conectados a una serie de colas (JMS Tibco) mediante el uso de procesos, uno por servicio, para recoger las peticiones que llegan por estos "conductos". Finalmente estas colas se conectan a un bus de conexiones denominado "ESB" el cual se encarga de conectarse a todos los servicios de los que dispone el banco y traduce todas las peticiones realizadas, por el mismo, de FIXML, o XML financiero, a un lenguaje que comprendan los sistemas de almacenamiento, XML.

Este funcionamiento quedará más claro con el siguiente gráfico del sistema actual que se encuentra implantado en el banco:

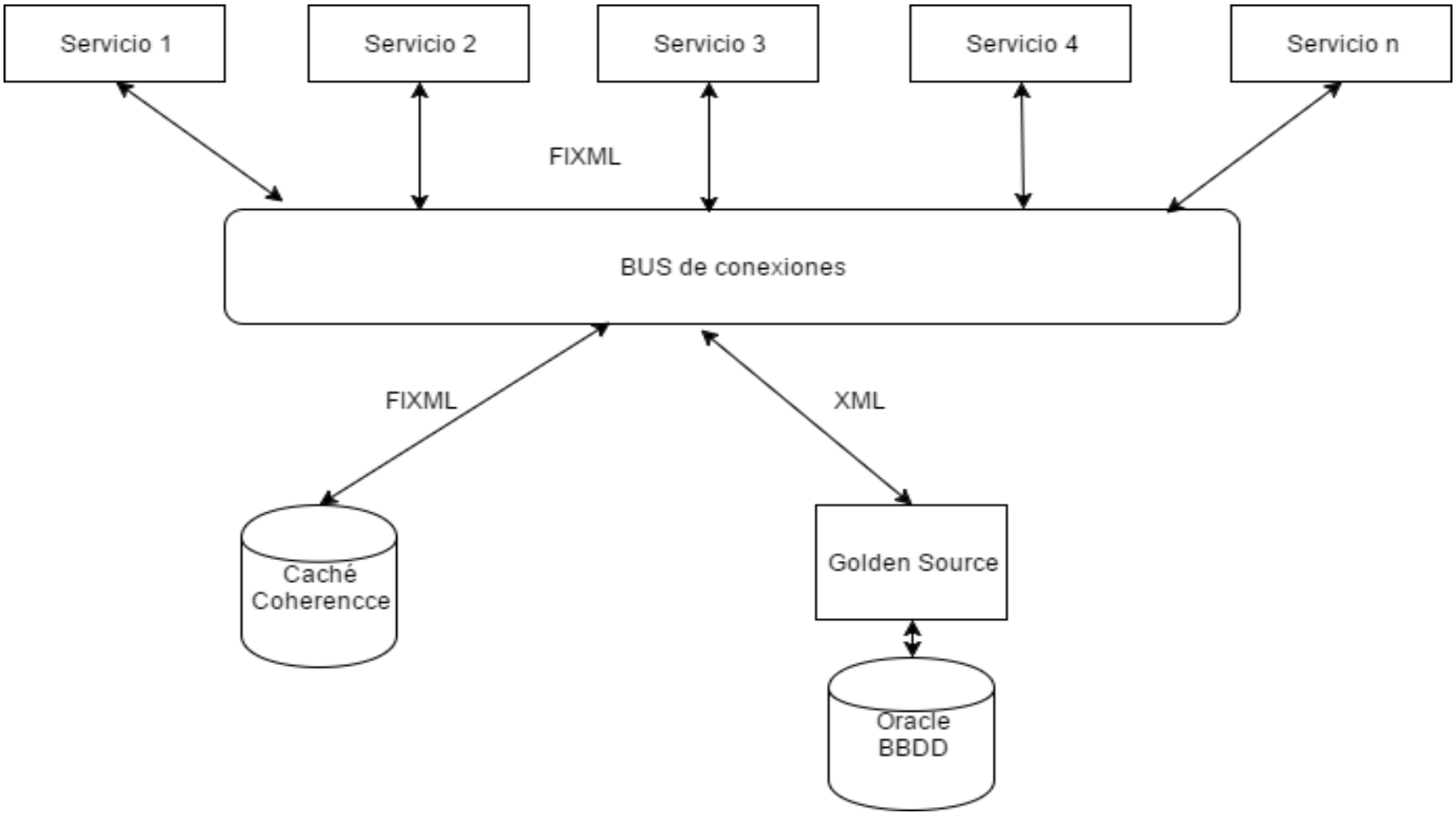



*Figura 1. Estado de la cuestión actualmente*

Un ejemplo de funcionamiento del proceso de obtención de información, sería el mostrado en el siguiente diagrama de flujo:

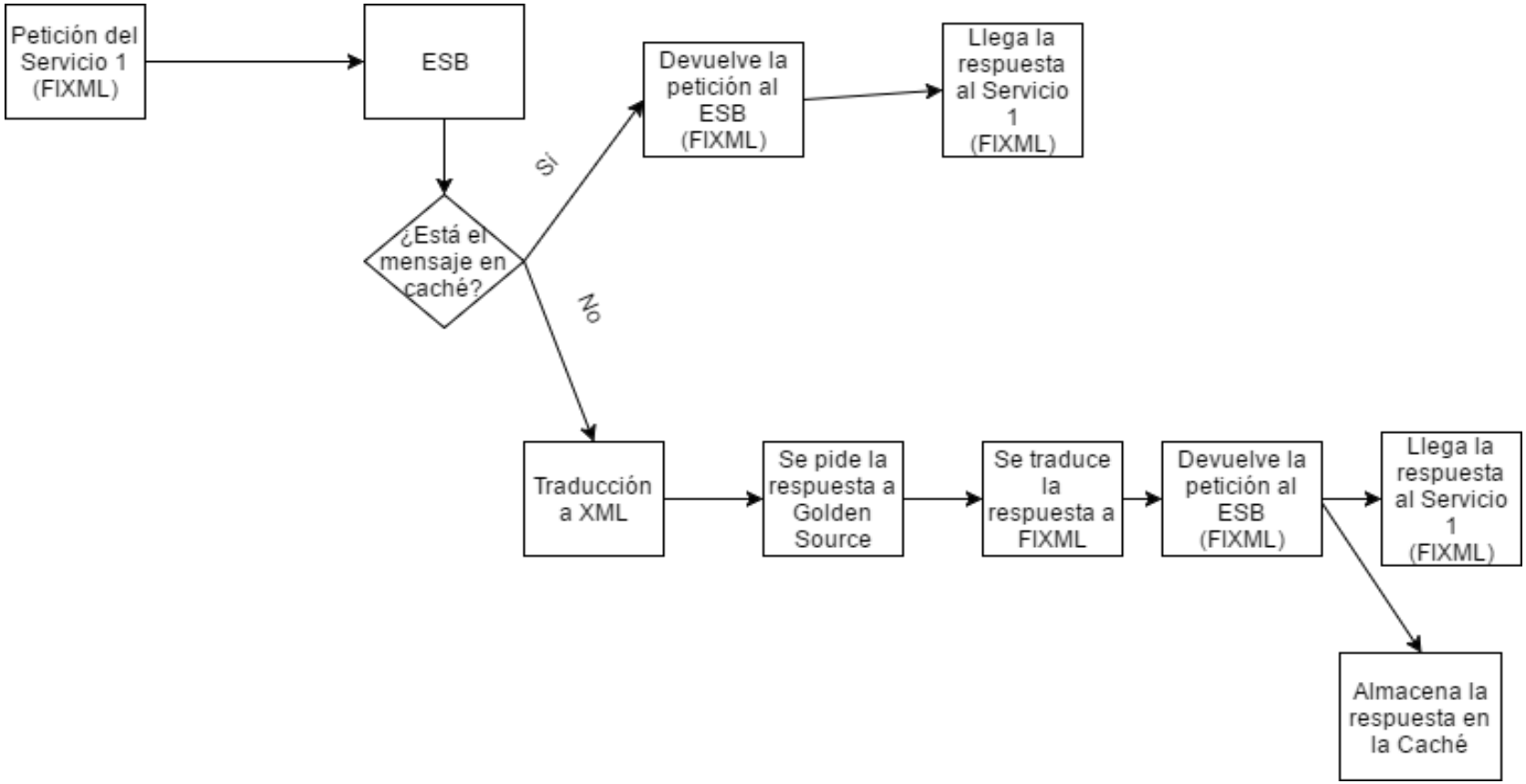



*Figura 2. Funcionamiento en la actualidad*

Como se puede observar, el flujo de información, describe un camino sencillo para obtener la respuesta de cualquier petición, si la respuesta se encuentra almacenada previamente en la caché de Oracle Coherence, y describe un camino más difícil, si la información se encuentra en la base de datos.

Esto se debe principalmente a dos causas, la primera es la necesidad de traducir la petición a lenguaje XML, para la comprensión de Golden Source, y la segunda es la lentitud que tiene obtener la petición de la base de datos frente a la obtención de la misma cantidad de información de la caché.

Además, cabe destacar que para almacenar la información, obtenida de la base de datos, en la caché, esta debe volver al ESB, traducirse a lenguaje FIXML y de ahí ir a la caché, no estando conectadas directamente la Base de Datos y la caché, ralentizando así el tiempo de almacenamiento en la memoria rápida.

Por lo tanto se encuentran diversos problemas a este modelo de funcionamiento:

- En primer lugar se tiene una dependencia muy alta de Golden Source, ya que si este sufriera algún problema se caerían todos los sistemas de suministro de respuesta.
- En segundo lugar, el rendimiento de la propia aplicación Golden Source, está por debajo del deseado, ralentizando las respuestas de la base de datos.
- Por otro lado se encuentra la problemática de traducción de la información para almacenarla en la caché, haciendo imposible la conciliación entre las dos tecnologías.
- El funcionamiento de un único proceso de escucha por cada servicio, disminuye el rendimiento de la obtención de información de la bases de datos, al poder llegar a colapsarse las colas de información si llegan muchas peticiones.
- Finalmente existen otras problemáticas desde el punto de vista de la administración como una trazabilidad del flujo de información complicada o una baja flexibilidad a la hora de levantar nuevos entornos de trabajo.

Todas estas problemáticas para llevar a cabo tareas críticas de la entidad, como obtención de información de gran valor para las acciones de la empresa, motivaron la evolución del tratamiento de esta información hacia una primera fase, la Fase I.

Para explicar los cambios de este primer evolutivo se añade la imagen de la Fase I para poder comprenderlo con mayor facilidad.

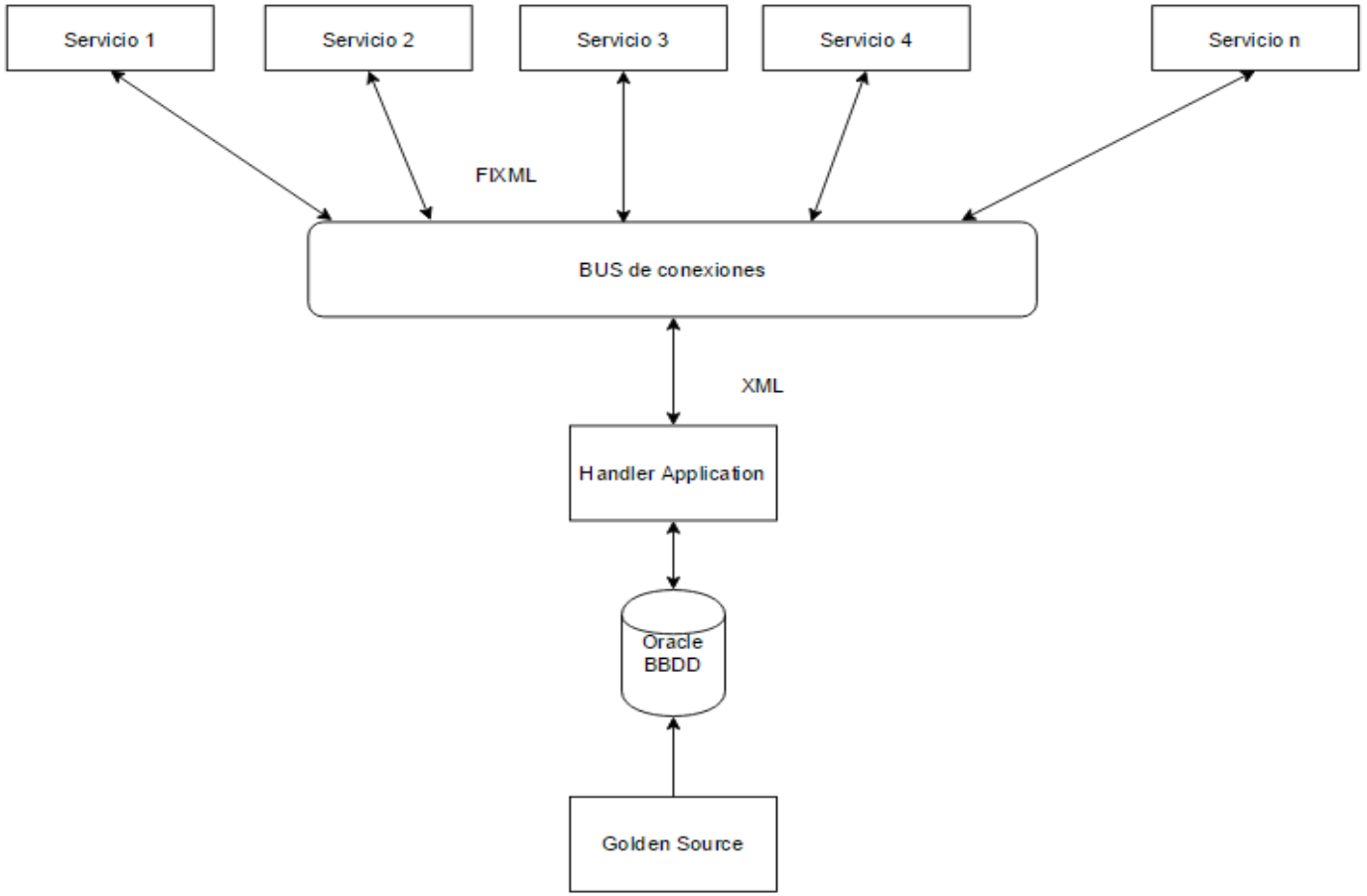



*Figura 3. Diagrama Fase I*

En este proyecto pueden apreciarse diversos cambios, entre ellos está la desaparición de la caché Coherence, la cual se planea reaparezca en una futura fase, la conexión por un único punto, en lenguaje XML, a la base de datos y la aparición de una aplicación Java que se encargará de gestionar todas las peticiones de servicios síncronos que llegan a través del ESB.

Para entender más fácilmente el funcionamiento de este nuevo modelo, se muestra a continuación un diagrama de flujo de una petición cualquiera lanzada por el Servicio número 1.

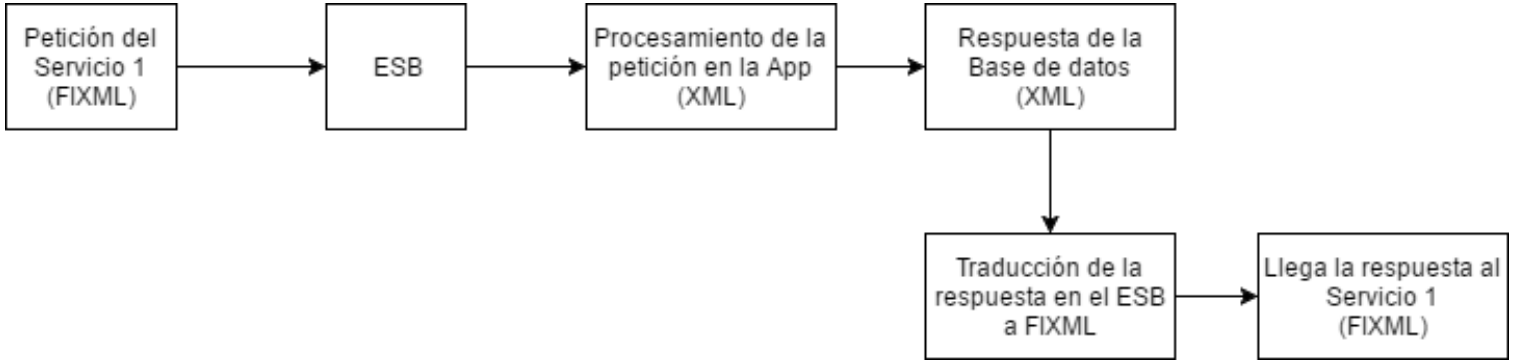



*Figura 4. Funcionamiento de la Fase I*

Como se puede observar, el funcionamiento es más sencillo, aunque parecido, al modelo actual. Los cambios más notables son:

- La traducción a lenguaje XML se realiza en un único punto, para luego distribuirse a todo el sistema de bases de datos.
- La aparición de la App Java encargada de gestionar las peticiones que llegan desde el ESB, de hacer la conexión a la base de datos y de devolver la respuesta obtenida de la base de datos al ESB.

Este nuevo modelo de almacenamiento abre muchas puertas a la hora de obtener la información, trayendo consigo múltiples ventajas que cubrirán muchos de los problemas del modelo actual:

- Se independiza de Golden Source a la hora de gestionar las peticiones procedentes de la base de datos y de realizar las respuestas, ya que todo el flujo de información va directamente entre la App Java y la Base de datos, teniendo Golden Source de respaldo.
- Como la nueva plataforma funciona de manera independiente a Golden Source, quita el problema de baja eficiencia de Golden Source, al depender de la nueva plataforma de gestión.
- La traducción de las peticiones se realiza en un único punto, en el ESB, y una sola vez, permitiendo así agilizar los procesos.
- Se flexibiliza el uso de la base de datos, ya que se permite la configuración de diferentes parámetros de gestión de las peticiones y de conexión a la base de datos: permite fijar el número de procesos de escucha abiertos; se puede variar el número de conexiones abiertas con la Base de datos; por último, se puede configurar la memoria asignada a la Java Virtual Machine, configurando el gasto de memoria que realizará la App Java en la máquina que esté corriendo. Así si se produce un error por *out of memory* en cualquiera de las máquinas no influirá sobre el funcionamiento de cualquier otra.

- Se implementa una capacidad de trazabilidad en el desarrollo del flujo de información, desde que se ejecuta el programa hasta que se para, gracias al sistema de log del que dispone la aplicación, el cual informa en todo momento de el transcurso de la aplicación.
- Permite una alta disponibilidad al poder estar corriendo en varias máquinas al mismo tiempo, mediante un proceso de instalación sencillo, lo cual permite que siempre haya muchos procesos escuchando todas las peticiones que llegan de cada servicio.

Además de todos estos puntos positivos, del nuevo modelo, se añade la aparición de una aplicación, también Java, que permite realizar pruebas, en diferentes entornos, automáticas o controladas, del correcto funcionamiento de la App Java encargada de la gestión del sistema de almacenamiento y de Golden Source, al comprobar si coinciden las respuestas suministradas con las respuestas que debería de dar, para cada petición particular. Así mismo, esta herramienta se encontrará implementada dentro de la integración continua, esto es, el software se implementará tanto en entornos del funcionamiento normal de la empresa, producción, y en entornos de pruebas antes de su implementación final.

Finalmente, y aunque este modelo solventa muchos de los problemas que aparecían en el sistema actual, también siguen existiendo diferentes riesgos como la aparición de un posible cuello de botella, al haber un único punto de acceso a la información o la posibilidad de pérdidas de memoria, al realizar un desarrollo “in-house”.

Estos posibles riesgos favorecerán al desarrollo de un Modelo II del cual se tratará en detalle en el apartado de ***Trabajos futuros*** del presente trabajo.

# Capítulo 4. DEFINICIÓN DEL TRABAJO

En el presente capítulo se dará respuesta a diferentes características del proyecto, así como la motivación, los objetivos, la metodología y planificación y la estimación económica. Estos puntos son claves para entender el porqué de realizar este proyecto y la realidad de cómo ha podido llevarse a cabo, y a qué precio.

## *4.1 MOTIVACIÓN*

La información, como ya se ha comentado en el capítulo *introducción*, es la piedra angular en torno a la cual gira la sociedad actual siendo las empresas con mayores crecimientos económicos las empresas dedicadas al intercambio de información personal, como son las redes sociales.

A esto hay que añadir que tratar la información se está convirtiendo en un impulsor de la economía y de la investigación, sobresaliendo campos como el *big data*, o el *IOT*. Por lo tanto, el tratamiento de la información se tiene que tomar muy en cuenta en cualquier empresa actual.

Los problemas que pueden surgir a la hora de manejar la información son dos: la congestión que se produce en la infraestructura de comunicaciones y la congestión que se causa en el acceso a la información almacenada en las bases de datos.

El primer problema se puede solucionar mejorando las infraestructuras ya existentes, investigando en otras más eficientes, o duplicando los canales de comunicación que hay en la actualidad.

El segundo problema tiene más difícil solución, esto se debe a que el acceso a la información en las bases de datos siempre se da por un mismo camino, el cual acaba formando el denominado *cuello de botella*, es decir, se acaba formando atascos de la información.

Para mejorar esta congestión en el acceso a la información, han aparecido las denominadas memorias caché, o memorias de acceso rápido, las cuales permiten obtener un acceso más rápido a la información.

La dificultad principal que radica del acceso a la información, mediante memorias caché, es que estas tienen almacenada una cantidad limitada de todo lo que está guardado en la base de datos, ya que sino se ralentizaría y lo único que se conseguiría sería duplicar la base de datos, no solucionando así el problema original del cuello de botella de información.

Por lo tanto, los esfuerzos en esta área deben centrarse en la información que debe de estar almacenada en la memoria caché, normalmente se basa en algoritmos estadísticos, y en ser capaces de aunar los esfuerzos de la memoria de la base de datos y la caché, para dar la respuesta en la mayor brevedad posible y ser capaz de responder a la mayor cantidad de peticiones en el menor tiempo.

Todo ello hace que la investigación en este campo se muestre como un área llena de posibilidades, y que además este demandada en el mundo laboral internacional.

## *4.2 OBJETIVOS*

El desarrollo de este proyecto se basa en las necesidades de agilizar los procesos de comunicación que existen actualmente en la obtención de información de la base de datos, por lo que los objetivos del proyecto girarán en torno a estas necesidades.

Como sabemos, la problemática se centra en tener un cuello de botella formado por el acceso a la información que contiene un único servidor y comparten 13 servicios. Por ello, el principal objetivo es facilitar el acceso de las aplicaciones que lo requieran a la información que contiene la base de datos, agilizando el proceso y, por lo tanto, "descongestionar" el tráfico que se forma en la entrada de la base de datos para que cualquier operación que requiera de la misma se realice con la mayor brevedad posible.

Así mismo, tratarán de realizarse diferentes funcionalidades y utilidades para facilitar los trabajos de gestión y de administración de los responsables del mantenimiento y seguimiento del correcto funcionamiento del sistema a implementar.

Además, tratará de aumentarse la disponibilidad de la información, ya que aunque la base de datos se caiga, en un momento determinado, toda la información que esté guardada en la caché seguirá estando disponible y accesible, funcionando así como respaldo de la propia base de datos. Esto se debe a que la aplicación java correrá independientemente a la plataforma de la base de datos.

El último objetivo o finalidad del proyecto, será asegurar la conciliación entre Golden Source y Oracle Coherence, esto es, asegurarse de que se cumpla el correcto funcionamiento entre dos programas diferentes.

## *4.3 METODOLOGÍA Y PLANIFICACIÓN*

Uno de los puntos más importantes a la hora de desarrollar un proyecto es realizar una buena planificación de los trabajos que se van a realizar y de cómo se van a llevar a cabo. Esto es de gran importancia, debido a que el éxito del proyecto dependerá de realizar una buena planificación de cada uno de las tareas que hay que hacer para poder cumplir con los tiempos de entrega del proyecto.

Para la planificación del proyecto se ha hecho un pequeño estudio de las diferentes metodologías para implementar en el transcurso del proyecto.

Se conoce una metodología como una plantilla o “framework”, la cual seguir, para llegar al desarrollo de un proyecto en concreto.

El proyecto que se está desarrollando, en este caso, se compone de un alto grado de programación, por lo cual es interesante buscar las metodologías que más se adapten y faciliten la labor del desarrollo software.

Entre las metodologías más destacadas para el desarrollo software encontramos el Modelo en cascada, el Prototipado, el Incremental, el Espiral, el Rapid Application Development y la metodología Ágil.

Tras una evaluación de las diferentes ventajas y desventajas que suministra cada una de las metodologías, se concluyó, que la que más se adecuaba a las necesidades del proyecto era la metodología Ágil.

La metodología Ágil, se basa en partir el desarrollo del software completo en pequeñas partes o iteraciones que se tendrán que realizar. Estas iteraciones se dividirán en a todo el equipo que participa en el proyecto, y cada uno de ellos tendrá que encargarse de terminar las tareas que le son asignadas. Las tareas podrán encontrarse en diferentes fases, pendiente, haciéndose, en pausa o terminada, dependiendo del grado de avance en el que se encuentre. Finalmente, todas las tareas se juntan para dar forma al proyecto final y poder llevar a cabo su implementación.

En la ilustración mostrada a continuación, se podrá ver la planificación general del proyecto, mediante un diagrama de GANTT.

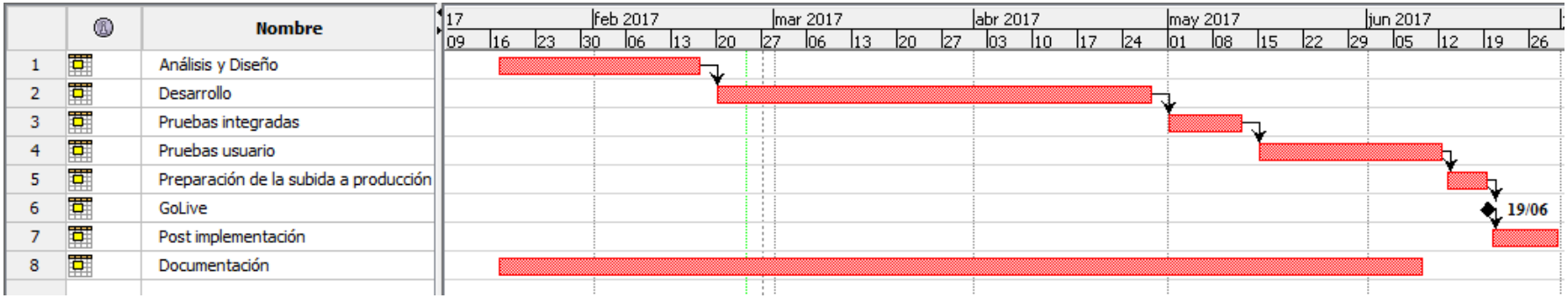



*Figura 5. Diagrama de Gantt*

Como se puede observar, el proyecto se desglosa en cada una de las tareas resumen que conlleva el mismo, haciendo una distribución a lo largo de seis meses, desde enero hasta junio del 2017. Además se puede ver que en paralelo se encuentra la tarea de documentación de la memoria del proyecto.

Además de esta planificación, del proyecto, se ha realizado una planificación de los sprints que contiene cada una de estas tareas resumen, mostradas en el diagrama de GANTT, mediante el uso del programa Trello, comentado en el capítulo de *Descripción de las Herramientas*.

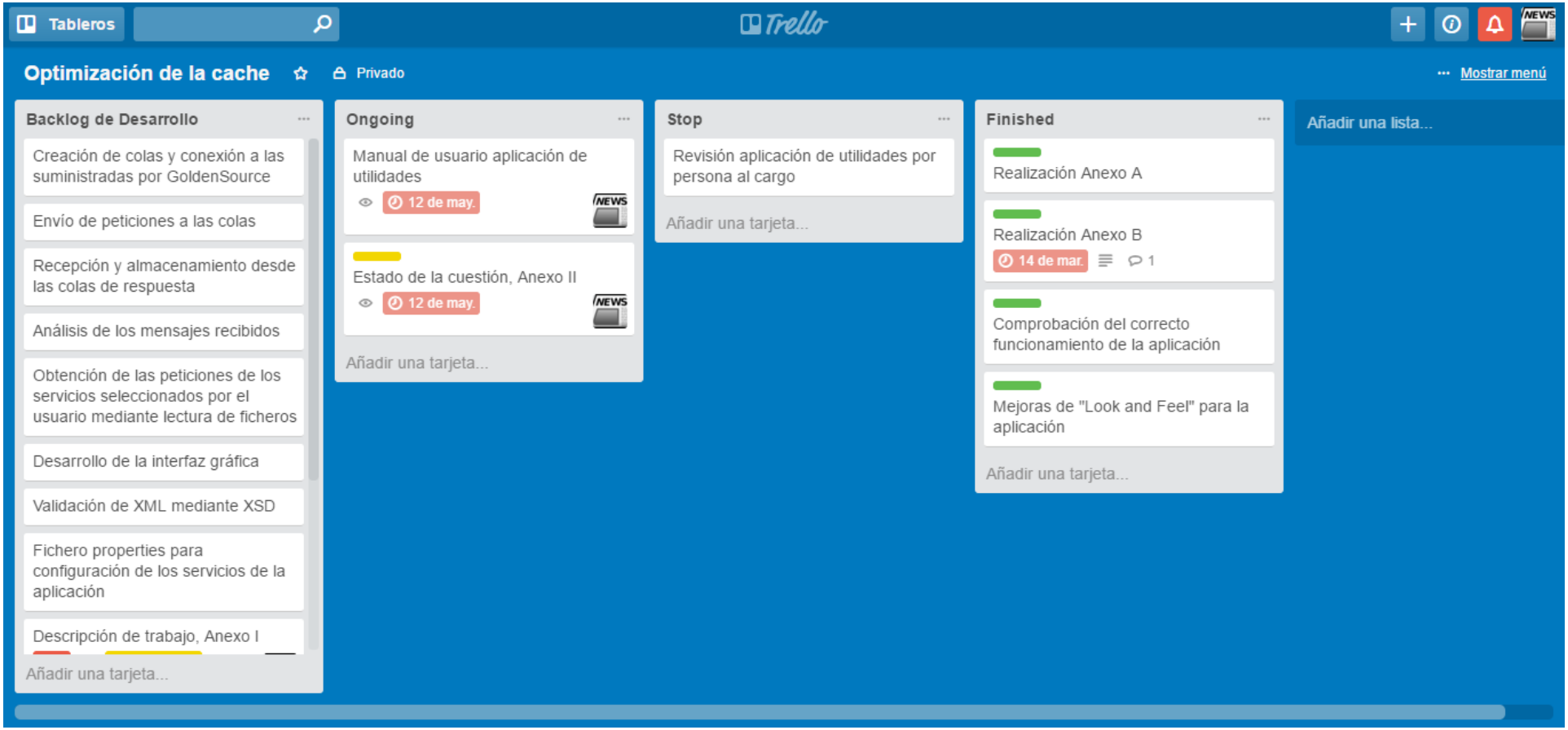



*Figura 6. Planificación metodología Ágil*

Es esta imagen se pueden diferenciar diferentes tableros, en los que se contienen cada uno de los sprints:

- En el tablero **Backlog de Desarrollo** se encuentran todas las historias de usuario que hay que realizar hasta el final del proyecto, es decir, cada una de las tareas que quedan por hacer para terminar el trabajo.
- En el tablero **Ongoing** se encuentran las historias de usuario que están actualmente en desarrollo, en este caso se encuentra la documentación de la aplicación de administración y gestión, y un Anexo del presente documento.
- En el tablero **Stop** se encuentran las historias de usuario que están paradas hasta la ocurrencia de un evento, en este caso un desarrollo está parado hasta que un superior acepte la versión que se le ha mostrado del programa.

En el tablero **Finished** se encuentran las historias de usuario que ya han sido realizados y finalizados de manera correcta.

## *4.4 ESTIMACIÓN ECONÓMICA*

Uno de los aspectos más importantes a tener en cuenta a la hora llevar a cabo un proyecto, dentro de cualquier entidad, es la realización de una estimación económica para cubrir todos los gastos del mismo.

Para ello es necesario hablar del tiempo de duración del proyecto, el cual tendrá una duración total de 6 meses. Durante estos meses, suponiendo todos los meses de 4,3 semanas y con un trabajo semanal de 20 horas se obtiene, fácilmente, que el total de horas trabajadas es de 516 horas (6 meses * 4,3 semanas cada mes * 20 horas por semana).

Teniendo una claro el tiempo que se ha empleado en el desarrollo del proyecto, el siguiente paso es realizar una tabla con cada uno de los gastos que se producirán para el desarrollo del proyecto.

| Concepto | Justificación | Coste |
|---|---|---|
| Ordenador | Se ha usado un ordenador de sobremesa con un procesador i7, 16 GB de RAM y 2 TB de disco duro | 986 € |
| Monitor del ordenador | Se han usado un monitor de ordenador HP de gama media | 153 € |
| Trabajadores | Se ha empleado un trabajador en la realización del proyecto a 55,59€ la hora | 28.684,44 € |
| Herramientas | Se han empleado todas las herramientas descritas en el apartado ***descripción de herramientas*** | 0 € |
| **Total** | | **29.823,44 €** |

*Tabla 1. Estimación económica del proyecto*

# Capítulo 5. SISTEMA DESARROLLADO

En el presente capítulo, se procederá a realizar una explicación exhaustiva sobre el funcionamiento de las aplicaciones que se han llevado a cabo en el transcurso del proyecto.

Para realizar dicha explicación, es preciso recordar que hay dos aplicaciones programadas, por un lado se encuentra la aplicación de pruebas regresivas, y por otro la nueva plataforma de gestión de las peticiones de los servicios.

## *5.1 APLICACIÓN DE PRUEBAS REGRESIVAS*

Debido a la aparición de la nueva plataforma de gestión de peticiones de los servicios del banco, era necesaria la creación de una herramienta que se encargada de comprobar el correcto funcionamiento de la plataforma. Esto se debe a que cuando se realiza cualquier cambio en el software, se debe asegurar que no existan regresiones en el mismo.

La herramienta de pruebas maneja un sistema de ficheros, almacenados de manera local, en los cuales guarda las peticiones con sus respuestas respectivas, en un archivo para cada servicio, mediante el estándar csv.

El estándar csv permite guardar, de manera ordenada, las peticiones y respuestas de la forma “petición”,”respuesta”. Por lo tanto, el usuario que emplee la herramienta, tendrá siempre disponibles todas las peticiones y respuestas a través de la herramienta. La gestión de todas estas peticiones se puede ejercer a través de la propia aplicación.

Se muestra un ejemplo de la disposición de peticiones y respuestas, en un archivo .csv para el servicio Calendars.

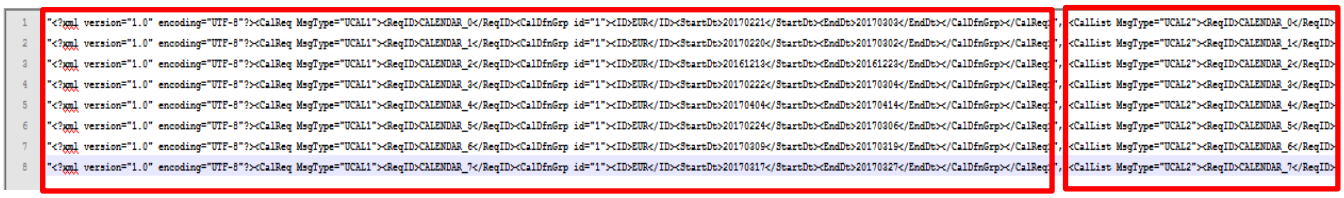



*Figura 7. Peticiones y respuestas en estándar csv*

Como se puede observar, se forman dos columnas de peticiones y respuesta para cada una de las peticiones, así se facilitará su almacenamiento en un HashMap durante el funcionamiento del programa.

Además para que estas peticiones puedan ser guardadas en los ficheros, antes debe ser comprobada su estructura mediante estándares xsd que la propia aplicación recupera de la base datos. Estos esquemas xsd son estructuras básicas xml que una petición o respuesta debe cumplir para que esta sea válida, en este caso para que sea válida para el servicio Calendars.

Una vez se han guardado todas las peticiones y respuestas que se quieran utilizar, el usuario ya está en disposición de realizar las pruebas que considere necesarias. Para entender el correcto funcionamiento de la aplicación, se muestra a continuación un diagrama del flujo del funcionamiento.

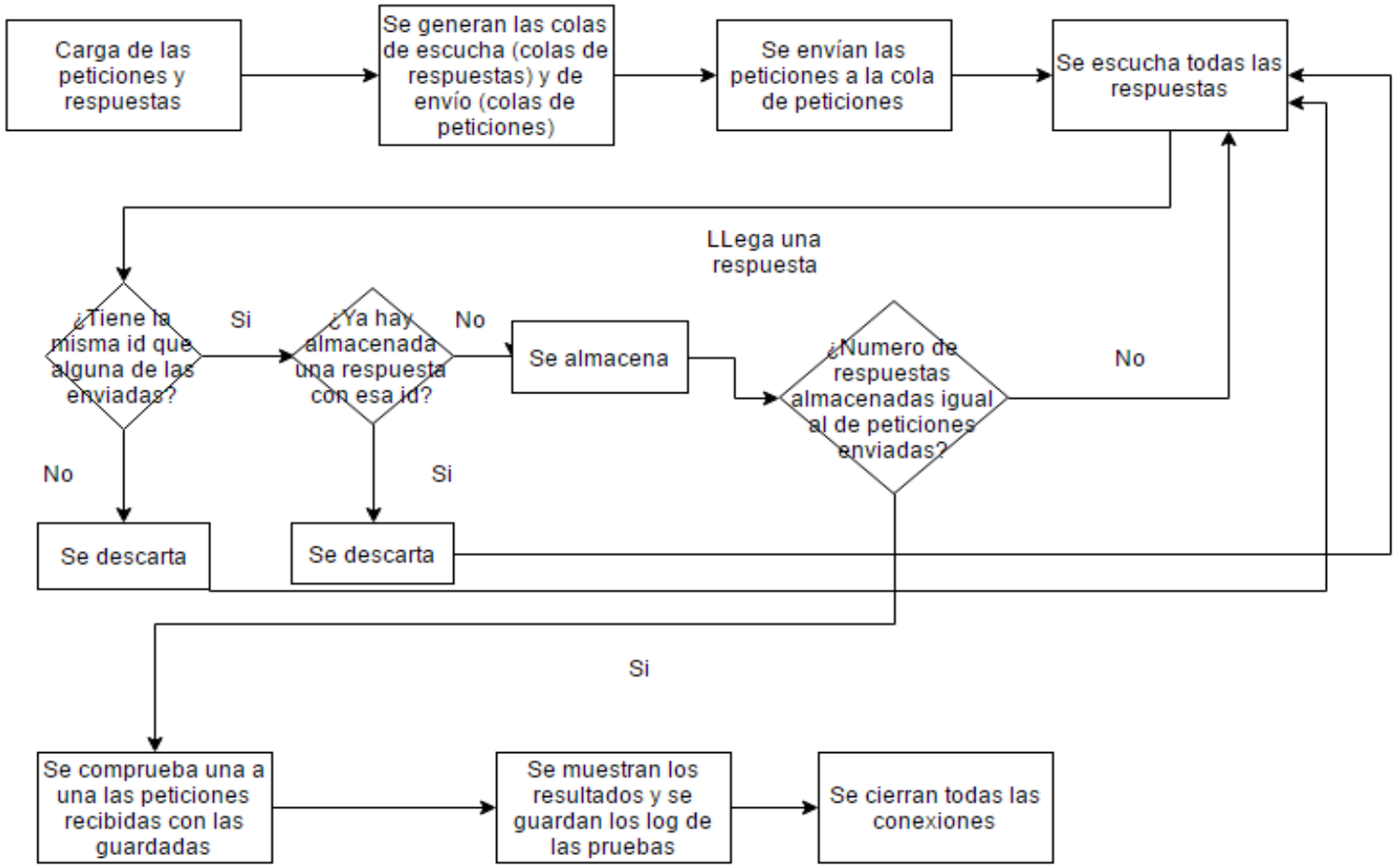



*Figura 8. Diagrama de funcionamiento de Testing Tool modo test*

Antes de comenzar el flujo de información, mostrado en el esquema anterior, el usuario debe haber ingresado el archivo de credenciales, en el cual se encuentra toda la información necesaria para realizar la conexión con la base de datos y con los servidores de las colas.

Para cargar estas credenciales será necesario indicar la ruta o "path" del archivo credentials en el ordenador donde esté funcionando el programa. Cabe destacar que esta indicación solo será necesario realizarla la primera vez que se arranque el programa o cada vez que se quiera cambiar de entorno de trabajo, ya que el programa aprende de manera automática dicha dirección, almacenándola en un archivo properties, interno al programa.

Una vez el archivo con las credenciales ha sido ingresado, puede comenzar el flujo mostrado anteriormente que, como puede apreciarse en el diagrama, todo comienza cuando el usuario elige el servicio, o servicios que quiere probar. En ese momento, la aplicación, lee de los ficheros todas las peticiones y respuestas, que tiene almacenadas para ese servicio y comprueba que estén bien formadas, mediante esquemas XSD, y les sobrescribe el id con la siguiente nomenclatura, "NOMBRESERVICIO_NUMERODEPETICIÓN", descartándose todas las peticiones que no cumplan con el XSD.

Para finalizar el procesado de lectura de peticiones y respuestas, estas vuelven a ser escritas en el fichero, borrando todo el contenido previo, para escribir el fichero de manera correcta, por si alguien o algo pudiera haber manipulado el fichero.

Además, estas peticiones y respuestas serán almacenadas en dos HashMaps diferentes, relacionados mediante las id dadas anteriormente, produciendo así que se vuelva a comprobar si hay duplicidad de peticiones o de respuestas, ya que los HashMaps no permiten elementos duplicados.

Esta comprobación es importante realizarla, debido a que cualquier usuario puede introducir peticiones mal formadas directamente en los ficheros y, estas peticiones mal formadas, puede producir la caída de los servidores de respuesta.

A continuación se produce la formación de las dos colas JMS:

- La cola de peticiones, es la cola a la que se envían las peticiones que se quieren hacer al servidor y de donde estará escuchando la plataforma de gestión de peticiones y respuestas.
- La cola de respuestas, en la cual se produce el envío de las respuestas de las peticiones que se recibieron por la cola anterior.

Tanto la petición como su respuesta están relacionadas entre ellas por un campo llamado "<ReqId>", es decir, un campo de identificación que permitirá saber si las peticiones enviadas han sido respondidas y, si ha sido así, comprobar si la respuesta ha sido correcta.

Una vez establecidas las colas, mediante la información proporcionada por el archivo de credenciales, se procede al envío de las peticiones a la cola de peticiones. Este envío se produce mediante la creación de hilos, para cada una de las peticiones, para que el programa pueda continuar su funcionamiento y poder enviar todas las peticiones al mismo tiempo. Esto permitirá la agilización del funcionamiento de todos los procesos del programa

Cabe destacar que hay diferentes procesos enfrentándose para la escucha de las peticiones que se envían, por lo que cada una de estas peticiones es enviada cinco veces, esto es debido a que así se trata de asegurar que una de estas peticiones es escuchada por un proceso de la plataforma de gestión de peticiones y respuestas y no otra aplicación que esté escuchando en estas mismas colas.

Una vez enviadas las peticiones se inicia la cuenta de un timeout. Este time out es ajustable por la persona que ejecuta el programa mediante la edición del archivo properties que contiene el programa. Mediante este time out se realiza una espera de un tiempo predeterminado escuchando la cola, JMS, de respuesta para el servicio, o servicios, que se trata de comprobar. Cuando dicho time out llega a 0, se presupone que la petición que se envió ha sido perdida o se produjo algún fallo en la cadena de formación de la respuesta desde que el mensaje salió de la aplicación, por lo que la respuesta se da por perdida o "LOST".

Al tiempo que se comienza el time out se comienzan a procesar las peticiones que llegan por la cola de respuesta y se comienza un cronómetro, para conocer el tiempo que se ha tardado en recibir todas las respuestas, lo cual podrá ofrecer estadísticas importantes a la hora de realizar futuras mejoras y comprobar la eficiencia del programa de gestión de plataformas o handler.

Para realizar la comprobación, o el procesamiento, de las respuestas recibidas, se realizan una serie de etapas:

- Cada vez que llega una respuesta se realiza la obtención de su id, mediante la obtención del campo <ReqId> del xml de entrada.
- En este momento se comprueba si esta id se encuentra contenida en el HashMap de peticiones enviadas, sabiendo así si es una respuesta a una de las peticiones enviadas o si es a una enviada por otra aplicación.
- Posteriormente se realiza una comprobación en el HashMap de respuestas recibidas para saber si este ya contiene esta id, si es así, se descarta dicha respuesta y, de lo contrario, se continua con el procesamiento de la respuesta. Esto se realiza, ya que podrían llegar, como mínimo, cinco respuestas iguales, ya que se enviaron cinco peticiones iguales.
- Además se comprueba si la petición contiene la marca <Snt> que simplemente es una marca de la fecha en la que se procesó la respuesta y, si la contiene, se elimina para poder realizar las comprobaciones posteriores.
- El siguiente paso a realizar es la obtención de la respuesta que se tiene almacenada, con dicha id, y se guarda en una variable, para poder dar paso a las comprobaciones de semejanza entre ambas respuestas.
- Ahora se lleva a cabo una comprobación de las dos respuestas, para ello se realiza una simple comprobación mediante el método equals de la clase String de Java, ya que, teóricamente, ambas peticiones deberían de ser exactamente iguales, incluso los espacios que se encuentran entre etiquetas. Cabe destacar que se realiza una transformación previa a la comprobación mediante la cual se borran espacios, saltos de línea o caracteres extraños antes y después del xml de la respuesta, por si se

pudiera haber colado cualquier tipo de carácter no deseado y estropeara la comprobación.

- El resultado de la comprobación anterior puede devolver tres tipos de resultados:
    - Por un lado se encontraría la respuesta "OK", lo cual indicaría que la comprobación resultó exitosa, es decir, las dos respuestas coinciden al 100%.
    - Por otro lado se encontraría la respuesta "NOK", lo que querría decir que se encontró algún tipo de fallo en la comprobación de las respuestas, por lo que la respuesta recibida se considera errónea.
    - Finalmente se encontraría la respuesta "LOST", lo que significaría que no se ha recibido respuesta ninguna para la petición enviada.
- Una vez realizada la comprobación, se daría paso al almacenamiento del resultado, en un HashMap para evitar duplicidad, cuya clave de mapa sería la id de la respuesta y el valor sería el resultado de la comprobación, esto es: "OK", "NOK" o "LOST".
- Este proceso se repetiría para cada una de las respuestas que se recibieran mediante la cola de respuestas hasta que se produzca cualquiera de los dos eventos siguientes:
    - Finaliza el time out, es decir, que el tiempo de espera para estar escuchando peticiones llega a su fin.
    - Se han recibido tantas respuestas como peticiones diferentes se han enviado, se resalta lo de diferentes, ya que cabe recordar que se enviaron cinco veces más peticiones de las necesarias.
- Una vez se den cualquiera de los dos eventos anteriores, otro hilo toma el control, para poder continuar con el funcionamiento del programa, mientras tanto, se realiza la cuenta de todas las respuestas 100% exitosas, las que contienen algún tipo de error y las peticiones que se han perdido y se realiza su escritura en un JFrame, si se está trabajando con el formato en interfaz gráfica, o se muestra el resultado por pantalla, si se trabaja con el formato por línea de comandos.

- Para finalizar, se guardan los resultados en cada uno de los archivos de log, cuya ruta se encuentra guardada en el archivo properties del programa y puede ser variada por la persona que ejecuta el programa.

Este almacenamiento de las pruebas es importante, sobre todo para la versión automatizada mediante líneas de comandos, para poder tener una trazabilidad visible de las pruebas realizadas.

Este es uno de los modos de funcionamiento de la aplicación, aunque también puede observarse otra forma de funcionamiento, la denominada simulación.

La simulación, como su propio nombre indica, permite recrear la existencia de un servicio ficticio, y permite el funcionamiento de la aplicación como si fuera una plataforma de gestión de peticiones y respuestas por sí mismo. Es mportante señalizar, que la aplicación solo responderá a las peticiones que tenga almacenadas previamente en sus ficheros de peticiones y respuestas de simulación, los cuales son independientes a los del funcionamiento normal, el de pruebas.

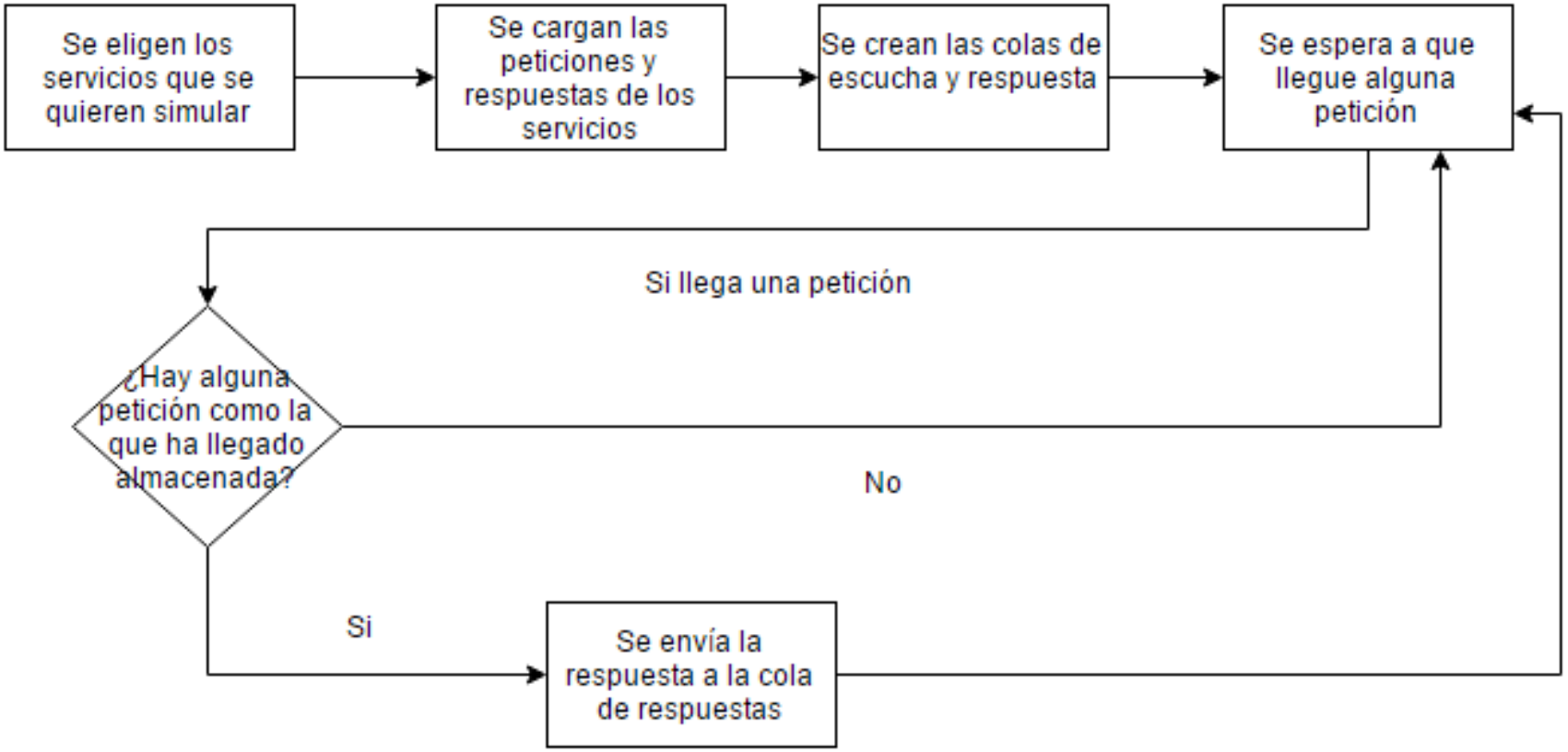



*Figura 9. Diagrama funcionamiento Testing Tool modo simulación*

Como se puede ver, en este diagrama, el funcionamiento de la parte de simulación funciona de una manera "inversa" a la del funcionamiento de las pruebas.

Para llevar a cabo este funcionamiento, el usuario debe elegir uno, o varios, de los servicios que hay en la lista, el cual puede haber creado, de servicios y pulsar el botón de comenzar la simulación del servicio.

Una vez comienza el modo simulación, este no finalizará hasta que se el programa se cierre, es decir, hasta que no se finalice el programa.

El funcionamiento que lleva la parte de simulación, puede explicarse con los siguientes puntos clave:

- La simulación comienza con la formación de las colas de escucha, para el servicio, o servicios, que ha seleccionado, mediante las direcciones almacenadas en el archivo properties y las credenciales de conexión almacenadas en el archivo credentials.
- Al igual que en el caso de las pruebas regresivas, cuando llega una petición, a la cola de escucha, lo primero que se hace es obtener el id de la petición, pero esta vez, la id, se almacena en una variable y se borra de la petición de entrada.
- Una vez que la petición no tiene id, esta se compara con cada una de las peticiones almacenadas en los ficheros de simulación, que ya no contienen id. De esta comprobación se pueden dar dos casos:
    - Por un lado, la comprobación puede ser errónea, lo que hará que la petición sea desechada y se pase a seguir escuchando más peticiones.
    - Por otro lado, la comprobación puede ser exitosa, y se continuará con el procesamiento de la petición.
- Si la comprobación ha sido exitosa, esto significará que se tiene almacenada una respuesta para dicha petición, por lo que se obtendrá dicha respuesta y se almacenará en una variable.

- Una vez que se tenga la respuesta, se le implantará la id de la petición, para que el programa que envió la petición pueda identificar la respuesta correspondiente, y se procede a crear la cola de respuesta para enviar dicha respuesta.

Esto permitirá que se puedan realizar pruebas de funcionamiento de hipotéticos servicios, incluso ejecutando la aplicación dos veces, al mismo tiempo, permitiendo así realizar pruebas de servicios antes de proceder a su implantación.

## *5.2 PLATAFORMA DE GESTIÓN DE PETICIONES Y RESPUESTAS*

La plataforma de gestión de peticiones y respuestas se trata de una aplicación que permitirá ejercer de intermediario entre la base de datos y las peticiones de información, a cada uno de los servicios, para responder a las mismas en la mayor brevedad posible y de la manera más eficiente.

Esta es totalmente configurable ya que permite ajustar el número de conexiones abiertas al mismo tiempo a la base de datos, en paralelo, o incluso el número de procesos escuchando en las colas para atender a todas las peticiones que llegan.

Así mismo, permite la creación de nuevos servicios a través del archivo de propiedades mediante la escritura de las características principales del nuevo servicio como son la dirección de la cola de entrada y salida y la dirección donde se encuentran los archivos XSD, entre otros parámetros.

Un ejemplo de configuración de un servicio genérico, se explicará a continuación qué es, sería el siguiente:

```
######################
#Service example
######################
generic.example.servicename=Example
generic.example.queue.request=KYRS.RDR.EXAMPLE.REQUEST
generic.example.queue.response=KYRS.RDR.EXAMPLE.RESPONSE

generic.example.xslt_getquery=db://resource/ExampleGetQuery.xslt
generic.example.xslt_getparams=db://resource/ExampleGetQueryParameters.xslt

generic.example.query.1=RDR_ExampleAllParameters
generic.example.query.2=RDR_ExampleCurrency
generic.example.query.3=RDR_ExampleProducts
generic.example.output_msg=EXP
generic.example.resp_node=ExmpResp
generic.example.check_result=/ExmpResp/Exmp

#XSD VALIDATION
generic.example.xsd.input=db://resource/QueueExampleRequest.xsd
generic.example.xsd.output=db://resource/QueueExampleResponse.xsd
```

*Figura 10. Creación nuevo servicio genérico Handler Application*

Por lo que la creación de un nuevo servicio, para que la plataforma comienza a responder a peticiones, para el mismo, se reduce a los siguientes campos:

- Servicename: este campo se refiere al nombre del servicio, el cual será empleado para identificar el mismo, mostrar mensajes informativos o guardados de logs de información.
- Queue.request: dirección de la cola de entrada en la cual la plataforma se encontrará a la escucha de peticiones, para su posterior procesamiento.
- Queue.response: dirección de la cola de salida, o de respuesta, a la cual se enviarán las respuestas producidas al procesar las peticiones de entrada recibidas.
- Xslt_getquery: se refiere a la dirección donde se encuentra almacenado el código xslt para la obtención de la query a partir de una petición de entrada.
- Xslt_getparamr: se refiere a la dirección donde se encuentra almacenado el código xslt para la obtención de los parámetros de la petición de entrada, para poder ejecutar la query, obtenida en el punto anterior.

- Query.número: nombre de una de las queries que puede ejecutar el servicio en cuestión, para la obtención de la respuesta.
- Output_msg: parámetro a añadir en la formación del xml de respuesta.
- Resp_node: nodo padre del xml de respuesta que se formará.
- Check_result: parámetro que permitirá obtener los nodos de las peticiones xml de los servicios.
- Xsd.input: dirección donde se encuentra el esquema de comprobación de la estructura de los mensajes de entrada, peticiones.
- Xsd.output: dirección donde se encuentra el esquema de comprobación de la estructura de los mensajes de salida, respuestas.

El tratamiento de las peticiones se realizará de manera diferente, para cada uno de los servicios. Así se pudo comprobar que hay una serie de servicios que tienen unas características peculiares y, otros, que tienen características muy similares entre ellos, dando así lugar a dos tipos de servicios:

- Se denominan servicios genéricos a todos aquellos que tienen unas características parecidas, entre ellos, y que por lo tanto pueden ser agrupados dentro de una misma categoría.
- Se denominan servicios especiales a aquellos que tienen unas características individuales diferentes a los demás, lo que hace que no puedan ser introducidos en ninguna categoría común.

Esta diferenciación, producirá la creación de clases Java diferentes, para cada uno de los servicios especiales, y una para los servicios genéricos.

Esto se debe a que cada uno de los servicios individuales necesitará de unos procedimientos diferentes, a la hora de realizar el procesamiento de una petición de entrada, para obtener la respuesta.

Sin embargo, los servicios genéricos disponen de las características comunes que permiten el procesamiento de las peticiones de cada uno de estos servicios, con el simple ca,bio de

datos como las queries de obtención de respuesta o la dirección de los esquemas de comprobación xsd.

Para realizar una mejor ejemplificación y distinguir de manera más sencilla entre estos dos tipos de servicios, a continuación se procederá a mostrar, en primer lugar, el ejemplo del funcionamiento de un servicio especial, en este caso “Calendars” y de un servicio genérico como puede ser “Contacts”.

Sin embargo, antes de esta ejemplificación, se procederá a mostrar el proceso de inicialización del programa, por el cual se abrirán todas las conexiones y se preparará toda la información necesaria para el procesamiento cualquier petición que llegue por cualquiera de los procesos de escucha que están activos.

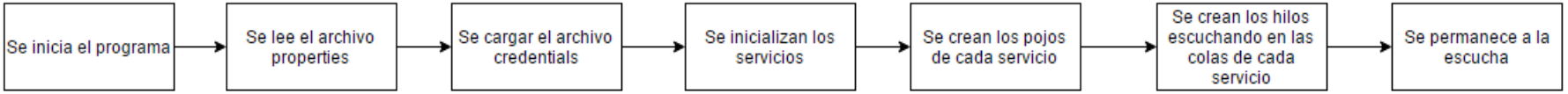



*Figura 11. Diagrama inicialización Handler Application*

Como se puede observar, mediante el diagrama anterior, el proceso de inicialización de la aplicación es rápido y fácil de comprender, aunque a continuación se realiza una explicación más detallada de cada uno de los pasos y de sus propósitos:

- En primer lugar, se realiza la lectura del archivo properties, mediante un FileInputStream, de todas las propiedades que hay en dicho archivo. Esto proporcionará toda la información necesaria de cada uno de los servicios. Así mismo, este archivo también contiene información importante para el normal, y óptimo, funcionamiento de la aplicación como son en número de hilos de escucha para cada servicio o el máximo número de conexiones en paralelo permitidas a la base de datos.
- La información de cada uno de los servicios es obtenida mediante las clases Config*NombreServicio* programadas en la aplicación, las cuales almacenarán la información contenida en las propiedades, para poder acceder a ellas en tiempo de ejecución de manera más ágil que si hubiera que hacerlo accediendo al fichero properties.
- El siguiente paso es cargar el archivo credentials, dicho archivo contendrá la información necesaria para poder establecer las conexiones con las colas JMS y para poder realizar la conexión a la base de datos.
- En siguiente lugar, se realiza la creación de las clases “pojo” que permitirán almacenar toda la información de entrada de las peticiones así como sus parámetros o su id.
- El siguiente paso a realizar es la creación de los hilos de conexión, los cuales estarán condicionados por la cantidad de ellos permitidos, en el archivo properties, mediante la información suministrada en el archivo credentials, que fue procesada en puntos anteriores.
- Finalmente, y una vez inicializados todos los servicios, se procede a permanecer a la escucha hasta la llegada de alguna petición que produzca el procesamiento de la misma.

Una vez que se han inicializado los servicios necesarios para el funcionamiento del programa, se permanece a la escucha, como dice el último punto, hasta que la llegada de una petición produzca que comience el procesamiento de la misma.

Como se adelantó, anteriormente, los servicios se podrán dividir en servicios genéricos o en servicios especiales. A continuación se procederá a mostrar el procesamiento de una petición que pertenezca a uno de estos servicios especiales como es "*Calendars"*, posteriormente, se realizará lo mismo con un servicio genérico.

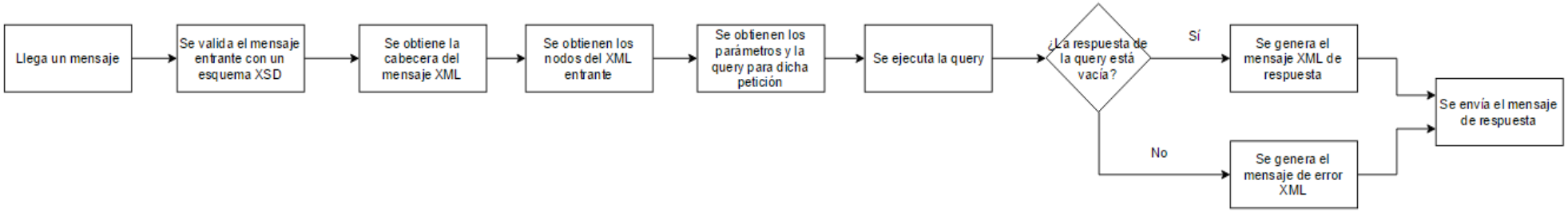



*Figura 12. Diagrama funcionamiento del servicio específico en Handler Application*

El diagrama anterior muestra el camino que sigue una petición hasta que se forma la respuesta que se enviará para atender a la petición que llegó, dicho camino se explica de manera más detallada a continuación:

- En primer lugar, y una vez que llega una petición por la cola de entrada del servicio *Calendars*, el paso que se da es comprobar si la estructura que tiene dicha petición es la correcta, para ello se empleará el esquema XSD de validación de mensajes de entrada que se cargó en la fase de inicialización del programa.
- Una vez, validado el mensaje de entrada, se pasa a obtener las cabeceras del mismo, esto servirá para obtener información como la id del mensaje o el tipo de mensaje de entrada que se ha recibido.
- El paso siguiente, a la obtención de la cabecera, será la obtención de los nodos que componen el mensaje xml, para que así se pueda obtener los parámetros que trae el mensaje.
- Ahora ya se está en disposición de poder obtener los parámetros y de la query a ejecutar, para ello se empleará, por un lado el xslt de obtención de parámetros, y por otro lado el xslt de obtención de la query.
- Una vez procesada la petición, se está en posición de poder producir la respuesta que nos es pedida, para ello se pasa a ejecutar la query, obtenida en el punto anterior, rellenando cada uno de los parámetros, para la ejecución de la query, con los parámetros obtenidos también en el punto anterior.
- Se comprueba si la respuesta suministrada por la base de datos está vacía o no, en el caso de que la respuesta de la query está vacía, se genera un mensaje de error, que se acoplará al xml de respuesta, de lo contrario, se empleará la respuesta obtenida para la generación de la respuesta.
- En este punto, se da paso a formar la respuesta, mediante un esquema xml prefijado, en el que se rellenan los datos con el mensaje de error, debido a que la petición no produce respuesta, o con la respuesta obtenida en la ejecución de la query.

- Para terminar, la respuesta que se ha formado, es enviada, mediante un hilo, por la cola JMS de respuesta, que se genera con los datos de conexión del credentials y con la dirección de envío del Config*Calendar*.

Cabe destacar que si hay varios hilos corriendo en paralelo, se podrá estar procesando más de una petición de un servicio al mismo tiempo.

Para continuar con la explicación de cada uno de los modos de funcionamiento del Handler Java, se procede a mostrar el funcionamiento de la aplicación si la petición que hubiera llegado perteneciera a un servicio genérico como sería el caso del servicio "*Contacts*":

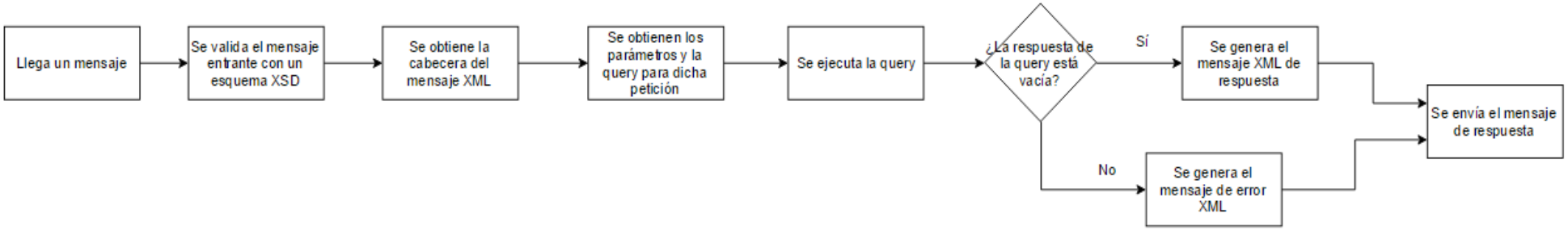



*Figura 13. Diagrama funcionamiento servicio genérico en Handler Application*

Como en los diagramas anteriores, a continuación se procede a dar una explicación más exhaustiva del funcionamiento de la aplicación, para que se comprenda de mejor manera:

- En primer lugar, y como en el servicio *Calendars* y una vez que llega una petición por la cola de entrada del servicio *Contacts*, el paso que se da es comprobar si la estructura que tiene dicha petición es la correcta, para ello se empleará el esquema XSD de validación de mensajes de entrada que se cargó en la fase de inicialización del programa.
- Una vez, validado el mensaje de entrada, se pasa a obtener las cabeceras del mismo, esto servirá para obtener información como la id del mensaje o el tipo de mensaje de entrada que se ha recibido.
- La diferencia de este servicio al específico, que no tendría por qué ser igual entre todos, es que no se realiza una obtención de los nodos de la petición, directamente se pasa a la obtención de parámetros.
- Ahora ya se está en disposición de poder obtener los parámetros y de la query a ejecutar, para ello se empleará, por un lado el xslt de obtención de parámetros, y por otro lado el xslt de obtención de la query.
- Una vez procesada la petición, se está en posición de poder producir la respuesta que nos es pedida, para ello se pasa a ejecutar la query, obtenida en el punto anterior, rellenando cada uno de los parámetros, para la ejecución de la query, con los parámetros obtenidos también en el punto anterior.
- Se comprueba si la respuesta suministrada por la base de datos está vacía o no, en el caso de que la respuesta de la query está vacía, se genera un mensaje de error, que se acoplará al xml de respuesta, de lo contrario, se empleará la respuesta obtenida para la generación de la respuesta.
- En este punto, se da paso a formar la respuesta, mediante un esquema xml prefijado, en el que se rellenan los datos con el mensaje de error, debido a que la petición no produce respuesta, o con la respuesta obtenida en la ejecución de la query.

- Para terminar, la respuesta que se ha formado, es enviada, mediante un hilo, por la cola JMS de respuesta, que se genera con los datos de conexión del credentials y con la dirección de envío del Config*Contacts*.

Igual que en el caso de los servicios específicos, este procesamiento se produce en un hilo, por lo que, si está determinado que hay más de un proceso escuchando por servicio, se podría dar respuesta a más de una petición, por servicio, al mismo tiempo.

## *5.3 PRUEBAS UNITARIAS*

Para comprobar el correcto funcionamiento del Handler se han llevado a cabo diferentes pruebas para garantizar su correcto funcionamiento una vez implantado. Estas pruebas consisten en el uso de JUnit para comprobar que cada uno de los métodos, de todas las clases que componen la plataforma, funciona de manera correcta.

Para esto se ejecutan todos y cada uno de los métodos ingresando parámetros que produzcan salidas que se comparan con las salidas esperadas, así mismo, también se introducen parámetros que produzcan errores y excepciones para comprobar el funcionamiento del tratamiento de errores de la aplicación.

Una vez realizados estos códigos de pruebas JUnit dirá si los métodos han pasado la prueba o no, además, mediante un plugin de Eclipse denominado Eclemma se puede observar la cantidad de código, del programa, que ha sido comprobado.

Esta cantidad de código comprobado de mide en porcentaje, sobre el total de código escrito, y se denomina cobertura del código, esta devolución de la cobertura la da Eclemma mediante la librería de cobertura de código denominada JaCoCo o Java Code Coverage.

Como objetivo de comprobación del código del Handler, se impuso una cobertura mínima del 90% sobre el código total, esta cobertura se puede observar fácilmente desde el

dashboard que muestra las estadísticas y gráficos del proyecto que proporciona la aplicación SonarQube.

Como se puede observar, en la imagen que se muestra a continuación, la cobertura obtenida, en el momento de la captura, es del 90%, lo cual supone que se cumple el objetivo propuesto para poder pasar a implantar la plataforma en el funcionamiento normal de la empresa.

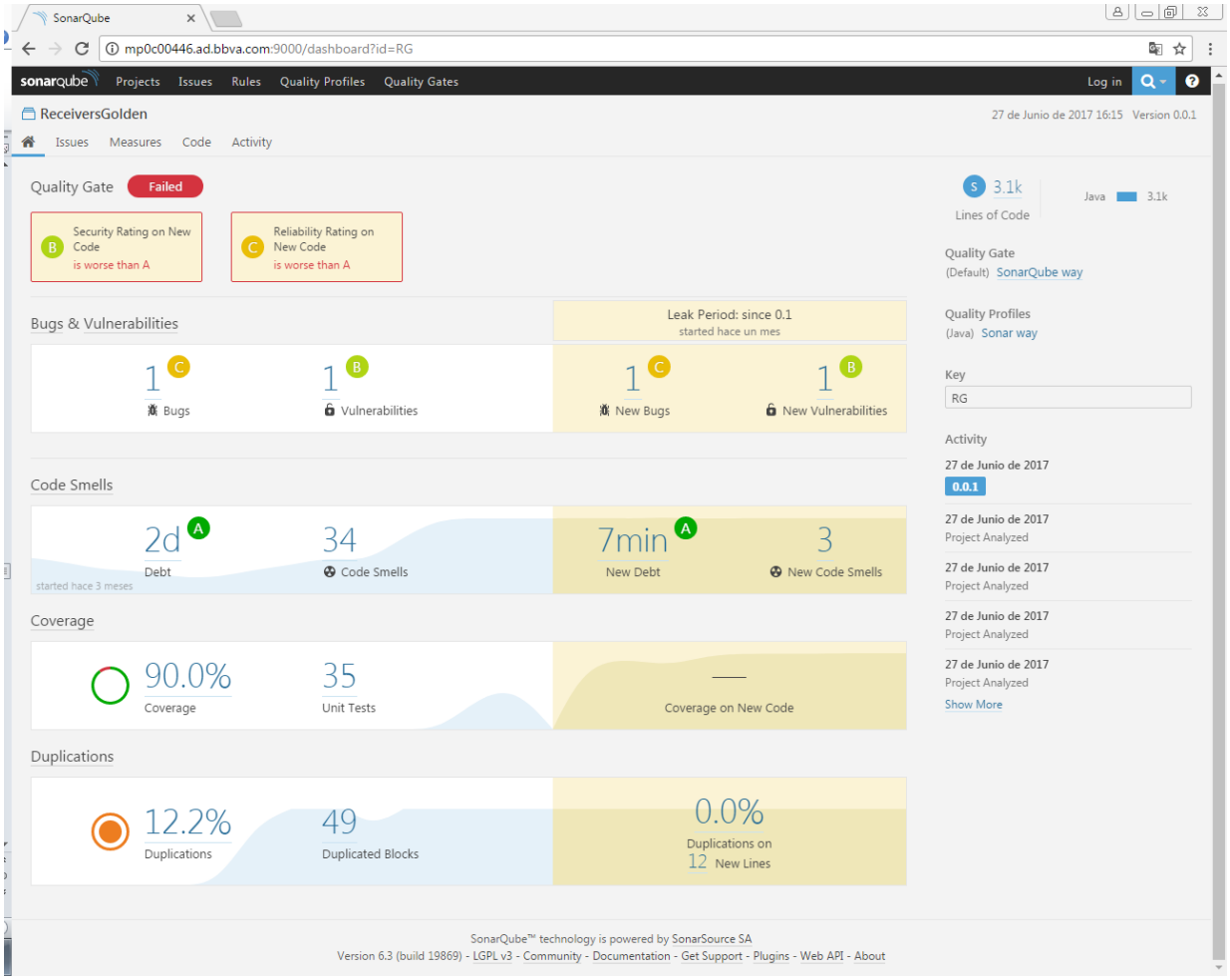



*Figura 14. Dashboard SonarQube*

Unos requerimientos tan altos para la implementación de esta plataforma se deben a que los requisitos para la implantación de un sistema tan crítico, como es el tratamiento de la información en una empresa como BBVA, son severos y es necesario comprobar que todo funciona según lo previsto antes de proceder a su implementación.

Para poder comprender, de manera más detallada el funcionamiento de estas pruebas unitarias, a continuación se muestra la imagen de la ejecución de una clase de pruebas que está comprobando el correcto funcionamiento de la clase principal del proyecto "*ServicesRDR*".

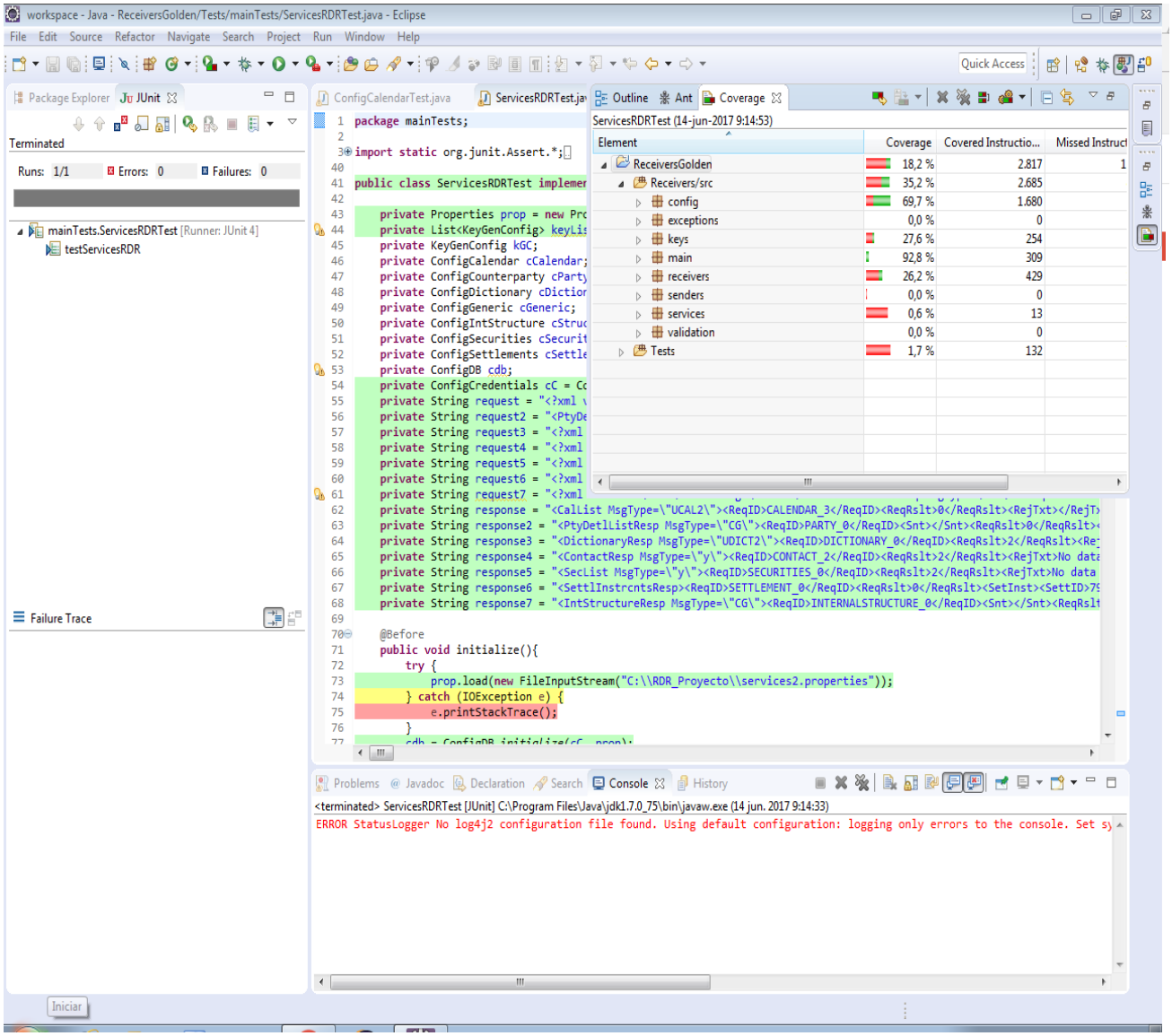



*Figura 15. Ejemplo pruebas unitarias JUnit*

Como se puede observar, en la caja de la izquierda, el único método que se ha comprobado ha funcionado sin problemas, es decir, la prueba ha sido exitosa. Así mismo, en la caja de la derecha, se muestra el porcentaje que cubre esta prueba sobre cada una de las clases del proyecto y, como se puede observar, donde cubre de mayor manera es en el package main, obviamente, ya que es el paquete que contiene la clase principal "*ServicesRDR*" llegando a cubrir un 92,8% de dicha clase.

Aunque esta prueba es positiva, también se puede dar el caso de realizar pruebas negativas, es decir, ejecutar un método ingresando parámetros que se conoce que dan errores y comprobar que dicho erros se ha producido, esto sirve para poder controlar los errores en la manera de lo posible, durante el normal funcionamiento de la plataforma.

Estas pruebas son muy importantes a la hora de comprobar cómo reaccionaría la aplicación ante cualquier situación de problemática a la que se pueda enfrentar. Esta respuesta, en una aplicación que tiene que estar levantada a todas horas y todos los días, es necesaria, ya que ninguna petición mal formada, ni ninguna excepción o fallo deben de frenar el funcionamiento de la aplicación.

# Capítulo 6. ANÁLISIS DE RESULTADOS

En el presente capítulo se tratará de analizar en profundidad los resultados obtenidos a la hora de ejecutar ambas aplicaciones, durante meses y comprobando su funcionamiento y utilidad.

Por lo tanto este apartado se dividirá en dos subapartados, por un lado los resultados de la herramienta de pruebas regresivas y, por el otro, los resultados de la plataforma de gestión de peticiones y respuestas.

## *6.1 HERRAMIENTA DE PRUEBAS REGRESIVAS*

La aplicación de pruebas regresivas se ha posicionado para cubrir una necesidad que ni siquiera se sabía que existía en la empresa.

Este sistema permite la realización de pruebas de manera automática y de manera múltiple, para la comprobación del funcionamiento de la plataforma de procesamiento de peticiones, tanto la anterior como la nueva, lo que permite aumentar la productividad de los empleados de la entidad, así como aumentar la eficiencia de su trabajo.

Así mismo, y dado que la aplicación se ha presentado plenamente operativa y sin errores ha permitido comenzar a usarla en todos los ámbitos que sean necesarios.

Además la facilidad de su uso, y lo visual de la aplicación, ha permitido que pueda aprenderse su funcionamiento de manera muy rápida, mediante una formación de apenas una hora, y su interfaz intuitiva permite que pueda aprender a usarse mediante la marcha.

También cabe destacar la realización del manual de usuario de la aplicación que permite el acceso a obtener respuestas sobre cualquier duda que pueda aparecer sobre el funcionamiento de la aplicación.

El funcionamiento de la herramienta se ha producido como todo un éxito en el ámbito de la empresa. Este éxito se debe al resultado de agilizar el proceso de comprobación del funcionamiento de la plataforma de gestión de peticiones, hasta ahora implantada (Golden Source), y ha permitido comprobar el funcionamiento de la nueva e incluso ayudar a la corrección de errores en el tratamiento de las peticiones.

Esto se debe a que ha automatizado el lanzamiento de peticiones, la cantidad de veces que se envían, cuantas se envían e incluso a cuantos servicios se envían; devolviendo resultados tan provechosos como son el tiempo que ha tardado en hacer frente, la plataforma de gestión, a todas las peticiones y ha permitido comprobar si son correctas.

Por un lado ha permitido ver que la actual plataforma, Golden Source, no respondía de manera óptima a las peticiones, ya que en algunos casos tardaba hasta más de 10 segundos en responder, siendo un hecho totalmente inaceptable para una empresa del tamaño de BBVA.

Esto ha permitido que las personas encargadas del departamento y de los departamentos de gestión de la información, dentro de la empresa, se dieran cuenta del serio problema, de la ineficiencia de la plataforma, y que promovieran el desarrollo de una nueva.

Por otro lado, se ha comprobado la eficacia de la herramienta a la hora de comprobación de errores en las respuestas recibidas, ya que se han podido detectar errores, en el tratamiento de las peticiones, en la nueva plataforma que se iba a implantar para sustituir a la antigua.

Este hecho ha dado lugar a la optimización de la nueva plataforma hasta el punto de no tardar más de unos pocos milisegundos a la hora de responder a 20 peticiones al mismo tiempo.

Además de los hechos anteriormente mostrados, el personal que trabaja en el banco ha alabado el funcionamiento de la herramienta, destacando su facilidad de uso y la simplificación de las tareas, que antes debían hacerse de manera individual y de una forma mucho más lenta.

También es importante el avance que la aplicación ha supuesto a la hora de generar entornos de simulación de manera local y en pruebas, que permita comprobar el funcionamiento de diferentes servicios pudiendo hacer pruebas sin tener que realizar todas las trabas administrativas que presenta el banco para hacerlas de manera general.

## *6.2 PLATAFORMA DE GESTIÓN DE PETICIONES Y RESPUESTAS*

La nueva plataforma de gestión de peticiones y respuestas se ha presentado como una novedad muy bien recibida dentro del área Corporate Investment Banking, .

Esto se ha debido gracias al alto rendimiento que está ofreciendo, frente a la plataforma anterior, y la eficiencia con la que está respondiendo a las peticiones que le llegan.

Esta eficiencia se está viendo plasmada en los tiempos de respuesta que está teniendo, los cuales son de milisegundos, es decir, mejora en segundos a la plataforma anterior, lo cual la posiciona como una apuesta de futuro para un mejor funcionamiento de las bases de datos del banco.

A continuación se muestra una captura de los tiempos conseguidos, en primer lugar con la plataforma antigua, y en segundo lugar con la Fase I implementada.

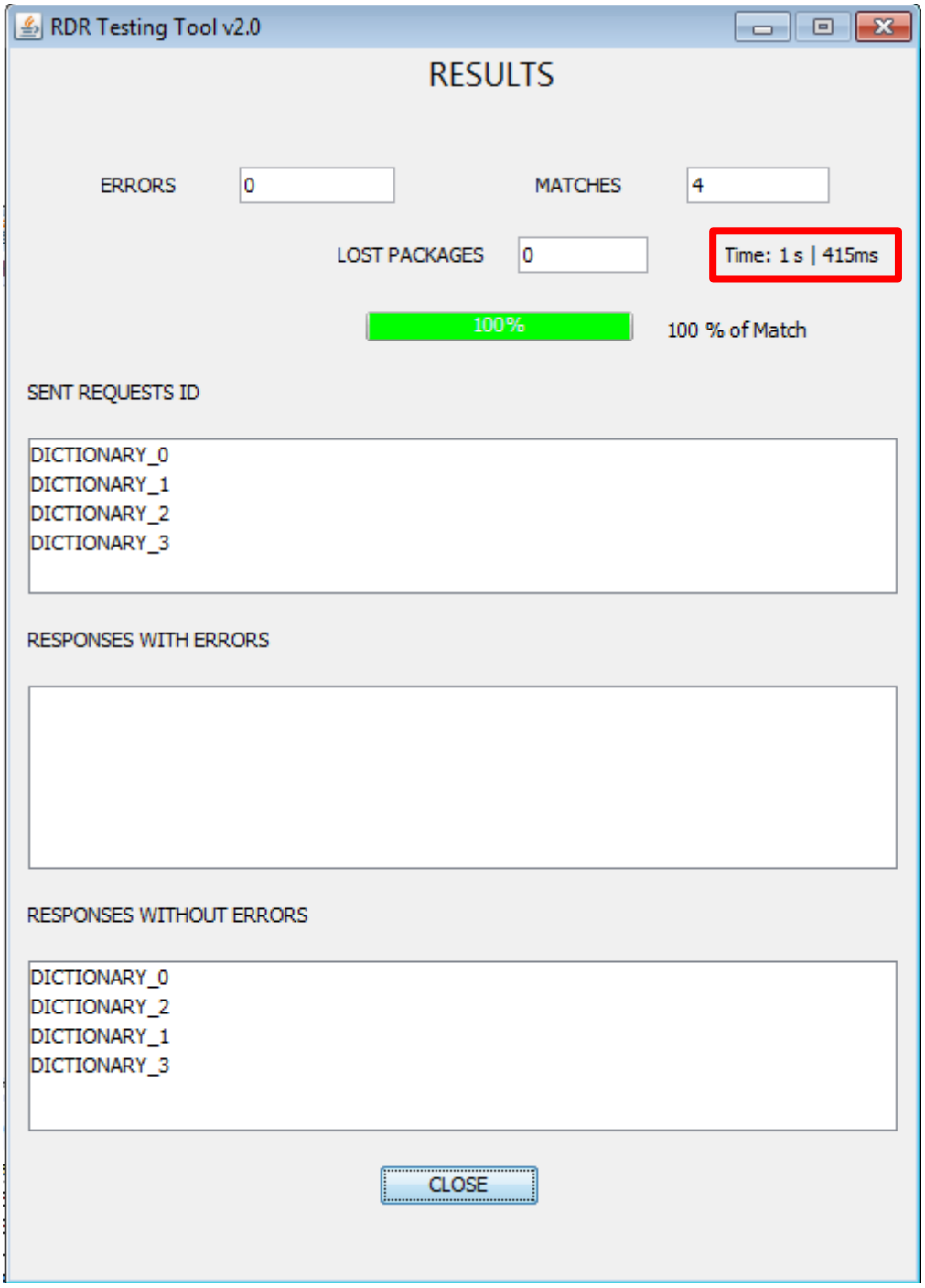



*Figura 16. Ejemplo resultado tiempos en GoldenSource con Testing Tool*

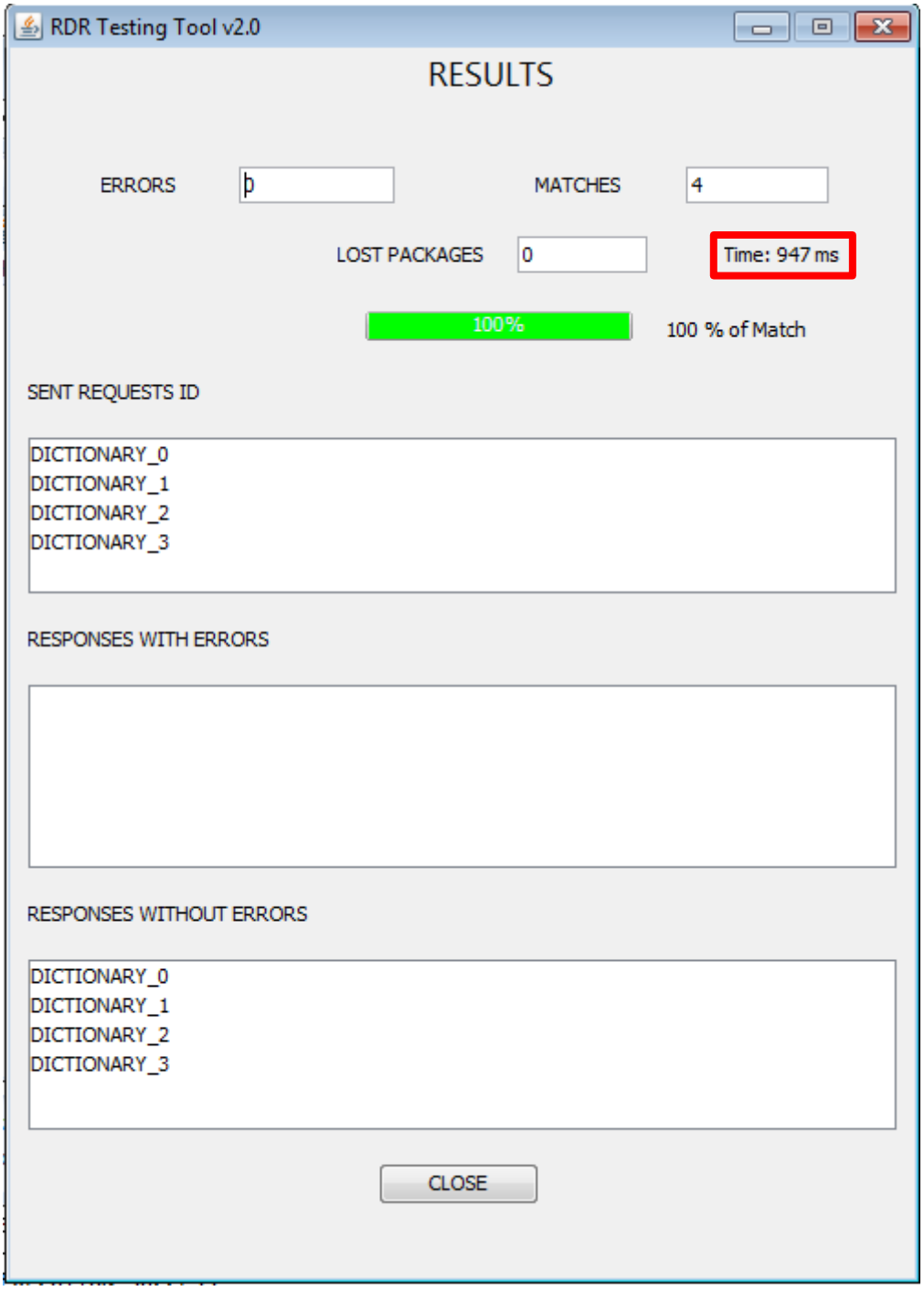



*Figura 17. Ejemplo resultado tiempos en Handler Application con Testing Tool*

Como se puede observar, el tiempo de respuesta se reduce en 468 milisegundos, lo cual es una cantidad muy apreciable, si se realizan muchas peticiones al mismo tiempo.

Así mismo, los trabajadores del departamento, que se van a encargar de emplearla, les ha parecido una gran idea la flexibilidad que permite la aplicación, haciendo posible la creación de nuevo servicios con la simple modificación de un archivo de propiedades de la aplicación. Además destacan la facilidad de su uso y la eficiencia de su funcionamiento al realizar demostraciones durante las reuniones que se hicieron para mostrar su potencial.

# Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

El proyecto desarrollado ha cumplido con creces cada uno de los objetivos que se tenían previsto realizar, con éxito. Así, en esta Fase I del proyecto, se ha conseguido que los tiempos de respuesta de la base de datos hayan disminuido del orden de entre 300ms y varios segundos la respuesta de las peticiones de los servicios de BBVA.

Esto hizo posible que uno de los retos más importantes fuese solventado, la falta de eficiencia en la velocidad de respuesta que estaba llevando a cabo la plataforma de gestión, Golden Source, de las peticiones y respuestas.

Además se ha permitido una flexibilidad muy alta, a la hora de crear o modificar nuevos servicios, ya que se ha automatizado esta tarea mediante la edición de un archivo properties, ajustando unos pocos parámetros.

Finalmente se ha permitido una facilidad de comprobación del funcionamiento de la nueva plataforma, y de la antigua, mediante la herramienta de administración y gestión, a través de la cual, se comparan las respuestas, recibidas, con las que deberían de ser recibidas, además de obtener los tiempos de respuesta.

La consecución, de manera satisfactoria, de todos estos objetivos, además de la impresión causada en todos los departamentos del entorno por la facilidad de uso y lo completo del funcionamiento de todas las herramientas que componen el proyecto, ha permitido que vaya a ser implantado en la mayor brevedad posible en el funcionamiento de la entidad empresarial.

Si a todos estos beneficios se les suma el ahorro monetario, y de retención de conocimiento, que supone la realización de un desarrollo “in-house” frente al encargo del mismo a una empresa externa, se presenta la posibilidad de evolucionar el proyecto hacia una segunda fase, la cual se denominará Fase II.

Para el desarrollo de la Fase II se volverá a añadir la caché de coherence al sistema, con el cambio del lenguaje de la información almacenada en la caché de FIXML a XML. En el siguiente diagrama se muestra más en detalle el nuevo sistema.

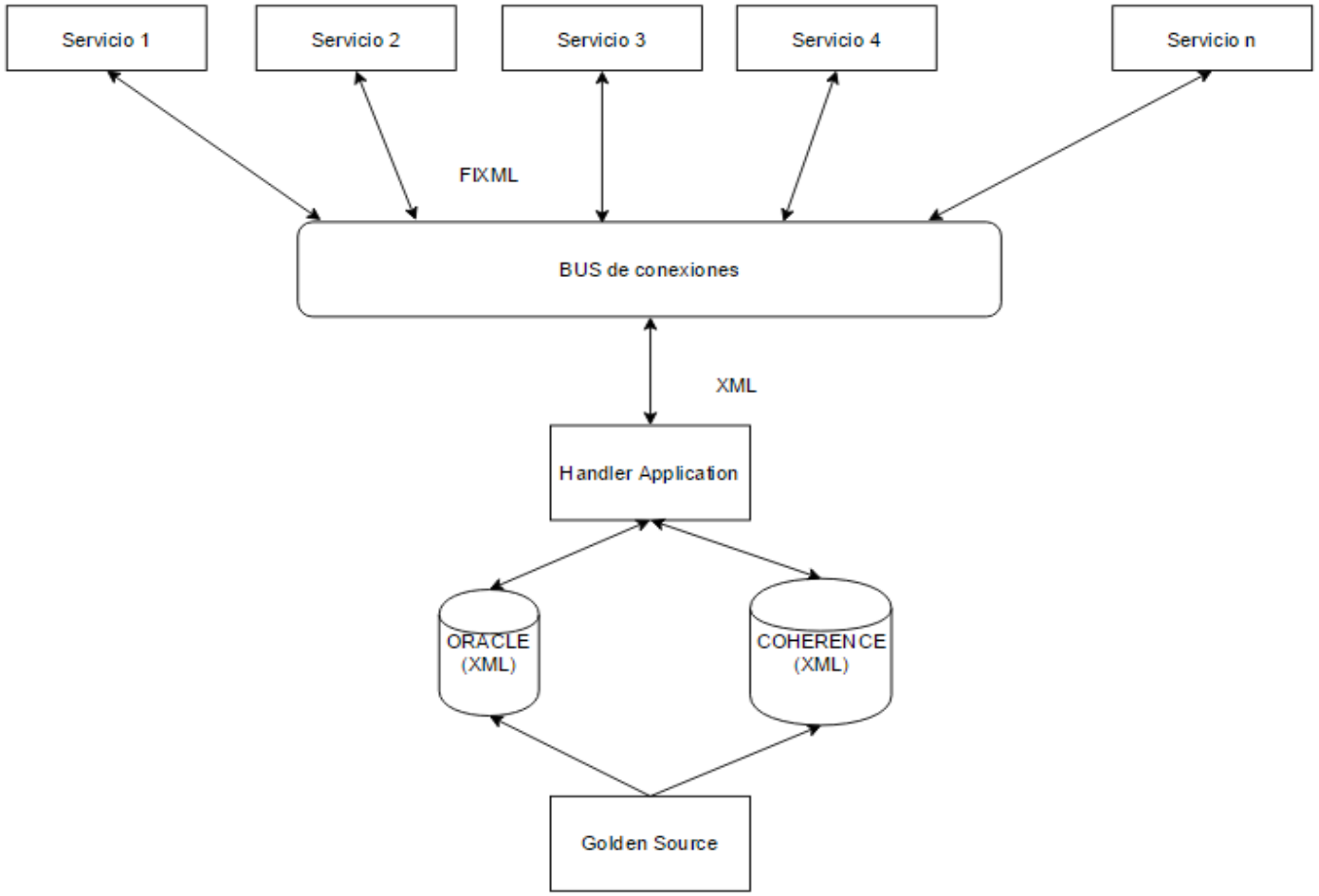



*Figura 18. Evolución en trabajos futuros, Fase II*

Como se puede apreciar, este modelo es muy parecido al de la Fase I con el único cambio de la agregación de la caché de Oracle Coherence, lo que proporcionará una respuesta más rápidas a las peticiones, que ya están almacenadas en la misma, utilizando un modelo de almacenamiento, en la caché, FIFO (First In First Out).

Este nuevo modelo de funcionamiento traerá consigo diversas ventajas que ayudarán a mejorar, aún más, el funcionamiento de la plataforma:

- Mejorará el rendimiento del sistema, al tener la caché y la base de datos el mismo lenguaje de comunicación, por lo que solo será necesario traducir la información una vez al entrar y otra al salir.
- Elimina el cuello de botella que se formaba en ORACLE, al poder obtener la información de la caché, si esta se encuentra guardada en ella con anterioridad, liberando así el atasco que se produce en la base de datos.

Este es un nuevo modelo que podría llegar a implantarse en un futuro en el cual la Fase I haya sido probada y su funcionamiento sea el esperado.

# Capítulo 8. BIBLIOGRAFÍA

[1] Página Oficial de Oracle: https://www.oracle.com

[2] Página Oficial SonarQube: https://www.sonarqube.org

[3] Página Oficial Oracle Coherence: http://www.oracle.com/technetwork/middleware/coherence/overview/index.html

[4] Página Oficial Golden Source: https://www.thegoldensource.com/

[5] Página Oficial Oracle Java: https://www.java.com/es/

[6] Página Oficial de Filezilla: https://filezilla-project.org/

[7] Página Oficial de Putty: http://www.putty.org/

[8] Página Oficial de Eclipse: https://eclipse.org/

[9] Página Oficial de NetBeans: https://netbeans.org/

[10] Página Oficial de SQLDeveloper: http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html

[11] Página Oficial de Trello: https://trello.com/

[12] "Software Engineering 9", Ian Sommerville, Pearson, 2010: https://www.pearson.com/us/higher-education/product/Sommerville-Software-Engineering-9th-Edition/9780137035151.html

[13] Tutorial Oficial de Java de Oracle: http://www.oracle.com/technetwork/java/index.html

[14] Tutorial de XML en w3schools: https://www.w3schools.com/xml/

# Capítulo 9. ANEXO A: MANUAL DE USUARIO

En el presente Anexo se tratará de realizar una explicación del funcionamiento de la aplicación de pruebas regresivas, tanto de la versión de interfaz gráfico como la versión por línea de comandos.

## *9.1 INTRODUCCIÓN*

RDR Testing Tool es el sistema mediante el cual los responsables del control de la integración de Oracle Coherence, podrán comprobar si la información devuelta por la caché es coherente y si devuelve la respuesta que debería de dar.

Este documento es una guía para el uso de todas las funciones de la RDR Testing Tool, enfocado desde el punto de vista del usuario.

## *9.2 CONOCIMIENTOS NECESARIOS*

Para llevar a cabo la ejecución de la aplicación de manera correcta, será necesario comprender primero la diferente terminología empleada.

### 9.2.1 SERVICIO

Referido a cada una de las posibles prestaciones de las cuales se podrá obtener información. En la actualidad se dispone de 12 servicios (Podrían ser aumentados en un futuro), los cuales se enumeran a continuación: Agreement, Calendar, Confirmations, Contact, Country, Dealmandates, Deatypes, Dictionary, Index, Party, Securities, Settlement.

## 9.2.2 "QUEUES" O COLAS

Se determina cola o "queue", por su terminología en inglés, a cada una de las conexiones a las que se puede recurrir, es decir, a cada uno de los canales a los que se puede realizar la petición. En este momento se dispone de 4 colas: Development o desarrollo (DE), Integrado (INT), Porducción (PR) y Preproducción (PP).

## 9.2.3 CREDENCIALES

Se empleará este término para referirse a los datos necesarios para realizar la conexión a cada una de las colas comentadas anteriormente. Las credenciales de conexión necesarias serán tres: servidor, usuario y contraseña. Estas credenciales estarán incluidas en el archivo "credentials.xml".

### *9.2.3.1 SERVIDOR*

Referido al servidor al cual se conecta la aplicación.

### *9.2.3.2 USUARIO*

Referido al usuario que se quiere conectar a la aplicación.

### *9.2.3.3 CONTRASEÑA*

Credencial que autentifica al usuario.

## 9.2.4 PARÁMETROS DEL MENSAJE

Se refieren a la id de correlación del mensaje y a la id del mensaje, solo para la versión gráfica (dejar por defecto).

## 9.2.5 NÚMERO DE "REQUEST" O PETICIÓN

Se refiere al número de petición que se quiere enviar, entre todas las posibles peticiones que se contienen en los respectivos archivos .csv, que se explicarán más adelante.

## 9.2.6 ARCHIVOS "CREDENTIALS.XML"

Referidos a cada uno de los archivos que contienen la información necesaria para realizar la conexión a cada una de las colas y otros parámetros de interés. Dentro de estos archivos xml destacan varios apartados a tener en cuenta.

### *9.2.6.1 ENVIRONMENT*

En esta sección se encontrará información correspondiente a parámetros de configuración de la aplicación. De estos parámetros serán interesantes dos de ellos.

#### 9.2.6.1.1 inputDataFolder

Referido a la dirección en la cual se guardarán los archivos ".csv" que contendrán las peticiones y las respuestas.

### *9.2.6.2 QUEUEFACTORY*

En esta sección se encontrará toda la información de las credenciales necesarias para realizar la conexión a las colas.

#### 9.2.6.2.1 Host

Referido al servidor al cual tiene que realizar la conexión la aplicación (Dejar el que está por defecto).

#### 9.2.6.2.2 User

Referido al usuario que trata de realizar la conexión (Dejar el que está por defecto).

#### 9.2.6.2.3 Password

Referido a la contraseña que autentifica al usuario (Dejar la que está por defecto).

### 9.2.7 ARCHIVOS "SERVICIO.CSV"

En la carpeta de la aplicación se encontrarán alojadas diferentes carpetas con los nombres de cada uno de los servicios disponibles. Dentro de estos servicios se encontrará un archivo con el nombre de dicho servicio.csv. Estos son archivos que contendrán las peticiones y respuestas almacenadas en el estándar CSV, esto es "petición","respuesta".

#### *9.2.7.1 PETICIONES O "REQUESTS"*

Se tratan de mensajes en lenguaje xml que contienen la información necesaria para que la base de datos la procese y genere una respuesta.

#### *9.2.7.2 RESPUESTAS O "RESPONSES"*

Se refiere a las respuestas devueltas por la base de datos, que responde a la petición que le llegó en concreto.

## *9.3 LOG*

Los ficheros log son aquellos en los cuales se almacenará toda la información proveniente de la aplicación, tanto la información que detalla el funcionamiento de la aplicación, como aquella que se destina a mostrar el resultado de las operaciones realizadas.

### 9.3.1 LOG DE APLICACIÓN

Se refiere a la información del funcionamiento de la aplicación. Esto es, detallará tanto el correcto funcionamiento de la aplicación ([INFO]), como los fallos ([WARN]) o los errores que provocan la finalización del programa ([ERROR]). Este archivo se producirá automáticamente en el trascurso de funcionamiento de la aplicación en un archivo llamado "log.log", en la dirección escrita en el archivo "credentials.xml/<environment>/<logPath>"

### 9.3.2 LOG DE RESULTADOS

Se refiere a toda la información detallada de los resultados de la aplicación. Este log da dos resultados, los resultados generales y los resultados de los errores cometidos.

#### *9.3.2.1 LOG GENERAL*

En este archivo se encontrará toda la información de los resultados que ha procesado el programa. En este archivo se encontrarán separados los resultados de cada servicio, por el nombre de servicio y fecha en la que se realizó la prueba, y los resultados para cada uno de las pruebas realizadas. Los resultados de estas pruebas pueden ser tres: OK, NOK, LOST. Este archivo se llamará "logGeneral.txt" y se encontrará en la dirección escrita en el archivo "credentials.xml/<environment>/<logpath>/results/logGeneral.txt"

##### 9.3.2.1.1 OK

Se mostrará al lado de una prueba cuando se ha realizado satisfactoriamente.

##### 9.3.2.1.2 NOK

Se mostrará al lado de una prueba cuando el resultado es erróneo.

##### 9.3.2.1.3 LOST

Se mostrará al lado de una prueba cuando no ha llegado ninguna respuesta, por parte de la base de datos, de la petición enviada.

#### *9.3.2.2 LOG DE ERRORES*

En este archivo se encontrará toda la información de los errores que se han encontrado al comparar los resultados en el programa. Este archivo se llamará "logFails.txt" y se encontrará en la dirección escrita en "credentials.xml/<environment>/<logpath>/results/logFails.txt"

En este archivo se encontrará, separado por fecha y hora, cada una de las veces que se ha ejecutado el programa con los errores obtenidos. Si no se produce ningún error, se mostrará el mensaje "NO ERRORS", por el contrario, si se produce algún error, se mostrará la siguiente información.

#### 9.3.2.2.1 Prueba

En primer lugar, se mostrará el nombre del servicio en el que se ha obtenido el error y se indicará el número de la petición en la que se ha obtenido dicho error.

#### 9.3.2.2.2 RECEIVED

Se mostrará la respuesta obtenida como respuesta a la petición enviada. En el caso de que no se haya recibido ninguna respuesta se mostrará el mensaje "ERROR".

#### 9.3.2.2.3 TESTED

Se mostrará la respuesta almacenada en el archivo .csv del servicio correspondiente.

## *9.4 TIPOS DE APLICACIONES*

La aplicación que se trata, en el presente documento, está disponible en dos formatos perfectamente distinguibles: la versión GUI con interfaz gráfico de usuario y la versión por línea de comandos.

### 9.4.1 LÍNEA DE COMANDOS

Ejecutable a través de línea de comandos con diferentes funcionalidades configurables a través de parámetros que se deben introducir en la ejecución del programa.

### 9.4.2 INTERFAZ GRÁFICO DE USUARIO

Estará compuesta por una ventana Java, archivo .jar, con toda la funcionalidad visual para el usuario.

## *9.5 LÍNEA DE COMANDOS*

Para la ejecución del programa a través de línea de comandos o la terminal del ordenador será necesario tener localizado el ejecutable "TestingToolTerminal.jar". En primer lugar

será necesario acceder a la terminal de la máquina en la que se encuentra. Una vez abierta, será preciso desplazarse a través de las carpetas del ordenador hasta llegar al archivo "TestingToolTerminal.jar" mediante el uso del comando "cd". Una vez llegado a este punto tendrá que escribirse la siguiente línea de código, sustituyendo los parámetros (<parámetro 1> <parámetro 2> … <parámetro n>) por el valor que se le quiere dar a dicha variable (este código está optimizado para el caso en el que se encuentre agregado el bin de java a las variables de entorno del sistema, en caso contrario es necesario escribir toda la ruta hasta el bin de java antes del comando añadiendo "/java" al final de la misma):

*java -cp TestingToolTerminal.jar testingtoolterminal/App <Entorno> <Ruta al archivo credentials.xml> <Ruta al archivo properties.properties> "<sercivio o servicios (separados por comas)>" <Tipo de comprobación> <Número de petición>*

```
xakytl1d@ldrdr602:/fichtemcomp/de/descargas/kytl/RDR_Testing_Tool$ /usr/local/de
/jdk1.6.0_29/bin/java -cp TestingToolTerminal.jar testingtoolterminal/App PRE cr
edentials_pre.xml config.properties "Contact, Confirmations" true 0
```

*Figura 19. Ejemplo lanzamiento Testing Tool por línea de comandos*

### 9.5.1 ENTORNO

En este parámetro debe de introducirse el tipo de entorno al que se quiere enviar la petición, DE, INT o PRE, comentados anteriormente.

### 9.5.2 RUTA AL ARCHIVO "CREDENTIALS.XML"

En este parámetro debe introducirse la ruta completa hasta el archivo "credentials.xml" correspondiente a la cola a la que se quiere conectar.

### 9.5.3 RUTA AL ARCHIVO "PROPERTIES.PROPERTIES"

En este parámetro debe introducirse la ruta completa hasta el archivo "properties.properties" correspondiente las propiedades de los servicios.

### 9.5.4 SERVICIO O SERVICIOS

En este parámetro debe de introducirse cada uno de los servicios, puede ser solo uno, a los que se pretende realizar la comprobación de funcionamiento, comentados anteriormente.

### 9.5.5 TIPO DE COMPROBACIÓN

En este parámetro debe introducirse “true”, si quiere realizarse una validación de todas las peticiones y responses de los servicios introducidos en el parámetro anterior, o “false” si quiere probarse solo una petición del servicio introducido en el parámetro anterior.

### 9.5.6 NÚMERO DE PETICIÓN

En este parámetro debe introducirse el número de petición que se quiere comprobar, siempre y cuando se haya introducido “false” en el parámetro anterior, sino será ignorado.

### 9.5.7 INTERFAZ GRÁFICO DE USUARIO

Para la ejecución del programa, en su versión de interfaz gráfico, solo será necesario pulsar dos veces en el archivo .jar del programa. Una vez abierto el programa, le aparecerá una ventana con dos partes claramente diferenciadas. Por un lado tendrá el menú de usuario, en la parte superior de la ventana, y por el otro, una serie de parámetros de la ventana “Configuración”.

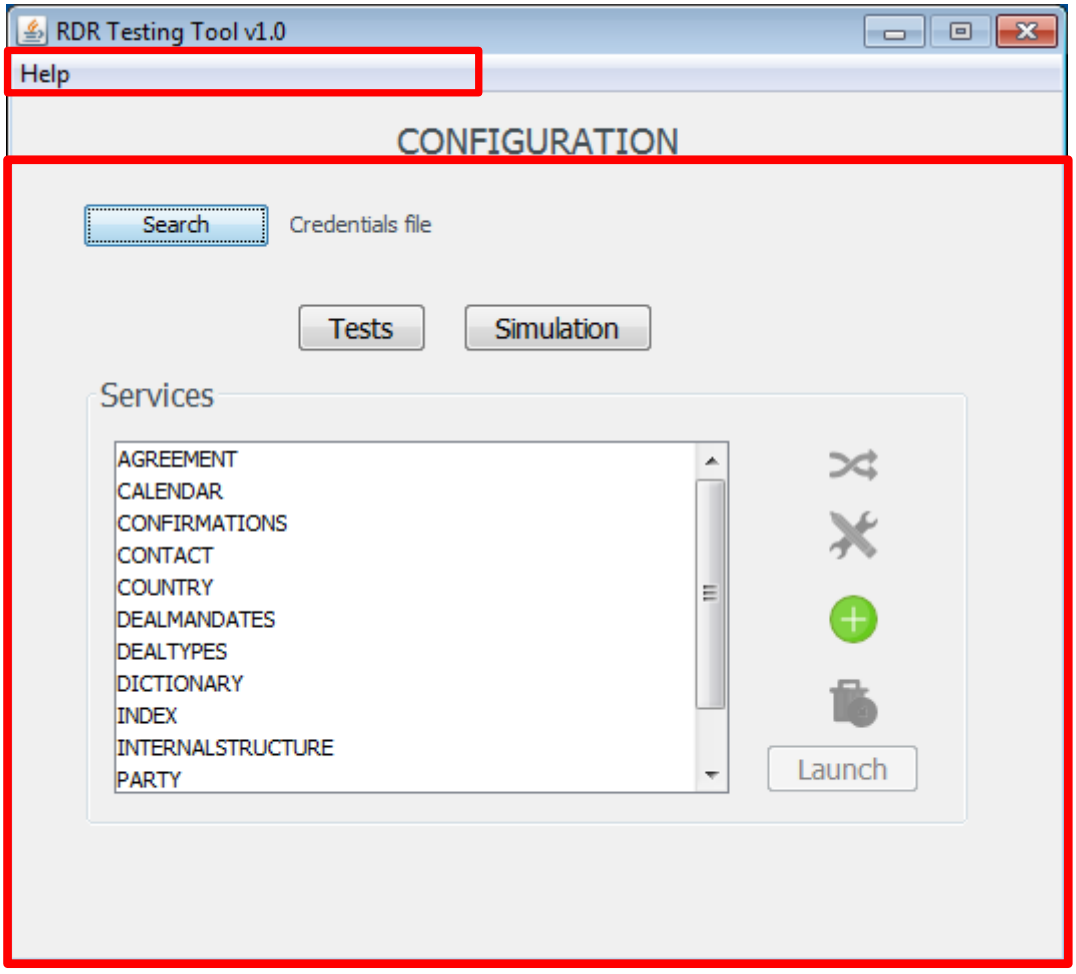



*Figura 20. Pantalla inicio Testing Tool*

## 9.5.8 MENÚ DE USUARIO

Este menú de usuario se encontrará en la barra superior horizontal de la ventana. En ella encontrará el apartado: "Help". Al pulsar sobre el mismo, se abrirá este Manual de Usuario para el programa.

### *9.5.8.1 BOTÓN DE "SEARCH"*

Al pulsar este botón se abrirá una ventana de exploración de archivos del ordenador en el que nos encontramos.

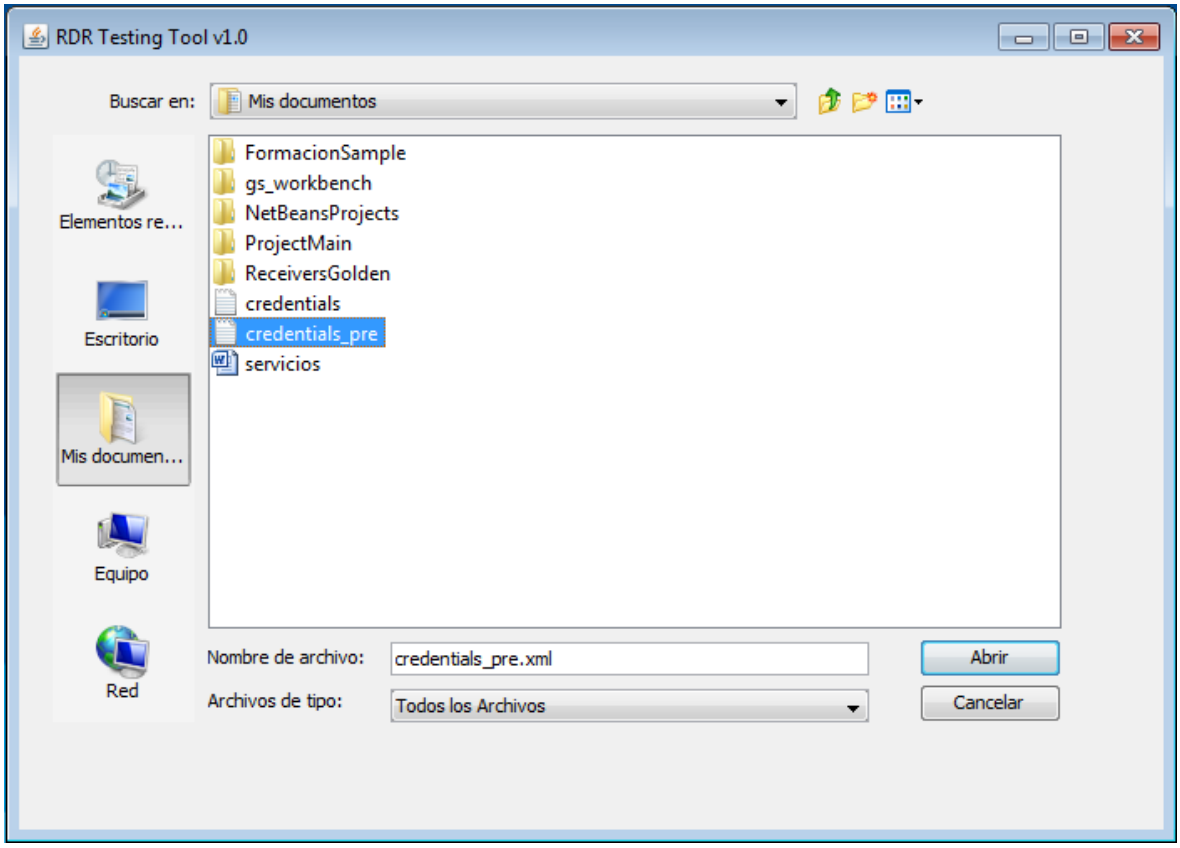



*Figura 21. Selección archivo Credentials en Testing Tool*

La finalidad de esta ventana es buscar el archivo "credentials.xml" que necesitamos para realizar la conexión a la cola específica en la que queremos realizar las pruebas. Una vez encontrado el archivo "credentials.xml", pulsar sobre él y pulsar el botón "abrir". Una vez realizada esta acción, verá como, al lado del botón "Search", aparecerá la dirección absoluta del archivo "credentials.xml" seleccionado. Esta opción sólo será necesario hacerla la primera vez que se abre el programa y cada vez que se quiera probar en otra cola, ya que la dirección del archivo quedará guardada en el programa.

### *9.5.8.2 TEST Y SIMULACIÓN*

Con estos botones se podrá elegir entre un entorno de pruebas o el entorno de simulación, a través del cual se podrá suplir a la plataforma de respuesta a peticiones.

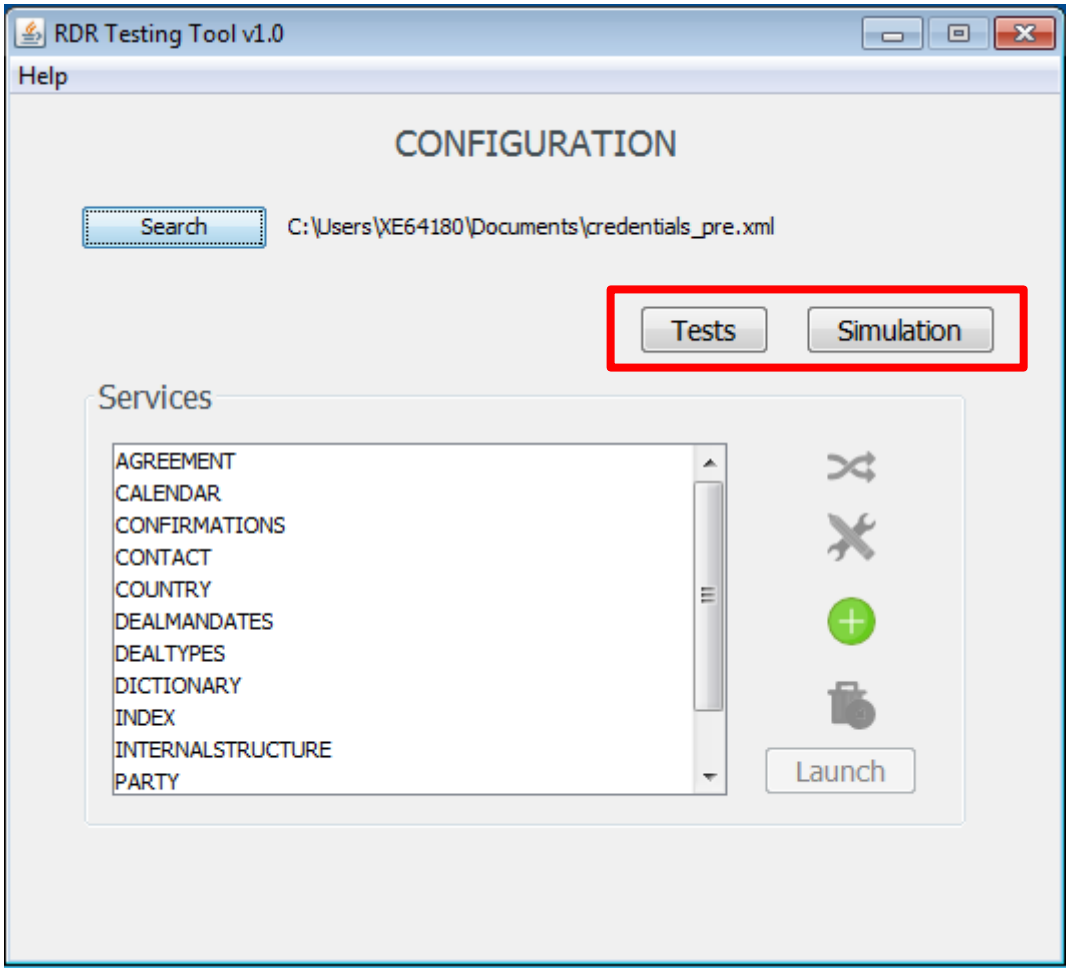



*Figura 22. Modos de funcionamiento en Testing Tool*

### *9.5.8.3 SERVICES*

La sección de "Services" será en la cual se puede elegir el servicio, o servicios, sobre el cual se quiere mandar las peticiones. Por un lado, se encuentra una lista de todos los servicios que hay disponibles, para lanzar las peticiones, y por el otro hay una imagen de "simular" (imagen de dos flechas entrelazadas), una imagen de "agregar" (imagen verde circular con un símbolo "+" blanco en el medio), una imagen de borrar (imagen de una

papelera con un círculo rojo y una "x", en el medio del mismo, en la parte inferior derecha), el cual estará deshabilitado por defecto, y el botón "Launch", deshabilitado por defecto.

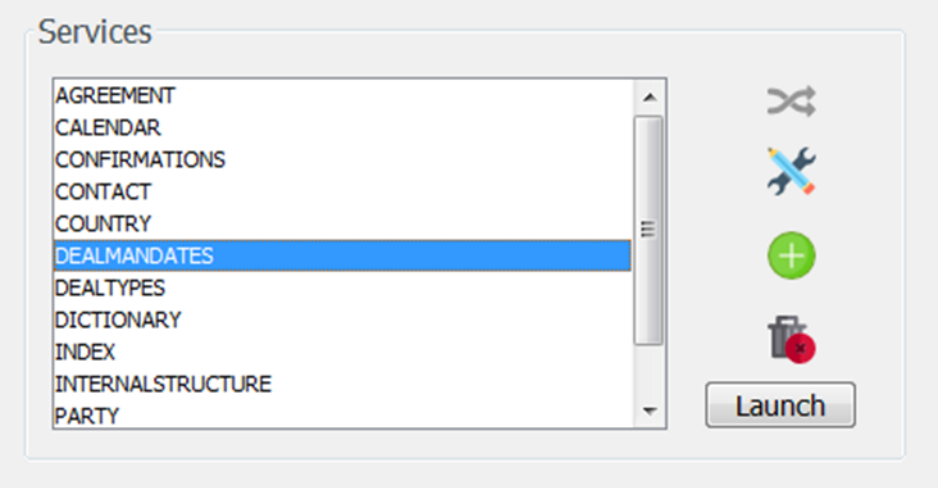



*Figura 23. Lista de servicios en testing Tool*

#### 9.5.8.3.1 Lista de servicios

En esta lista se encontrarán todos los servicios disponibles que podrá seleccionar pulsando sobre uno de ellos. Si se quiere probar en más de un servicio, simplemente deberá seleccionar todos los servicios que se desea probar manteniendo la tecla "Ctrl" o "Control" del teclado del ordenador. Si se pulsa un único servicio se habilitarán los botones de "Launch",el de "Eliminar", el de "Editar" y el de "Simular", si se pulsan varios servicios, se habilitará únicamente el botón "Launch" y el de "Simular". Finalmente, si pulsa un único servicio dos veces, se abrirá una nueva ventana con el título "REQUESTS & RESPONSE", donde se encontrarán todas las requests de dicho servicio y podrá trabajar sobre ellas.

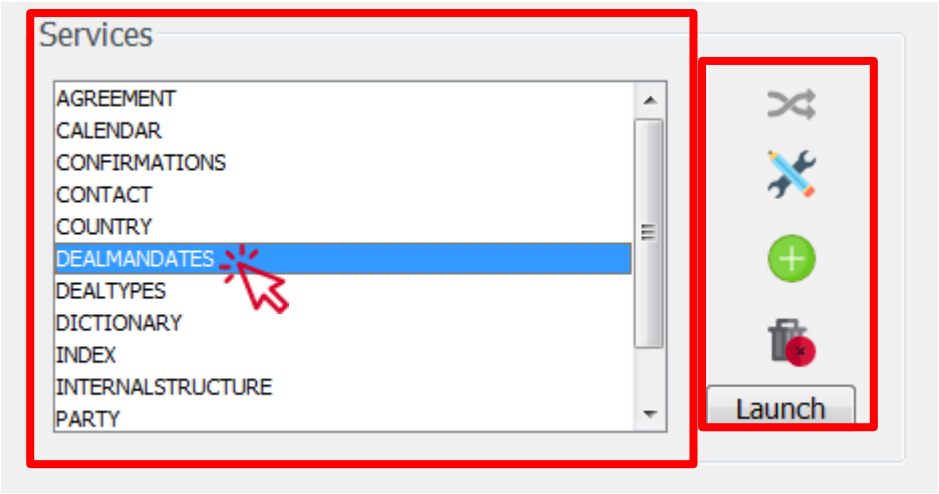



*Figura 24. Herramientas de servicios en Testing Tool*

- REQUESTS & RESPONSES

En esta ventana se encuentran dos secciones diferenciadas, por un lado están las imágenes de herramientas sobre las requests y responses, y por otro lado, dos listas con todas las requests y todas las responses que hay almacenadas sobre dicho servicio en el programa, con los nombres “REQUESTS” y “RESPONSES”, respectivamente.

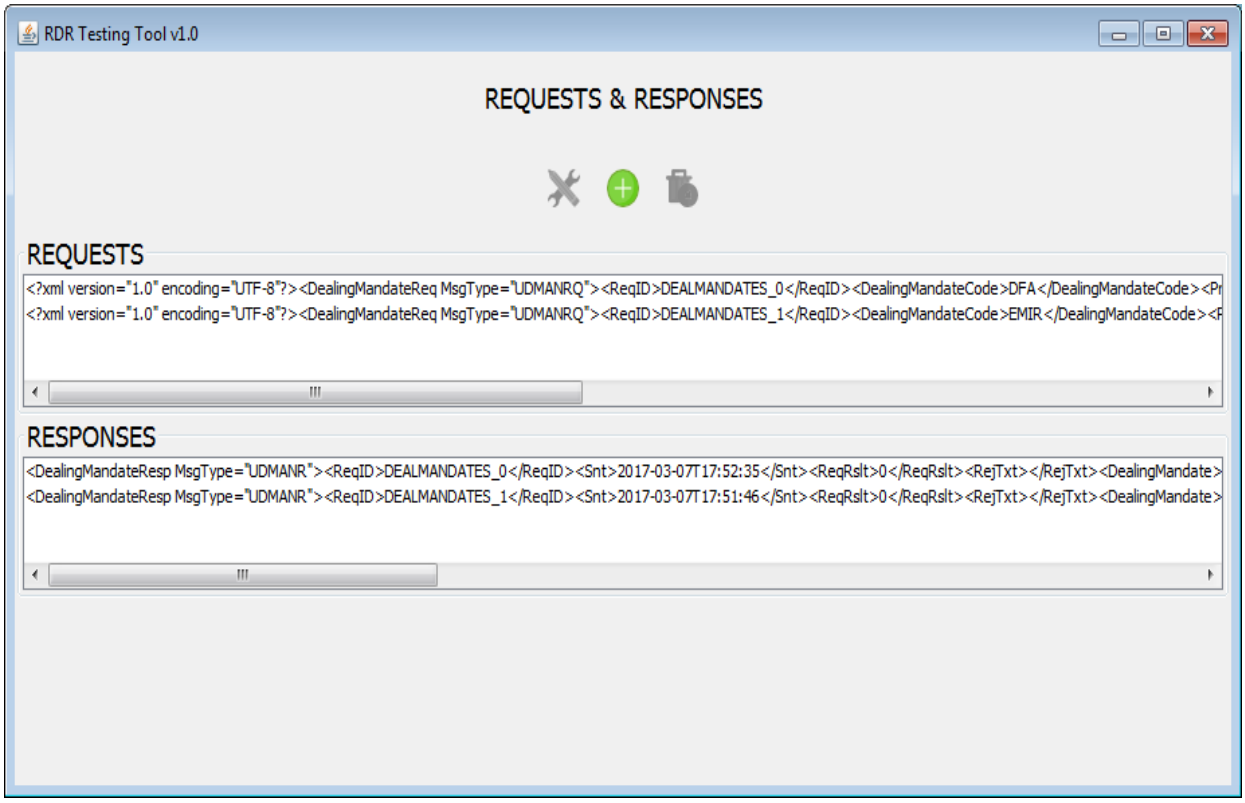



*Figura 25. Peticiones y respuestas en Testing Tool*

- HERRAMIENTAS

En la sección de herramientas, se encuentran tres imágenes: editar (representada por un lápiz y una llave inglesa), deshabilitada por defecto, crear, comentada en la sección 9.6.2.2.1, y eliminar, comentada en la sección 9.6.2.2.1.

- EDITAR

Esta imagen está deshabilitada hasta que no se selecciona una única request o una única response. Una vez seleccionada una request, o response, esta imagen ya será seleccionable. Cuando esta imagen es seleccionada se abrirá otra ventana, donde estará escrita la request, y su response correspondiente, las cuales se podrán editar y guardar mediante la pulsación del botón “OK”. Así mismo, también se dispone de un botón, llamado “FORMAT”, a

través del cual se formateará el texto escrito en el apartado request y response, hasta que tenga la misma estructura que un archivo xml.

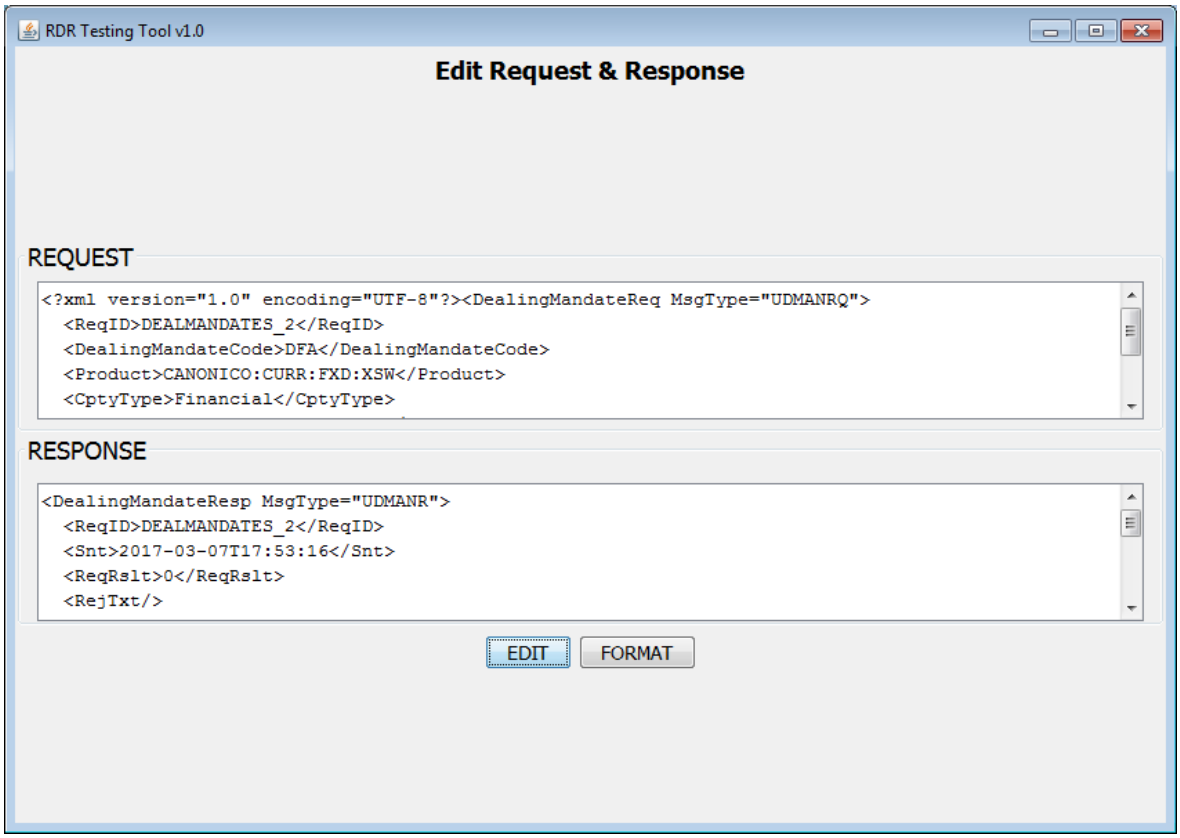



*Figura 26. Editar petición y respuesta en Testing Tool*

- CREAR

Una vez pulsada esta imagen, se abrirá otra ventana con el título "New Request & Response". En esta ventana se podrá escribir, o pegar, la nueva request en el apartado "REQUEST", y la nueva response, en el apartado "RESPONSE", y guardar la nueva pareja petición-respuesta al pulsar sobre el botón "CREATE NEW". Además, existe la opción de escribir únicamente la request, en el apartado request, y pulsar el botón "GET RESPONSE", para obtener la respuesta para dicha request que hay guardada en la base de datos.

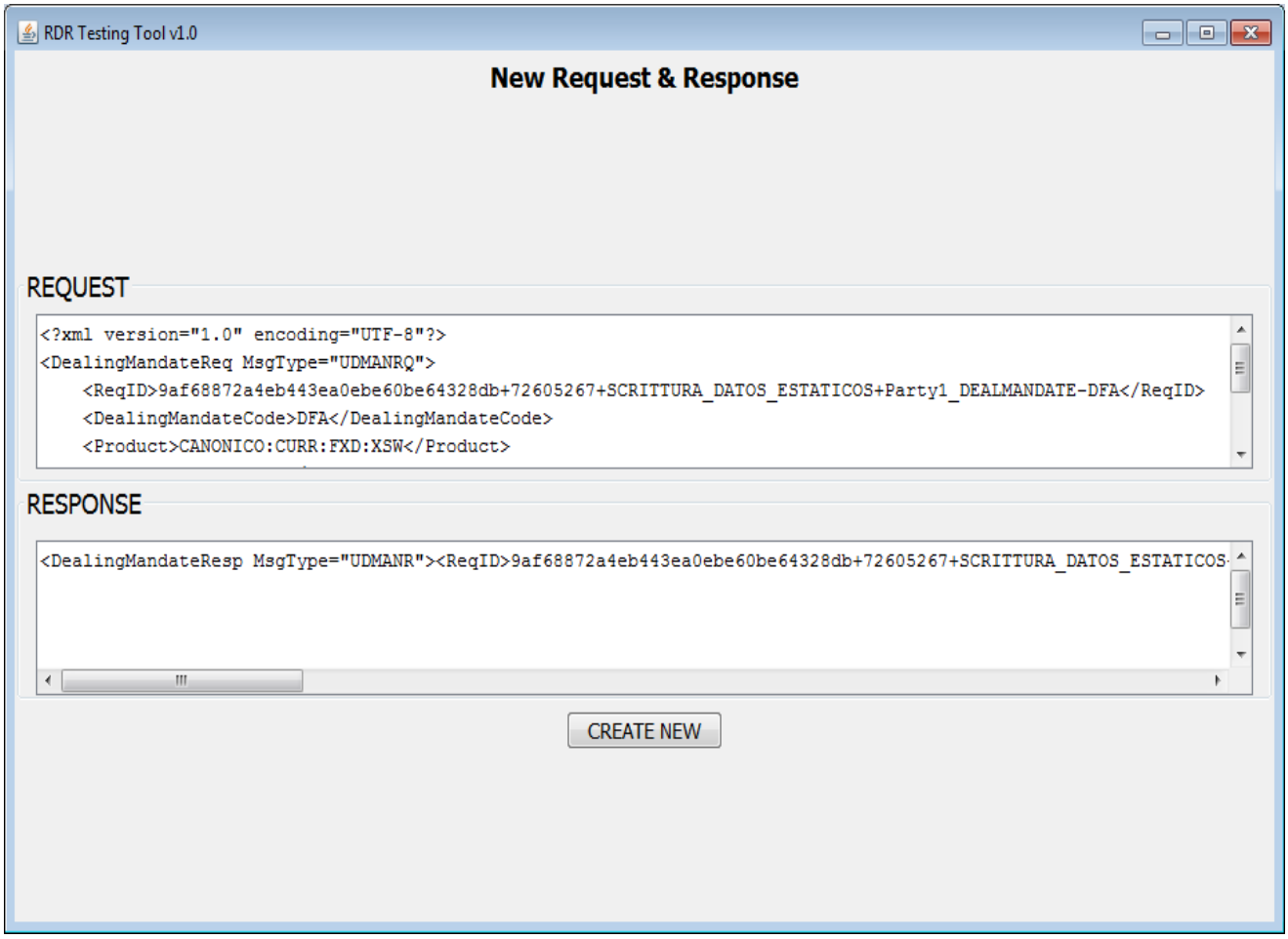



*Figura 27. Nueva petición y respuesta en Testing Tool*

- ELIMINAR

Esta imagen está deshabilitada hasta que no se selecciona una única request o una única response. Una vez seleccionada una request, o response, esta imagen ya será seleccionable. Cuando esta imagen es seleccionada se abrirá otra ventana con el texto "Are yo usure that you want to delete the request and response?", pulsar el botón "DELETE" si se quiere eliminar o "CANCEL", si no se quiere eliminar.

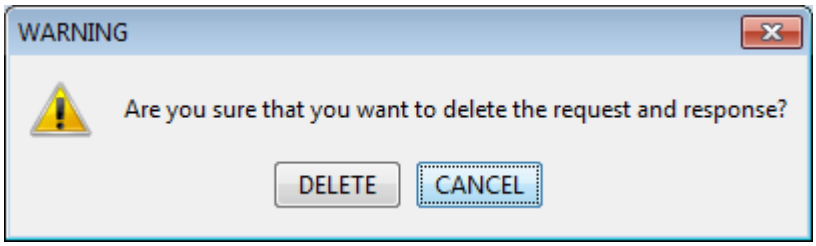



*Figura 28. Eliminar petición y respuesta en Testing Tool*

- REQUESTS Y RESPONSES

En esta sección se encontrarán dos listas con las peticiones y sus respuestas. Solo se podrá trabajar con dichas peticiones y respuestas una a una, por lo que para que se habiliten las herramientas deshabilitadas, se deberá pulsar en una de las requests o responses. Puede apreciarse que una vez que se pulsa en una request, automáticamente se seleccionará la response correspondiente y viceversa.

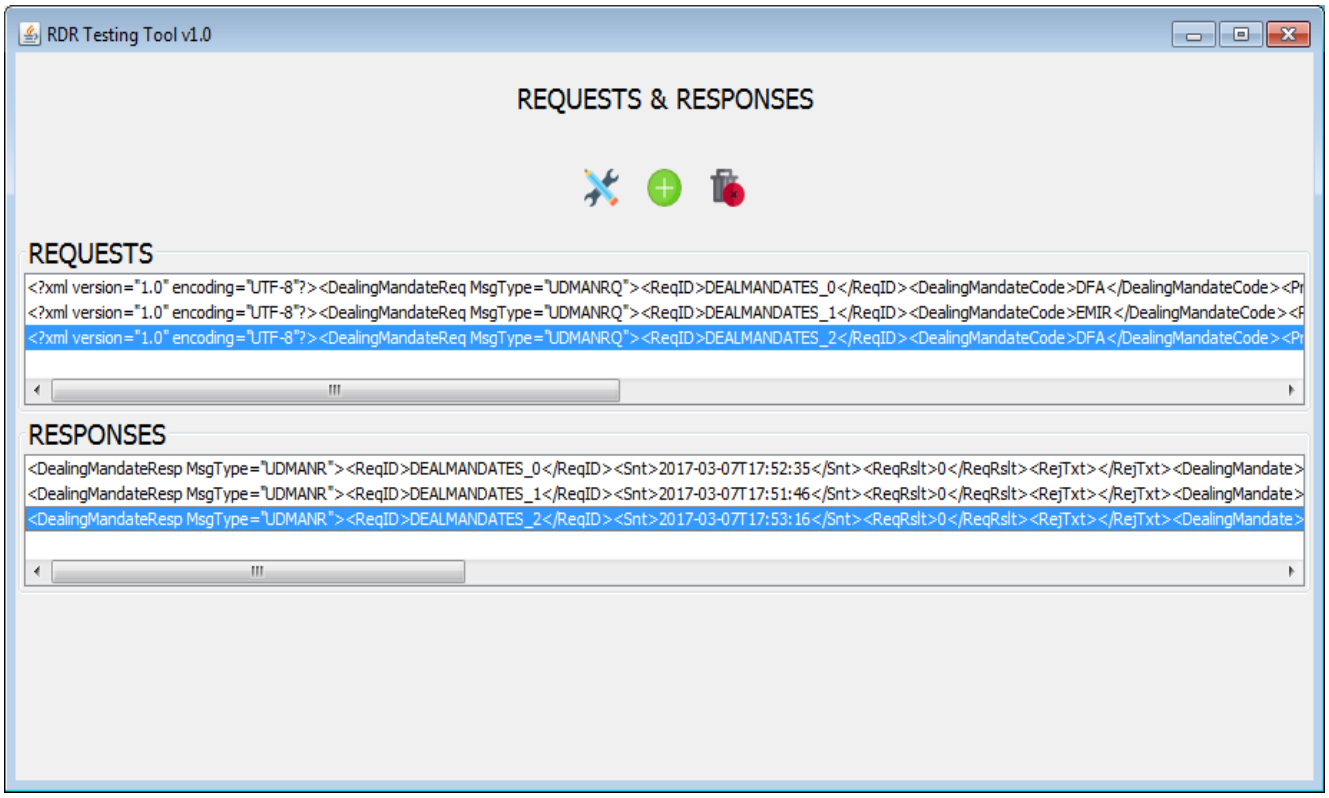



*Figura 29. Lista peticiones y respuestas despues de eliminar en testing Tool*

#### 9.5.8.3.2 Herramientas de servicios

En esta sección, se encuentran las acciones que se pueden realizar sobre los servicios y ya mencionadas en el apartado 9.6.2.2

- Simular

Esta imagen está deshabilitada hasta que no se selecciona una, o varias, request o una, o varias, response, con el modo simulación activado. Al pulsarla comienza a escuchar peticiones, en el entorno seleccionado en el credentials, y las responderá, si tiene dichas requests almacenadas, con las responses que tiene guardadas para dichas peticiones.

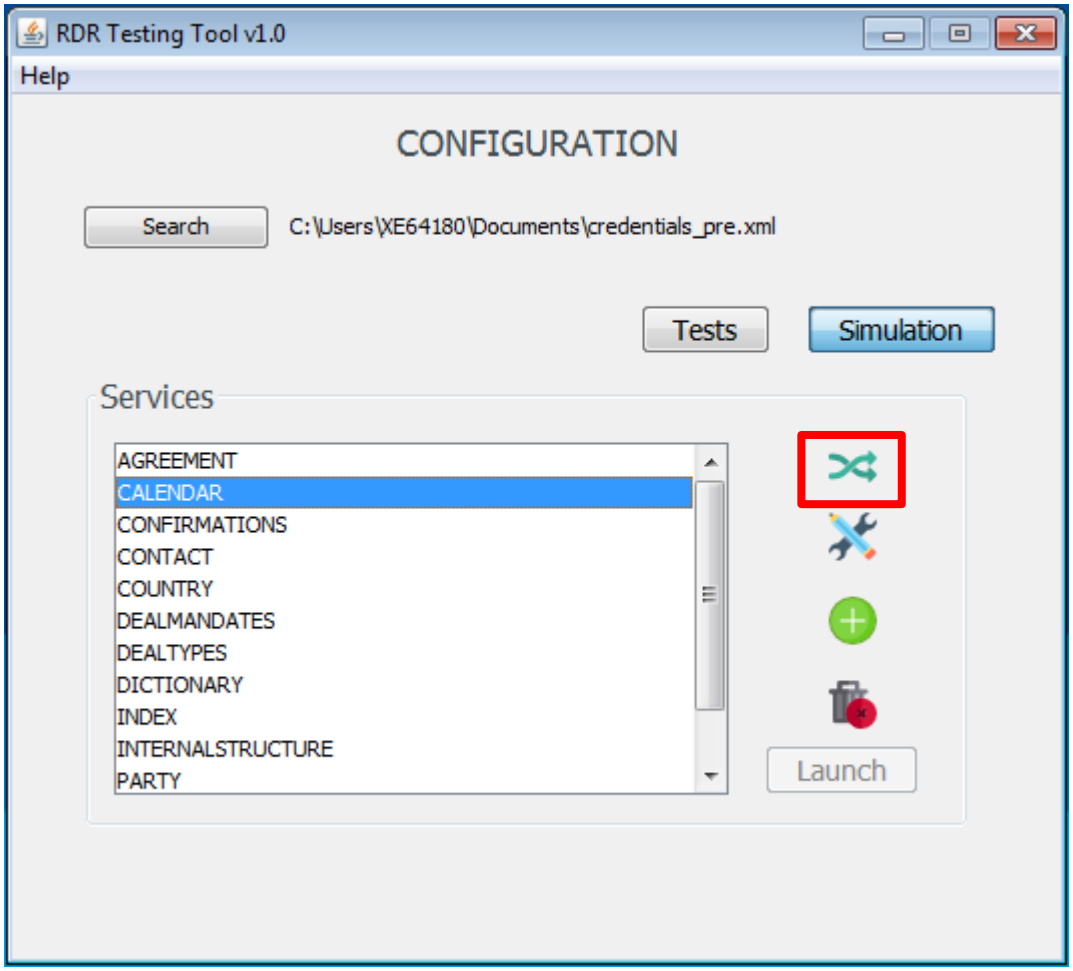



*Figura 30. Modo simulación en Testing Tool*

- Crear

Esta imagen permitirá crear nuevos servicios. Al pulsarla se abrirá una nueva ventana titulada "NEW SERVICE", en la cual, una vez escrito el nombre del servicio nuevo, se podrá crear un nuevo servicio al pulsar el botón "CREATE".

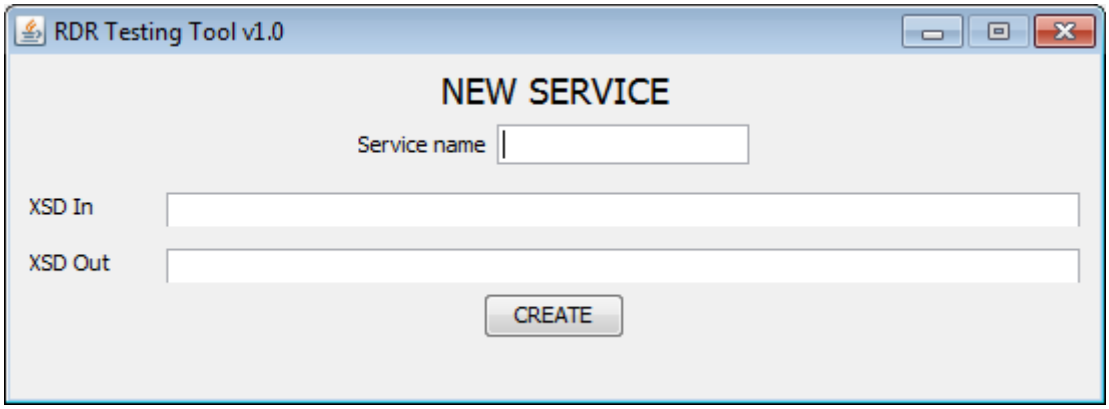



*Figura 31. Nuevo servicio en Testing Tool*

- Eliminar

Esta imagen se habilitará una vez pulsado un servicio de la lista. Cuando se elija un servicio y se pulse en eliminar, se abrirá un cuadro de diálogo con el texto "Are you sure that you want to delete this service?", y se debe pulsar en "DELETE", si se quiere eliminar el servicio, o el "CANCEL", en el caso contrario.

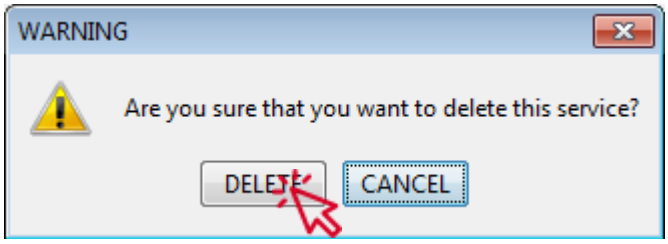



*Figura 32. Eliminar servicio en Testing Tool*

- Launch

Una vez elegido uno, o varios servicios, y elegido el archivo "credentials.xml", se habilitará el botón "Launch".

Este botón permitirá realizar las pruebas con las requests y responses almacenadas en el programa de los servicios seleccionados. Una vez pulsado sobre el botón, se abrirá una nueva ventana con la información, al momento, sobre la prueba.

Tanto si se ha seleccionado un servicio, como si se han seleccionado varios, la ventana abierta tendrá el mismo aspecto. Esta estará compuesta por los siguientes apartados.

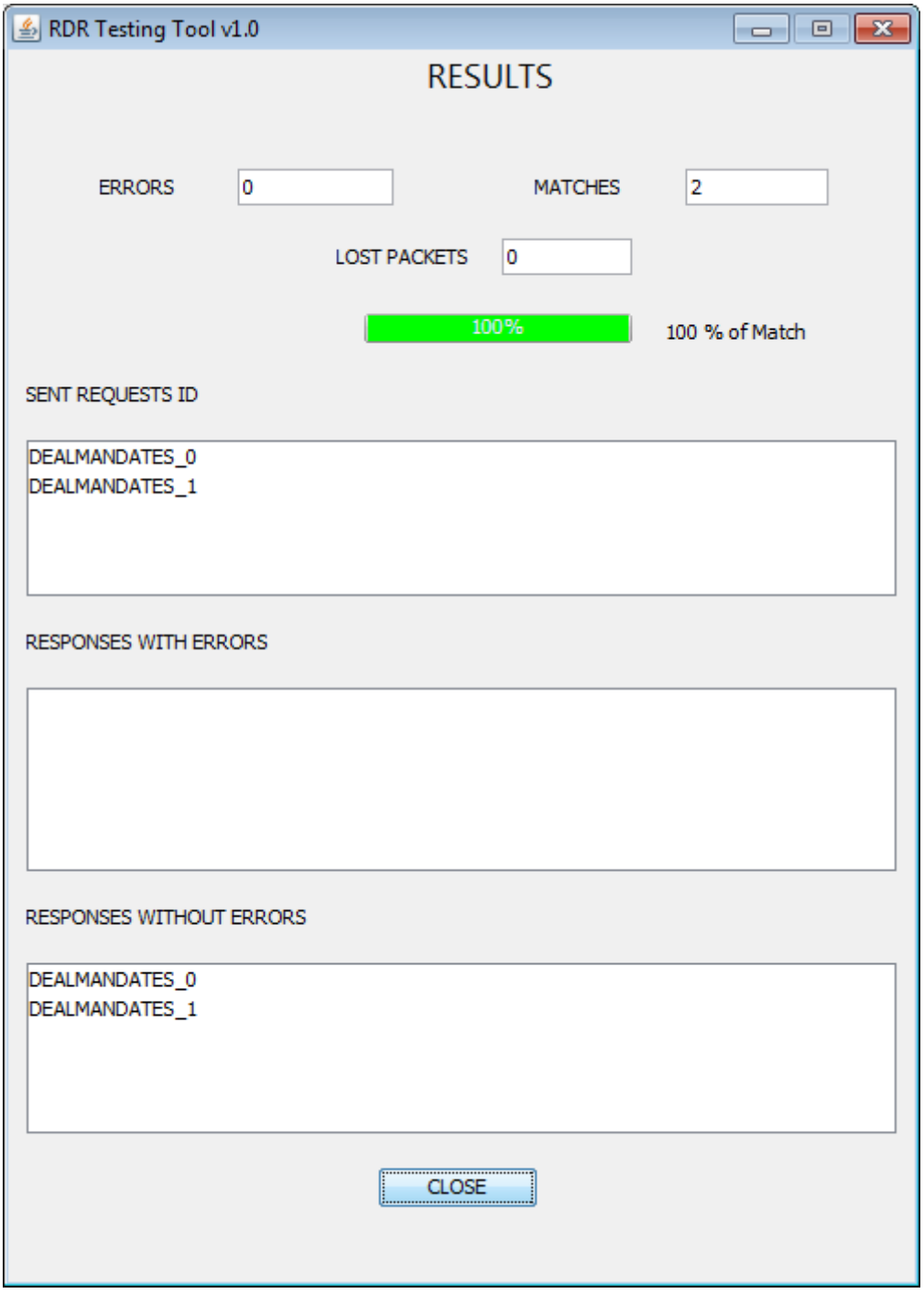



*Figura 33. Pantalla de resultados en Testing Tool*

- ERRORS

Esta celda, contendrá el número de peticiones erróneas con las que ha respondido el programa.

- MATCHES

Esta celda, contendrá el número de peticiones satisfactorias con las que ha respondido el programa.

- LOST PACKAGES

Esta celda, contendrá el número de respuestas que no se han recibido, esto es, que se han perdido.

- BARRA DE PORCENTAJE

En esta barra de progresión, se representará en verde, sobre el tanto por ciento, el número de aciertos sobre el número de pruebas realizadas que se han hecho. Además, está acompañado con un texto clarificativo con el tanto por ciento de éxito.

- LISTAS DE PETICIONES Y RESPUESTAS

A continuación, se ofrecen tres listas con los títulos "SENT REQUESTS ID", "RESPONSES WITH ERRORS" y "RESPONSES WITHOUT ERRORS", las cuales mostrarán, las id de las peticiones enviadas, las id de las respuestas erróneas recibidas y las id de las respuestas satisfactorias recibidas, respectivamente. Se puede apreciar que una vez pulsada en cualquiera de las tres listas, se seleccionará la misma id en la lista de respuestas o de peticiones contraria.

Además, si se pulsa dos veces sobre cualquiera de las peticiones o respuestas, se abrirá una ventana nueva titulada "CHECK", que contendrá la petición enviada, la respuesta recibida por la aplicación y la respuesta almacenada por el programa para dicha petición.

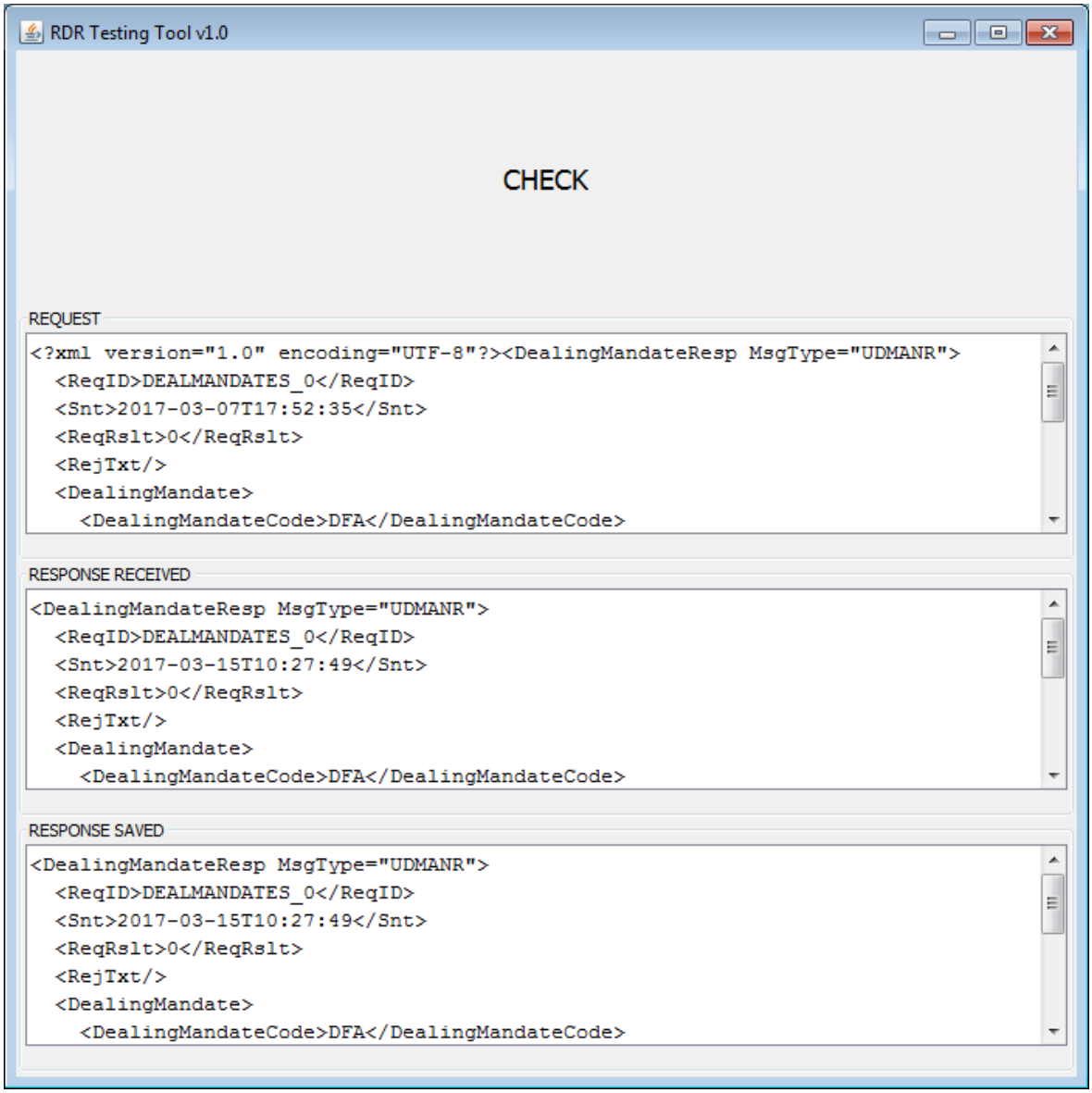



*Figura 34. Pantalla de comparación en Testing tool*

# Capítulo 10. ANEXO B: GUÍA DE INSTALACIÓN

En el presente anexo tratará de darse explicación a diferentes aspectos necesarios para la ejecución de los programas realizados, en el presente proyecto, de manera satisfactoria.

## *10.1 INSTALACIÓN DE JAVA*

Java es el lenguaje en el que están escritos los programas que se han realizado para el presente proyecto. Aunque este es un lenguaje multiplataforma, necesita de la instalación de la Java Virtual Machine, para poder ejecutar las aplicaciones.

Para la instalación de Java es necesario ir hasta la página web https://www.java.com/es/download, como se muestra en imagen siguiente.


*Figura 35. Página web oficial de Java*

En esta página debe pulsarse sobre el botón *"Aceptar e iniciar descarga gratuita"* y esperar a que comience la descarga.

Una vez ha finalizado la descarga, es necesario pulsar sobre el archivo ejecutable descargado.

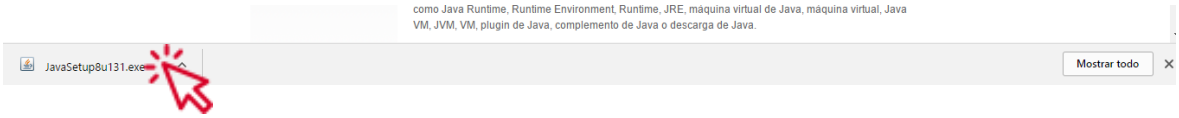



*Figura 36. Descarga Java*

Una vez ejecutado, seguimos los pasos de instalación de la aplicación. En primer lugar, se pulsa en "instalar".

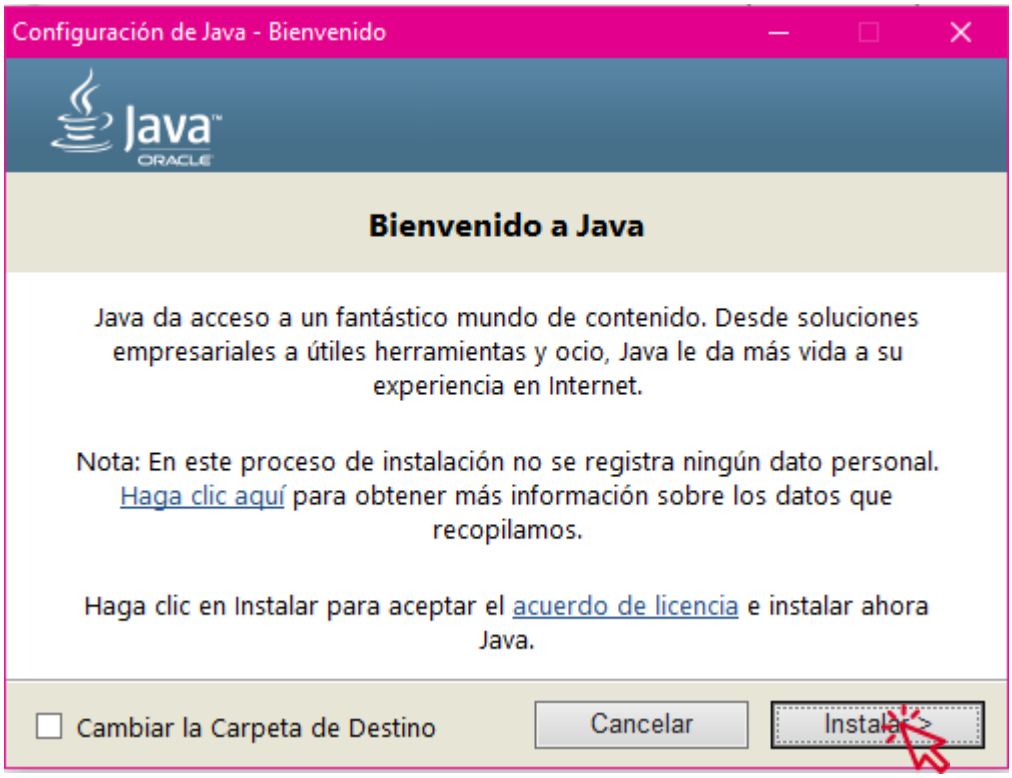



*Figura 37. Pantalla instalación Java*

Se espera a que la barra de progreso de instalación finalice.


*Figura 38. Progreso instalación Java*

Finalmente aparecerá una ventana notificando que la instalación se ha realizado de manera correcta y solamente será necesario pulsar en "cerrar".

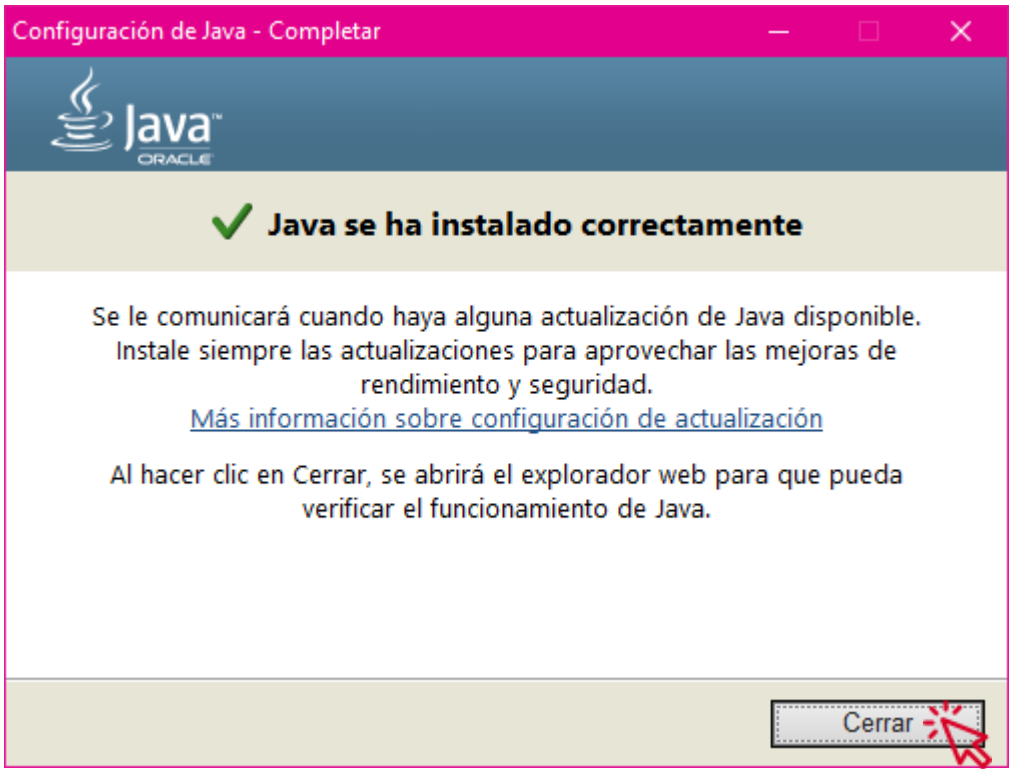



*Figura 39. Pantalla final instalación Java*

Con todo esto finalizado, ya se habrá realizado la instalación de Java correctamente.

### 10.1.1 INSTALACIÓN DEL JDK DE JAVA

Una vez finalizada la instalación de java, será necesaria la instalación del kit de herramientas para desarrolladores Java denominado JDK. Este será necesario para la compilación de aplicaciones y su ejecución.

Para la descarga de JDK será necesario acceder a la siguiente dirección web: http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html#javasejdk

Una vez se entra en esta página web, es necesario pulsar sobre la imagen de Java con la palabra "Download".

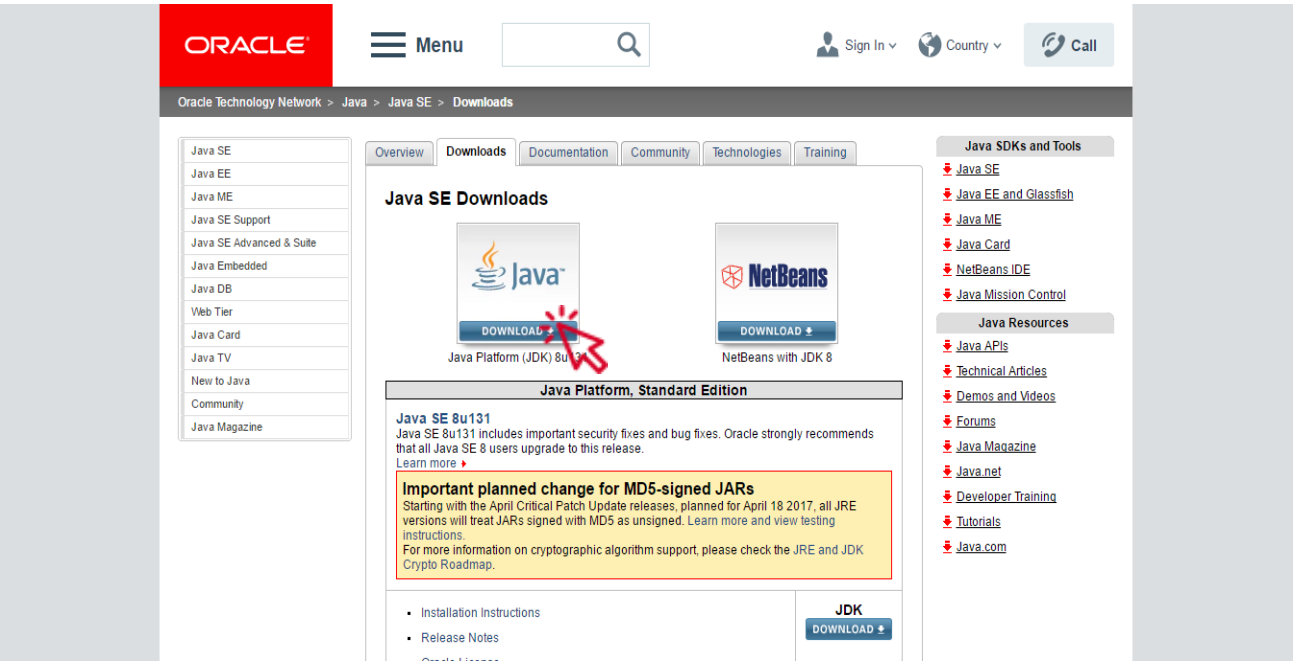



*Figura 40. Página web oficial JDK*

Esta acción redireccionará a otra página, donde se encuentra una lista con todos los JDK disponibles para cada uno de los sistemas operativos existentes en la actualidad. Una vez se está en esta página pulsar la opción que se adecue al dispositivo desde el que se está navegando.

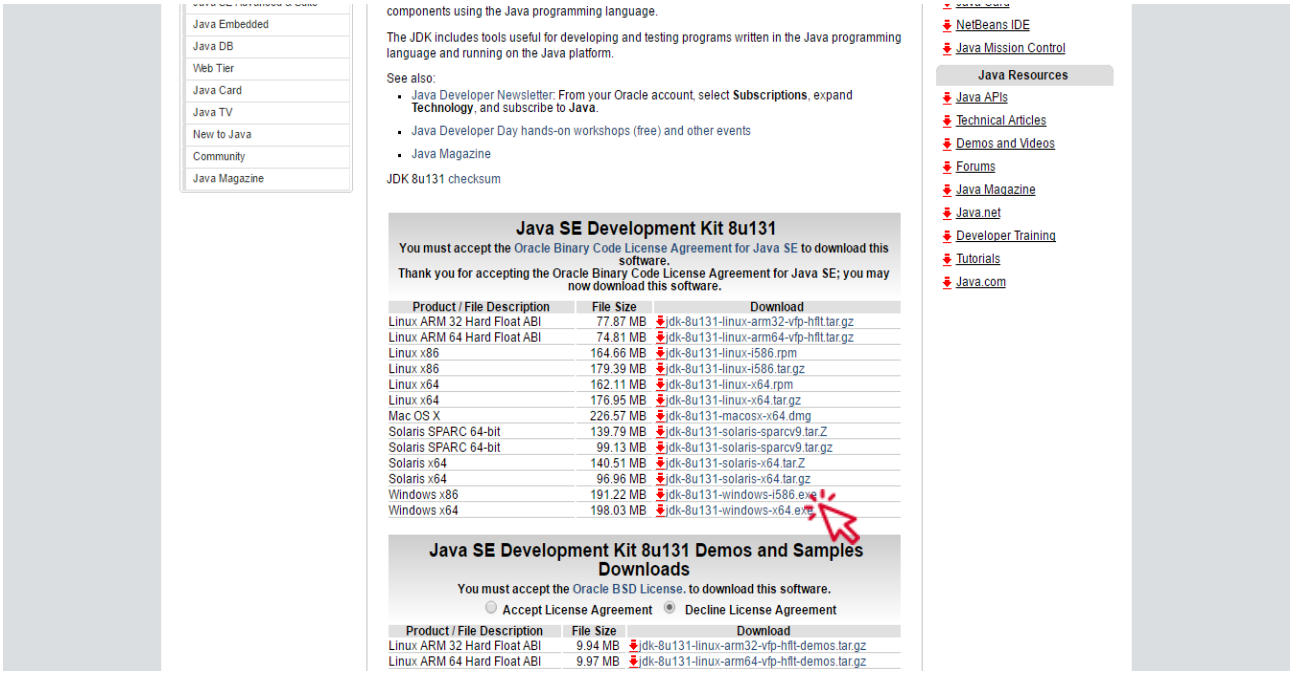



*Figura 41. Enlaces descargar JDK*

Cuando se ha pulsado en el enlace, es necesario esperar a que comience la descarga. Una vez finalizada la descarga, se debe pulsar en el ejecutable para que comience la instalación.

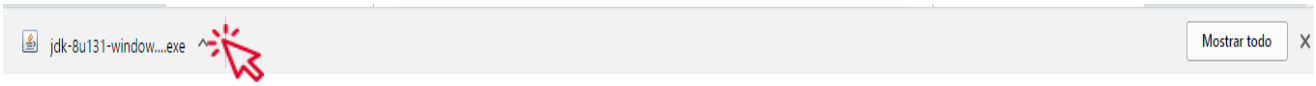



*Figura 42. Ejecutable JDK*

Una vez se abre la ventana de instalación, debe pulsarse en el botón “Next”.

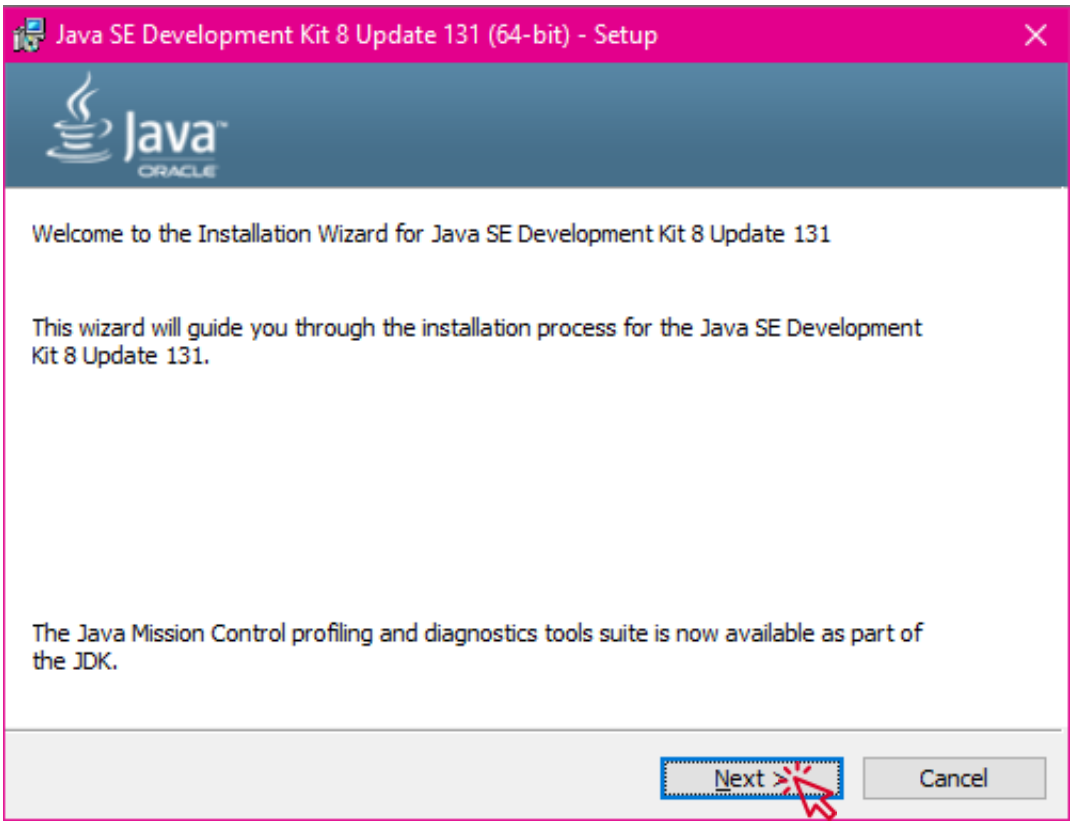



*Figura 43. Pantalla instalación JDK*

En la siguiente pantalla debe pulsarse sobre el botón “Next” de nuevo.

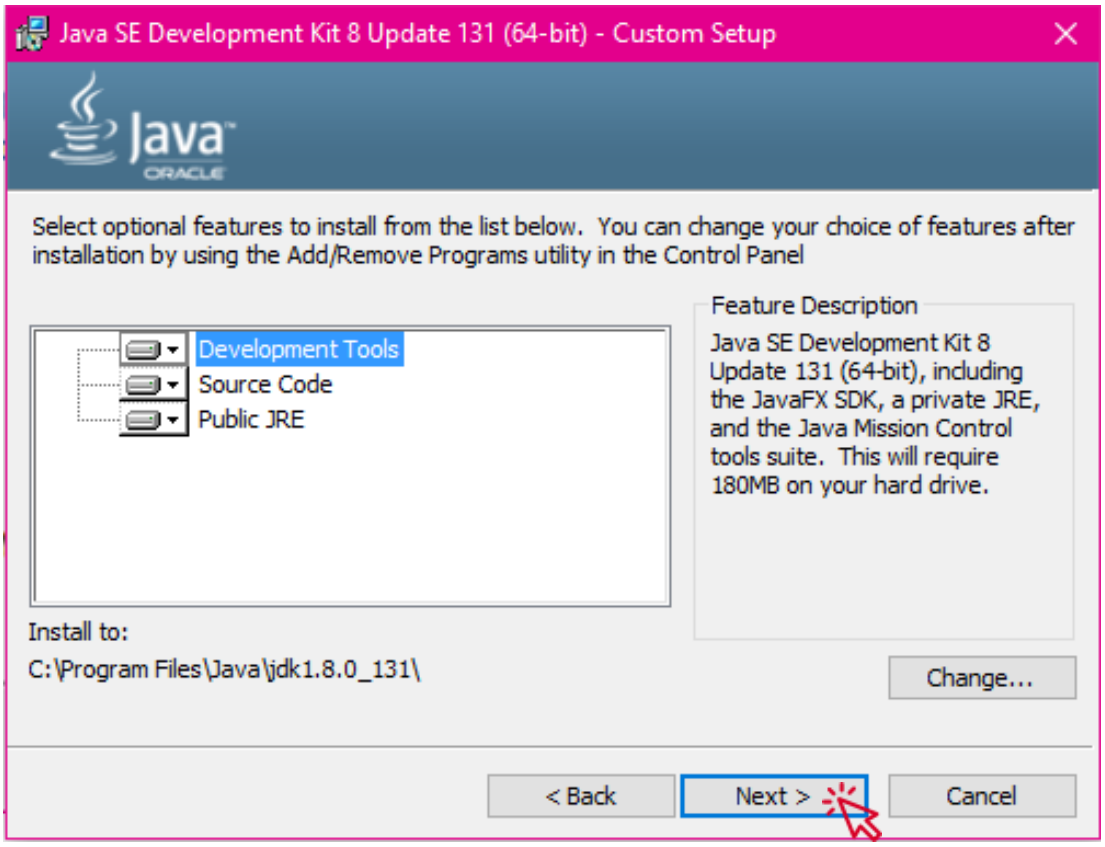



*Figura 44. Selección ruta JDK*

A continuación, debe esperarse a que la barra de progreso finalice.

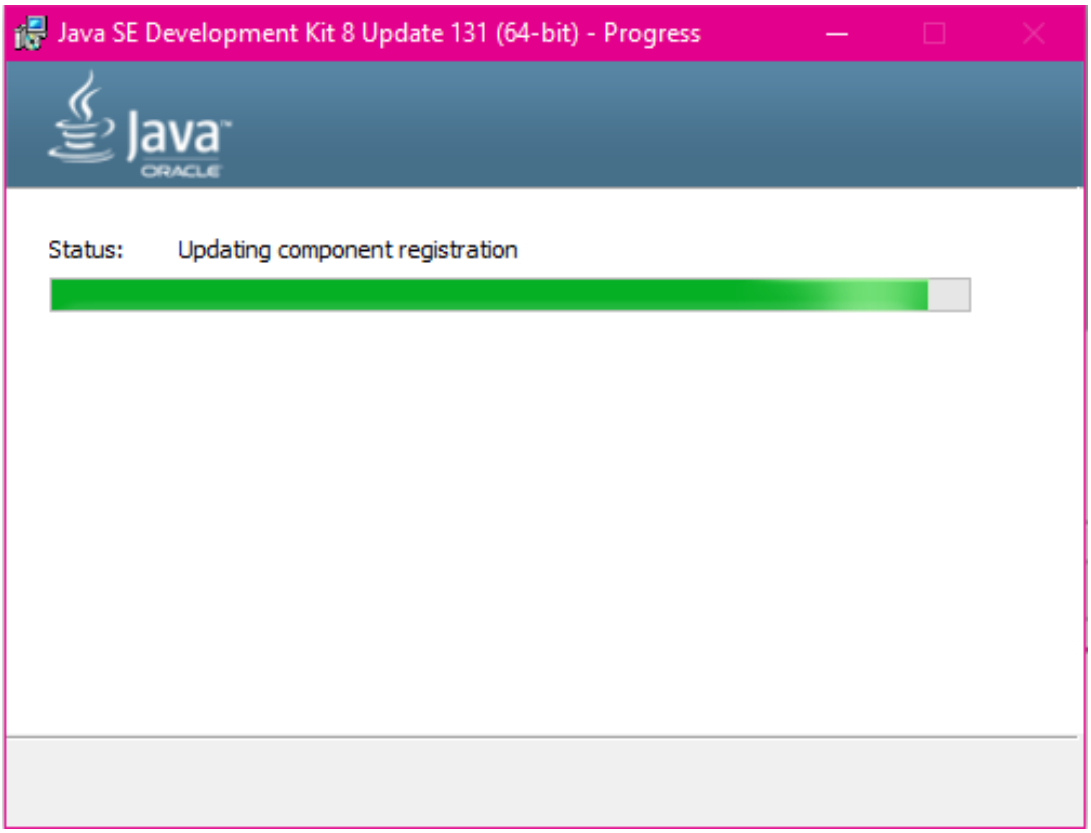



*Figura 45. Progreso instalación JDK*

Una vez la barra de progreso finalice, deberá pulsarse en *"Siguiente"* para elegir la carpeta de instalación.

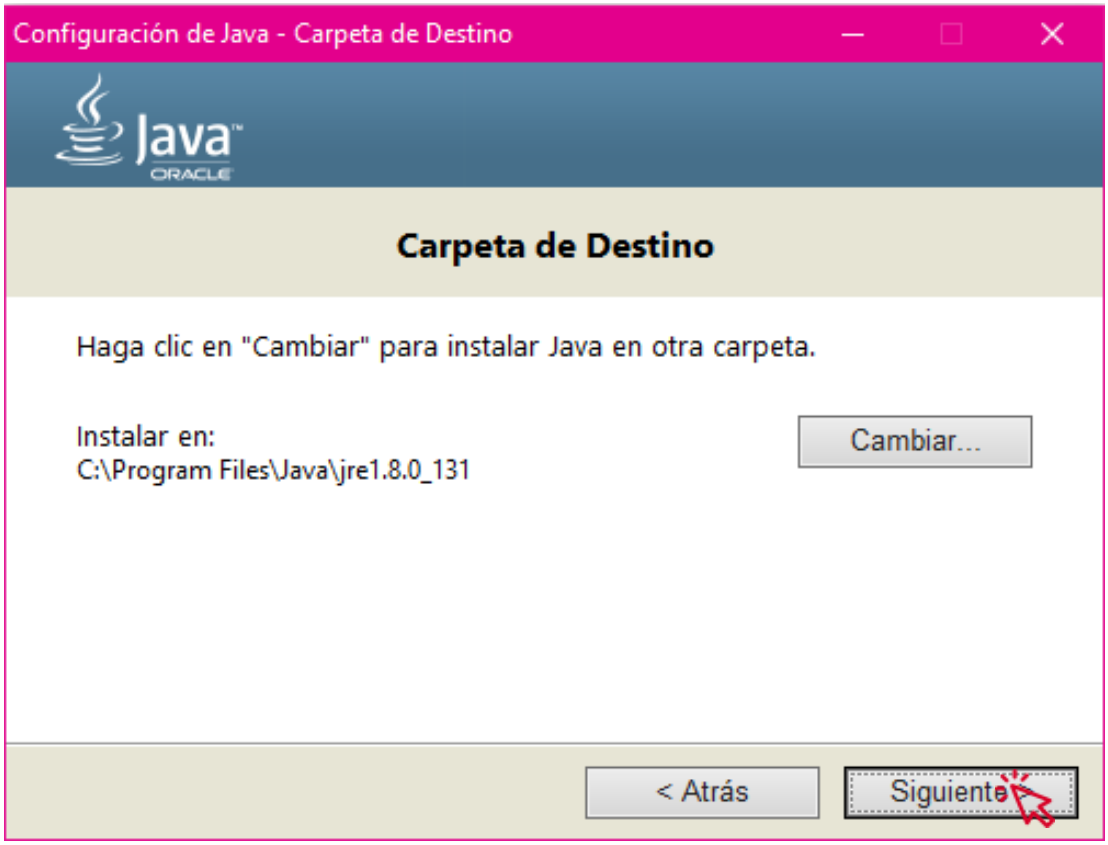



*Figura 46. Ruta instalación JRE*

Deberá esperarse a que finalice la barra de progreso de instalación de Java.


*Figura 47. Progreso instalación JRE*

Al finalizar la barra de progreso pulsar en "Close" y así finalizará la instalación.


*Figura 48. Finalizar instalación JDK*

## 10.1.2 VARIABLE DEL SISTEMA

El último paso para poder usar programas Java será fijar la variable del sistema con Java, para poder ser ejecutado como una variable del propio sistema operativo desde la ventana de línea de comandos.

Para fijar la variable de sistema será necesario acceder a la ventana de "Panel de Control" de la máquina y pulsar sobre el enlace "Sistema".

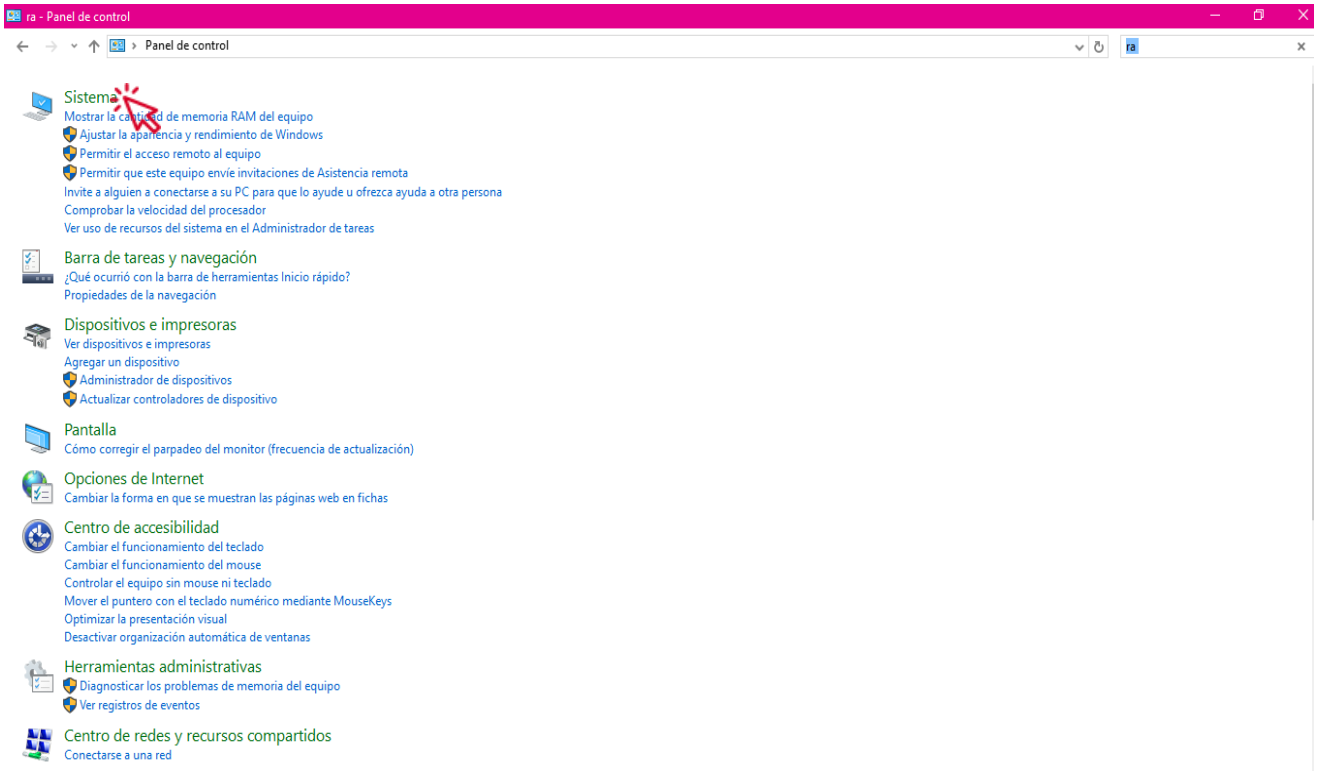



*Figura 49. Pantalla Panel de Control*

En la ventana de Sistema debe pulsarse sobre el enlace "Configuración avanzada del sistema".

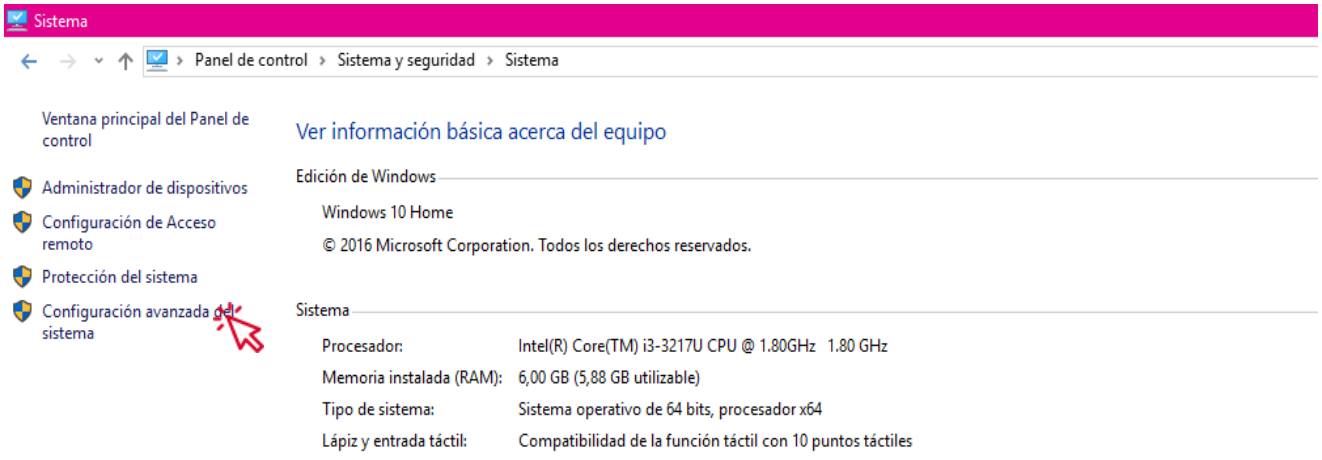



*Figura 50. Configuración avanzada del sistema*

En la ventana que se abre debe pulsarse en "Variables de entorno…".

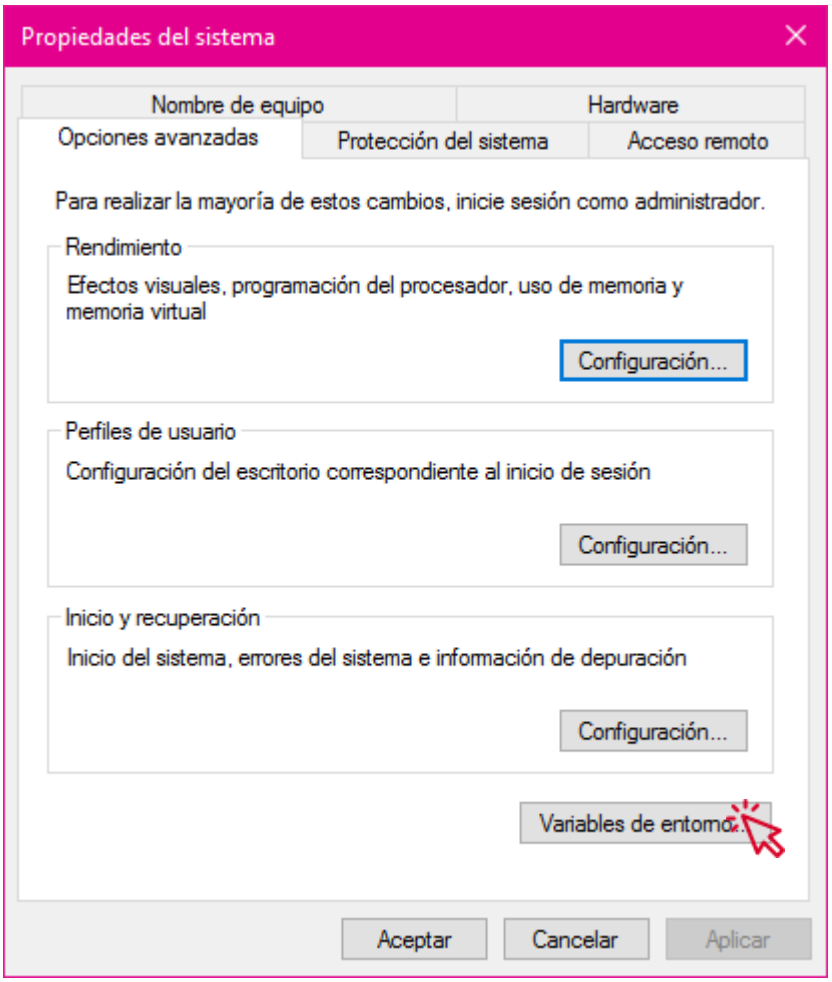



*Figura 51. Variables de entorno*

En la siguiente ventana abierta debe pulsarse, en la sección de "Variables del sistema", en el botón "Nueva".

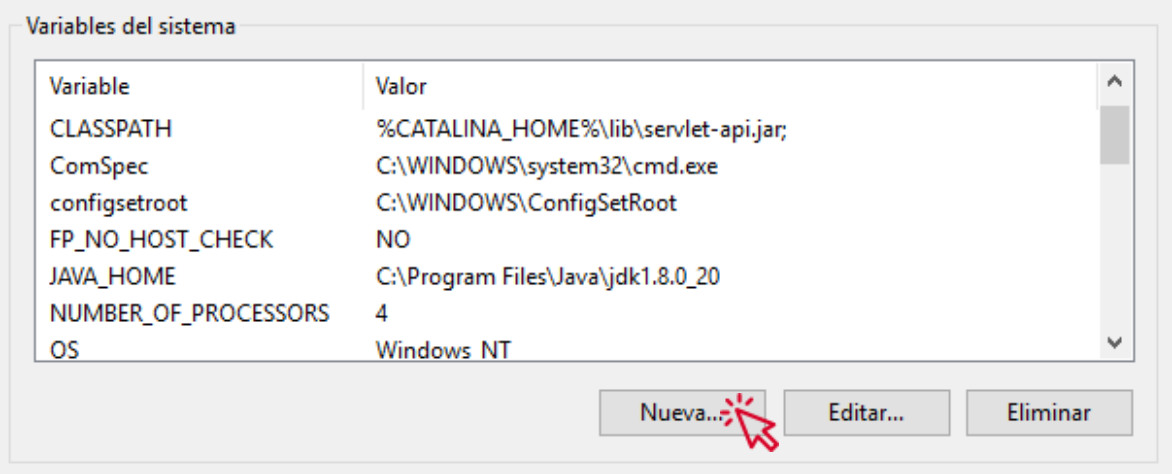



*Figura 52. Nueva variable del sistema*

En la nueva ventana, deberán rellenarse los campos:

- **Nombre de la variable**: con el texto *"JAVA_HOME"*.
- **Valor de la variable**: es necesario pulsar en *"Examinar archivo…"* y buscar la carpeta *bin* en la carpeta de instalación de *JDK* del paso anterior.

Una vez rellenado los campos, debe pulsarse en el botón *"Aceptar"*.

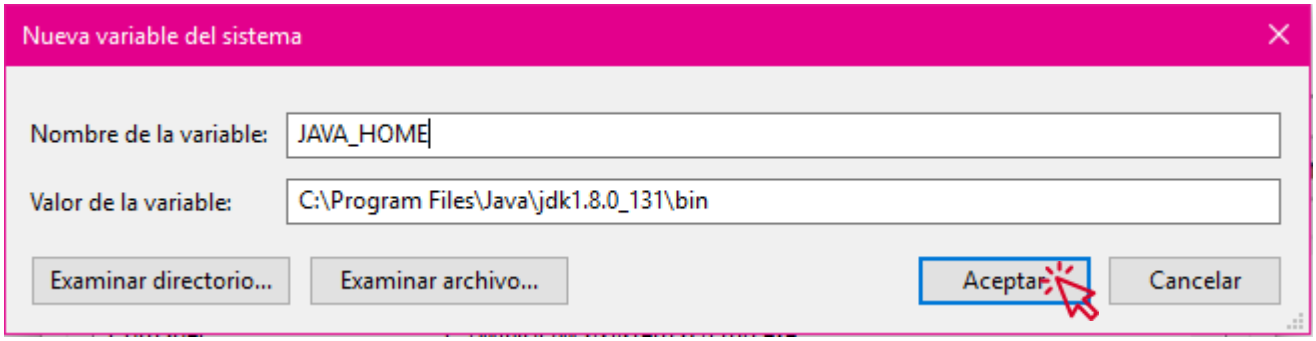



*Figura 53. creación JAVA_HOME*

Con estos pasos ya se habrá fijado la variable de sistema del JDK.