Universitat Autònoma de Barcelona

escola d'enginyeria

(5657 – FUNCTIONAL PROCESS ANALYZER)

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Alex Torras García
i dirigit per
Asier Ibeas Hernandez
Bellaterra, 19 de Maig de 2014

El sotasignat, Asier Ibeas Hernandez

professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en/na Alex Torras García

I per a que consti firma la present.

Signat:

Bellaterra, 19 de Maig de 2014

Agradecimientos

Quisiera mostrar mi respeto y agradecimiento a todos los profesores que me han enseñado, sin los cuales esto hubiera sido imposible.

También agradecer a mis compañeros y superiores que, en el mundo laboral, han colaborado poniendo un poco de ellos en mis conocimientos.

Agradecer la inestimable ayuda del director del proyecto, Asier Ibeas, colaborando en todo momento, haciendo cortas las distancias.

Quiero dar un agradecimiento especial a mi familia, mi mujer Silvia, mi hija Éire, mi madre Isabel, gracias por vuestra paciencia. Mis abuelos, Paulino y Juanita, que vieron el principio de esta aventura pero, lamentablemente, no vieron el final. A mi padre que sigue ahí, de alguna forma, en algún lugar del tiempo.

Tabla de contenido

Índice de figuras

Índice de tablas

1 Introducción

La utilización de *Frameworks* y herramientas que estandarizan los desarrollos facilitando el mantenimiento posterior de las aplicaciones y aportando un valor añadido es una necesidad en la actualidad. Al inicio de los proyectos se determinan las necesidades que va a tener y se establecen las herramientas que se van a utilizar para cubrir ciertos aspectos que son comunes en todos los proyectos, tales como la gestión de usuarios, acceso a base de datos, gestión de negocio… Uno de estos aspectos es la gestión de los errores asociados a procesos internos de la aplicación. Esta gestión es necesaria por varios motivos entre los que podríamos destacar los acuerdos de nivel de servicio que se establecen en los contratos de mantenimiento. Para poder aplicar estos acuerdos es necesario conocer exactamente cuándo se ha producido el problema y conocerlo en tiempo real para poder actuar rápidamente si el problema lo requiere.

Una y otra vez las aplicaciones repiten los mismos controles embebiendo en la aplicación controles en los propios procesos y sistemas que alerten hechos "a medida". Por otro lado se pueden encontrar en el mercado productos de monitorización de servicios, son aplicaciones que monitorizan los servicios que ofrecen otras aplicaciones o sistemas como bases de datos, servidores de aplicaciones, etc. Esta monitorización la realizan desde fuera sin atender a cuestiones funcionales.

El sistema que se propone se encarga de realizar tareas de monitorización de procesos a nivel funcional, desde dentro de las aplicaciones, estandarizando el método de trabajo y de forma que pueda alertar en tiempo real de problemas relativos a funcionalidades de las aplicaciones que estén fallando.

1.1 Descripción general del proyecto

El proyecto consiste en la realización de un sistema de análisis funcional de procesos en tiempo de ejecución. El sistema actuará como un monitor que controla la correcta ejecución de los procesos y, en caso de producirse algún problema, informará del mismo de la forma que se haya indicado. Con el sistema interactuarán distintos actores: las aplicaciones cuyos procesos se van a monitorizar, aplicaciones externas que inician los procesos, usuarios que consultan por las ejecuciones… El sistema será nombrado como MPROC (monitor de procesos) en adelante.

Para comprender mejor se va a trabajar sobre un ejemplo que va a servir de guía en toda la memoria: una tienda virtual. Una tienda virtual tiene procesos integración de datos con los vendedores (fabricantes, importadores…) que envían sus catálogos de productos a vender con los precios, ofertas… También tiene procesos de muestra de los catálogos y ventas de los productos con los clientes, procesos de pago de los productos vendidos y por último procesos que informan a los vendedores de los productos que han vendido y que tienen que enviar a los clientes. Todos estos procesos son susceptibles de errores de negocios controlados que pueden requerir intervención por parte del equipo de mantenimiento o, en otros casos, informar del problema para que se pueda subsanar. Por ejemplo, la integración del catálogo de ventas puede tener un problema interpretando el fichero con los datos, encontrarse algún producto con precio negativo, etc. El proceso de pago puede encontrarse con problemas de acceso a la plataforma bancaria, etc. Todos estos posibles errores deben estar inventariados en MPROC de forma que se pueda indicar como proceder en cada caso. La aplicación de la tienda virtual cada vez que va a iniciar un proceso informa a MPROC del inicio, de la finalización y de los errores que se produzcan. También se pueden definir puntos en el proceso por los que el proceso debe pasar de forma opcional u obligatoria. Por ejemplo, la integración de datos de ventas puede tener los

puntos de chequeo (o *checkpoints*) “interpretar documento de entrada” “validar lista de precios” “integrar lista de precios”… A través de estos *checkpoints* se puede estudiar si la ejecución de un proceso ha sido correcta o no.

Básicamente estos serán los elementos que manejará MPROC: Procesos, errores y *checkpoints*. El último concepto que maneja es el concepto de alerta: frente a determinados eventos como el registro de un error se pueden lanzar alertas asociadas a ese evento de forma que se notifique en tiempo real a las personas que se hayan definido de la situación que se está dando, ya sea un problema integrando el catálogo o un problema accediendo a la banca electrónica.

Juntamente con los conceptos del sistema están los flujos de trabajo. Básicamente hay 3 flujos de trabajo:

- Preparación previa del sistema: definición del catálogo de procesos. Son las tareas administrativas que se deben realizar previamente a la puesta en producción de la monitorización de cualquier aplicación. En esta fase se definirán las aplicaciones, los procesos, los puntos de chequeo, los errores, las alertas… todos los conceptos que se han descrito anteriormente.
- Registro de información en tiempo de ejecución: Una vez se han definido los procesos estos pueden empezar a enviar información al sistema de sus ejecuciones. Esta información será registrada y analizada en tiempo real.
- Explotación de la información: El sistema contendrá formularios que permitirán la búsqueda de información de las ejecuciones por parte de usuarios o del mismo sistema de alertas para el envío de alertas.

1.2 Estructura de la memoria

La memoria se haya estructurada de la siguiente forma:

En el primer capítulo se ha realizado una introducción al proyecto haciendo un resumen y se introduce algún ejemplo para comprender mejor el concepto de la aplicación. El ejemplo se irá revisando durante toda la memoria.

En el segundo capítulo se realiza un estudio de la viabilidad del proyecto. Marcando los objetivos del proyecto se estudian las distintas opciones similares que se encuentran actualmente en el mercado analizando las similitudes y diferencias con el sistema propuesto. Así mismo se hace una breve descripción de las tecnologías que van a utilizarse para cumplir los objetivos pues sin la tecnología adecuada la realización del proyecto es una tarea imposible.

En el tercer capítulo se recogen las especificaciones del proyecto. Una vez realizada una descripción global se entra en el detalle de los distintos requisitos funcionales de la aplicación modelándolos en UML. Se realiza un estudio de la arquitectura tecnológica requerida para la implementación del sistema en función de los requisitos. Una vez definida la arquitectura se realiza una descripción del diseño de la aplicación mediante diagramas de clases UML.

En el cuarto capítulo se realiza una descripción de los elementos destacables de la construcción de la aplicación: cuestiones de configuración, de seguridad, de logs, descripción de la base de datos, de las interfaces…

En el quinto capítulo se realiza una demostración completa de la aplicación. Utilizando el ejemplo descrito en la introducción se escenifica en la aplicación distintos procesos relacionados con el ejemplo.

En el sexto capítulo se resumen las conclusiones, haciendo un análisis de las partes de la aplicación que pudieran ampliarse y del nivel de cumplimentación de objetivos.

En el séptimo capítulo se ha incluido una bibliografía global de la memoria.

2 Estudio de viabilidad del proyecto

La viabilidad del proyecto puede ser estudiada desde distintas perspectivas: económica, tecnológica, comercial…

La viabilidad económica no aplica en este proyecto. No obstante se puede decir que si el proyecto es realizable con los recursos disponibles para el mismo, es viable económicamente. Esto dependerá de la planificación del proyecto, desarrollada en el siguiente punto.

La viabilidad tecnológica podemos desarrollarla de la siguiente manera:

- La aplicación debe ser accesible desde distintos puntos de la organización que la implemente así como desde fuera de la organización. Esto es realizable mediante una aplicación Web.

    - La parte que deba ser accesible por un humano será una aplicación Web con un frontal HTML accesible desde la intranet de la organización o desde internet. Siendo una aplicación WEB será accesible desde cualquier sistema, no habrá una dependencia con la plataforma que se use en los clientes. Tanto Windows como cualquier otro sistema como Linux, Unix o Apple disponen de navegadores web que hacen que la aplicación esté accesible para ellos.

    - La parte que deba ser accesible para otras aplicaciones (aplicaciones que son monitorizadas o aplicaciones externas que acceden para comprobar la ejecución de ciertos procesos) pueden ser desarrolladas en un escenario SOA (*Service Oriented Architecture*) mediante REST o SOAP o con soluciones más dependientes de una tecnología concreta como EJBs remotos en JAVA

- La aplicación no debe manejar una gran cantidad de datos ya que los datos que maneja tienen una caducidad. Es decir, la consulta de ejecución de un proceso no es algo que se vaya a hacer pasado un tiempo razonable desde la ejecución. Por lo tanto, tecnológicamente no existe ninguna limitación que comprometa la viabilidad del proyecto en cuanto a volumen de datos.

- La aplicación debe alertar a los usuarios si algo no va según lo previsto. Esto se puede realizar de varias formas. Las soluciones tecnológicas que se aportan para este tipo de problemas son básicamente dos: mediante SMS o mediante email. La solución SMS es una solución que aporta un coste económico extra, aunque hay varias compañías que la ofrecen e incluso ofrecen APIs para el envío de SMSs no sería una opción viable por el coste económico que comporta. La opción de envío de alertas mediante email si es una opción viable puesto que hay multitud de servicios que ofrecen el envío de emails gratuitamente.

Dicho esto, tendríamos un sistema basado en un servidor de aplicaciones WEB con una base de datos. Como servidor de aplicaciones optaría por un servidor WebSphere de IBM y como base de datos un Oracle.

Se escogen estos entornos pues son entornos muy robustos, ampliamente utilizados y que disponen de versiones gratuitas. Todo esto facilita el desarrollo y el mantenimiento al disponer de multitud de información.

Los desarrollos serian en JAVA, usando como IDE de desarrollo Eclipse. Se escoge Eclipse al ser un entorno gratuito y ampliamente desarrollado en la actualidad.

En los siguientes apartados se va a realizar un análisis de los distintos elementos relacionados con la viabilidad del proyecto: Objetivos, estado del arte, planificación, herramientas, metodologías y análisis de riesgo.

En los siguientes puntos dentro de este apartado se desarrollarán todas estas cuestiones.

2.1 Objetivos del proyecto

El objetivo final del proyecto es la realización de una aplicación web que permita la administración de procesos (gestión del catálogo de procesos) y alertas asociadas a los procesos, el registro de los procesos que se están monitorizando, la generación de alertas en tiempo real y la explotación de todas los datos registrados.

La gestión administrativa y de consulta debe ser accesible desde la red corporativa de la compañía que implante la solución y desde cualquier otro lugar a través de internet.

El acceso a los servicios de registro de ejecución de procesos debe ser independiente de cualquier plataforma permitiendo la monitorización de procesos independientemente de la plataforma donde esos procesos estén desarrollados: se podrá monitorizar procesos de .net, procesos JAVA, PHP… que estén en un Windows, un Linux, un Solaris…

Debe ser un sistema robusto, en alta disponibilidad, rápido, eficiente y fiable.

Como objetivos intermedios se requiere la justificación de las decisiones sobre las herramientas, *software*, *frameworks*, etc que se hayan seleccionado para su desarrollo. También se deberá realizar una exposición sobre el diseño arquitectónico utilizado explicando que se está haciendo y por qué. También se deberá realizar un diseño de la aplicación mediante diagramas de clases.

Como objetivos iniciales se requerirá la toma de requisitos en formato UML así como una descripción de cada uno.

2.2 Estado del arte

Existen multitud de sistemas para la monitorización de otros sistemas. Algunos de estos son sistemas propietarios, realizados por los mismos fabricantes de *software* sobre el que corren las aplicaciones. Estos sistemas están muy orientados a monitorizar desde una perspectiva de sistemas más que desde una perspectiva funcional. Por ejemplo, para Oracle existe el "Enterprise Monitoring" (http://www.oracle.com/technetwork/ es/oem/sys-mgmt/index.html) que es un sistema orientado a monitorizar la base de datos: el espacio que ocupa, el tiempo de ejecución de sentencias SQL, problemas generales que puedan ocurrir en la base de datos… Este *software* no entra en las funciones de negocio o funcionales, su función es comprobar que la base de datos esté activa y funcionando correctamente.

A nivel de servidor de aplicaciones existen herramientas que se acercan un poco más al sistema propuesto, como el "IBM Business Monitor" para WebSphere (http://www-03.ibm.com /software/products/es/business-monitor/)

No obstante este *software* monitoriza ciertos elementos del sistema de negocio que se encuentre en ejecución en un WebSphere Application Server pero no desde un punto de vista funcional, sino desde un punto de vista de sistema: si el proceso de negocio está activo, cuánto tarda en ejecutarse, número de ejecuciones…

De todas formas se acerca bastante a lo que se propone. Pero tiene carencias a la hora de evaluar funcionalmente las ejecuciones. Por ejemplo, si un proceso integra un fichero de datos el cual tiene un formato predefinido y el proceso descarta un fichero por no ajustarse al formato,

la ejecución del proceso ha terminado correctamente desde un punto de vista de sistema pero el proceso del fichero ha fallado funcionalmente pues no ha integrado los datos. Este tipo de sistemas no son capaces de capturar eventos como este (a no ser que desde la aplicación se empiecen a lanzar excepciones para gestionar cosas así, algo que va contra las *best practices* ya que no es conveniente usar excepciones para realizar controles de flujo o gestiones puramente funcionales)

Por último, hay herramientas generalistas que nos ofrecen varios fabricantes para realizar monitorizaciones de sistemas. Dentro de estas herramientas quizá la más grande y conocida es Tivoli, de IBM. Tivoli es una gran suite de programas para realizar una gran cantidad de tareas dentro del área de sistemas: lanzamiento de procesos *batch*, servicios de *cloud computing* y, dentro de lo que nos ocupa, "Tivoli Monitoring" para la monitorización de sistemas (http://www-03.ibm.com/software/products/es/tivomoni) Este sistema es capaz de comprobar que los sistemas estén activos, de lanzar páginas de test, de examinar logs en busca de determinadas secuencias o excepciones…

No obstante adolece de las mismas carencias que los sistemas anteriores: monitoriza los sistemas desde un punto de vista de sistema y no desde un punto de vista funcional.
Se pueden utilizar parte de las herramientas que nos ofrece para realizar comprobaciones funcionales, pero eso en cierta forma es "trampear" el sistema, usar creativamente las opciones que nos ofrece para dar una solución al problema usando herramientas que han sido diseñadas para otras cosas. Por ejemplo, se podrían usar los logs para lanzar ciertas alertas funcionales.
También existen sistemas externos que pueden monitorizar aplicaciones on-line como http://www.monitor.us
Estos sistemas son similares, comprueban que el sistema responda y poco más.

A excepción de este último sistema (que es un sistema muy básico) el resto tienen una serie de limitaciones importantes:

- Son sistemas muy vinculados al *software* que se esté usando: Oracle, WebSphere… dejarían de funcionar si realizamos un cambio, por ejemplo, de Oracle a SQL-Server
- Son sistemas altamente especializados que requieren un conocimiento amplio de aquello que monitorizamos, bien sea una base de datos o un servidor de aplicaciones
- Son sistemas con un coste económico elevado
- Son sistemas cerrados, en tanto que no es conveniente que sean accesibles desde fuera de la organización que los implemente

En contra partida, el sistema propuesto podría aportar:

- Sistema totalmente independiente del software que se esté usando: se podría monitorizar cualquier sistema que pueda acceder por WebServices, incluso plataformas HardWare tales como líneas de producción…
- Sistema generalista focalizado en "lo que hacen" los sistemas monitorizados más que en como lo hacen o que solución tecnológica utilizan para hacerlo.
- El sistema puede abrirse para las consultas desde internet

2.3 Planificación

El proyecto se dividiría de forma temporal en las fases que habitualmente conforman un proyecto: Requisitos, análisis, diseño construcción y pruebas. Dejamos las fases de implantación y *roll-out* fuera del alcance del proyecto ya que en este caso no aplica, y añadimos una fase que consistiría en la realización de la memoria del proyecto.

Cada fase constaría con el conjunto de tareas, a alto nivel, que se detallan a continuación.

En posteriores fases del proyecto estas tareas se verán más detalladas.

- **Tareas:**

Tarea	Descripción
Fase 1.- REQUISITOS	
Toma de requisitos	Se definen todos los requisitos que debe cumplir la aplicación y se diseñan los diagrames de casos de uso.
Fase 2.- ANALISIS	
Definición de entidades	Se definen las distintas entidades persistentes que componen la aplicación
Análisis de procesos	Se definen los distintos procesos que componen la aplicación
Fase 3.- DISEÑO	
Diseño de arquitectura	
Requerimientos no funcionales	Se definen los requisitos no funcionales de la aplicación: Tiempos máximos de respuesta, idiomas…
AppServer (instalación y parcheado)	Se define el servidor de aplicaciones que se va a usar, la versión, el nivel de parcheado, los procedimientos de instalación…
DDBB (instalación, configuración y parcheado)	Se define la base de datos que se va a usar, la versión, el nivel de parcheado, los procedimientos de instalación…
Framework de persistencia	Se define el *framework* de persistencia que se va a usar, la versión, la forma de funcionamiento… se realiza una prueba de concepto para examinar su correcto funcionamiento.
Framework de negocio	Se define el *framework* de negocio que se va a usar, la versión, la forma de funcionamiento… se realiza una prueba de concepto para examinar su correcto funcionamiento. Se realiza una prueba integrada con el *framework* de persistencia
Framework de presentación	Se define el *framework* de presentación que se va a usar, la versión, la forma de funcionamiento… se realiza una prueba de concepto para examinar su correcto funcionamiento. Se realiza una prueba integrada con el *framework* de negocio y persistencia
Arquitectura SOA	Se decide que arquitectura SOA se va a usar: *WebService*, REST… Se realiza una prueba de concepto para evaluar su correcto funcionamiento con el resto de la arquitectura.

Arquitectura global de la aplicación (módulos…)	Se describe de forma global la arquitectura de la aplicación: que módulos la componen, la paquetización que va a tener, como exponer los servicios, como manejar las excepciones…
Diseño Global	
Diseño de interfaces de usuario	Se diseñan las *interfaces* de usuario
Diseño de base de datos	Se realiza el diagrama de entidad-relación del modelo de datos que compone la aplicación.
Diseño de aplicación	Se diseñan los distintos módulos de la aplicación que interaccionan con la *interface* de usuario y el modelo de datos.
Diseño de pruebas	Se diseñan el conjunto de pruebas funcionales que la aplicación tendrá que superar
Fase 4.- CONSTRUCCIÓN	
Construcción de interfaces	
Interface Web	Se implementa la interface Web (interface humana) para la aplicación
Interface *WebService*	Se implementa la interface *WebService* (interface para aplicaciones) de la aplicación
Interface JMS	Se implementa la interface de mensajería de la aplicación
Construcción de la administración	Se implementan el funcionamiento de las pantallas de administración de la aplicación
Construcción de procesos BATCH	Se implementan los diferentes procesos *batch* de la aplicación, los procesos de análisis de datos…
Construcción de procesos on-line	Se implementan los diferentes procesos *on-line* de la aplicación, los procesos de consulta de ejecuciones de procesos…
Fase 5.- PRUEBAS	
Pruebas funcionales	Se ejecutan las pruebas funcionales diseñadas en la fase de diseño. Se comprueba su correcta ejecución
Fase 6.- MEMORIA	
Elaboración de la memoria	Elaboración de la memoria del proyecto que se llevará a cabo durante todo el proyecto

- **TimeLine**: Línea de tiempo global del proyecto. En la figura 1 podemos ver la línea de tiempo prevista.

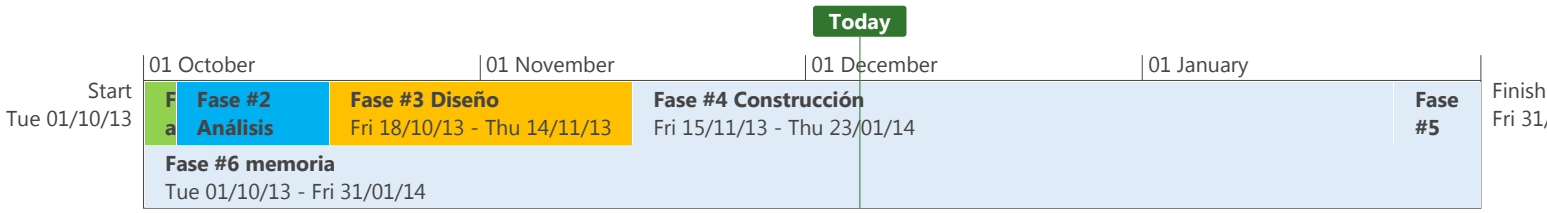
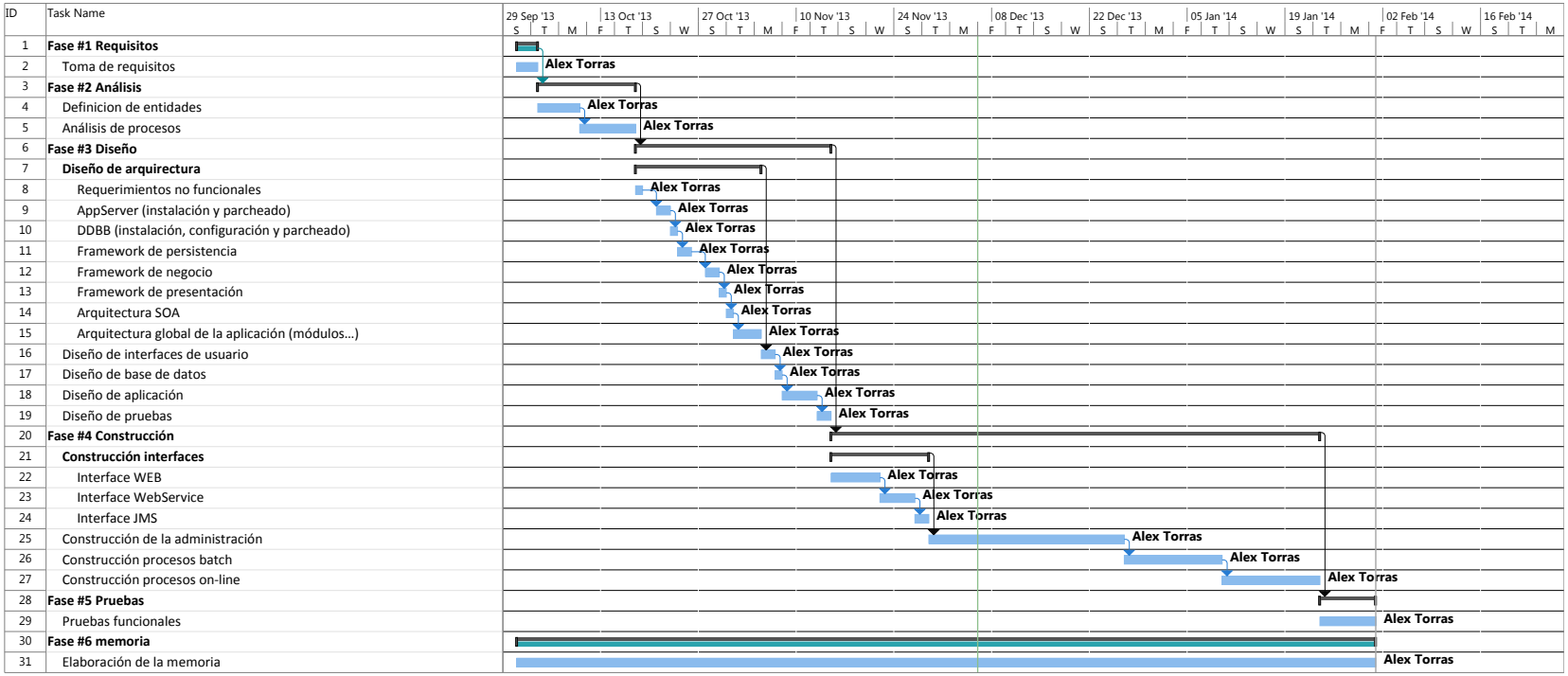


Figura 1 - TimeLine

- **Conjunto de tareas, duración y planificación:** Diagrama de gant


ID
Task Name
29 Sep '13
13 Oct '13
27 Oct '13
10 Nov '13
24 Nov '13
08 Dec '13
22 Dec '13
05 Jan '14
19 Jan '14
02 Feb '14
16 Feb '14
1 Fase #1 Requisitos
2 Toma de requisitos
3 Fase #2 Análisis
4 Definicion de entidades
5 Análisis de procesos
6 Fase #3 Diseño
7 Diseño de arquirectura
8 Requerimientos no funcionales
9 AppServer (instalación y parcheado)
10 DDBB (instalación, configuración y parcheado)
11 Framework de persistencia
12 Framework de negocio
13 Framework de presentación
14 Arquitectura SOA
15 Arquitectura global de la aplicación (módulos...)
16 Diseño de interfaces de usuario
17 Diseño de base de datos
18 Diseño de aplicación
19 Diseño de pruebas
20 Fase #4 Construcción
21 Construcción interfaces
22 Interface WEB
23 Interface WebService
24 Interface JMS
25 Construcción de la administración
26 Construcción procesos batch
27 Construcción procesos on-line
28 Fase #5 Pruebas
29 Pruebas funcionales
30 Fase #6 memoria
31 Elaboración de la memoria
Alex Torras

2.4 Tecnología empleada – Herramientas

Es este apartado se describirán las herramientas y tecnologías necesarias para realizar el proyecto.

2.4.1 Java

Figura 2 - Logotipo Java

Es un lenguaje de código intermedio desarrollado por James Gosling y publicado en el año 1995 por Sun Microsystems. Al ser un lenguaje de código intermedio puede ejecutarse en cualquier máquina virtual de java (en adelante JVM) que interprete el b*ytecode* generado por el compilador de java, lo que lo hace independiente de la plataforma en la que se ejecute. Es un lenguaje basado en objetos. Otra de las características importantes de java es el *garbage collector*, un mecanismo implícito de gestión de memoria que abstrae de las tareas de alocación y desalocación de memoria al programador. Actualmente es uno de los lenguajes más utilizados para aplicaciones cliente-servidor, y hay disponibles en el mercado servidores de aplicaciones que automatizan parte de las funciones necesarias para la realización de aplicaciones cliente-servidor y proporcionan un conjunto de herramientas con funcionalidades útiles y comunes en este tipo de desarrollos. En la figura 2 podemos ver el logotipo de Java.

2.4.2 IBM Rational Software Architect (RSA)

Figura 3 - Logotipo IBM RSA

Es una herramienta para el desarrollo de aplicaciones Web de IBM, ideal para trabajar con WebSphere. La herramienta está basada en Eclipse, por lo que es fácil de adaptarse a ella si se está acostumbrado a Eclipse. La versión utilizada será la 7.5 que es la versión que apareció para trabajar con WAS 7.0. Si no se dispone de licencia existen licencias de trial de IBM que pueden ir renovándose siempre y cuando no se utilicen con fines empresariales. Para más información ver la página de IBM http://www.ibm.com/developerworks/downloads/r/architect/index.html En la figura 3 se muestra el logotipo de IBM RSA.

2.4.3 IBM WebSphere

Figura 4 - Logotipo IBM WebSphere

Es el servidor de aplicaciones de IBM. Utilizaremos la versión 7.0. IBM tiene las distribuciones para desarrolladores disponibles para descargar en su WEB sin coste alguno. Se ha seleccionado este servidor de aplicaciones por varios motivos: integra toda la arquitectura escogida para el desarrollo sin tener que implantar soluciones de terceros, es una plataforma estable altamente utilizada, es un sistema conocido para el equipo de desarrollo. En la figura 4 se muestra el logotipo de IBM WAS.

2.4.4 Oracle

Figura 5 - Logotipo Oracle

Como DBMS se ha elegido Oracle al ser una base de datos potente altamente utilizada. Oracle dispone de las Express Edition que es una versión reducida de la base de datos que se puede descargar, instalar y distribuir gratuitamente. Esta es la versión seleccionada para el desarrollo de la aplicación, Oracle XE 11g. Se puede descargar aquí:

http://www.oracle.com/technetwork/database/database-technologies/express-edition/overview/index.html

En la figura 5 se muestra el logotipo de Oracle.

2.4.5 SoapUI

Figura 6 - Logotipo SoapUI

SoapUI es una herramienta gratuita, open source, que permite el testeo de servicios web utilizando una interface gráfica sencilla. Con esta herramienta se realizarán test funcionales sobre los WebServices. Se puede descargar gratuitamente aquí:

http://www.soapui.org/

En la figura 6 se muestra el logotipo de SoapUI

2.4.6 SQL Developer

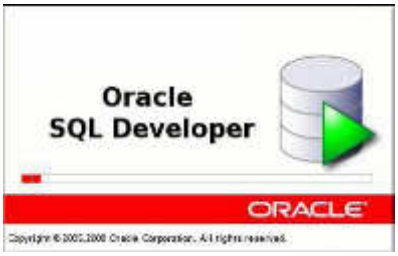


Figura 7 - Logotipo SQL Developer

Oracle SQL Developer es un entorno de desarrollo integrado que simplifica el desarrollo y la administración de una base de datos Oracle. Permite el desarrollo en PL-SQL, la ejecución de sentencias SQL y DDL y ciertas tareas de administración necesarias.

Es un entorno gratuito realizado por Oracle. Se puede descargar aquí:

http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index-097090.html

En la figura 7 se muestra el logotipo de SQL Developer.

2.5 Metodología ágil - SCRUM

Scrum es una metodología ágil de desarrollo de proyectos basada en los estudios sobre nuevas prácticas de producción de Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80.

Básicamente es un modelo de construcción incremental basado en iteraciones y revisiones. Es un modelo adaptable más que predictivo, busca adaptarse a las necesidades en las diferentes iteraciones, lo cual nos lleva a un desarrollo incremental.

Como idea principal en el desarrollo se busca tener un prototipo plenamente funcional bajo mínimos, que en las distintas iteraciones se va a ir incrementando en funcionalidad.

Scrum define una serie de roles, reuniones o *meetings* diarios de planificación de trabajo, revisión… que en este proyecto no tienen mucho sentido al ser desarrollado por una sola persona, no obstante la filosofía general de Scrum si es válida: nos centramos en el desarrollo de un requisito funcional tras otro enteramente y una vez finalizados entramos en otra iteración para ir perfilando los requisitos funcionales, los datos de las pantallas, etc.

Este ha sido el modelo de desarrollo seleccionado para este proyecto.

2.6 Análisis de riesgos

Básicamente los riesgos de un proyecto se pueden clasificar en tres apartados:

- Riesgos funcionales: Son los riesgos asociados a la falta del conocimiento funcional de la aplicación que va a desarrollarse. Es la respuesta a la pregunta “Que debe hacer la aplicación”. Si no se tiene claro que debe hacer la aplicación o se toman decisiones equivocadas el proyecto puede llevarse a cabo enteramente y fracasar ya que puede no ser de utilidad.
- Riesgos tecnológicos: Son los riesgos asociados a la tecnología, la arquitectura y la implementación de la solución. Es la respuesta a la pregunta “Cómo debe hacerse la aplicación”. Se puede fracasar debido a una falta de conocimientos necesarios para la implementación de la aplicación.

- Riesgos en los recursos: Son los riesgos asociados a los recursos necesarios para la realización de la aplicación. Es la respuesta a la pregunta “que necesito para realizar la aplicación”.

Siempre se trabaja con estimaciones: estimaciones en cuanto a las funcionalidades que se van a desarrollar, al coste que va a tener su desarrollo en función de los recursos necesarios y del tiempo.

En este caso existen riesgos funcionales: se va a desarrollar una aplicación para el uso de terceros basándose exclusivamente en supuestos pues no se dispone de ningún tercero dispuesto a implantar la aplicación que pueda colaborar en la toma de requisitos. Es posible que los supuestos o necesidades que se intentan cubrir sean insuficientes en la práctica, o no justifiquen la implantación y mantenimiento del producto.

Por la parte tecnológica se conoce sobradamente la tecnología a utilizar por lo que no se determina ningún riesgo.

Por la parte de recursos se dispone de un solo recurso así que el riesgo que se puede inferir sería en cuanto al tiempo. Es posible que no se cumplan las fechas previstas debido a múltiples causas: fechas no realistas, enfermedad o cualquier problema que paralice al recurso, problemas ajenos como un problema H/W en el equipo de desarrollo…

3 Especificaciones del proyecto

3.1 Descripción general

En este punto se describen los factores, funciones y restricciones del sistema que influyen en el desarrollo de la aplicación. También las características de los usuarios para una mejor compresión de los requisitos.

3.1.1 Descripción de las funciones de la aplicación

Las funciones de la aplicación pueden clasificarse en tres grandes grupos. Veremos que esta distinción en tres grupos es constante en distintos apartados (usuarios, subsistemas…). Esto se debe a que a nivel funcional se diferencian claramente estos tres grupos y, por lo tanto, en los distintos niveles de requisitos y diseño se verán reflejados.

- En el primer grupo se engloban todas las tareas administrativas de la aplicación. Esto es en primer lugar tareas de mantenimiento de aplicaciones. Las aplicaciones son el nivel más básico al que se deben asociar los procesos. Una aplicación contiene procesos, por lo que sería el siguiente nivel de mantenimiento que la administración realiza. Seguidamente un proceso contiene puntos de chequeo y errores. Las alertas estarán asociadas a estos puntos de chequeo, errores o bien a los procesos. Las distintas tareas administrativas se encargarán de gestionar toda esta información, que será nuestro catálogo de datos. A este grupo lo llamaremos “Definición de procesos”
- En el segundo grupo engloba las funciones relativas a las ejecuciones de procesos. Desde distintas aplicaciones enviarán datos a MPROC que deberá procesar. Estos datos básicamente serán el inicio de un proceso, el paso por los distintos puntos de chequeo, la emisión de algún error y la finalización del proceso. Aquí también estaría la generación y gestión de alertas. A este grupo lo llamaremos “Ejecución de procesos”
- En el tercer grupo todas las funciones relativas a la consulta de ejecuciones de procesos, generaciones de informes… a este grupo lo llamaremos “Consulta de procesos”

3.1.1.1 Definición de procesos

La definición de procesos forma el catálogo de datos de MPROC. En este catálogo tendremos las siguientes entidades: Aplicaciones, procesos, punto de chequeo, errores alertas y personas.

- **Aplicaciones**: son el conjunto de aplicaciones que van a ser monitorizadas. Cada aplicación tendrá un conjunto de procesos y una serie de datos que la definen: un nombre o identificador, un responsable...
- **Procesos**: son el objetivo de MPROC, los procesos que van a ser monitorizados. Cada proceso contiene una lista de puntos de chequeo, una lista de errores, una lista de alertas, un conjunto de datos que lo definen tales como nombre, identificador... y un conjunto de datos estadísticos (número de ejecuciones, tiempo medio de ejecución, tiempo máximo...). Los procesos pueden tener asociadas alertas de distintos tipos.
- **Puntos de checheo**: son un conjunto de puntos por los que un proceso puede pasar. Tendremos unos puntos de chequeo que son comunes a todos los procesos, tales como inicio o fin, que indican que un proceso ha iniciado o finalizado su ejecución. Luego tendremos los distintos puntos que son específicos para cada proceso. Cada punto de chequeo indica si el paso por él dentro de la ejecución de un proceso es obligatorio o no. Esto quiere decir que en la ejecución del proceso debe marcar el paso por el punto

de checheo, cosa que si no ocurre puede ser monitorizada y alertada siempre y cuando el proceso tenga asociada el tipo de alerta correspondiente con paso obligatorio. Un paso también puede tener asociada una alerta de aviso de paso. Esta alerta se dispararía cuando un proceso informa de ese paso que contiene la alerta. Esto sirve para alertar de ciertos eventos (por ejemplo, un proceso de pago que supere cierta cantidad de dinero puede disparar un paso opcional de “pago cuantioso“ que puede tener asociada una alerta de forma que un grupo de personas asociadas a esa alerta sean avisados automáticamente cuando se produce un pago de este tipo)

- **Errores**: se definen el conjunto de errores que se desean gestionar desde MPROC. Al dar de alta un proceso siempre se da de alta un error de tipo UNK (*Unknown* o desconocido) de forma automática para recoger los errores de las aplicaciones monitorizadas que no deseen ser inventariados en MPROC. De forma alternativa se pueden inventariar los errores específicos que se desee. Los errores pueden tener asociada una alerta que se dispare en el momento que el error se produzca, o bien se puede asociar una alerta al proceso de forma que alerte siempre que el proceso termine con errores.
- **Alertas**: Las alertas van a ir asociadas a procesos, puntos de chequeo o errores. Pueden ser de diferente tipo, así cada tipo de alerta comprueba una cosa distinta. Las alertas tienen asociadas un conjunto de personas a las que avisar de la forma que sea pertinente. Los tipos de alertas son los siguientes:
    - **Alerta de proceso sin finalizar**: es un tipo de alertas que se asocia a un proceso e indica que el proceso no ha finalizado en un tiempo determinado. El objetivo de esta alerta es alertar de procesos que empiezan y no terminan, es una alerta frente a una ejecución presuntamente fallida.
    - **Alerta de tiempo de proceso**: es un tipo de alerta que se asocia a un proceso e indica que un proceso debe ejecutarse en un momento determinado del día.
    - **Alerta de proceso finalizado con errores**: es un tipo de alerta que se asocia a un proceso o a un error. Alerta de los procesos finalizados con errores de algún tipo.
    - **Alerta de paso obligado**: Es una alerta que se asocia a un proceso e indica que el proceso ha finalizado sin haber registrado el paso por un punto de chequeo definido como obligatorio.
    - **Alerta de aviso de paso**: Es una alerta que se asocia a un punto de chequeo y se dispara cada vez que un proceso informa del paso por ese punto de chequeo.
- **Personas**: Son los receptores de las alertas y se asocian a las alertas. Se definen los datos necesarios para la recepción de mensajes de alerta.

En la figura 8 se muestra en nivel de anidamiento de las distintas entidades que componen el sistema.

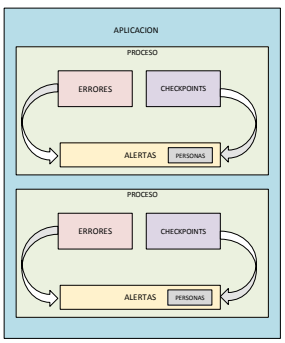


Figura 8 - Anidamiento de conceptos

El flujo de trabajo en este grupo seria el siguiente:

- Se dan de alta las aplicaciones
- Para cada aplicación se dan de alta los procesos
- Para cada proceso se dan de alta los puntos de chequeo
- Para cada proceso se dan de alta los errores
- Para cada proceso se dan de alta las alertas
- Para cada punto de chequeo se dan de alta las alertas
- Para cada error se dan de alta las alertas
- Para cada alerta se dan de alta las personas

Igual que se realiza el alta se realizan las bajas, modificaciones…

3.1.1.2 Ejecución de procesos

Una vez están definidos los procesos MPROC entra en ciclo de producción y empieza a recibir información: datos de ejecución de los distintos procesos que tiene en su catálogo. Esta información determina la planificación de un proceso, el inicio, los distintos pasos y errores y la finalización.

- Planificación de proceso: La aplicación informa a MPROC de la planificación de un proceso. Indica que aplicación es y qué proceso se está planificando. Informa también de una referencia única del proceso para futuras referencias. El proceso queda en estado PENDIENTE.
- Inicio del proceso: La aplicación que ha informado a previamente de la planificación de un proceso indica a MPROC que la ejecución del proceso ha iniciado. La aplicación informa de la aplicación, el proceso y la referencia del proceso. El proceso queda en estado INICIADO.
- Paso por punto de chequeo: la aplicación indica el paso por un punto de chequeo. Para ello indica la aplicación, el proceso, el punto de chequeo y la referencia al proceso. El proceso mantiene el estado INICIADO.
- Error: la aplicación indica que se ha producido un error. Para ello indica la aplicación, el proceso, el error y la referencia al proceso. El proceso mantiene el estado INICIADO.
- Fin del proceso: La aplicación indica a MPROC que la ejecución del proceso ha finalizado. La aplicación informa de la aplicación, el proceso y la referencia del proceso. El proceso queda en estado FIN OK si ha finalizado si errores, o FIN KO si ha finalizado con algún error.

3.1.1.3 Consulta de procesos

Como consulta de procesos se entienden las consultas que se realizan sobre la ejecución de los procesos que están siendo monitorizados. Estas consultas pueden ser realizadas por personal con permisos dentro de la compañía o bien por personal externo a la compañía relacionado con las aplicaciones que han emitido los datos asociados al proceso.

Por otra parte, esta información se puede consultar para saber cuál ha sido el resultado de una ejecución concretamente, o para ver que procesos ha fallado, o bien con carácter estadístico, con lo cual la aplicación cuando se busque información sobre procesos determinados mostrará la información por pantalla mientras que cuando se busque información con fines estadísticos exportará los datos en formato CSV de forma que puedan ser explotados por las herramientas qué se desee.

3.1.2 Características de los usuarios

Las características de los usuarios potenciales de la aplicación van a indicar las necesidades que va a tener la aplicación en cuanto a las interacciones con estos usuarios. En cierta forma van a condicionar el diseño de una interface gráfica. Cuanto más básicos sean sus conocimientos de aplicaciones informáticas más ayudas van a requerir por parte de la interface: ayudas en línea, notas que indiquen que pueden o no hacer o por que no han podido realizar las instrucciones que han solicitado. En cambio, cuanto más avanzados sean los usuarios menos ayudas requieren y necesitan de pantallas más eficientes en cuanto al tiempo que se debe invertir para realizar las acciones que se requieren. Para tener una idea vamos a realizar un estudio previo de los distintos tipos de usuarios que van a interaccionar con MPROC.

Los distintos usuarios de la aplicación se pueden clasificar en 3 grandes grupos:

- Sistemas: son distintas aplicaciones que acceden al sistema MPROC. Estas aplicaciones pueden ser aplicaciones internas de la compañía o aplicaciones externas. También pueden ser otros elementos no humanos que interaccionan con el sistema tales como planificadores. Al ser aplicaciones el nivel de estos usuarios es indiferente, no obstante estas aplicaciones requerirán de desarrollos o adaptaciones para interaccionar con MPROC por lo que los distintos equipos de desarrollo y mantenimiento de estas aplicaciones serán interlocutores de este tipo de usuarios. Al ser equipos de desarrollo de aplicaciones informáticas estos usuarios serian usuarios altamente cualificados.
- Usuarios administradores: son usuarios que se encargarán de las tareas administrativas de MPROC. Estos usuarios pueden tener un conocimiento informático medio o un conocimiento funcional de las aplicaciones que interaccionarán con MPROC. Son usuarios acostumbrados a tratar con aplicaciones informáticas puesto que la responsabilidad que deben ocupar para realizar tareas de mantenimiento que puede destruir información de forma accidental requiere de usuarios con cierto grado de conocimiento de aplicaciones informáticas.
- Usuarios de consulta: con usuarios que se encargarán de realizar consultas en MPROC. Estos usuarios pueden pertenecer a la compañía o a entidades externas que requieran consultar datos sobre la ejecución de procesos relacionados con ellos. Son usuarios con un conocimiento medio aunque pueden representar el grupo con conocimiento más básico de todos. Estos usuarios solo tendrán acceso a funciones de consulta por lo que no pueden eliminar información importante del sistema.

Como conclusión, a nivel general esta aplicación será usada por usuarios con conocimientos avanzados en aplicaciones informáticas. Por ello se recomienda una interface sencilla y rápida, de forma que se pueda realizar las funciones necesarias de forma rápida y directa sin necesidad de pasar por varias pantallas informativas.

3.2 Requisitos

3.2.1 Subsistemas

El sistema MPROC está dividido en varios subsistemas. Esto obedece a una organización funcional de los distintos elementos que lo componen. Simplemente es una organización funcional, no tiene ningún impacto ni en la arquitectura ni en el diseño. La tabla 1 muestra una lista de los subsistemas.

Nombre	Descripción
PROCESOS	Es el sistema que se encarga de gestionar todas las tareas administrativas de gestión de aplicaciones, procesos y demás. También se encarga de recibir y procesar los datos de las distintas aplicaciones que interactúan con MPRO. Este sistema acaba generando las alertas.
ALERTAS	Es el sistema que procesa las alertas generadas por los procesos y realiza los envíos de emails en caso necesario. Este sería el sistema que se podría ampliar para el envío de alertas en otro formato, como SMS.
INFORMES	Es el sistema que se encarga de la realización de informes para fines estadísticos.

Tabla 1 - Subsistemas

3.2.2 Actores

Los actores son los elementos de un sistema que inician las acciones dentro del sistema. Pueden corresponder con usuarios, con aplicaciones o con algunos elementos como un planificador de tareas. Un actor modela un rol jugado por una entidad que interactúa con el sistema pero es externo al sistema.

La Tabla 2 muestra la lista de los actores del sistema. Se incluye una descripción de los mismos.

Nombre	Descripción
Administrador	Representa al usuario administrador, que es el usuario que gestiona el sistema: realiza el mantenimiento de los datos maestros de las aplicaciones, procesos y alertas.
Aplicación	Representa las aplicaciones de la compañía que registran datos sobre ejecuciones de procesos en el sistema. Es un usuario de tipo “sistema”.
Aplicación emisora	Representa a las aplicaciones externas a la compañía que emiten datos o generan eventos que desencadenan procesos en los sistemas de la compañía. Estas aplicaciones pueden consultar datos relativos a los procesos que han generado.
Planificador	Representa a un actor que dispara procesos en el sistema de forma planificada. Estos procesos servirán para realizar tareas de mantenimiento (eliminación de datos obsoletos) o realizar análisis de los procesos que se están gestionando.
Usuario emisor	Representa a un usuario humano de la aplicación emisora.
Usuario supervisor	Representa a los usuarios de la compañía que acceden a consultar datos de procesos.

Tabla 2 - Actores

La Figura 9 muestra los actores y las interacciones con los subsistemas en formato UML.

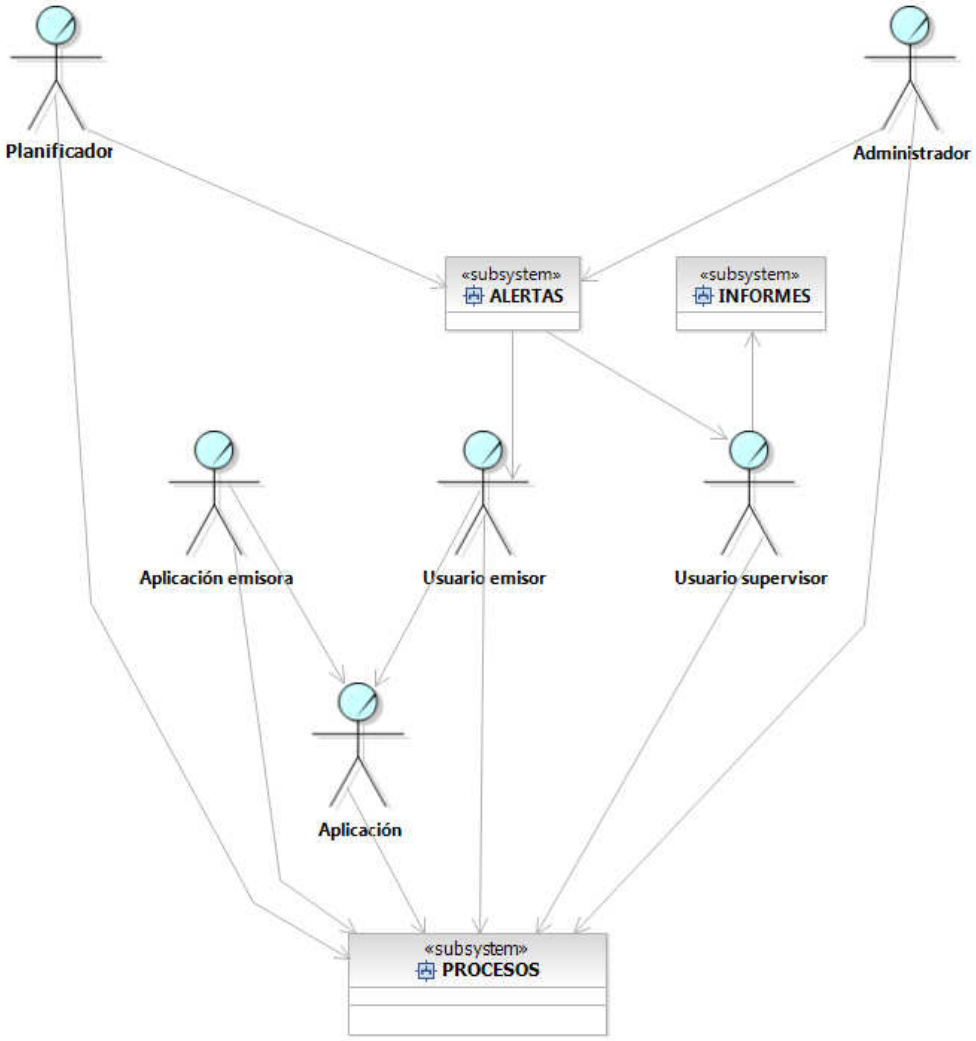


Figura 9 - UML Actores

3.2.3 Requisitos funcionales de la aplicación

En este apartado se listan los requisitos funcionales que tiene la aplicación. Se realiza un inventario de estos requisitos, se describen explicando brevemente los puntos más importantes y se especifican las relaciones que tienen con otros requisitos funcionales. Estas relaciones marcarán las interacciones entre los requisitos de forma que frente a una modificación de un requisito funcional se puede visualizar el alcance que puede tener. Para ello disponemos de dos tipos de relaciones: inclusión y ampliación. Una inclusión de R2 sobre R1 indica que siempre que se ejecute R1 se ejecutará también R2, es una relación no condicional y se expresaría así: R1 → R2. Una ampliación de R2 sobre R1 indica que cuando se ejecute R1 puede ejecutarse R2. Es una relación condicional y se expresa así: R1 ← R2

El objetivo de los requisitos funcionales y su modelado en forma de casos de uso es delimitar el alcance del proyecto. Se va a especificar qué relaciones tiene el sistema con otros sistemas, que relaciones tienen los diferentes elementos y que deben hacer, pero no como debe hacerlo ni por qué. Por esto los requisitos funcionales se alejan de conceptos técnicos. Estos modelos se realizan de forma que sean legibles para personas ajenas al sector de la informática.

Los grupos funcionales son una forma de agrupar los requisitos funcionales atendiendo a su funcionalidad y relaciones. De esta forma los requisitos funcionales que está fuertemente interrelacionados ya que tratan sobre un mismo concepto están agrupados. A este nivel se realizan los diagramas de casos de uso, no obstante si un caso de uso tiene relación con otro caso de uso de otro grupo funcional se incluye también en los diagramas de casos de uso aunque sean de otro grupo funcional.

Subsistema de procesos

- Tabla 3: GF-01 Administrar procesos - GF-01.1 Administrar aplicaciones

Nombre	Descripción
RF-01 Alta de aplicación	Requisito funcional que consiste en la realización del alta de una aplicación en el sistema. Muestra el formulario de alta, recoge los datos, los valida, comprueba que no exista otra aplicación con el mismo identificador y procede con el alta.
RF-02 Consulta de aplicación	Requisito funcional que consiste en la búsqueda de una aplicación en función de su identificador. En caso de no existir la aplicación muestra el error correspondiente, si la encuentra muestra los datos de la aplicación por pantalla. Al obtener los datos de la aplicación obtiene también un listado de los procesos de la aplicación que se muestra por pantalla, por lo que tiene una relación de inclusión con el *RF-10 Listar procesos*. Por otra parte para poder seleccionar la aplicación a consultar se debe mostrar previamente un listado de las aplicaciones del sistema, por eso este requisito amplia al *RF-05 Listado de aplicaciones*
RF-03 Modificación de aplicación	Requisito funcional que realiza la modificación de una aplicación. Muestra una pantalla con los datos modificables de una aplicación, recoge los datos, los valida y realiza la modificación. Para poder modificar una aplicación esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-02 Consulta de aplicación*
RF-04 Baja de aplicación	Requisito funcional que realiza la eliminación de una aplicación. Al eliminar la aplicación se eliminan todos sus datos asociados, tales como procesos, alertas, errores, puntos de checheo… Estos datos están relacionados en la base de datos a través del modelo de entidad-relación por lo que la eliminación de los datos asociados mediante claves primarias – claves foráneas es gestionada desde la base de datos al realizar un borrado en cascada. Para poder eliminar una aplicación esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-02 Consulta de aplicación*
RF-05 Listado de aplicaciones	Requisito funcional que se encarga de listar las aplicaciones dadas de alta en el sistema.

Tabla 3 - GF-01.1 Administrar aplicaciones

- Tabla 4: GF-01 Administrar procesos - GF-01.2 Administrar procesos

Nombre	Descripción
RF-06 Alta proceso	Requisito funcionar encargado de dar de alta un proceso para una aplicación en el sistema. Muestra el formulario de alta con un resumen de los datos de la aplicación, recoge los datos, los valida, comprueba que no exista otro proceso con el mismo identificador y procede con el alta. Con el alta de un proceso se dan de alta automáticamente dos checkpoints (INICIO y FIN) y un error (UNK). En caso de producirse un error lo muestra por pantalla. Este requisito funcional se realiza dentro del ámbito de una aplicación, por eso debe ampliar al *RF-02 Consulta de aplicación* ya que para realizar el alta de un proceso primero debe buscarse la aplicación sobre la que se va a crear el proceso.
RF-07 Consulta proceso	Requisito funcional que consiste en la búsqueda de un proceso en función de su identificador y de la aplicación a la que pertenece. En caso de no existir el proceso muestra el error correspondiente, si la encuentra muestra un resumen de la aplicación y los datos del proceso por pantalla. Al obtener los datos del proceso obtiene también un listado de los puntos de chequeo y de los errores asociados. Por ello tiene una relación de inclusión con el *RF-15 Listado de checkpoints* y *RF-20 Listado de errores*. Para poder consultar un proceso es necesario haberlo localizado previamente, por lo que este requisito tiene una relación de ampliación del *RF-10 Listar procesos*
RF-08 Modificación de proceso	Requisito funcional que realiza la modificación de un proceso. Muestra una pantalla con un resumen de la aplicación y los datos modificables del proceso, recoge los datos, los valida y realiza la modificación. Para poder modificar un proceso este debe haber sido seleccionado previamente, por lo tanto este requisito amplia al *RF-07 Consulta proceso*
RF-09 Baja proceso	Requisito funcional que realiza la eliminación de un proceso. Al eliminar el proceso se eliminan todos sus datos asociados, tales como alertas, errores, puntos de checheo… Estos datos están relacionados en la base de datos a través del modelo de entidad-relación por lo que la eliminación de los datos asociados mediante claves primarias – claves foráneas es gestionada desde la base de datos al realizar un borrado en cascada. Para poder eliminar un proceso este debe haber sido seleccionado previamente, por lo tanto este requisito amplia al *RF-07 Consulta proceso*
RF-10 Listar procesos	Requisito funcional que se encarga de listar las aplicaciones dadas de alta en una aplicación.

Tabla 4 - GF-01.2 Administrar procesos

- Tabla 5: GF-01 Administrar procesos - GF-01.3 Administrar checkpoints

Nombre	Descripción
RF-11 Alta de checkpoint	Requisito funcionar encargado de dar de alta un punto de chequeo para un proceso en el sistema. Muestra el formulario de alta con un resumen de los datos de la aplicación y el proceso, recoge los datos, los valida, comprueba que no exista otro *checkpoint* con el mismo identificador y procede con el alta. En caso de producirse un error lo muestra por pantalla. Este requisito funcional se realiza dentro del ámbito de un proceso, por eso debe ampliar al *RF-07 Consulta proceso* ya que para realizar el alta de un *checkpoint* primero debe buscarse el proceso sobre la que se va a realizar el alta.
RF-12 Consulta de *checkpoint*	Requisito funcional que consiste en la búsqueda de un *checkpoint* en función de su identificador y de la aplicación y proceso al que pertenece. En caso de no existir el *checkpoint* muestra el error correspondiente, si lo encuentra muestra un resumen de la aplicación, el proceso y los datos del *checkpoint* por pantalla. Al obtener los datos del *checkpoint* obtiene también un listado de las alertas asociadas. Por ello tiene una relación de inclusión con el *RF-33 Listado de alertas.* Para poder consultar un *checkpoint* es necesario haberlo localizado previamente, por lo que este requisito tiene una relación de ampliación del *RF-15 Listado de checkpoints*
RF-13 Modificación de *checkpoint*	Requisito funcional que realiza la modificación de un *checkpoint*. Muestra una pantalla con un resumen de la aplicación, el proceso y los datos modificables del *checkpoint*, recoge los datos, los valida y realiza la modificación. Para poder modificar un *checkpoint* este debe haber sido seleccionado previamente, por lo tanto este requisito amplia al *RF-12 Consulta de checkpoint*
RF-14 Baja de *checkpoint*	Requisito funcional que realiza la eliminación de un *checkpoint*. Al eliminar el *checkpoint* se eliminan todos sus datos asociados, tales como alertas, ejecución de puntos de checheo… Estos datos están relacionados en la base de datos a través del modelo de entidad-relación por lo que la eliminación de los datos asociados mediante claves primarias – claves foráneas es gestionada desde la base de datos al realizar un borrado en cascada. Para poder eliminar un *checkpoint* este debe haber sido seleccionado previamente, por lo tanto este requisito amplia al *RF-12 Consulta de checkpoint*
RF-15 Listado de checkpoints	Requisito funcional que se encarga de listar los checkpoints dados de alta en un proceso.

Tabla 5 - GF-01.3 Administrar checkpoints

- Tabla 6: GF-01 Administrar procesos - GF-01.4 Administrar errores

Nombre	Descripción
RF-16 Alta de error	Requisito funcionar encargado de dar de alta un error para un proceso en el sistema. Muestra el formulario de alta con un resumen de los datos de la aplicación y el proceso, recoge los datos, los valida, comprueba que no exista otro error con el mismo identificador y procede con el alta. En caso de producirse un error lo muestra por pantalla. Este requisito funcional se realiza dentro del ámbito de un proceso, por eso debe ampliar al *RF-07 Consulta proceso* ya que para realizar el alta de un error primero debe buscarse el proceso sobre la que se va a realizar el alta.
RF-17 Consulta de error	Requisito funcional que consiste en la búsqueda de un error en función de su identificador y de la aplicación y proceso al que pertenece. En caso de no existir el error solicitado muestra el error correspondiente, si lo encuentra muestra un resumen de la aplicación, el proceso y los datos del error por pantalla. Al obtener los datos del error obtiene también un listado de las alertas asociadas. Por ello tiene una relación de inclusión con el *RF-33 Listado de alertas.* Para poder consultar un error es necesario haberlo localizado previamente, por lo que este requisito tiene una relación de ampliación del *RF-20 Listado de errores*
RF-18 Modificación de error	Requisito funcional que realiza la modificación de un error. Muestra una pantalla con un resumen de la aplicación, el proceso y los datos modificables del error, recoge los datos, los valida y realiza la modificación. Para poder modificar un error este debe haber sido seleccionado previamente, por lo tanto este requisito amplia al *RF-17 Consulta de error*
RF-19 Baja de error	Requisito funcional que realiza la eliminación de un error. Al eliminar el error se eliminan todos sus datos asociados, tales como alertas, ejecución de errores… Estos datos están relacionados en la base de datos a través del modelo de entidad-relación por lo que la eliminación de los datos asociados mediante claves primarias – claves foráneas es gestionada desde la base de datos al realizar un borrado en cascada. Para poder eliminar un error este debe haber sido seleccionado previamente, por lo tanto este requisito amplia al *RF-17 Consulta de error*
RF-20 Listado de errores	Requisito funcional que se encarga de listar los errores dados de alta en un proceso.

Tabla 6 - GF-01.4 Administrar errores

- Tabla 7: GF-02 Introducir datos de procesos

Nombre	Descripción
RF-21 Crear ejecución de proceso	Requisito funcional que realiza la inserción de datos de una ejecución de un proceso. Desde una aplicación externa se reciben los datos del proceso, se verifica que exista la aplicación, el proceso, que el proceso pertenezca a la aplicación. También se verifica que la

	referencia indicada sea única para ese aplicación-proceso-emisor. Si no es así se comunica un error a la aplicación emisora, si los datos son correctos se inserta la ejecución del proceso en estado INICIAL.
RF-22 Informar *checkpoint*	Requisito funcional que realiza la inserción de un *checkpoint*. Se comprueba que exista la aplicación, el proceso y el *checkpoint*, que existan datos de la ejecución de un proceso con la referencia que indican para una aplicación-proceso-emisor. Si no se superan las validaciones se informa al emisor, si se superan se inserta el *checkpoint* y se asocia la ejecución del proceso. Si el *checkpoint* insertado es ***INCIO*** se cambia el estado del proceso a ***INICIADO***, si es Fin se cambia a ***FIN OK*** si no se han producido errores o a ***FIN KO*** si se han producido. Para poder insertar un *checkpoint* debe haberse insertado previamente una ejecución de proceso, por eso este requisito tiene una relación de ampliación con *RF-21 Crear ejecución de proceso*
RF-23 Informar error	Requisito funcional que realiza la inserción de un error. Se comprueba que exista la aplicación, el proceso y el error, que existan datos de la ejecución de un proceso con la referencia que indican para una aplicación-proceso-emisor. Si no se superan las validaciones se informa al emisor, si se superan se inserta el error y se asocia la ejecución del proceso. Para poder insertar un error debe haberse insertado previamente una ejecución de proceso, por eso este requisito tiene una relación de ampliación con *RF-21 Crear ejecución de proceso*
RF-24 Adjuntar fichero	Requisito funcional que se encarga de la recepción de ficheros adjuntos a una ejecución. Se comprueba que exista la aplicación, el proceso y que exista una ejecución de proceso con la referencia informada para esa aplicación-proceso-emisor. Si no se superan las validaciones se informa al emisor, si se superan se insertan los ficheros adjuntos para su posterior consulta. Para poder adjuntar ficheros debe haberse insertado previamente una ejecución de proceso, por eso este requisito tiene una relación de ampliación con *RF-21 Crear ejecución de proceso*

Tabla 7 - GF-02 Introducir datos de procesos

- Tabla 8: GF-03 Explotar datos de procesos

Nombre	Descripción
RF-25 Listar procesos	Requisito funcional encargado de listar la ejecución de procesos que se han registrado en el sistema. Mediante un formulario con un conjunto de filtros los usuarios pueden realizar búsquedas de las ejecuciones de procesos mostrándole en un listado los distintos procesos que cumplen con los criterios de búsqueda.

RF-26 Ver detalle de proceso	Requisito funcional encargado de realizar una consulta de una ejecución de proceso a partir de su identificador. Muestra los datos de ejecución del proceso así como los *checkpoint* y errores registrados. Para ver una ejecución de proceso previamente se ha tenido que buscar, por lo que este requisito tiene una relación de ampliación con *RF-25 Listar procesos*
RF-27 Consultar ficheros	Requisito funcional encargado de realizar una consulta de los ficheros adjuntos de una ejecución de proceso. Para poder consultar los ficheros de un proceso previamente se ha tenido que ver el detalle de un proceso ya que la consulta de sus adjuntos será una opción del detalle de fichero. Por lo tanto este requisito tiene una relación de ampliación con el *RF-26 Ver detalle de proceso*

Tabla 8 - GF-03 Explotar datos de procesos

- Tabla 9: GF-04 Mantener datos de procesos

Nombre	Descripción
RF-28 Purgar procesos no vigentes	Requisito funcional encargado de la eliminación de los datos obsoletos de ejecución de procesos. En el catálogo de procesos se indica los días de permanencia en el sistema y en la ejecución de un proceso se indica la fecha de ejecución. A partir de la fecha y del número de días de permanencia se purgan los datos de ejecución de procesos.

Tabla 9 - GF-04 Mantener datos de procesos

Subsistema de alertas

- Tabla 10: GF-05 Administrar alertas

Nombre	Descripción
RF-30 Consulta de alerta	Requisito funcional que consiste en la búsqueda de una alerta en función de su identificador. En caso de no existir la aleta solicitada muestra el error correspondiente, si la encuentra muestra un resumen de la aplicación, el proceso y los datos de la alerta por pantalla. Al obtener los datos de la alerta obtiene también un listado de las personas asociadas. Por ello tiene una relación de inclusión con el *RF-37 Listado de destinatarios.* Para poder consultar una alerta es necesario haberla localizado previamente, por lo que este requisito tiene una relación de ampliación del *RF-33 Listado de alertas*
RF-31 Modificación de alerta	Requisito funcional que realiza la modificación de una alerta. Muestra una pantalla con un resumen de la aplicación, el proceso y los datos modificables de la alerta, recoge los datos, los valida y realiza la modificación. Para poder modificar una alerta esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-30 Consulta de alerta*
RF-32 Baja de alerta	Requisito funcional que realiza la eliminación de una alerta. También elimina todos los datos asociados a la

	alerta tales como ejecuciones de alertas, emails enviados… Realiza este borrado de datos asociados mediante un borrado en cascada en la base de datos. Para poder eliminar una alerta esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-30 Consulta de alerta*
RF-33 Listado de alertas	Requisito funcional que realiza un listado de las alertas de una aplicación-proceso, error o *checkpoint*.
RF-34 Alta de alerta	Requisito funcionar encargado de dar de alta una alerta. Muestra el formulario de alta con un resumen de los datos de la aplicación y el proceso, recoge los datos, los valida, comprueba que no exista otra alerta con el mismo identificador y procede con el alta. En caso de producirse un error lo muestra por pantalla.
RF-35 Añadir destinatario	Requisito funcional que se encarga de dar de alta un destinatario para la alerta seleccionada y asociarlo a ella. Para poder dar de alta un destinatario a una alerta esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-30 Consulta de alerta*
RF-36 Eliminar destinatario	Requisito funcional que se encarga de dar de baja un destinatario para la alerta seleccionada y asociarlo a ella. Para poder dar de baja un destinatario a una alerta esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-30 Consulta de alerta*
RF-37 Listado de destinatarios	Requisito funcional que se encarga de obtener un listado de destinatarios para la alerta. Para poder obtener un listado de destinatarios de una alerta esta debe haber sido seleccionada previamente, por lo tanto este requisito amplia al *RF-30 Consulta de alerta*

Tabla 10 - GF-05 Administrar alertas

- Tabla 11: GF-06 Operación de alertas

Nombre	Descripción
RF-38 Generar alerta en *runtime*	Requisito funcional que se encarga de la generación de alertas en tiempo de ejecución. Estas alertas se generan en el momento de insertar un *checkpoint* o un error, en caso de que estos tengan una alerta asociada, o cuando un proceso finaliza su ejecución, si ha excedido el tiempo de ejecución o ha finalizado con errores y tiene alertas asociadas.
RF-39 Generar alerta en batch	Requisito funcional que se encarga de generar las alertas en batch, es decir, alertas que no se pueden generar en tiempo de ejecución. Las alertas que se producen cuando un proceso lleva cierto tiempo sin finalizar son alertas que no dependen de un evento que las dispare, sino de un análisis periódico de los procesos que no han finalizado.
RF-40 Procesado de alertas	Requisito funcional que se encarga de procesar todas las alertas que se han generado y enviar los mensajes correspondientes a los destinatarios indicando lo que ha ocurrido en el sistema. Un proceso batch examina

	periódicamente la pila de alertas pendientes de procesar y las va procesando, para cada alerta busca los destinatarios y genera los mensajes.

Tabla 11 - GF-06 Operación de alertas

- Tabla 12: GF-07 Mantener datos de alertas

Nombre	Descripción
RF-41 Purgar alertas	Requisito funcional encargado de la eliminación de los datos obsoletos de alertas. Las alertas se mantienen en el sistema un número determinado de días, este requisito funcional se encarga de buscar las alertas que llevan más de esos días en el sistema y las elimina.

Tabla 12 - GF-07 Mantener datos de alertas

Subsistema de informes

- Tabla 13: GF-08 Generar informes

Nombre	Descripción
RF-42 Generar informes	Requisito funcional encargado de la exportación de datos para fines estadísticos. Mediante un formulario el usuario filtra los datos que quiere exportar. Esos datos serán exportados en formato CSV.

Tabla 13 - GF-08 Generar informes

3.2.4 Casos de uso

- Figura 10: GF-01 Administrar procesos - GF-01.1 Administrar aplicaciones

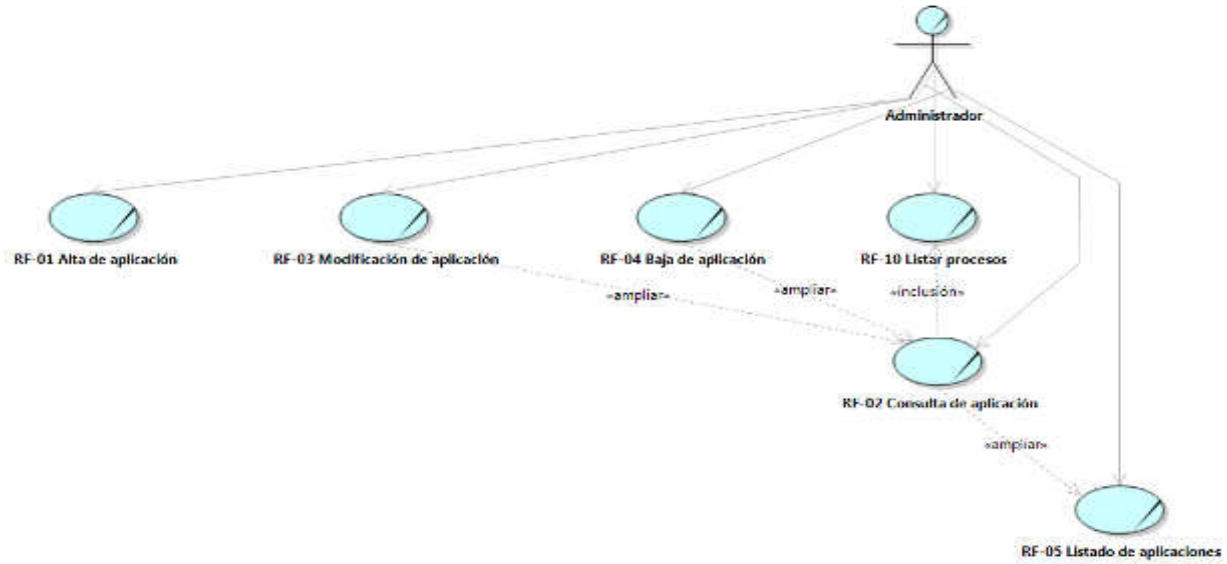


Figura 10 - UML Use Case - GF-01.1 Administrar aplicaciones

- Figura 11: GF-01 Administrar procesos - GF-01.2 Administrar procesos

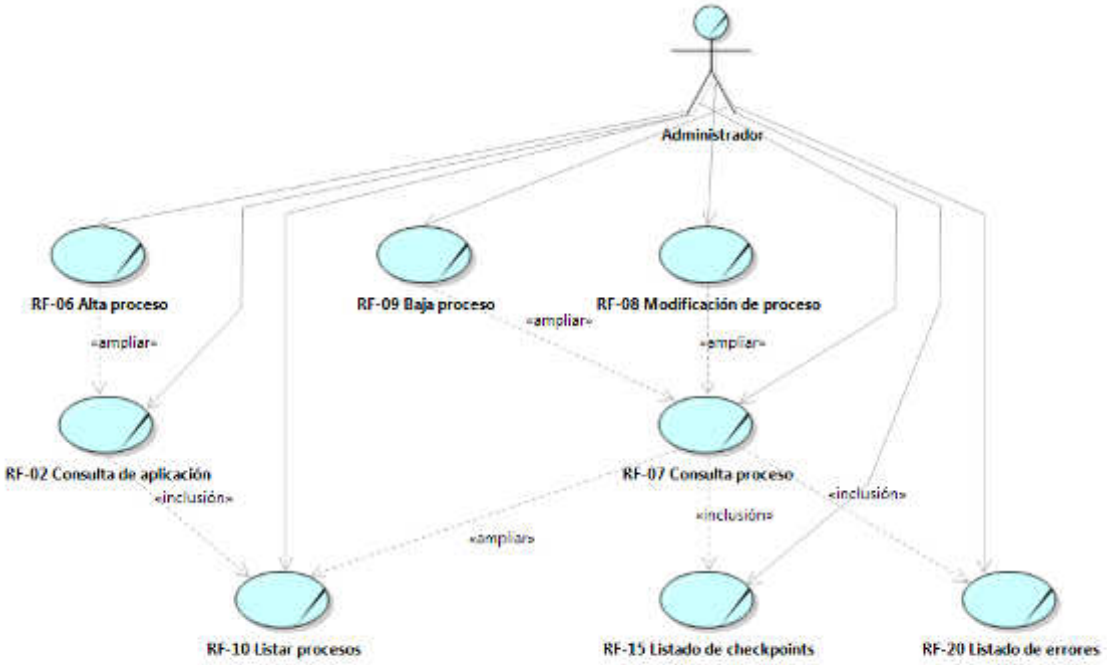


Figura 11 - UML Use Case - GF-01.2 Administrar procesos

- Figura 12: GF-01 Administrar procesos – GF-01.3 Administrar *CheckPoints*

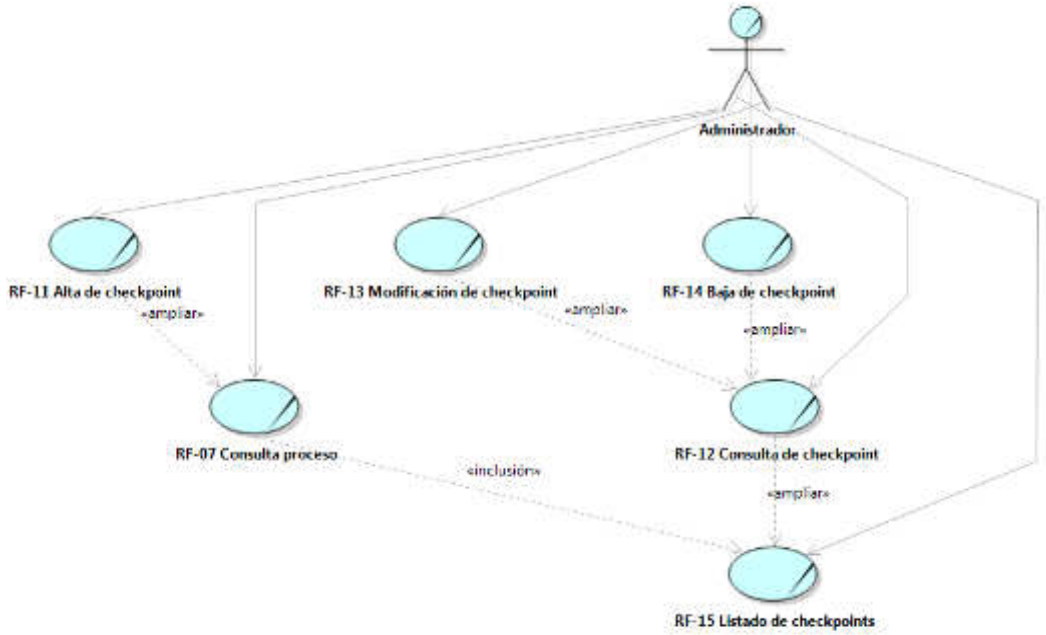


Figura 12 - UML Use Case - GF-01.3 Administrar CheckPoints

- Figura 13: GF-01 Administrar procesos - GF-01.4 Administrar errores

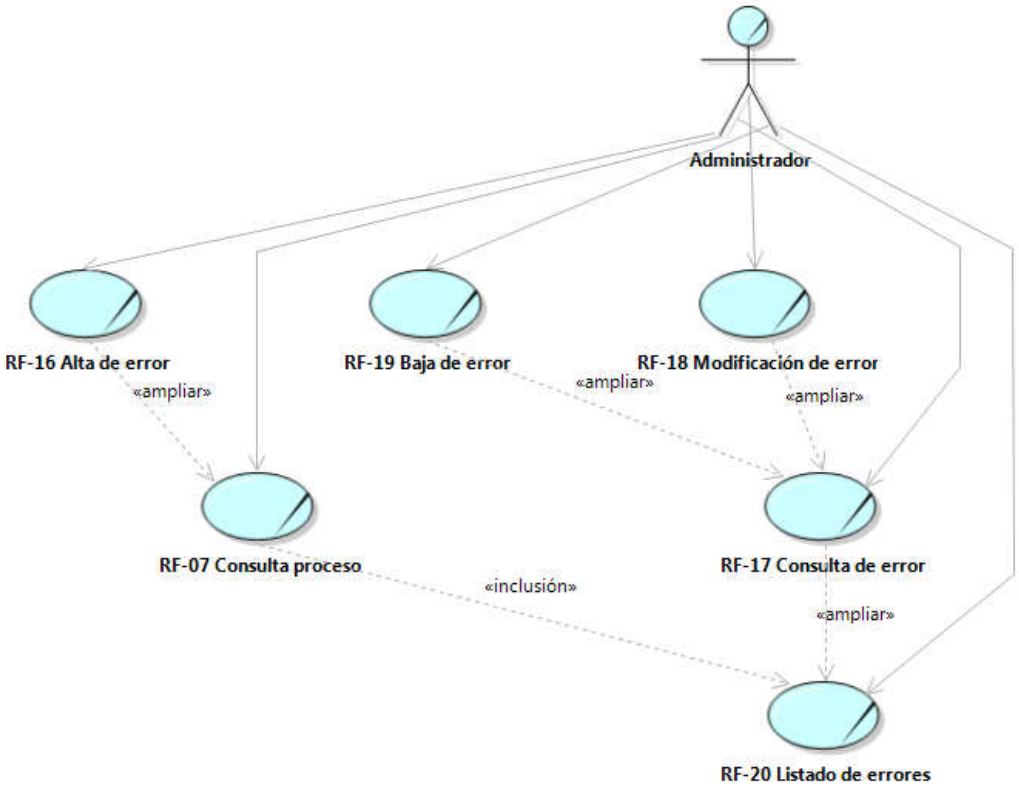


Figura 13 - UML Use Case - GF-01.4 Administrar errores

- Figura 14: GF-02 Introducir datos de procesos

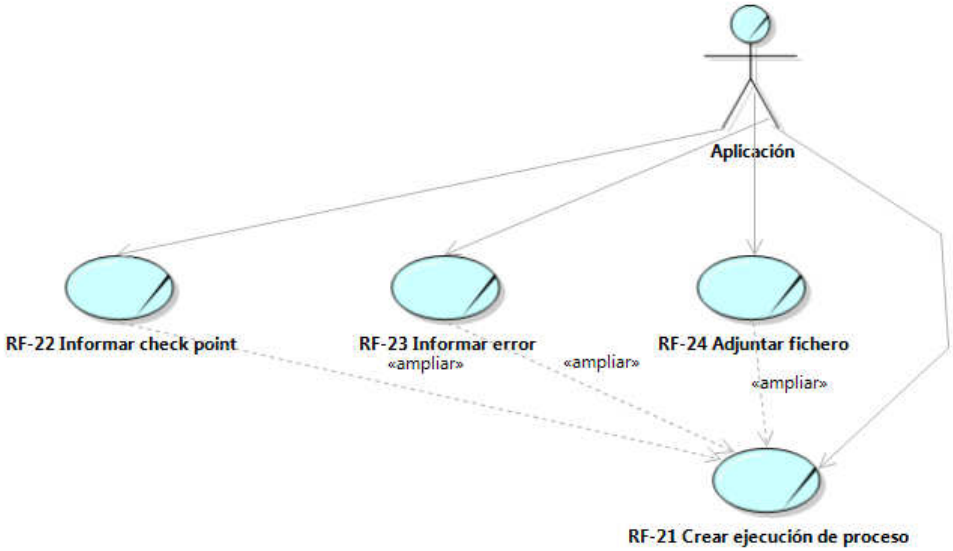


Figura 14 - UML Use Case - GF-02 Introducir datos de procesos

- Figura 15: GF-03 Explotar datos de procesos

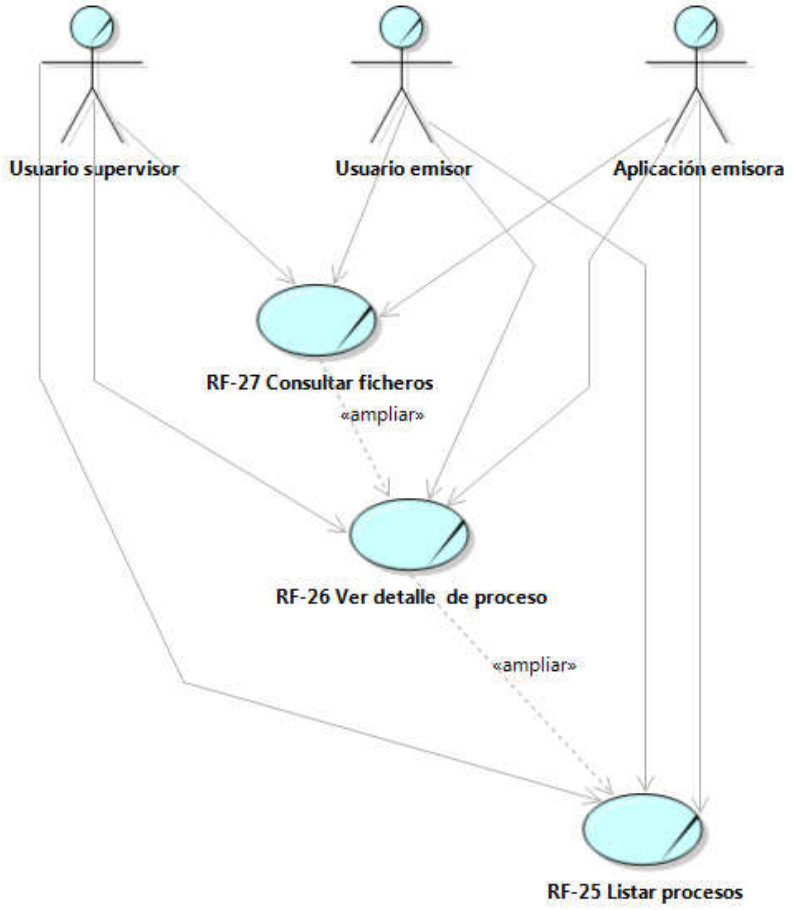


Figura 15 - UML Use Case - GF-03 Explotar datos de procesos

- Figura 16: GF-04 Mantener datos de procesos

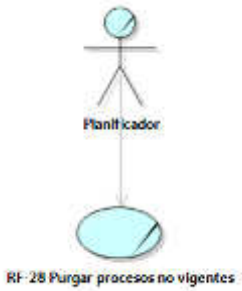


Figura 16 - UML Use Case - GF-04 Mantener datos de procesos

- Figura 17: GF-05 Administrar alertas

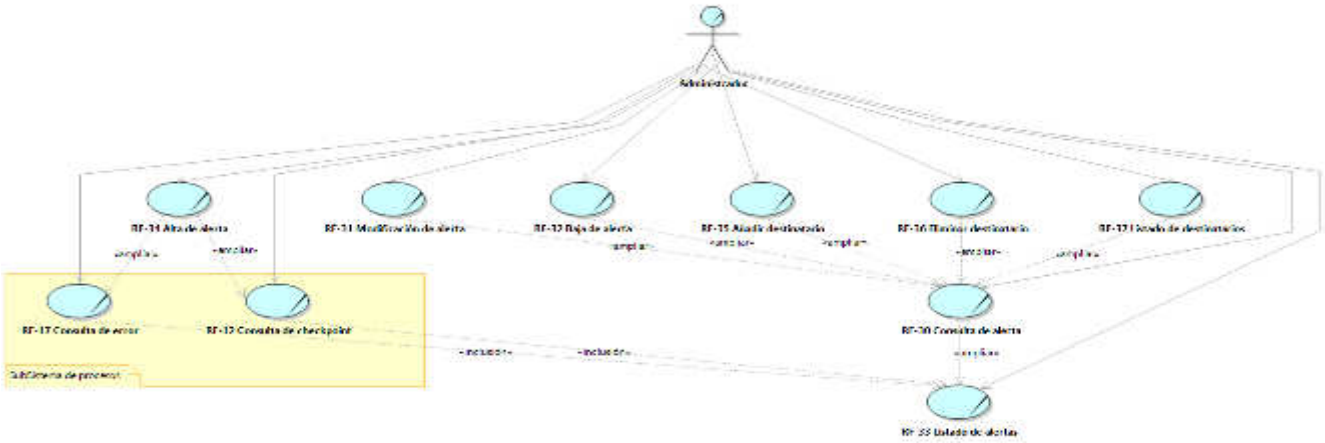


Figura 17 - UML Use Case - GF-05 Administrar alertas

- Figura 18: GF-06 Operación de alertas

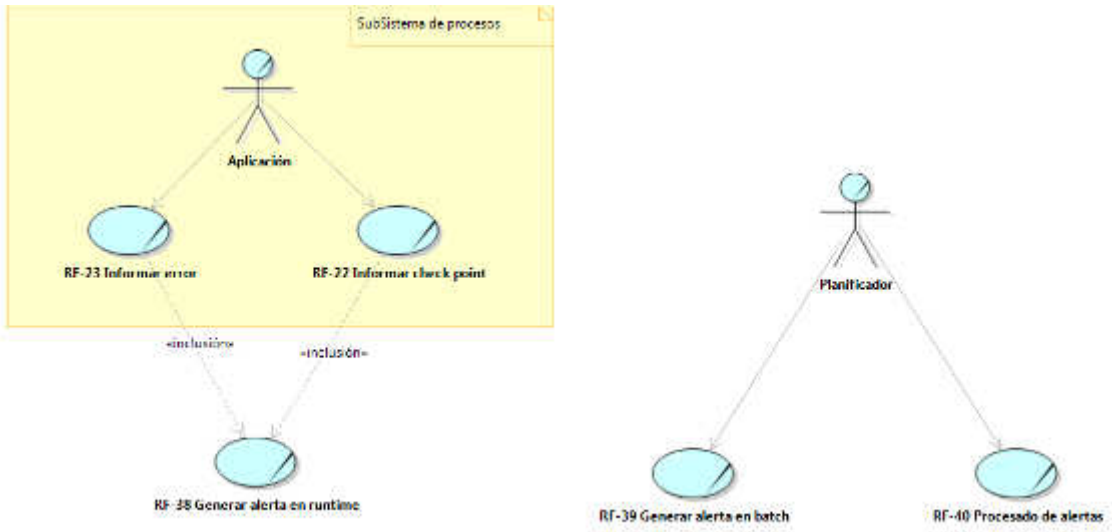


Figura 18 - UML Use Case - GF-06 Operación de alertas

- Figura 19: GF-07 Mantener datos de alertas

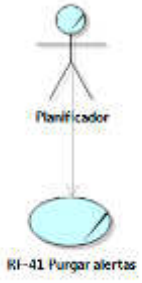


Figura 19 - UML Use Case - GF-07 Mantener datos de alertas

- Figura 20: GF-08 Generar informes

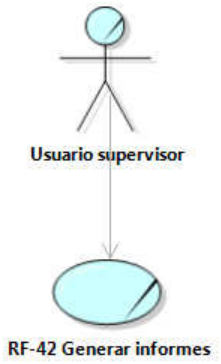


Figura 20 - UML Use Case - GF-08 Generar informes

3.2.5 Definición del plan de pruebas

- Pruebas unitarias: Son pruebas que se realizan durante el desarrollo para probar las distintas piezas de software de forma unitaria.
- Pruebas funcionales: Son pruebas que se realizan para evaluar las distintas funcionalidades que ofrece la aplicación. Se deben probar los casos de éxito como mínimo (que la aplicación haga lo que se espera que haga en el caso correcto)
- Pruebas de rendimiento:
    - Pruebas de carga: son unas pruebas que se llevan a cabo para evaluar la respuesta del sistema (tiempo de respuesta) en una situación de rendimiento real (una simulación de usuarios / nº de transacciones esperados en un entorno productivo). El objetivo es evaluar si la aplicación mantiene unos tiempos de respuesta aceptables en un entorno real.
    - Pruebas de resistencia: son unas pruebas que se realizan para evaluar si bajo unos parámetros de carga preestablecidos la aplicación mantiene los índices de respuesta durante un periodo de tiempo prolongado. El objetivo es evaluar que la aplicación es estable en el tiempo, que no degrada el rendimiento.
    - Pruebas de escalabilidad: las pruebas de escalabilidad son unas pruebas que se realizan para evaluar la capacidad del sistema frente a un aumento de escala (número de usuarios o transacciones)
- Pruebas exploratorias: Son pruebas que se realizan sobre la aplicación sin un objetivo concreto. Se introducen distintos valores en los campos de entrada de datos para evaluar si la aplicación responde correctamente (poner letras en campos numéricos…)

Para poder realizar pruebas de rendimiento se requiere un entorno igual al entorno productivo y alguna herramienta que permita simular una cantidad determinada de usuarios con un número de transacciones por usuario realizando tareas en la aplicación. La herramienta también debe realizar mediciones en los tiempos de respuesta. Por otro lado se realizan monitorizaciones del sistema (uso de memoria, CPU, uso de la red y similares) y de los distintos elementos que forman la aplicación (servidor de base de datos, servidor de aplicaciones…)

Como herramienta utilizada para estas pruebas es destacable HP LoadRunner http://en.wikipedia.org/wiki/HP_LoadRunner que realiza gráficas y se integra perfectamente con HP Quality Center http://en.wikipedia.org/wiki/HP_Quality_Center

Lamentablemente estas herramientas son licenciadas y tienen un coste elevado. Por otra parte tampoco se dispone de un entorno estable para poder realizar este tipo de pruebas. Por lo tanto, valga como reseña este apunte, pero no se van a realizar pruebas de rendimiento. La intención es dejar constancia de la importancia de estas pruebas para poder plantearse poner un sistema en un entorno productivo y que en este caso no se han realizado estas pruebas.

Las pruebas unitarias se realizarán en tiempo de desarrollo de la forma que el desarrollador considere oportuno. Una técnica sencilla para realizar pruebas unitarias en un entorno fuertemente dependiente del servidor de aplicaciones y que, por lo tanto, no permite el uso de herramientas como JUNIT ya que estas se ejecutan fuera del servidor de aplicaciones, es realizarlas a través de un Servlet o WebService creado para dicho propósito.

Las pruebas funcionales se realizarán en el caso de éxito y en los casos de error que parezca evidentes (por ejemplo: control de duplicados, se comprueba que no permita dar de alta dos aplicaciones con los mismos identificadores y que el error que da sea controlado).

Las pruebas de exploración se realizarán una vez la aplicación esté terminada y las pruebas funcionales hayan sido superadas.

3.3 Arquitectura

El propósito de este apartado es definir y describir los diferentes elementos arquitectónicos que van a utilizarse para el diseño y desarrollo de la aplicación. Se realizará una descripción de los distintos modelos de arquitectura del software que se van a implementar y se definirá cuál es su funcionamiento en términos generales sin entrar en el detalle de la implementación de cada uno de estos modelos. La figura 21 muestra distintos *frameworks* por orden de aparición.

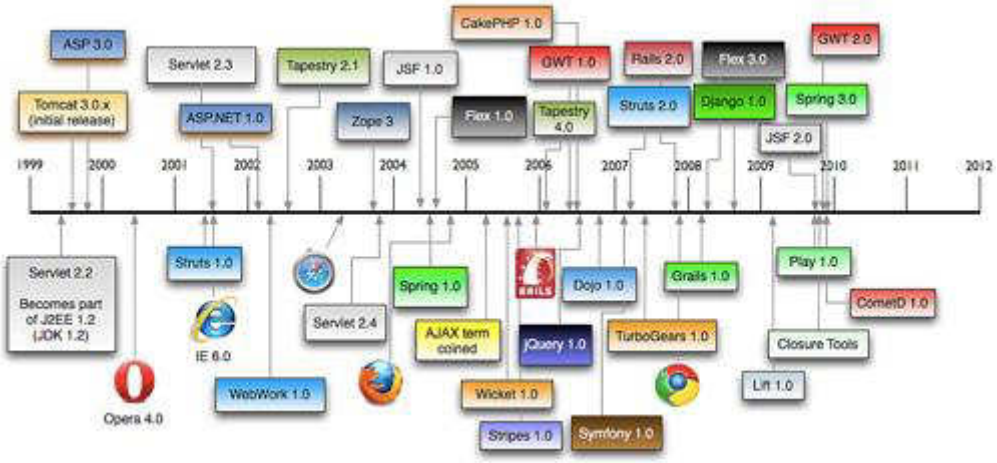


Figura 21 - Java Frameworks

3.3.1 Patrón de Arquitectura de Software

El patrón elegido para la arquitectura de la aplicación es conocido como *arquitectura a tres capas*. Para entender esto primero vamos a explicar que es un modelo de capas.

Cuando se habla en arquitectura de aplicaciones de un modelo de capas implica que la aplicación está diseñada a diferentes niveles de forma que cada nivel desconoce completamente la implementación del nivel inmediatamente inferior. Los componentes de un nivel solo pueden hacer referencia a componentes del nivel inmediatamente inferior. Un nivel no puede referenciar al nivel superior ni a niveles más inferiores que el inmediatamente inferior. Esto es una organización que reduce y organiza las dependencias de forma que simplifica la complejidad de los desarrollos a la vez que mantiene los componentes fuertemente desacoplados de forma que puedan ser reutilizados.

Aplicando el modelo de capas a un modelo de tres capas se obtienen los tres niveles que conforman una aplicación: nivel de presentación, nivel de negocio, nivel de persistencia. La figura 22 muestra un modelo de tres capas.

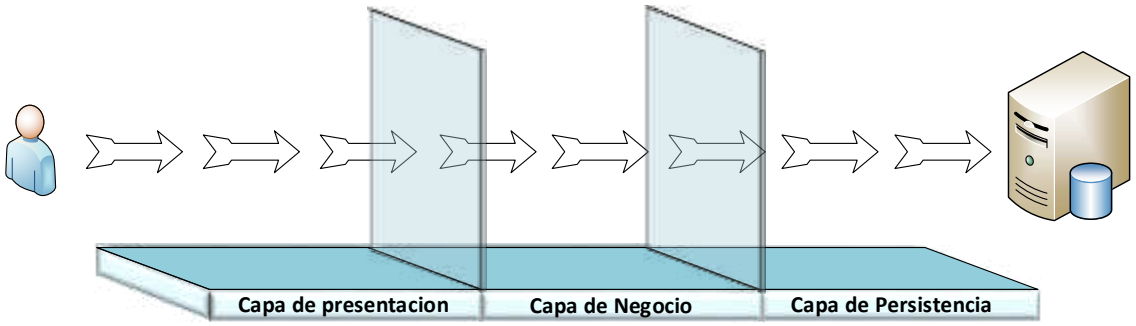


Figura 22 - Modelo de arquitectura a tres capas

De estos tres niveles básicos pueden aparecer niveles paralelos por diversos temas de arquitectura, pero siempre se basarán en el mismo principio y realmente no son más que diferentes implementaciones del mismo concepto en la capa en la que se encuentren. Por ejemplo, una aplicación puede tener diferentes vistas para diferentes dispositivos cliente, lo que indicará que en la capa de presentación tenga diferentes componentes independientes. La figura 23 muestra un modelo a tres capas con dos implementaciones distintas de la capa de presentación.

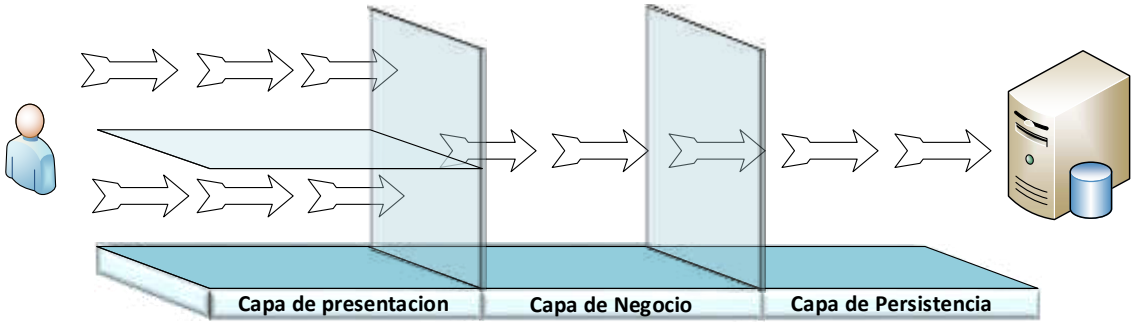


Figura 23 - Modelo de arquitectura a tres capas

En Java la forma de mantener las capas fuertemente desacopladas es realizando los accesos o referencias entre capas mediante interfaces que modularmente se encuentran separadas de sus implementaciones. Estas interfaces representan un conjunto de servicios entre las capas, estos servicios transmiten objetos de entrada y/o salida entre las capas. El conjunto de objetos de entrada y salida entre capas es lo que conocemos como el dominio. Así el dominio de una aplicación son el conjunto de objetos que la aplicación expone a través de sus servicios con el exterior. Análogamente el dominio de una capa son el conjunto de objetos que una capa expone a través de sus servicios a la capa inmediatamente superior.

Los objetos de un dominio son lo que se conocen por DTOs: Objetos de transferencia de datos*, Data Transformation Objet*. Son objetos planos que transportan datos entre procesos. Se les dice planos, también conocidos como POJO *(Plain Old Java Objects*), pues son objetos que solo contienen propiedades y sus métodos de obtención y establecimiento de datos (*getters and setters*). La figura 24 muestra los distintos dominios de una aplicación, un dominio por cada capa.

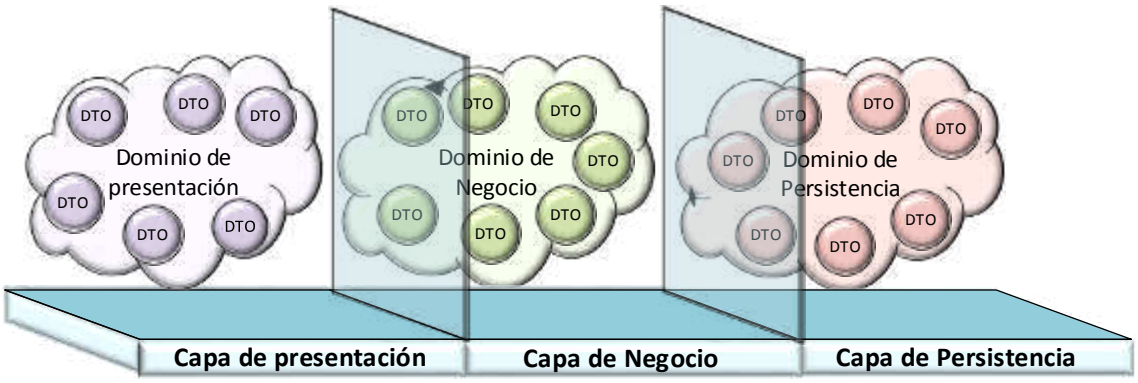


Figura 24 - Patrón DTO

Un modelo de diseño de arquitectura estricto diría que para que las capas estén fuertemente desacopladas el dominio de una capa solo puede ser referenciado desde la capa inmediatamente superior de forma que cada capa tuviera su propio dominio y este fuera distinto del dominio de la aplicación. Esto se conoce como modelo dominio-adaptador. Hay un adaptador que se encarga de convertir los dominios entre capas o del dominio de la aplicación al dominio de capas. Esto reduce significativamente el impacto que pueda tener una modificación de un DTO por un tema horizontal en una capa pues solo se vería afectado el DTO, la capa del dominio de ese DTO y el adaptador. Si la modificación obedece a un tema transversal que afecta a varias capas se deberá modificar el DTO en cada dominio y los adaptadores correspondientes. La figura 25 muestra el patrón de dominio – adaptador.

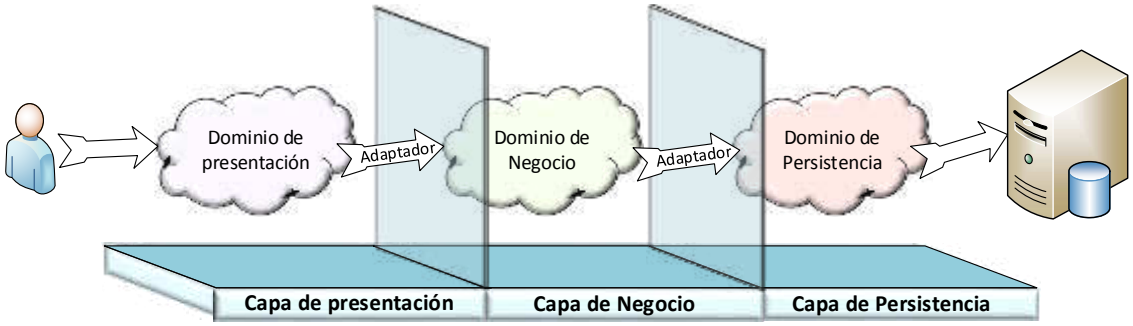


Figura 25 - Patrón Dominio-Adaptador

La idea que subyace en todos estos conceptos es siempre la misma: mantener las capas el máximo de desacopladas posible bajo la premisa de mantener la simplicidad y la reutilización de componentes.

3.3.2 Modelo MVC

Modelo Vista Controlador, en adelante MVC, en un patrón de arquitectura que separa los datos y la lógica de negocio de una aplicación (lo que se conoce como Modelo) de la interface de usuario (lo que conocemos como Vista) y la lógica de la vista (lo que conocemos como Controlador). La figura 26 muestra esquemáticamente el patrón MVC. En este patrón de diseño se diferencian las siguientes partes:

- El controlador es la parte encargada de decidir que hacer frente a un evento determinado: mostrar una vista, ejecutar una determinada función de negocio… Reacciona frente a las órdenes del usuario.
- La vista corresponde con el conjunto de elementos de la interface gráfica del usuario. Es una representación gráfica del modelo.
- El modelo representa toda la gestión de negocio interna de la aplicación.

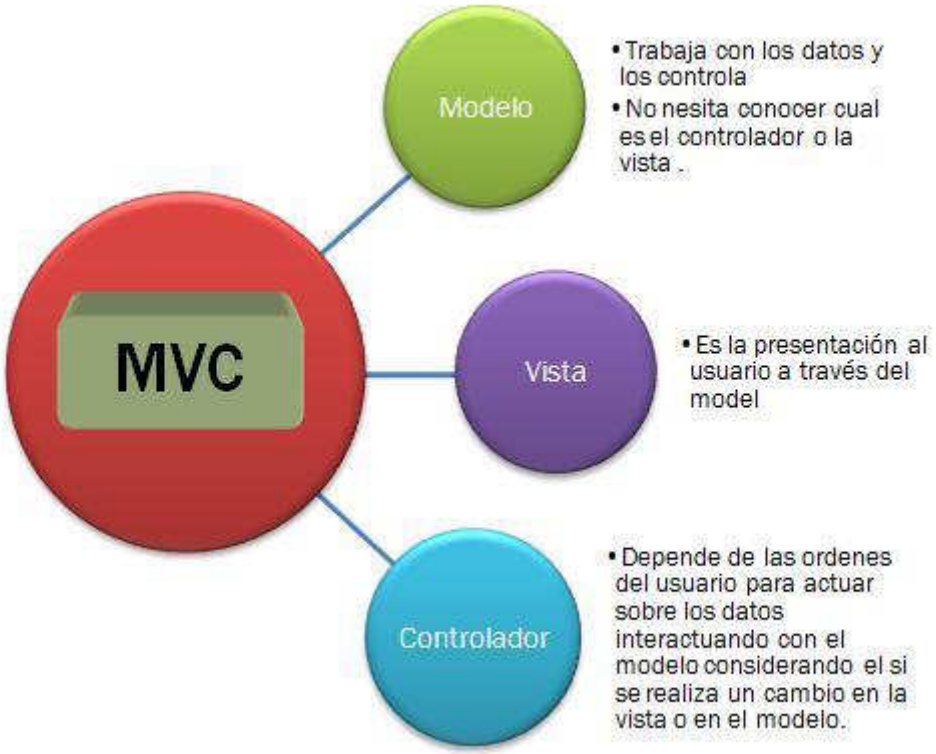


Figura 26 - Modelo MVC

La interacción entre estas partes la muestra el diagrama de la figura 27:

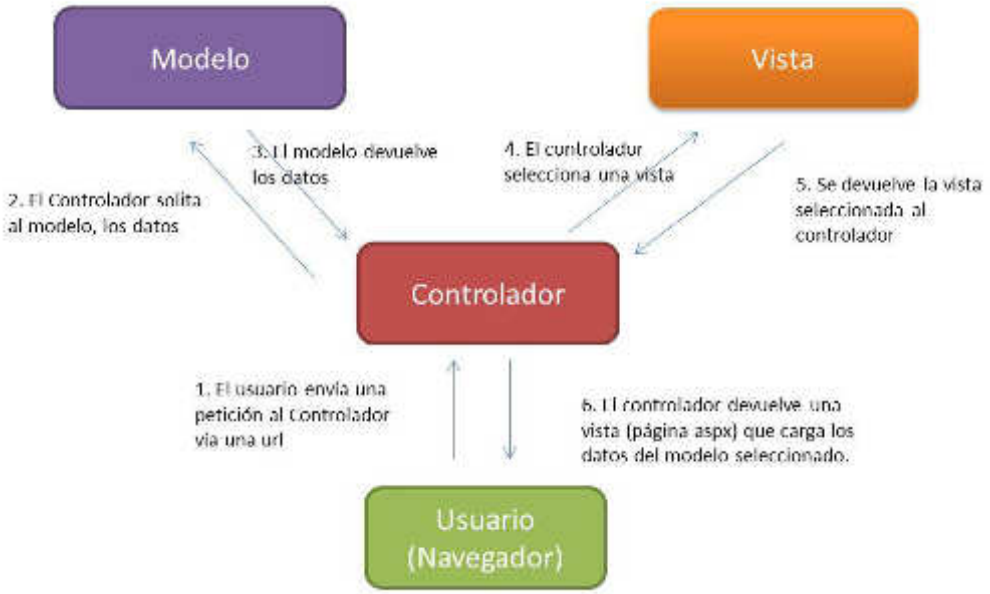


Figura 27 - Modelo MVC

Es un modelo que aplica a la capa de presentación y su relación con la capa de negocio, como se muestra en la figura 28.

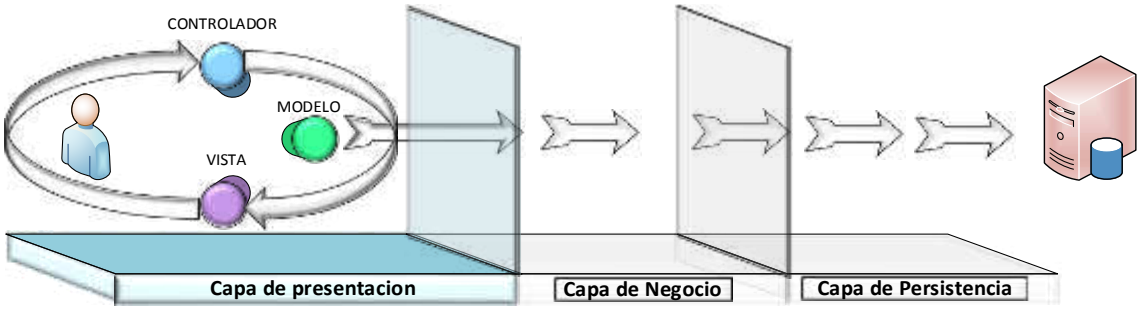


Figura 28 - MVC en presentación

Hay que tener en cuenta que el modelo MVC no es una tecnología determinada sino un patrón de diseño. Hay varios *frameworks* para java que implementan este modelo. Los más utilizados son: Apache Struts, Apache JavaServerFaces. Spring también tiene su *framework* para MVC. El *framework* MVC que vamos a utilizar para la aplicación es JavaServerFaces al venir ya integrado con WebSphere.

3.3.3 Capa de presentación: Java Server Faces

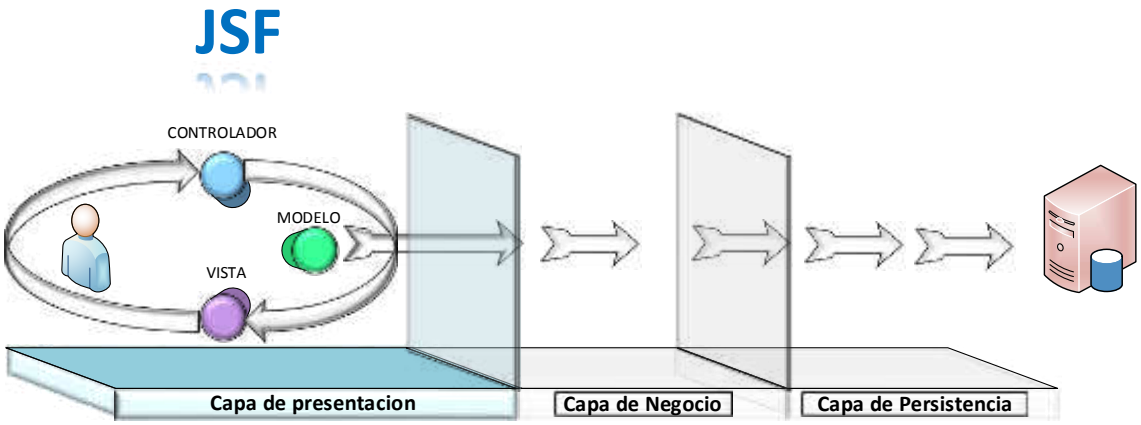


Figura 29 - JSF en presentación

La figura 29 muestra como interviene JSF en la arquitectura a tres capas. JSF Implementa una solución dentro del modelo MVC. Concretamente cubre la vista y el controlador. La figura 30 muestra como JSF realiza esta implementación de forma esquemática.

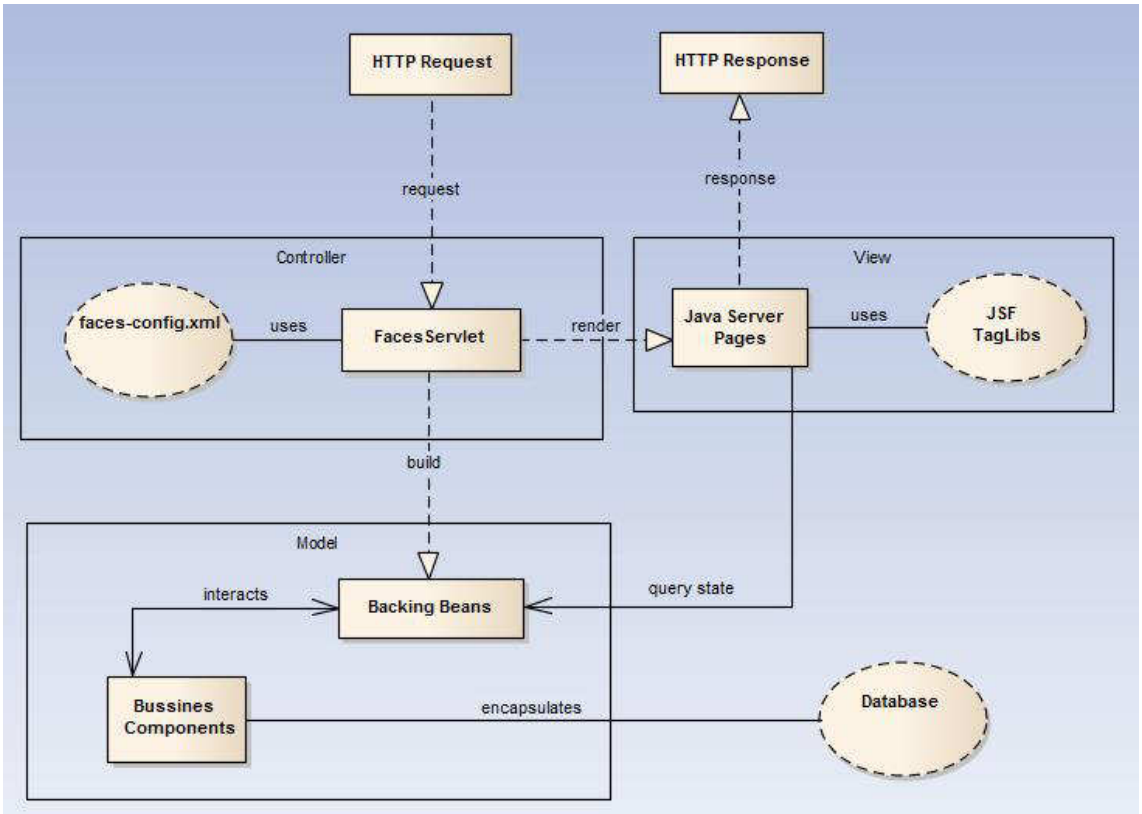


Figura 30 - JSF

Es una solución que viene integrada en el servidor de aplicaciones WAS 7.0

JSF incorpora:

- Un conjunto de componentes para renderizar la interface del usuario. A parte de las que incorpora por defecto al ser un estándar es fácil encontrar componentes que se ajusten a las necesidades.
- Un modelo de control que funciona definiendo los eventos que pueden ocurrir en el sistema y nos permite definir un esquema de navegación entre páginas.
- Unas APIs que permiten definir validadores, internacionalización, accesibilidad, conversores de datos, interceptores de eventos,
- Un conjunto de extensiones con multitud de funcionalidad extra como RichFaces, ICEFaces, jQuery, PrimeFaces, OpenFaces…

JSF representa el controlador como una máquina de estados que permite definir cuál es la acción que hay que tomar frente a un evento determinado. Las propias acciones del software son eventos que el controlador maneja, así podemos controlar cuando se produce una acción en el sistema (por ejemplo, se ha efectuado una alta correctamente) que respuesta debe dar (mostrar una pantalla notificando que el alta se ha efectuado correctamente). De la misma forma puede controlar las acciones que realiza el usuario: pulsar un botón o cualquier otra interacción.

3.3.4 Capa de negocio: Enterprise JavaBeans 3.0

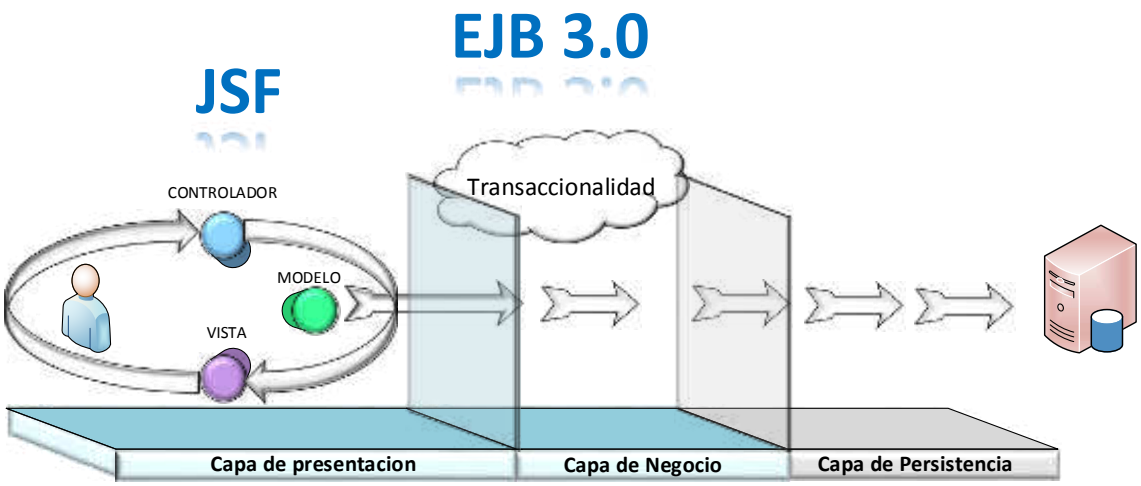


Figura 31 - Capa de negocio

Los Enterprise JavaBeans (EJB en adelante) forman parte de las especificaciones de J2EE. Proporcionan un modelo de componentes distribuido estandarizado que permite abstraerse de ciertas tareas generales en una aplicación empresarial, tales como el manejo de la concurrencia y la transaccionalidad, entre otros, pare centrarse en el desarrollo de la lógica de negocio en sí. La figura 31 muestra donde encaja EJB dentro de la arquitectura que se ha definido.

Hay tres tipos de EJB:

- EJB de Entidad (*Entity* EJB): encapsulan los objetos que almacenan los datos en la base de datos. Son objetos de la capa de persistencia. Hoy en día en desuso ya que están siendo reemplazados por JPA. Ver apartado de *capa de persistencia*

- EJB de sesión (sesión EJB): gestionan el flujo de información en el servidor. Es el tipo de EJB que vamos a utilizar para la implementación de la lógica de negocio. Sirven a los clientes como una fachada de los servicios que se proporcionan (servicios de negocio) Hay de dos tipos:
    - Con estado (*stateful*); Son EJBs de sesión que mantienen los valores de las propiedades durante la conexión con el cliente. El estado termina cuando la sesión con el cliente termina, por tanto el ámbito del estado es la sesión. Dentro de su ámbito no pueden ser llamados concurrentemente y la gestión del estado es compleja y consume recursos por lo que su uso no es recomendable. Se establece como norma que se utilicen cuando el mantenimiento del estado dentro de la sesión sea imprescindible.
    - Sin estado (*stateless*): Son EJBs de sesión que proporcionan una fachada de servicios (servicios de negocio) al cliente sin mantener ningún estado dentro de la sesión. Al no mantener ningún estado pueden ser llamados concurrentemente.
- EJB dirigidos por mensaje (Message-driven EJB): Son EJBs con funcionamiento asíncrono. Están relacionados con las colas JMS (Java Messaging System). En las colas JMS (JMS Queue) se encolan mensajes que el EJB asociado a la cola va procesando. Es un potente sistema de proceso asíncrono que funciona de forma totalmente paralela tanto dentro de un servidor como en un *cluster*.

Los EJBs de sesión sin estado, que son los que vamos a utilizar para implementar la capa de negocio, pueden contener los siguientes elementos:

- Interface Local: es la interface para ser usada localmente que define el servicio. Cualquier componente de la aplicación que requiera un servicio de negocio de forma local accederá a él mediante su interface local. Así se accederá a la fachada de servicios de negocio desde la capa de presentación o la capa de servicios que la aplicación ofrece al exterior. Las interfaces locales se identifican con la anotación @Local y al ser usadas por la propia aplicación no deben distribuirse. No obstante para mantener las capas fuertemente desacopladas se empaquetarán las interfaces locales de forma independiente, así desde la capa de presentación solo se referenciará el paquete con las interfaces locales independizando la capa de presentación de la implementación de los servicios.
- Interface Remota: Es una interface similar a la interface local pero que define los servicios a ser usados de forma remota, tanto por otras aplicaciones dentro del mismo servidor de aplicaciones como por otras aplicaciones existentes en otros servidores de aplicaciones. En nuestro caso vamos a limitar las llamadas a EJBs remotos dentro de la red corporativa donde se instale la aplicación. Así otras aplicaciones de la red corporativa podrán acceder mediante EJB remotos a la aplicación. Las interfaces remotas se identifican con la anotación @Remote y al ser interfaces pensadas para usarse por otras aplicaciones deben empaquetarse en un proyecto a parte para que puedan ser distribuidas.
- Implementación del EJB: Son las clases JAVA que implementan los servicios definidos por las interfaces locales y remotas de EJB.

Sobre las características importantes de los EJBs de sesión sin estado que ya se han comentado y que hacen de ellos una herramienta potente sobre la que crear los servicios de negocio hay una de ellas que es importan comentar y detallar un poco más: la transaccionalidad.

Los EJBs se ejecutan dentro del contenedor de EJB del servidor de aplicaciones y una de las cosas que el contendor puede realizar es gestionar toda la transaccionalidad de los servicios de negocio. Esto es: abrir transacciones, realizar *commit* si ha terminado correctamente o *roolback* si se ha producido alguna excepción marcada como excepción transaccional. Cada servicio de negocio que se expone a través de una interface local puede definir su propia transaccionalidad dentro del siguiente grupo:

- MANDATORY: Debe haber una transacción previa creada por el cliente que llama al servicio. La transacción del servicio se ejecutará en ese contexto transaccional. De no ser así el contenedor lanzará una excepción.
- NEVER: El servicio debe ejecutarse fuera de un contexto transaccional. Si existe un contexto transaccional creado por el cliente al llamar al servicio el contenedor lanzará una excepción
- NOT SUPORTED: El servicio se ejecutará fuera de un contexto transaccional. Si existiera un contexto transaccional creado por el cliente en el momento de llamar al servicio este contexto quedará en suspenso hasta que la ejecución del servicio termine.
- REQUIRED: Debe existir un contexto transaccional. Si ya existe uno el contenedor continuará con ese contexto, si no lo creará. El contexto transaccional no será propagado desde el cliente. Es la opción por defecto.
- REQUIRES NEW: El contenedor creará un contexto transaccional para el servicio independientemente del contexto que tuviera el cliente.
- SUPPORTS: Si el cliente tiene un contexto transaccional el servicio se ejecutará dentro de ese contexto, si no el contenedor creará uno. Es similar a REQUIRED solo que en este caso el contexto si es propagado desde el cliente.

Otra característica importante de los EJB es la inyección de dependencias y la inversión de control (IoC – *Inversion of Control*). Se dice que hay una inversión de control cuando el control del objeto que se va a usar en un punto del código no depende del código que utiliza el objeto sino de un tercer elemento que gestiona ese control, de ahí que se llama inversión por que el control queda invertido.

De forma práctica, si en un objeto A creamos un objeto B haciendo un new dentro de A:

```
Class A{
    …
   Obj b = New B();
    …
}
```

Tenemos un control clásico de A sobre B (es A quien decide que objeto B crea). Pero si A solo define un objeto B pero no lo crea sino que un elemento externo crea B y lo inyecta en A tenemos inversión de control. En la figura 32 se muestra la inyección de dependencias entre EBJs:

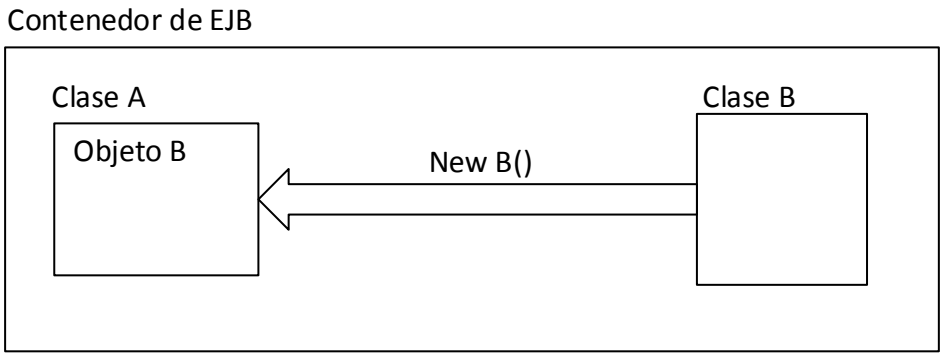


Figura 32 - EJB Inyección de dependencias

Ahora el control de A sobre B no está en A sino en ese "elemento externo". En el caso de EJB el elemento externo es el propio contenedor de EJB. Se puede indicar que objeto B debe crear para inyectar en A de varias formas: mediante anotaciones, mediante un fichero de configuración de EJB, mediante la incorporación de una u otra implementación de servicio EJB… Sea como sea esta será la forma mediante la cual se inyectarán los servicios de negocio desde la capa de negocio en la capa de presentación, y de los servicios de acceso a datos desde la capa de persistencia en la capa de negocio. Esto una vez más persigue la independencia entre capas, que sean fuertemente desacopladas. Cambiando la configuración de la inyección podemos cambiar la lógica de negocio en la capa de presentación sin modificar nada en esta última.

Para que esto sea posible el acceso entre las distintas capas se debe realizar mediante interfaces. Así, en el ejemplo, el objeto A accede al objeto B a partir de una interface del objeto B donde se describen los distintos servicios que B ofrece a A. La descripción de estos servicios es lo que se conoce como contrato del servicio.

3.3.5 Capa de persistencia: Java Persistence Api

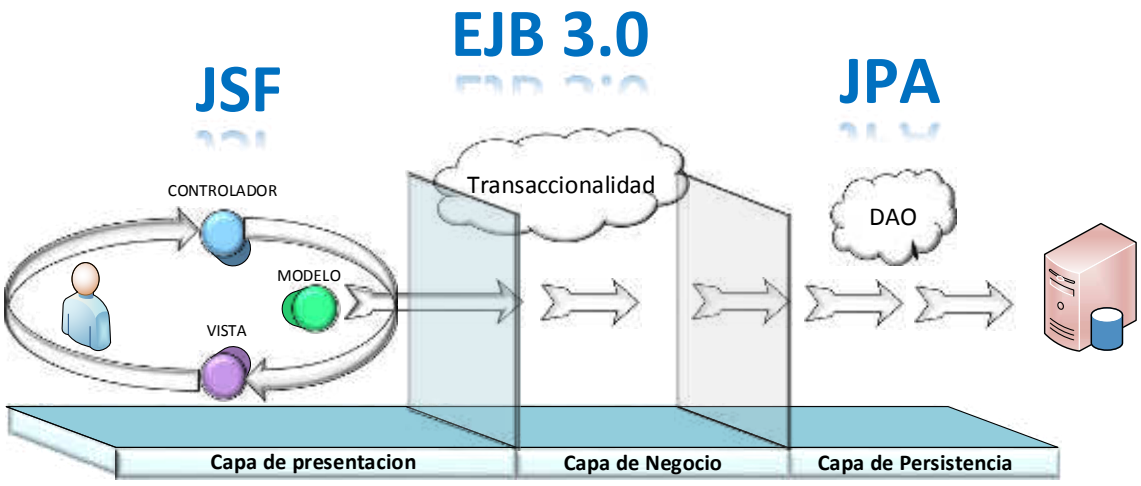


Figura 33 - Capa de persistencia

La capa de persistencia en JAVA implementa una solución ORM (*Object Relational Maping*) para java. ORM es un modelo diseñado para la conversión de datos inter-relacionados a objetos. Los datos son inter-relacionados ya que se obtienen de una base de datos relacional a través de un motor de persistencia. ORM es una arquitectura implementable en cualquier plataforma, para JAVA se define el estándar JPA (*Java Persistence API*). JPA está definida dentro de EJB 3.0 y tiene diferentes implementaciones tales como Hibernate, TopLink y OpenJPA que es la implementación ofrecida por Apache Software Foundation (http://openjpa.apache.org/) OpenJPA está soportado de forma nativa en el WAS 7.0 y por eso ha sido el motor de persistencia elegido para la realización del proyecto. La figura 33 muestra cómo encaja JPA dentro de la arquitectura definida. La figura 34 muestra esquemáticamente la filosofía ORM en Java.

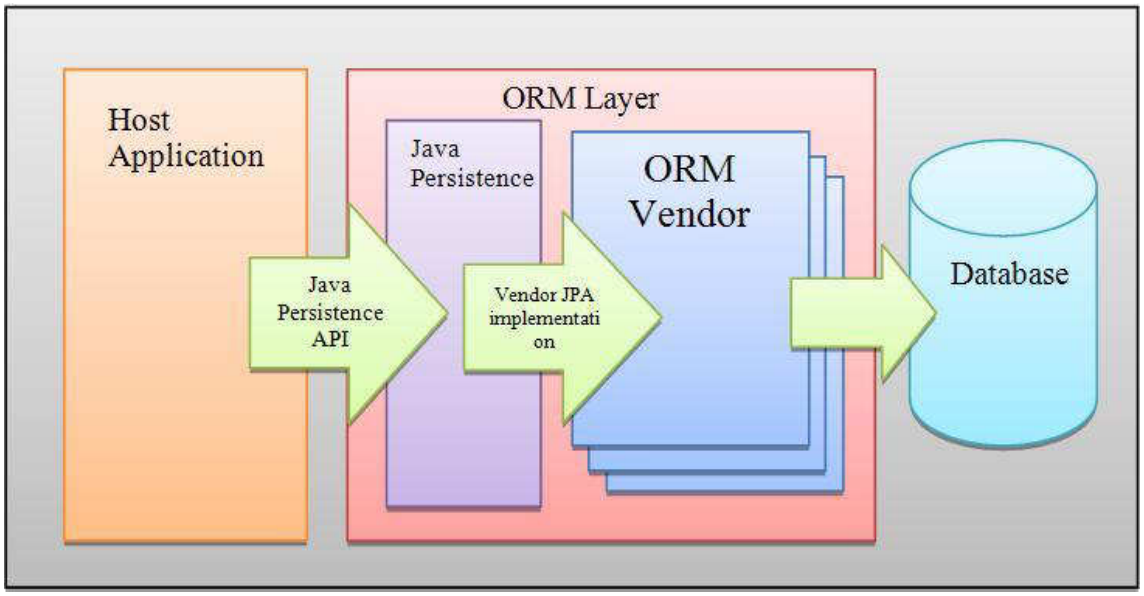


Figura 34 - ORM

JPA representa en objetos Java la misma estructura de datos que representa la base de datos. Se encarga de la persistencia de los datos (inserción, modificación y eliminación) así como de la realización de consultas mediante JPQL (lenguaje similar a SQL mediante el cual se realizan las consultas a la base de datos a través de OpenJPA). Otra ventaja de OpenJPA es que independiza los desarrollos del DBMS utilizado.

JPA es una herramienta para convertir de objetos java a registros en la base de datos o al revés. Implementa ese mapeo de datos a objetos, pero no implementa ninguna lógica o función a nivel de programación. Necesitamos una capa que utilizando JPA obtenga los datos que la capa de negocio le pida o persista los datos que la capa de negocio le dé. Para ello utilizaremos el patrón DAO (*Data Transformation Object*) cuyo concepto de base es independizar la gestión física de los datos de la gestión de negocio de la aplicación. El dato puede obtenerse de un WebService, de una conexión JDBC a través de JPA o de un fichero de texto, los DAO se encargarán de obtener los datos de forma que donde residen los datos y con qué formato es transparente para la gestión del negocio.

DAO es un patrón, no una tecnología. Como patrón que es podemos implementarlo de varias formas. La forma elegida es como un conjunto de servicios EJB 3.0 de sesión sin estado.

Se ha escogido esta forma porque los gestores de negocio son EJBs por lo que al mantener esta tecnología podemos utilizar las facilidades que nos proporciona: mantener la transacción, inyección de dependencias en los gestores de negocio (se inyectarían los DAO) y similares.

3.3.6 Servicios externos: WebServices con JAX-WS

Una parte imprescindible de la aplicación es el conjunto de servicios que debe ofrecer a otras aplicaciones tanto de la propia compañía como aplicaciones externas. Esto es lo que se conoce como arquitectura orientada a servicios (*Service Oriented Architecture*, SOA en adelante). Los *WebServices* son una implementación de SOA. WebSphere 7.0 ofrece varias tecnologías para exponer *WebServices*: JAX-RPC, JAX-WS… también se podría integrar una solución de terceros, pero lo más lógico es utilizar las soluciones que el propio WAS ofrece. En este caso la tecnología más actual y potente de las que ofrece es JAX-WS que implementa de forma muy sencilla y cómoda para los desarrolladores los servicios WEB. Simplemente indicando con anotaciones las clases que implementas los W*ebServices* y los métodos que queremos exponer el propio WAS en tiempo de despliegue de la aplicación crea los servicios web basándose en las especificaciones de los métodos Java expuestos. También desde la misma consola del WAS se puede configurar todo lo relativo a la seguridad que nos convenga implementar para cada servicio web de forma que toda esa gestión es transparente a la programación. WAS implementa *Web Service Security* con XML-DSign, es decir, mediante firma electrónica de los mensajes SOAP que se intercambian en las llamadas a los servicios WEB.

Mediante JAX-WS la aplicación expondrá los servicios web hacia las aplicaciones de la misma compañía que informarán de la ejecución de los procesos que van a registrarse. También expondrá los servicios de consulta de procesos a aplicaciones externas a la compañía que quieran consultar el resultado de un proceso.

La figura 35 muestra los distintos accesos mediante *WebServices* que va a suministrar el sistema (por internet y por intranet)

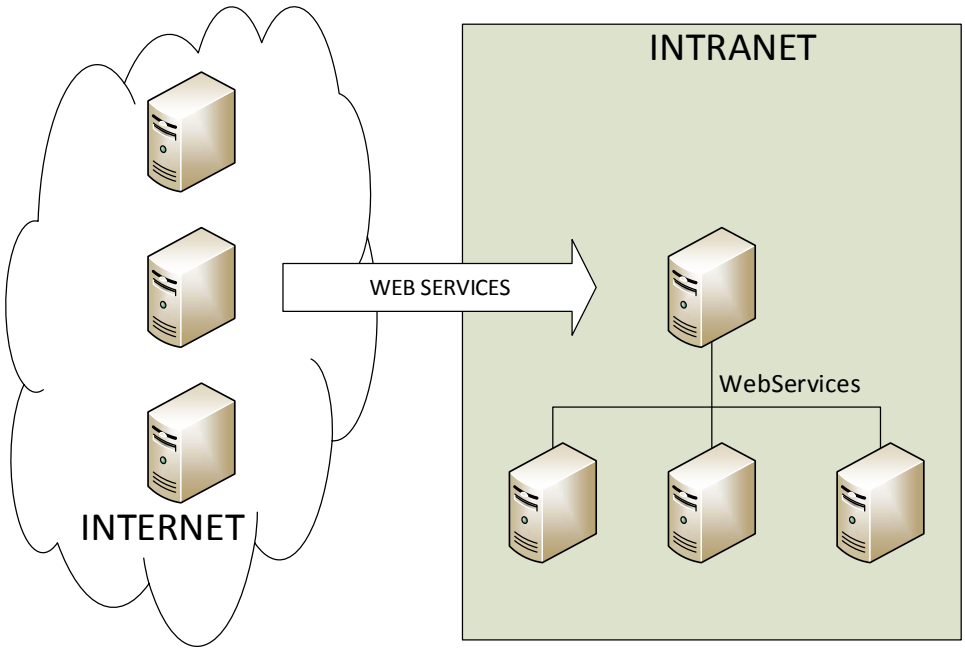


Figura 35 - WebServices

3.3.7 Servicios internos: EJBs remotos

Sobre los EJB ya se ha habado en la capa de negocio donde se ha expuesto las distintas interfaces de acceso que tienen los EJBs, las interfaces locales y las interfaces remotas. Los mismos servicios web internos que se exponen a otras aplicaciones de la compañía para que informen de las ejecuciones de procesos van a ser expuestos mediante EJB remoto. Exponemos estos servicios como EJB Remoto además de como *WebServices* básicamente por motivos de rendimiento: el acceso a EJBs remotos es más rápido y menos costoso en términos de uso de recursos que los accesos mediante *WebService* (que requieren parsear XML y des serializar los objetos del SOAP). EJB es una tecnología exclusivamente Java por lo que es óptima en Java. Por otro lado debemos ofrecer a demás los servicios mediante *WebServices* para dar acceso a aquellos sistemas realizados fuera de Java. La figura 36 muestra los accesos mediante EJB remotos que va a tener el sistema.

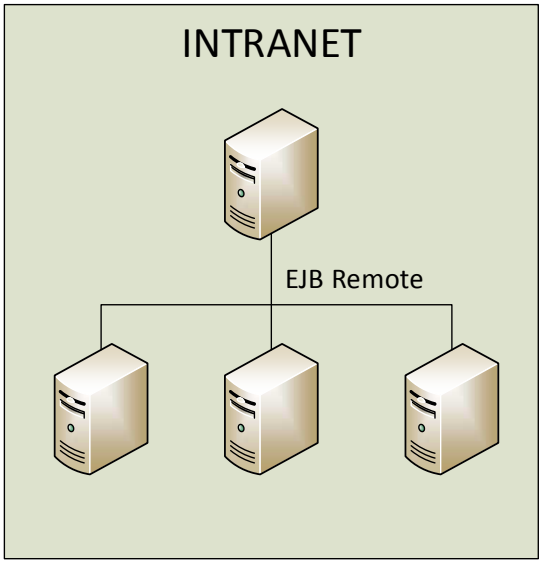


Figura 36 - EJB Remotos

3.3.8 Arquitectura de sistema

La figura 37 presenta una arquitectura básica de una infraestructura de *hosting* de aplicaciones Web. La infraestructura está abierta tanto a internet como a la red corporativa de la compañía para el acceso a través de la intranet a alas aplicaciones por parte del personal de la compañía.

Presenta una primera capa con servidores Alteon que exponen las URLs y realizan el balanceo de carga hacia los distintos servidores WEB. La siguiente capa está compuesta por los servidores WEB donde se almacena el contenido estático de las aplicaciones y se realiza la gestión de la sesión WEB. La siguiente capa está compuesta por los servidores de aplicaciones que es donde realmente residen las aplicaciones. Por último tenemos los servidores de base de datos formando la capa de datos. Los accesos entre las distintas capas se realizan a través de *firewalls* que protegen la infraestructura.

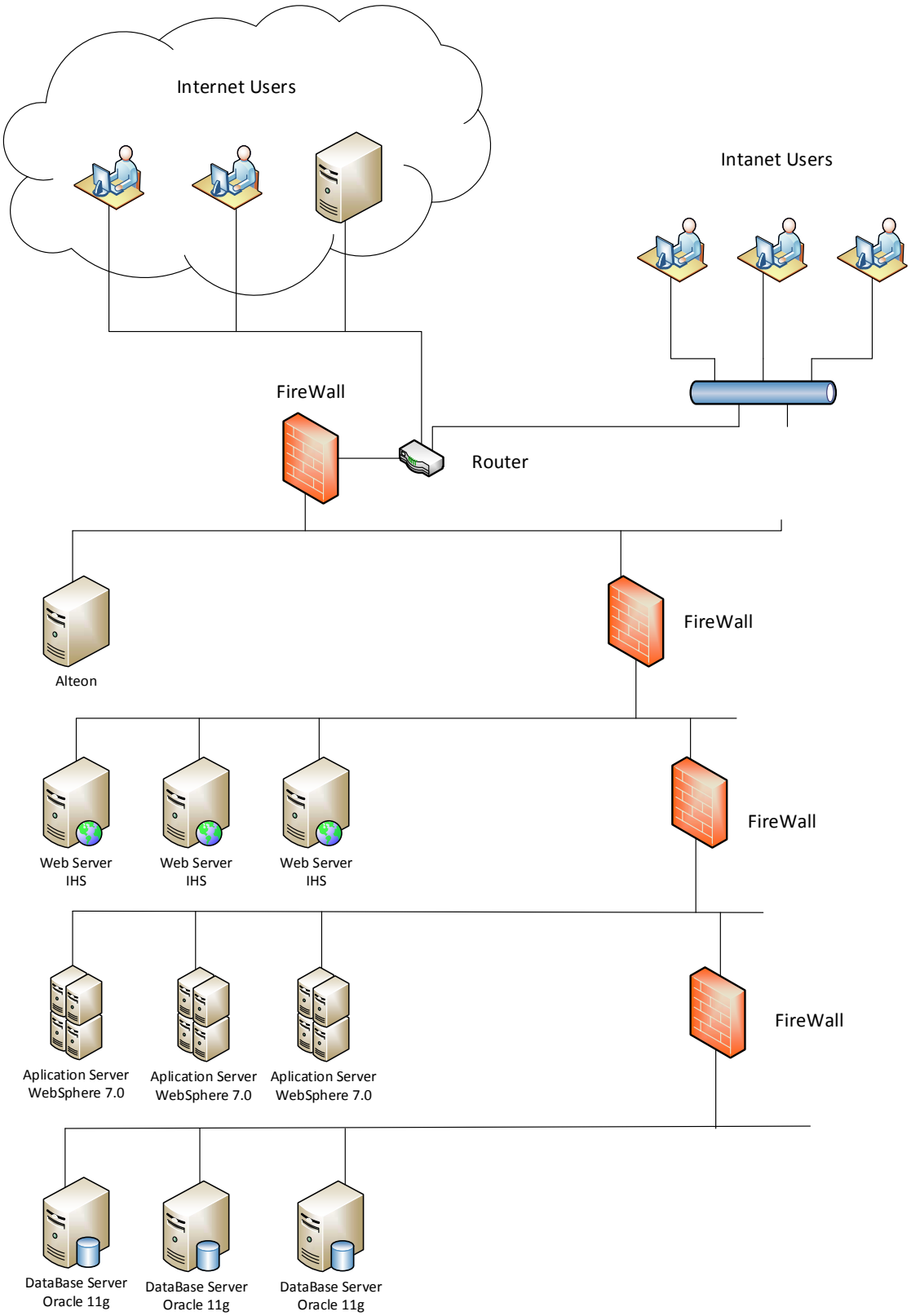


Figura 37 - Diagrama de arquitectura de sistemas

3.4 Diseño

En este apartado se va a realizar un diagrama de clases por cada grupo funcional. Se intenta representar cada capa las clases que contiene así como los accesos entre las capas. Se podría realizar un diagrama de secuencia por cada caso de uso pero se considera que sería demasiado extenso para la información que aporta.

3.4.1 Diseño de la aplicación

Subsistema de procesos

- Figura 38: GF-01 Administrar procesos - GF-01.1 Administrar aplicaciones

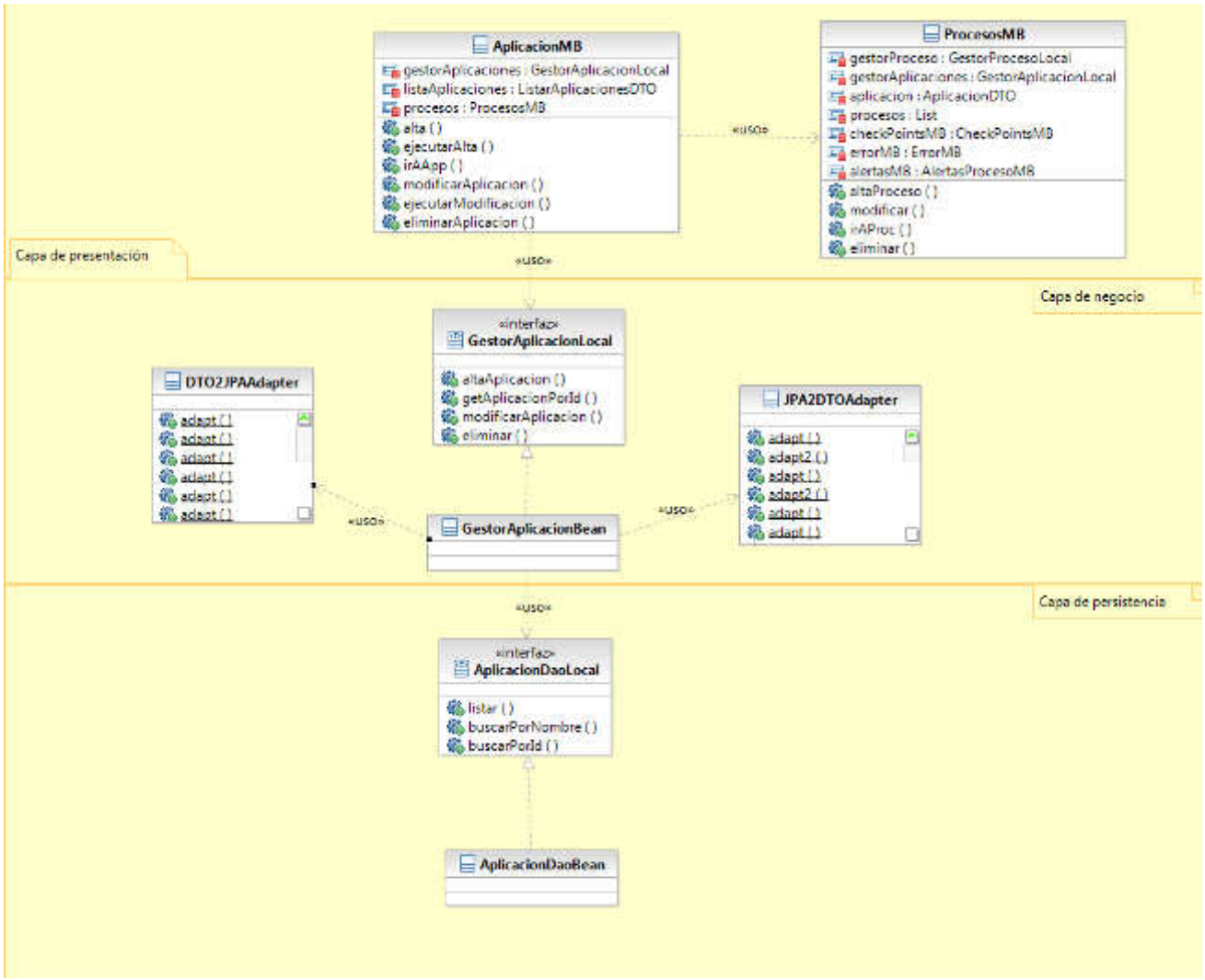


Figura 38 - Diagrama de clases - GF-01.1 Administrar aplicaciones

- Figura 39: GF-01 Administrar procesos - GF-01.2 Administrar procesos

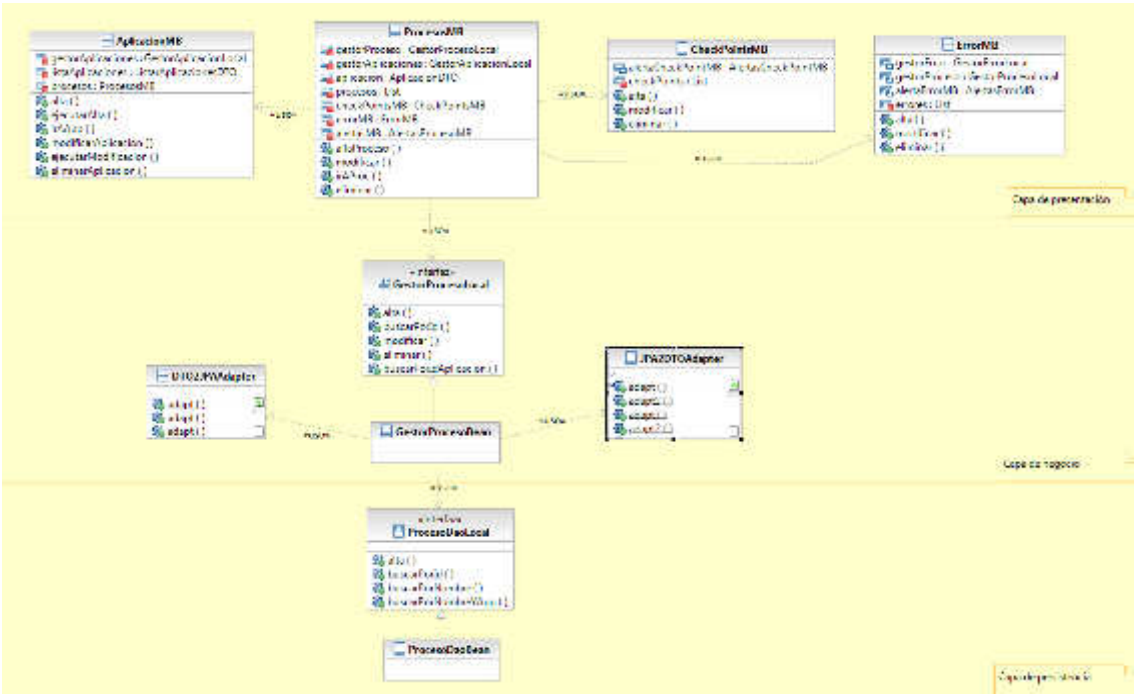


Figura 39 - Diagrama de clases - GF-01.2 Administrar procesos

- Figura 40: GF-01 Administrar procesos - GF-01.3 Administrar checkpoints

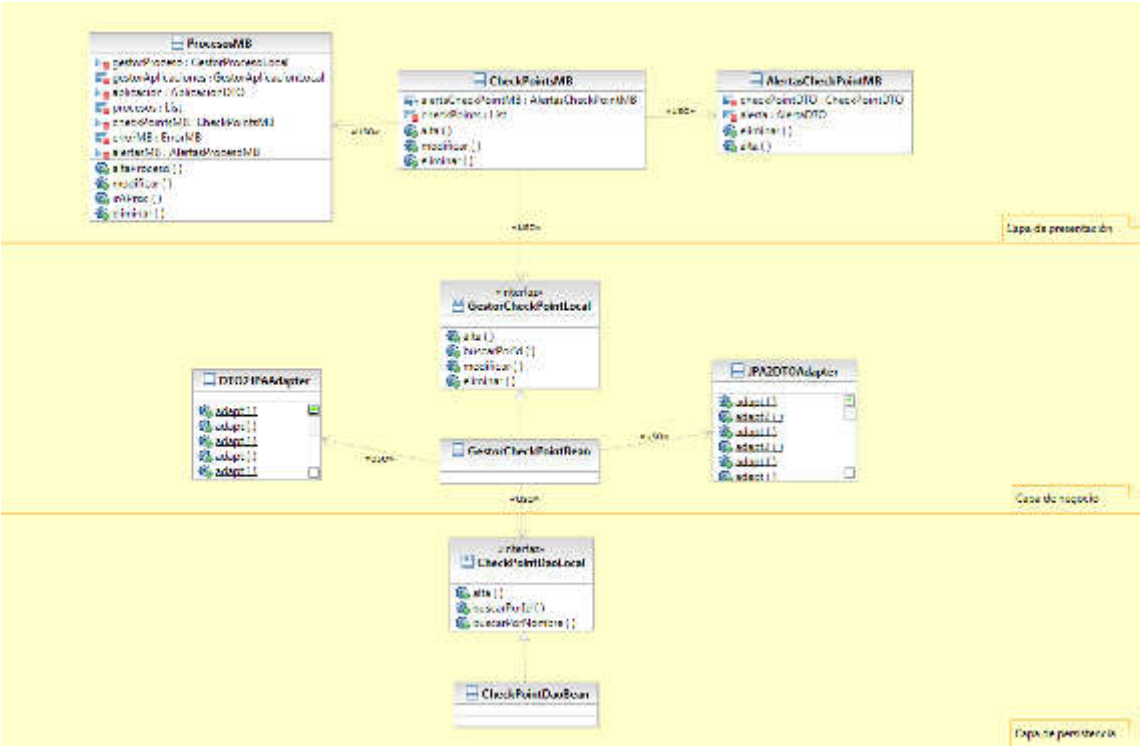


Figura 40 - Diagrama de clases - GF-01.3 Administrar checkpoints

- Figura 41: GF-01 Administrar procesos - GF-01.4 Administrar errores

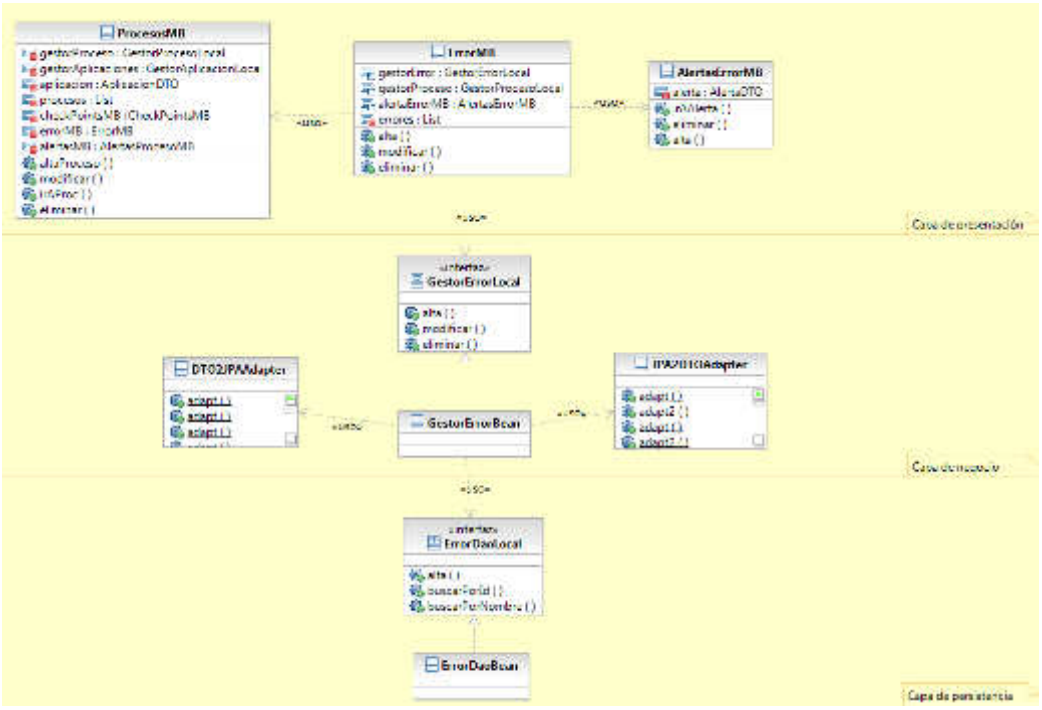


Figura 41 - Diagrama de clases - GF-01.4 Administrar errores

- Figura 42: GF-02 Introducir datos de procesos

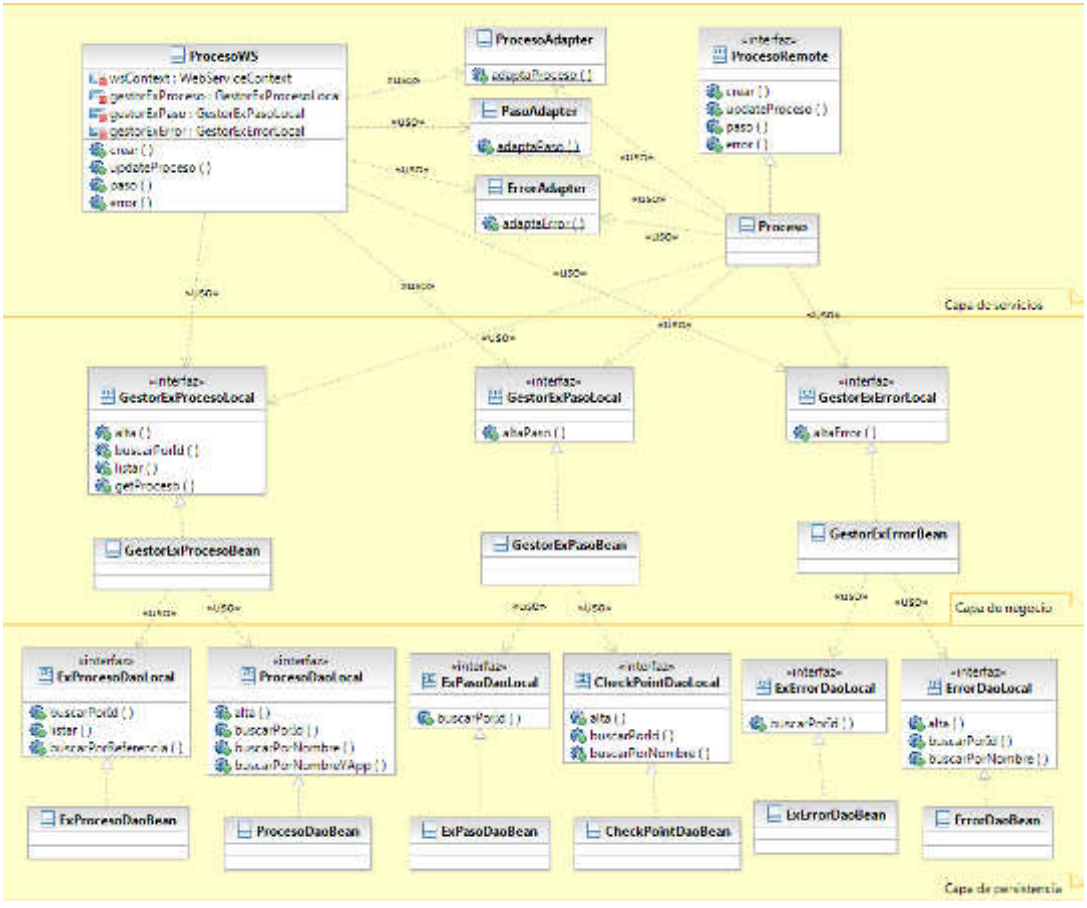


Figura 42 - Diagrama de clases - GF-02 Introducir datos de procesos

- Figura 43: GF-03 Explotar datos de procesos

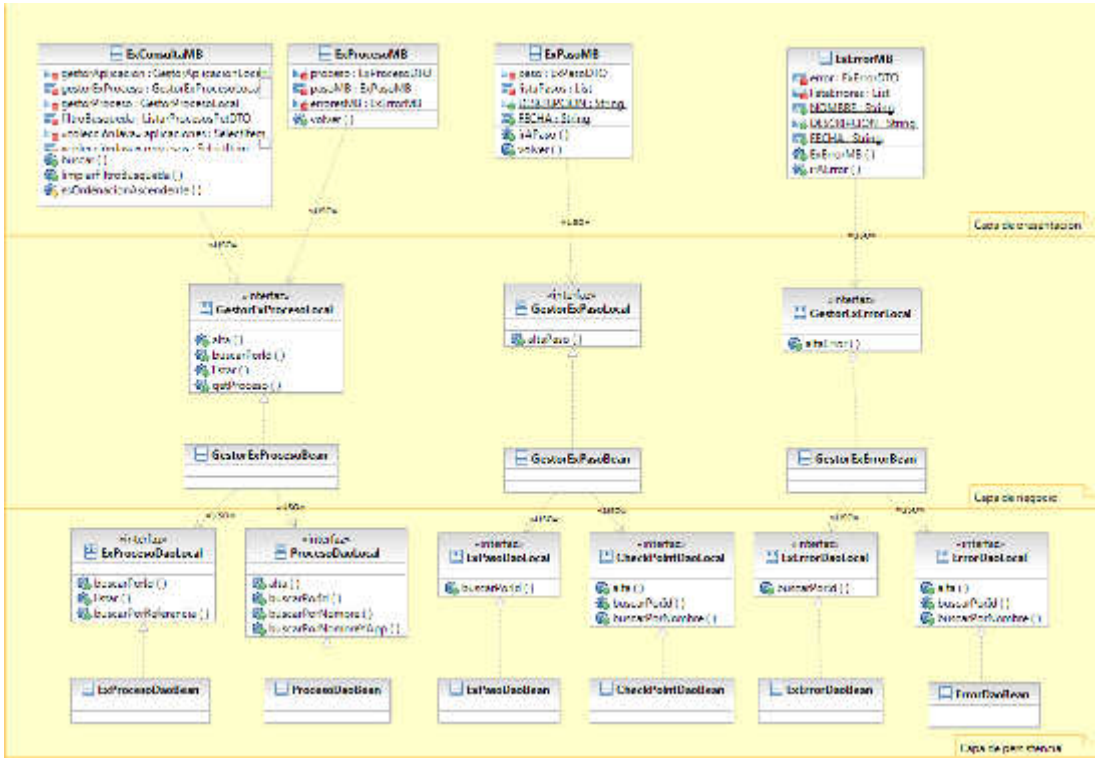


Figura 43 - Diagrama de clases - GF-03 Explotar datos de procesos

- Figura 44: GF-05 Administrar alertas

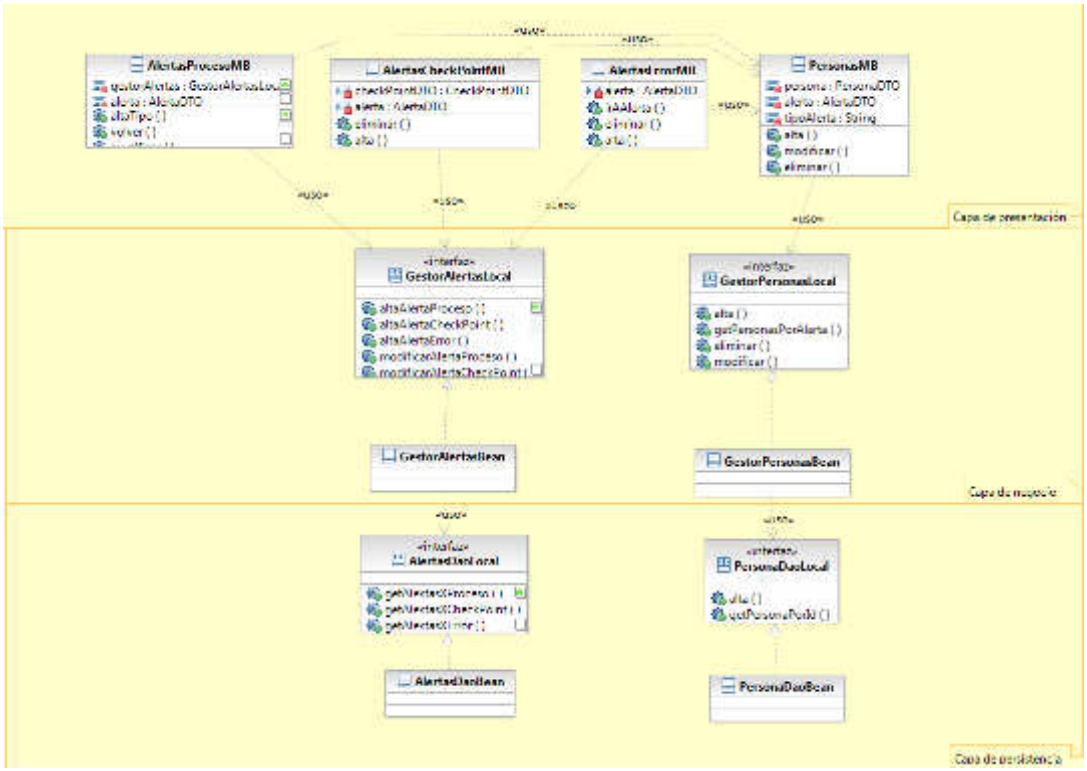


Figura 44 - Diagrama de clases - GF-05 Administrar alertas

- Figura 45: diagrama de Entidades JPA

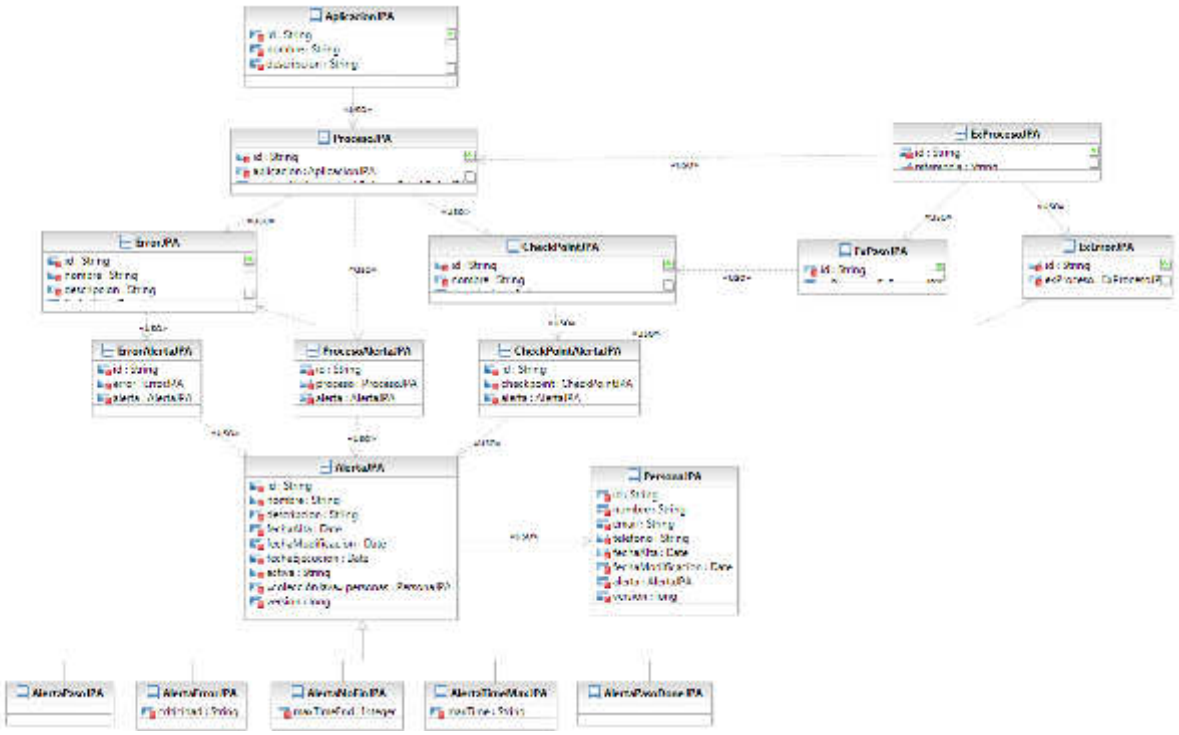


Figura 45 - Diagrama de entidades

- Figura 46: diagrama de excepciones

Figura 46 - Diagrama de excepciones

3.4.2 Modelo de datos

En este apartado se va a definir el modelo físico que la aplicación implementa.

3.4.2.1 Definición de las tablas

En este apartado se van a definir las distintas tablas que la aplicación implementa. Antes de entrar en la lista de las tablas se van a aclarar unos conceptos que determinan y condicionan el modelo de datos.

- Herencia: se puede implementar un modelo similar al concepto existente de herencia en un modelo referencial de forma que una tabla sea "padre" y otra sea "hija". La tabla padre representa un concepto más genérico, las tablas hijas representarán un concepto más específico. Estas tablas se mapearán con objetos JAVA que mantendrán esta relación de herencia. Como realizar este mapeo dependerá de la implementación JPA que se esté usando, en nuestro caso Open-JPA. Para poder realizar este mapeo en Open-JPA necesitamos que la tabla padre y la tabla hija compartan ID y, por lo tanto, el ID en la tabla hija sea clave primaria y clave foránea. También necesitamos un discriminador en la tabla padre que le indique al JPA donde encontrar el registro hijo. En el modelo las tablas ALERTA_TIPO_1, ALERTA_TIPO_2, ALERTA_TIPO_3, ALERTA_TIPO_4, ALERTA_TIPO_5 tienen una relación de herencia con ALERTA: definen alertas específicas aumentando los datos en caso necesario.
- ID: los Ids que hacen de clave primaria (y, por lo tanto, foránea) se pueden implementar de varias formas. En este caso los va a gestionar Open-JPA y se definen como CHAR de 32.
- OpenJPA controla que el registro que va a modificar no haya sido modificado por otra transacción desde que se inició la transacción. Para ello requiere de un campo específico para esta función que llamamos "VERSION". La versión se modifica desde JPA en cada update por lo tanto si la versión del registro a modificar ha cambiado es que otra transacción ha modificado el registro y, por lo tanto, hay una inconsistencia en los datos.

La tabla 14 muestra la lista de tablas de la aplicación:

Nombre	Descripción
ALERTA	Contiene las alertas y sus datos genéricos. Representa una alerta sin entrar en detalles del tipo de alerta que es ni de los datos específicos de la alerta ya que estos se definen por relación de herencia en las tablas ALERTA_TIPO_X
ALERTA_PERSONA	Contiene las personas que deben recibir información sobre las alertas.
ALERTA_TIPO_1	Contiene las aletas de tipo "proceso sin finalizar". Alertas que deben dispararse cuando un proceso no ha finalizado en un tiempo predeterminado
ALERTA_TIPO_2	Contiene las alertas de tipo "tiempo máximo de proceso superado". Son alertas que deben disparase cuando un proceso ha finalizado pero ha superado el tiempo que se haya definido como máximo.
ALERTA_TIPO_3	Contiene las alertas de tipo "Proceso finalizado con errores". Son alertas que deben dispararse cuando un proceso ha finalizado con algún error registrado.

ALERTA_TIPO_4	Contiene las alertas de tipo "alerta de *checkpoint* obligatorio no alcanzado". Son alertas que se disparan cuando un proceso finaliza sin haber pasado por un *checkpoint* obligado.
ALERTA_TIPO_5	Contiene las alertas de tipo "alerta de aviso de *checkpoint*". Son alertas que se asocian a un *checkpoint* y deben dispararse cuando se registra la el paso por un *checkpoint*.
APLICACION	Contiene la lista de aplicaciones que interaccionan con el sistema y van a enviar datos para monitorizar.
CHECKPOINT	Contiene una lista de los puntos de paso predefinidos para cada proceso.
CHECKPOINT_ALERTA	Es una tabla de relación entre *CheckPoint* y alerta: define las alertas que aplican a un punto de chequeo.
EMAIL	Contiene una lista de los mails que se han enviado o están pendientes de enviar. Un proceso batch va procesando esta lista y enviando los mails.
ERROR	Contiene una lista de errores predefinidos para cada proceso.
ERROR_ALERTA	Es una tabla de relación entre error y alerta: define las alertas que aplican a un error.
EX_ALERTA	Contiene las alertas que se generan en tiempo de ejecución.
EX_ERROR	Contiene los errores que se registran en tiempo de ejecución.
EX_PASO	Contiene los pasos que se registran en tiempo de ejecución.
EX_PROCESO	Contiene los procesos que se registran en tiempo de ejecución.
PROCESO	Contiene la lista de procesos de cada aplicación. Los procesos que deban ser monitorizados deben estar en esta tabla que tiene una relación con la tabla de aplicaciones.
PROCESO_ALERTA	Es una tabla de relación entre proceso y alerta: define las alertas que aplican a un proceso.

Tabla 14 - Lista de tablas

3.4.2.2 Descripción de las tablas

- **Tabla 15: Tabla ALERTA**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
NOMBRE	VCHAR2(32)	Nombre de la alerta.
DESCRIPCION	VCHAR2(50)	Descripción de la alerta.
TIPO	CHAR(1)	Tipo de la alerta. Campo que sirve de discriminador para la herencia con las tablas ALERTA_TIPO_X. Valores posibles: 1,2,3,4,5
FECHA_ALTA	DATE	Fecha de alta de la alerta
FECHA_MODIFICACION	DATE	Fecha de última modificación de la alerta
FECHA_EJECUCION	DATE	Fecha de la última ejecución de la alerta
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
ACTIVA	CHAR(1)	Indica si la alerta está activa.
NUM_EJECUCIONES	NUMBER	Número total de ejecuciones.

Tabla 15 - Definición de tabla Alerta

PK: ID

- **Tabla 16: ALERTA_PERSONA**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
NOMBRE	VCHAR2(100)	Nombre de la persona
EMAIL	VCHAR2(100)	Email de la persona
TELEFONO	VCHAR2(20)	Teléfono de la persona
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
ID_ALERTA	CHAR(32)	Clave foránea de la alerta
FECHA_ALTA	DATE	Fecha de alta de la persona
FECHA_MODIFICACION	DATE	Fecha de última modificación de la persona

Tabla 16 - Definición de tabla Alerta_Persona

PK: ID

FK ALERTA_PERSONA.ID_ALERTA = ALERTA.ID

- **Tabla 17: ALERTA_TIPO_1**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
TIEMPO	NUMBER	Tiempo máximo para alerta de no finalización, en segundos.

Tabla 17 - Definición de tabla Alerta_Tipo_1

PK: ID
FK: ALERTA_TIPO_1.ID = ALERTA.ID

- **Tabla 18: ALERTA_TIPO_2**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
TIEMPO	NUMBER	Tiempo máximo para alerta de tiempo máximo superado. En segundos.

Tabla 18 - Definición de tabla Alerta_Tipo_2

PK: ID
FK: ALERTA_TIPO_2.ID = ALERTA.ID

- **Tabla 19: ALERTA_TIPO_3**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
CRITICIDAD	CHAR(1)	Criticidad para alertas de proceso finalizado con errores.

Tabla 19 - Definición de tabla Alerta_tipo_3

PK: ID

FK: ALERTA_TIPO_3.ID = ALERTA.ID

- **Tabla 20: ALERTA_TIPO_4**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA

Tabla 20 - Definición de tabla Alerta_Tipo_4

PK: ID
FK: ALERTA_TIPO_4.ID = ALERTA.ID

- **Tabla 21: ALERTA_TIPO_5**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA

Tabla 21 - Definición de tabla Alerta_Tipo_5

PK: ID
FK: ALERTA_TIPO_5.ID = ALERTA.ID

- **Tabla 22: APLICACION**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
NOMBRE	VCHAR2(32)	Nombre de la aplicación. Debe ser único.
DESCRIPCION	VCHAR2(500)	Descripción de la aplicación
NOMBRE_RESPONSABLE	VCHAR2(50)	Nombre de la persona responsable de la aplicación.
EMAIL_RESPONSABLE	VCHAR2(50)	Email de la persona responsable de la aplicación.
CRITICIDAD	CHAR(1)	Criticidad de la aplicación.
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
ACTIVA	CHAR(1)	Indica si la aplicación está activa.
AREA	VCHAR2(50)	Área de negocio de la aplicación.
FECHA_ALTA	DATE	Fecha de alta de la aplicación
FECHA_MODIFICACION	DATE	Fecha de última modificación de la aplicación

Tabla 22 - Definición de tabla Aplicacion

PK: ID

- **Tabla 23: CHECKPOINT**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
NOMBRE	VCHAR2(32)	Nombre del *checkpoint*, debe ser único dentro del proceso.
DESCRIPCION	VCHAR2(150)	Descripción del *checkpoint*.
ID_PROCESO	CHAR(32)	Clave foránea del proceso.

OBLIGADO	CHAR(1)	Indica si el paso por el *checkpoint* es mandatorio dentro del proceso.
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
FECHA_ALTA	DATE	Fecha de alta del *checkpoint*
FECHA_MODIFICACION	DATE	Fecha de última modificación del *checkpoint*
FECHA_EJECUCION	DATE	Fecha de la última ejecución del *checkpoint*.
NUM_EJECUCIONES	NUMBER	Número total de ejecuciones del *checkpoint*.

Tabla 23 - Definición de tabla Checkpoint

PK: ID
FK: CHECKPOINT.ID_PROCESO = PROCESO.ID

- **Tabla 24: CHECKPOINT_ALERTA**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_CHECKPOINT	CHAR(32)	Clave foránea de *checkpoint*.
ID_ALERTA	CHAR(32)	Clave foránea de la alerta.

Tabla 24 - Definición de tabla Checkpoint_Alerta

PK: ID
FK: CHECKPOINT_ALERTA.ID_CHECKPOINT = CHECKPOINT.ID
CHECKPOINT_ALERTA.ID_ALERTA = ALERTA.ID

- **Tabla 25: EMAIL**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_EX_ALERTA	CHAR(32)	Clave foránea de la ejecución de alerta
ID_EX_PROCESO	CHAR(32)	Clave foránea de la ejecución de proceso
DESTINO	VCHAR2(100)	Email del destinatario
ASUNTO	VCHAR2(200)	Asunto del email
CUERPO	VCHAR2(4000)	Cuerpo del email
FECHA_GENERACION	DATE	Fecha en la que el email ha sido generado
FECHA_ENVIO	DATE	Fecha en la que el email ha sido enviado

Tabla 25 - Definición de tabla Email

PK: ID
FK: EMAIL.ID_EX_ALERTA = EX_ALERTA.ID
EMAIL.ID_EX_PROCESO = EX_PROCESO.ID

- **Tabla 26: ERROR**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
NOMBRE	VCHAR2(32)	Nombre del error, debe ser único dentro del proceso
DESCRIPCION	VCHAR2(150)	Descripción del error

ID_PROCESO	CHAR(32)	Clave foránea del proceso
FECHA_ALTA	DATE	Fecha de alta del error
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
CRITICIDAD	CHAR(1)	Criticidad asociada al error
FECHA_MODIFICACIONES	DATE	Fecha de última modificación del error
FECHA_EJECUCION	DATE	Fecha de última ejecución del error
NUM_EJECUCIONES	NUMBER	Número total de ejecuciones

Tabla 26 - Definición de tabla Error

PK: ID

FK: ERROR.ID_PROCESO = PROCESO.ID

- **Tabla 27: ERROR_ALERTA**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_ERROR	CHAR(32)	Clave foránea del error
ID_ALERTA	CHAR(32)	Clave foránea de la alerta

Tabla 27 - Definición de tabla Error_Alerta

PK: ID

FK: ERROR_ALERTA.ID_ERROR = ERROR.ID

ERROR_ALERTA.ID_ALERTA = ALERTA.ID

- **Tabla 28: EX_ALERTA**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_ALERTA	CHAR(32)	Clave foránea de la alerta
FECHA_CREACION	DATE	Fecha de creación del registro de ejecución de alerta
FECHA_PROCESO	DATE	Fecha de ejecución del registro del proceso.
ID_EX_PROCESO	CHAR(32)	Clave foránea de la ejecución del proceso
TEXTO	VCHAR2(2000)	Texto registrado con la ejecución de la alerta para formar el mensaje.

Tabla 28 - Definición de tabla Ex_Alerta

PK: ID

FK: EX_ALERTA.ID_ALERTA = ALERTA.ID

EX_ALERTA.ID_EX_PROCESO = EX_PROCESO.ID

- **Tabla 29: EX_ERROR**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_EX_PROCESO	CHAR(32)	Clave foránea de la ejecución del proceso.
ID_ERROR	CHAR(32)	
FECHA	DATE	
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA

TEXTO	VCHAR2(500)	
CODIGO_ERROR	VCHAR2(50)	
EXCEPCION	VCHAR2(4000)	

Tabla 29 - Definición de tabla Ex_Error

PK: ID
FK: EX_ERROR.ID_EX_PROCESO = EX_PROCESO.ID
EX_ERROR.ID_ERROR = ERROR.ID

- **Tabla 30: EX_PASO**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_EX_PROCESO	CHAR(32)	Clave foránea de la ejecución del proceso.
ID_PASO	CHAR(32)	Calve foránea del *checkpoint*
FECHA	DATE	Fecha de ejecución del paso.
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
TEXTO	VCHAR2(500)	Texto libre asociado al paso que genera el emisor.

Tabla 30 - Definición de tabla Ex_Paso

PK: ID
FK: EX_PASO.ID_EX_PROCESO = EX_PASO.ID
EX_PASO.ID_PASO = CHECKPOINT.ID

- **Tabla 31: EX_PROCESO**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
FICHERO_ORIGEN	BLOB	Fichero origen del proceso
FICHERO_DEST_USR	BLOB	Fichero para el usuario
FICHERO_DEST_CAU	BLOB	Fichero para el CAU
REFERENCIA	VCHAR2(50)	Referencia de la ejecución del proceso. Única para el proceso - emisor
ID_PROCESO	CHAR2(32)	Clave foránea del proceso
EMISOR	VCHAR2(20)	Identificador del emisor de los datos / origen del proceso. Normalmente un CIF
RECEPTOR	VCHAR2(20)	Identificador del receptor de los datos / destino del proceso. Normalmente un CIF
PERFIL_EMISOR	VCHAR2(10)	Datos del perfil del emisor que permiten diferenciar los datos dentro de la compañía del emisor
CANAL	VCHAR2(10)	Canal de envío del fichero de origen: WEB, WEBSERVICE, FTP…
MSG_RECIBIDOS	NUMBER	Número total de mensajes recibidos en el fichero origen, si es un fichero con lotes.

MSG_CORRECTOS	NUMBER	Número total de mensajes procesados correctamente dentro de los mensajes recibidos.
MSG_ERRONEOS	NUMBER	Número total de mensajes procesados con error dentro de los mensajes recibidos.
FECHA_ENVIO	DATE	Fecha de envío del fichero de origen.
FECHA_PROCESO	DATE	Fecha de inicio de proceso del fichero de origen.
ESTADO	VCHAR2(5)	Estado del proceso: PENDIENTE → 0 INICIADO → 1 FIN OK → 2 FIN KO → 3
ID_APLICACION	CHAR2(32)	Clave foránea de la aplicación.
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
TIEMPO	NUMBER	Tiempo total de proceso en segundos.
FECHA_PROCESO_FIN	DATE	Fecha de final del proceso.

Tabla 31 - Definición de tabla Ex_Proceso

PK: ID

FK: EX_PROCESO.ID_PROCESO = PROCESO.ID
EX_PROCESO.ID_APLICACION = APLICACION.ID

- **Tabla 32: PROCESO**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_APLICACION	CHAR(32)	Clave foránea de la aplicación a la que pertenece el proceso.
NOMBRE	VCHAR2(32)	Nombre del proceso. Identificador único mediante el cual se referirán al proceso en la ejecuciones de procesos.
DESCRIPCION	VCHAR2(100)	Descripción del proceso.
PURGA	NUMBER	Número de días en que la ejecución de procesos se mantendrá en el sistema.
ACTIVO	CHAR(1)	Indicador de proceso activo.
FECHA_ALTA	DATE	Fecha de alta del proceso
CRITICIDAD	CHAR(1)	Criticidad del proceso.
URGENCIA	CHAR(1)	Urgencia del proceso.
VERSION	NUMBER(12,0)	Campo para control de modificados del JPA
FECHA_MODIFICACION	DATE	Fecha de última modificación del proceso
FECHA_EJECUCION	DATE	Fecha de la última ejecución del proceso.
NUM_EJECUCIONES	NUMBER	Número total de ejecuciones.
TIEMPO_MEDIO	NUMBER	Tiempo medio del proceso.
DESVIO_TIPICO	NUMBER	Desviación típica respecto al tiempo medio del proceso.
NUM_OK	NUMBER	Número total de procesos finalizados correctamente.

NUM_KO	NUMBER	Número total de procesos finalizados con error.

Tabla 32 - Definición de tabla Proceso

PK: ID
FK: PROCESO.ID_APLICACION = APLICACION.ID
EX_PROCESO.ID_APLICACION = APLICACION.ID

- **Tabla 33: PROCESOS_ALERTA**

Nombre	Tipo	Descripción
ID	CHAR(32)	Identificador de la tabla, generado y gestionado por JPA
ID_PROCESO	CHAR(32)	Clave foránea del proceso
ID_ALERTA	CHAR(32)	Clave foránea de la alerta

Tabla 33 - Definición de tabla Procesos_Alerta

PK: ID
FK: PROCESO_ALERTA.ID_PROCESO = PROCESO.ID
PROCESO_ALERTA.ID_ALERTA = ALERTA.ID

3.4.2.3 Modelo de base de datos

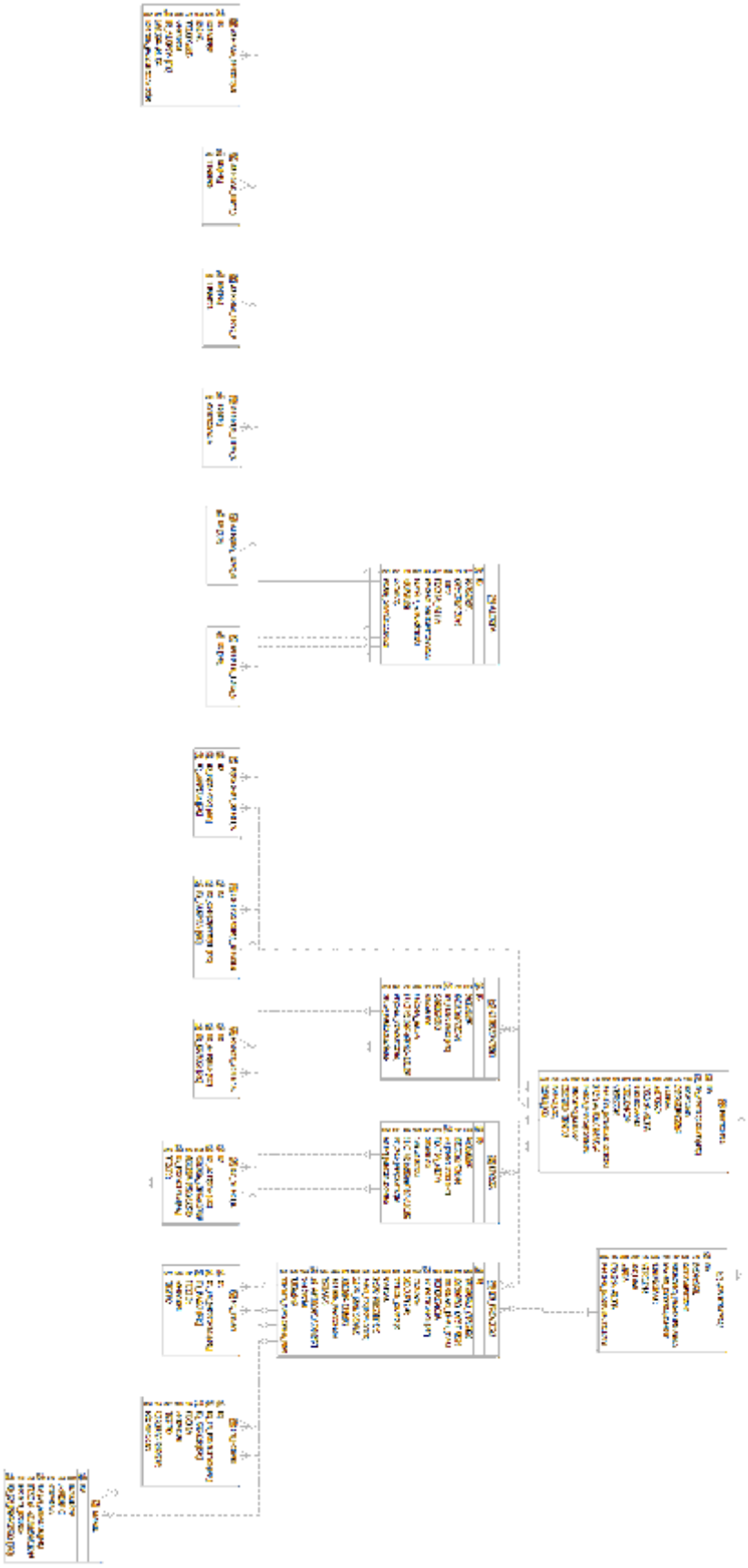


Figura 47 - Diseño de datos

4 Construcción del proyecto

4.1 Configuración del entorno de desarrollo

En este punto se va a describir el conjunto de acciones que deben realizarse en el software base para que la aplicación pueda funcionar. Esto viene determinado por las necesidades de la arquitectura de la aplicación o bien por las propias especificaciones del software. Son tareas previas a la construcción del proyectos ya que sin el software base funcionando correctamente no se puede empezar los desarrollos.

4.1.1 Configuración del servidor de aplicaciones

La versión del WAS utilizado es la siguiente:

IBM WebSphere Application Server for Developers 7.0

WebSphere Application Server for Developers 7.0 es un producto completamente licenciado disponible para descargas sin cargo. Es la versión gratuita del WAS para desarrolladores. Para más información visitar la página de IBM

https://www.ibm.com/developerworks/downloads/ws/wasdevelopers/

Para la descarga de la versión 7 ir directamente a este link:

https://www14.software.ibm.com/webapp/iwm/web/reg/pick.do?source=swg-wsasfd&S_TACT=109J84JW&S_CMP=web_dw_rt_swd&lang=en_US

O a este otro:

http://www.myeclipseide.com/module-htmlpages-display-pid-448.html

Una vez descargado el WAS proceda a su instalación.

Una vez instalado necesitaremos estas dos herramientas para instalar los fix-packs y las extensiones necesarias.

- IBM Update Installer: IBM Update Installer V7.0.0.17 for WebSphere Software for Windows

http://www-01.ibm.com/support/docview.wss?uid=swg24020448&wv=1

32-bit x86 AMD/Intel	7.0.0.17-WS-UPDI-WinIA32.zip

- IBM Installation Manager: IBM Installation Manager 1.7.1

http://www-01.ibm.com/support/docview.wss?uid=swg24035764

32-bit x86 AMD/Intel	agent.installer.win32.win32.x86

- Fix Packs: Descargar los siguientes Fix-pack para el WAS 7.0:

7.0.0.15: WebSphere Application Server V7.0 Fix Pack 15 for Windows

http://www-01.ibm.com/support/docview.wss?uid=swg24029075&wv=1

32-bit x86 AMD/Intel AppServer	7.0.0-WS-WAS-WinX32-FP0000015.pak

7.0.0.15: Java SDK 1.6 SR9 Cumulative Fix for WebSphere Application Server

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24028881&wv=1

Win 32-bit x86 AMD/Intel Java SDK	7.0.0-WS-WASSDK-WinX32-FP0000015.pak

- Extensiones:

http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/index.html

OSGi y JPA2 Feature pack	ariesjpa_and_import_repositories
Instrucciones de instalación	ariesjpa_gettingstarted.zip

- Procedimiento a seguir para la configuración:

- Instalar el IBM WebSphere Application Server
- Instalar el IBM Update Installer
    - IBM Update Installer V7.0.0.17 for WebSphere Software for Windows
- Instalar el IBM Installation Manager
    - IBM Installation Manager 1.7.1
- Instalar fix packs usando el Update Installer
    - 7.0.0.15: WebSphere Application Server V7.0 Fix Pack 15 for Windows
        - Solo el 7.0.0-WS-WAS-WinX32-FP0000015.pak
    - 7.0.0.15: Java SDK 1.6 SR9 Cumulative Fix for WebSphere Application Server
        - 7.0.0-WS-WASSDK-WinX32-FP0000015.pak
- Instalar Feature Packs
    - Desde el Installation Manager (archivo -> preferencias) agregar repositorio apuntando al fichero ariesjpa_and_import_repositories\local-repositories\repository.config.
    - En Installation Manager aparece una opción de Importar, hacer click y especificar el directorio donde se encuentra instalado el WAS, por ejemplo: D:\Archivos de programa\IBM\WebSphere\AppServer
        - Si diera error de conexión con el repositorio al importar, utilizar el siguiente repositorio: http://public.dhe.ibm.com/software/websphere/repositories
    - En Installation Manager ir a opción instalar, Instalar: Paquete de características de IBM WAS V7 para aplicaciones OSGi y Java Persistence API 2.0. Versión 1.0.0.0
    - Desde la Herramienta de Gestión de Perfiles:
        - Seleccionar los perfiles creados y sobre ellos hacer click en Aumentar e instalar capacidades OSGi y JPA2 en ese orden.

En caso de necesitar mayores detalles, recurrir a las instrucciones de instalación descriptas en el fichero: docs/GettingStarted.html contenido dentro del: ariesjpa_gettingstarted.zip

En este punto tenemos el WAS preparado para trabajar. Ahora debemos crear un perfil para poder desplegar nuestras aplicaciones en él. Para crear un perfil en el WAS está la herramienta de gestión de perfiles. Si no está disponible en el menú de inicio de Windows, puede encontrarse en C:\Archivos de programa\IBM\WebSphere\AppServer\bin\ProfileManagement

4.2 Control de acceso

El control de acceso consta de dos partes: autenticación y autorización. La autenticación es el proceso mediante el cual un usuario informa de quien es, la autorización es el proceso mediante el cual se decide si un usuario previamente autenticado puede acceder a un recurso.

Hay dos formas de definir el control de acceso en aplicaciones web: programática o declarativa.

Se entiende por seguridad programática una solución desarrollada a medida para la aplicación programáticamente: la aplicación tiene sus propias tablas de usuarios en la base de datos y maneja internamente quien puede acceder a que. También es la propia aplicación quien solicita al usuario que se identifique.

La seguridad declarativa funciona definiendo unos recursos y unos roles. Para acceder a un recurso determinado se requiere un rol. Un elemento externo a la aplicación gestiona los accesos: si quien intenta acceder tiene un rol que le permite acceder al recurso (autorización). En este caso los usuarios suelen estar en un LDAP o algún repositorio accesible por el sistema que esté realizando la gestión de accesos, y es el propio sistema que controla los accesos el encargado de solicitar identificación al usuario y validarla (autenticación). En JAVA estos sistemas se conocen como JAAS (*Java Autentication and Autorization System*). WebSphere implementa seguridad mediante JAAS, que es la implementada en la aplicación.

Para implementar la seguridad declarativa se han seguido los siguientes pasos:

- Se comprueba que una dirección determinada no está securizada y se puede acceder libremente. En la figura 48 se puede ver el acceso a la aplicación sin autenticación.

Figura 48 - Control de acceso - Acceso no securizado

- Se añade un rol de seguridad al web.xml (Figura 49 y 50)

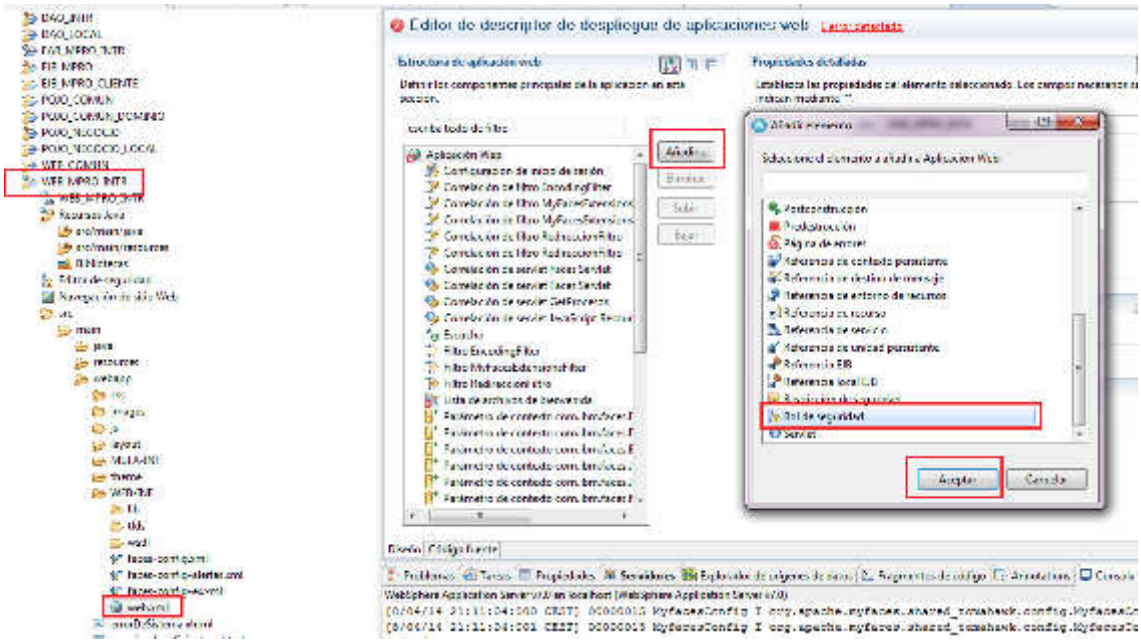


Figura 49 - Control de acceso - Roles en web.xml

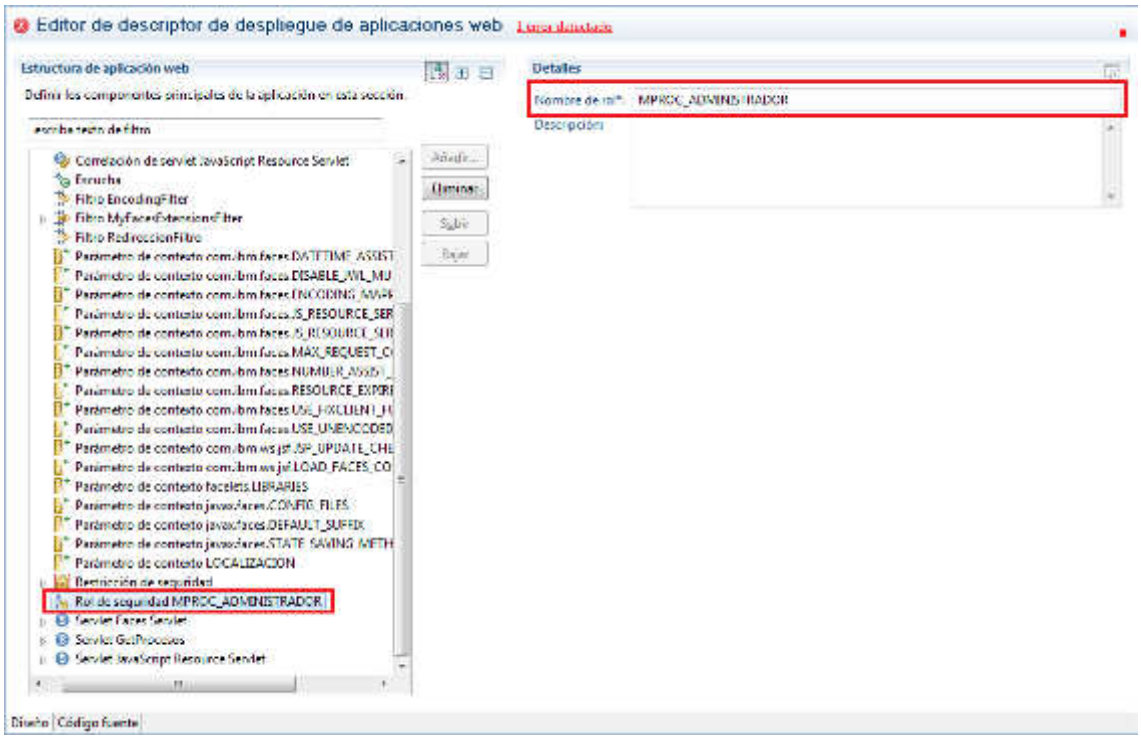


Figura 50 - Control de acceso - Roles en web.xml

- Se añade una restricción de seguridad en el Web.xml del proyecto (Figuras 51, 52 y 53):

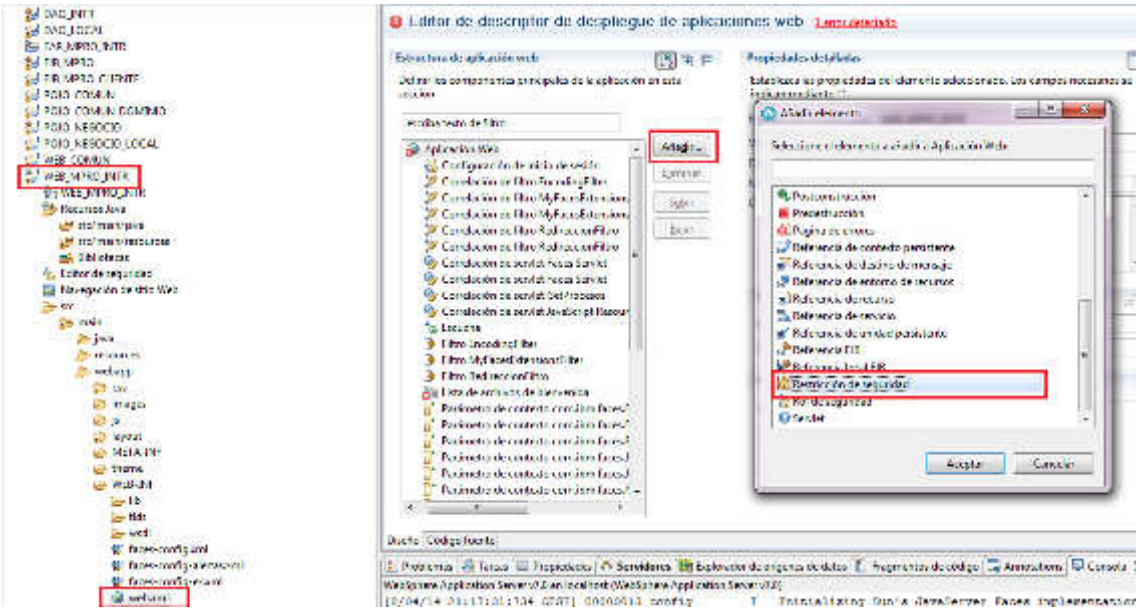


Figura 51 - Control de acceso - Restricciones de seguridad en web.xml

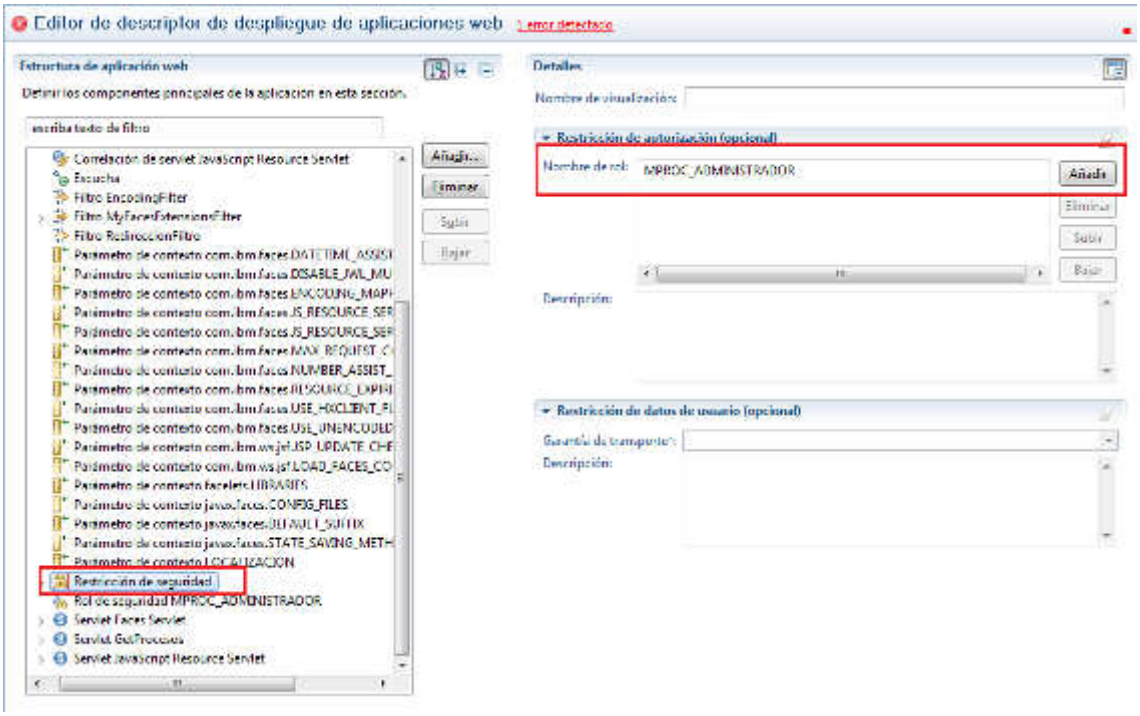


Figura 52 - Control de acceso - Restricciones de seguridad en web.xml

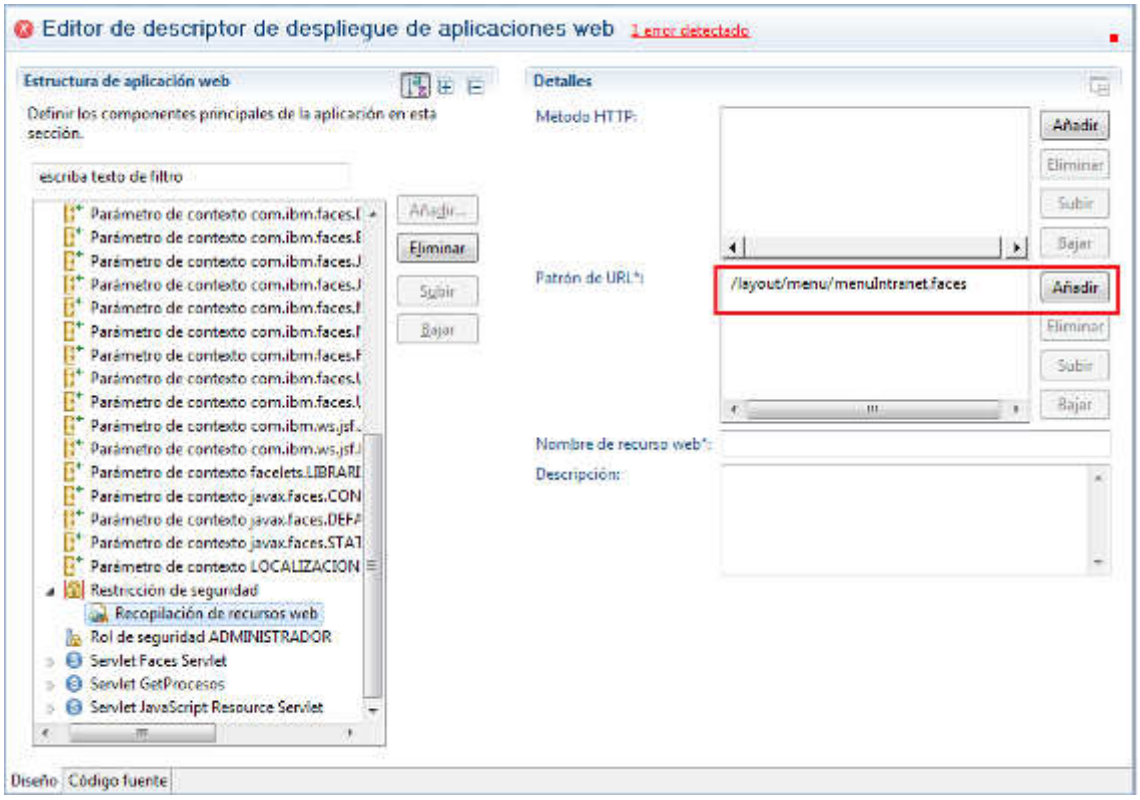


Figura 53 - Control de acceso - Restricciones de seguridad en web.xml

En la última pantalla se puede ver que el recurso /layout/menu/menuIntranet.faces queda protegido de forma que solo un usuario con rol MPROC_ADMINISTRADOR puede entrar.

- Ahora se puede ver que para acceder al recurso solicita autenticarse (Figura 54):

Figura 54 - Control de acceso - Acceso securizado

- En el WAS creamos el grupo MPROC_ADMINISTRADOR (Figuras 55 y 56):

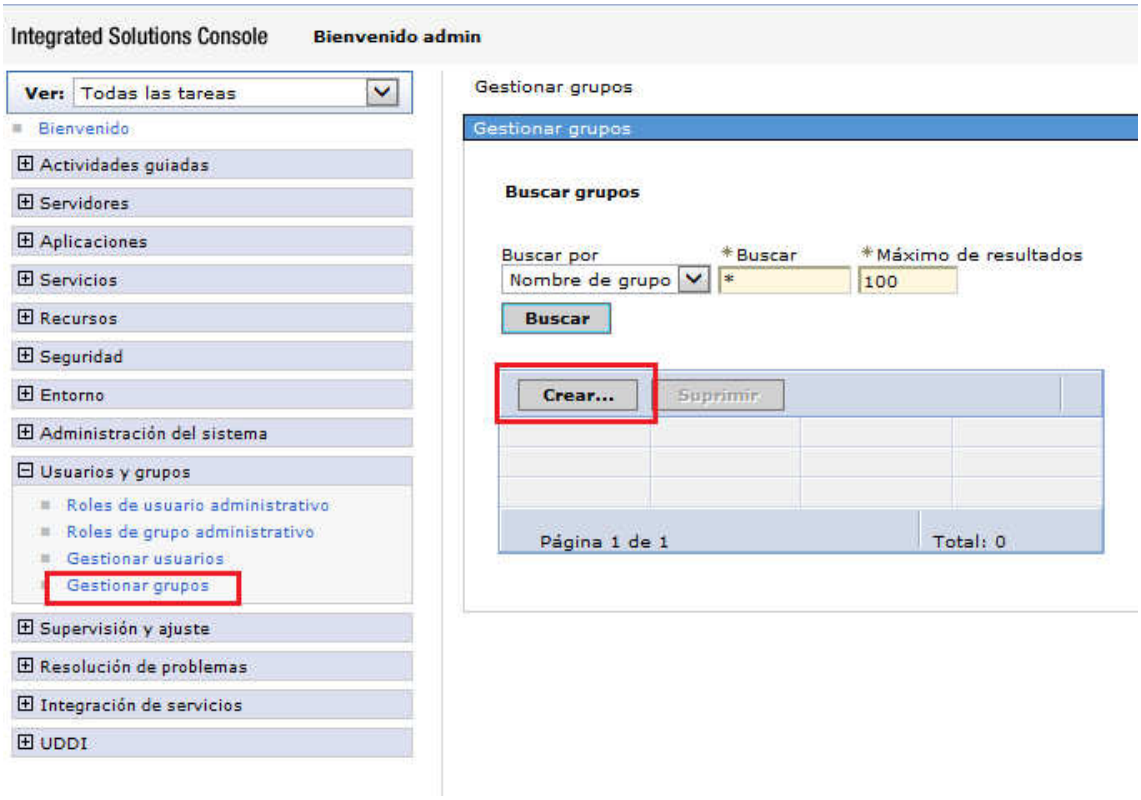


Figura 55 - Control de acceso - Creación de grupos en WAS

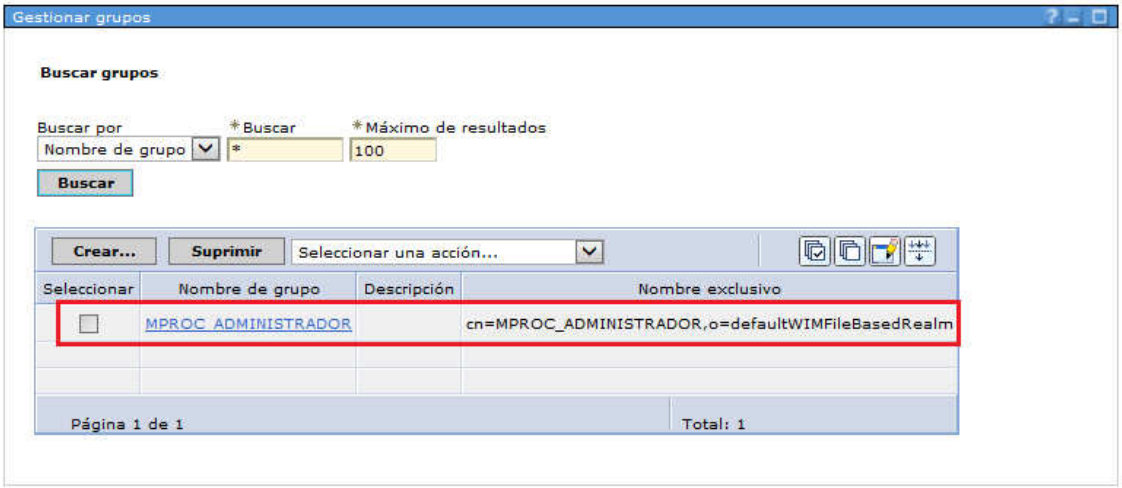


Figura 56 - Control de acceso - Creación de grupos en WAS

- Creamos un usuario dentro del grupo (Figuras 57 y 58):

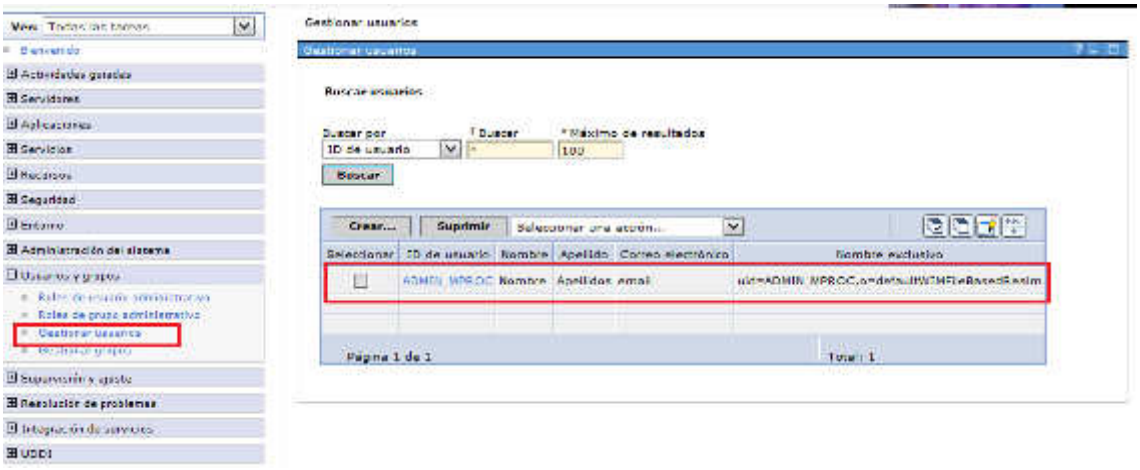


Figura 57 - Control de acceso - Creación de usuarios en WAS

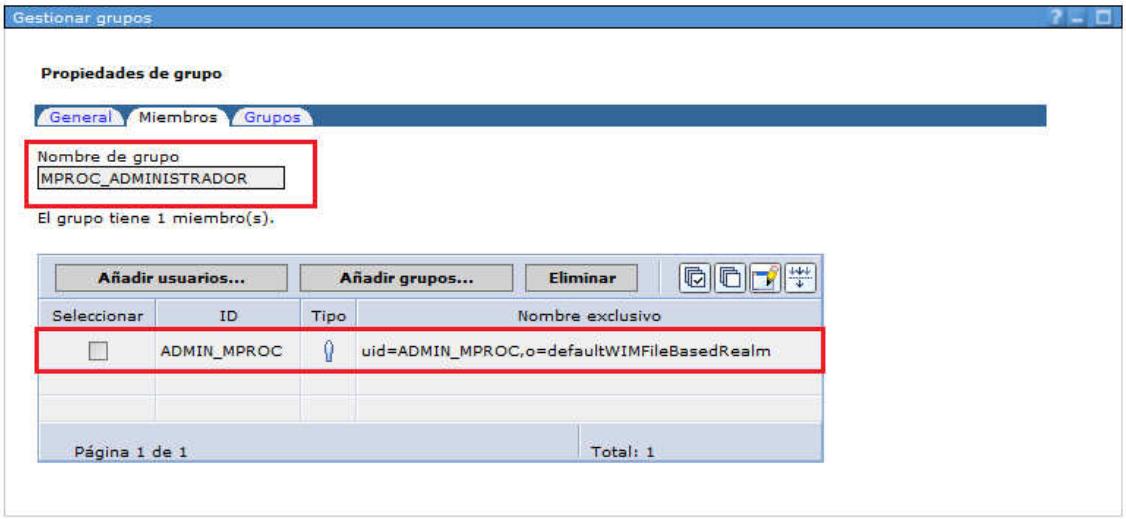


Figura 58 - Control de acceso - Creación de usuarios en WAS

- En la aplicación, asociamos el ROL al grupo que hemos creado (Figuras de 59 a 62):

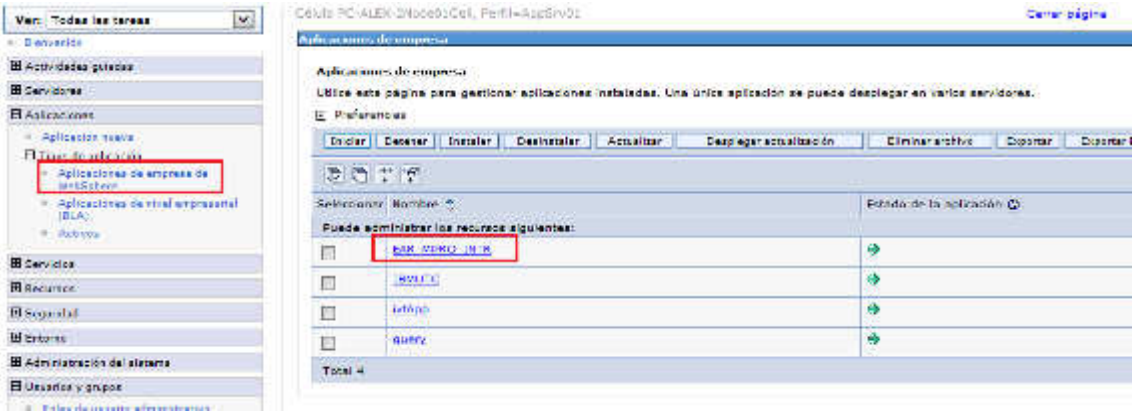


Figura 59 - Control de acceso - Asociación de grupos con roles

Figura 60 - Control de acceso - Asociación de grupos con roles

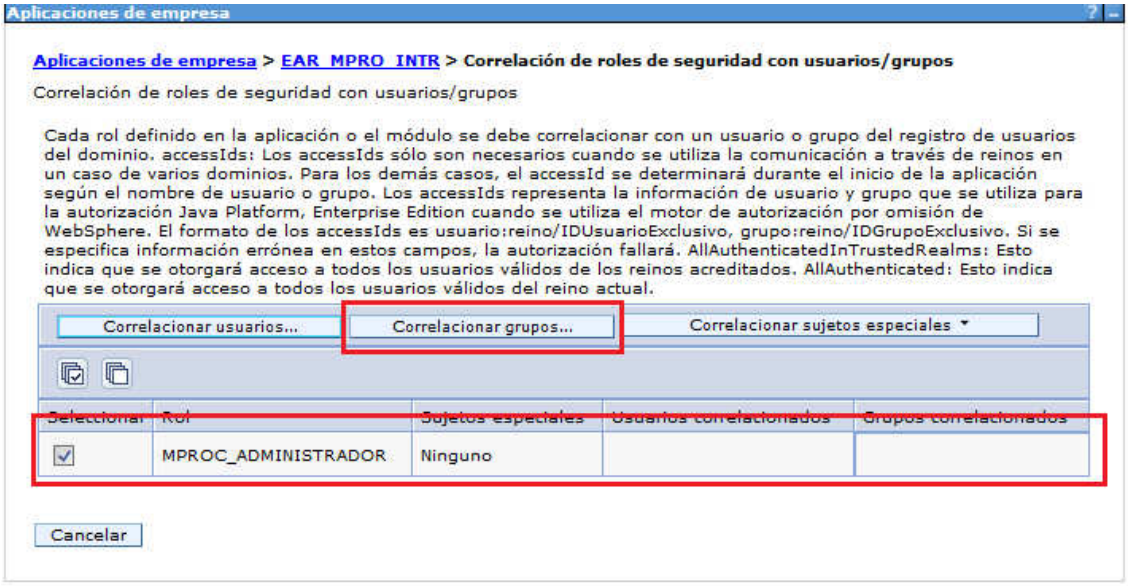


Figura 61 - Control de acceso - Asociación de grupos con roles

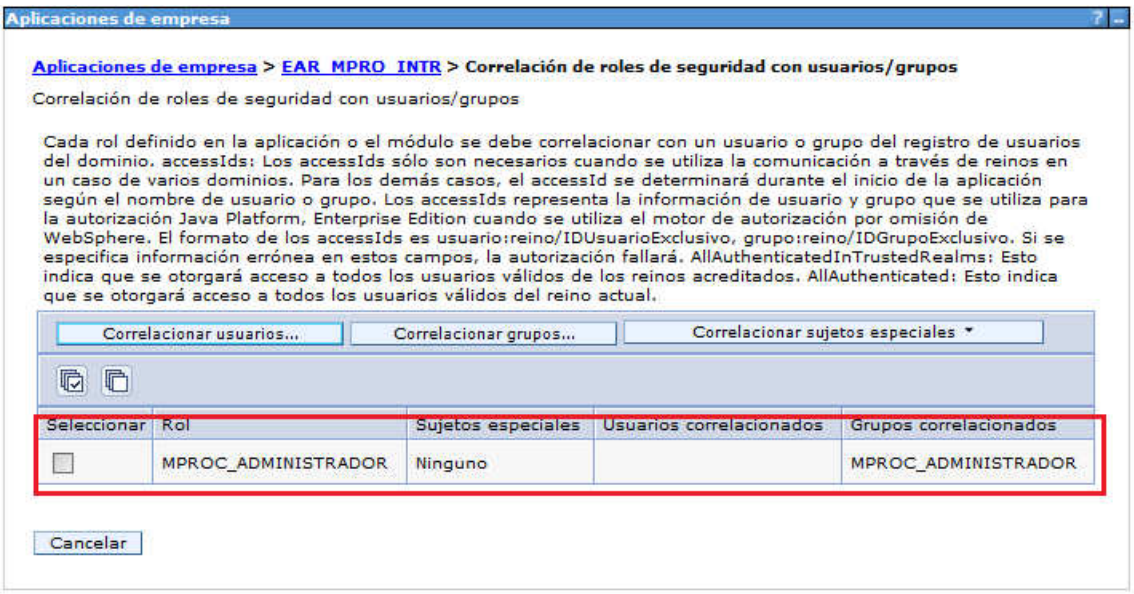


Figura 62 - Control de acceso - Asociación de grupos con roles

Así queda vinculado el grupo con el ROL de forma que todos los usuarios del grupo puedan acceder a los recursos asignados a su rol.

Se requieren crear los grupos (o roles) mostrados en la Tabla 34:

GRUPO	ROL	DESCRIPCION
MPROC_ADMIN	MPROC_ADMIN	Grupo de administración con acceso a todos los recursos.
MPROC_SUPERVISOR	MPROC_SUPERVISOR	Grupo supervisor, corresponde con el usuario de consulta de la compañía, con acceso a la pantallas de consulta
MPROC_EXTERNO	MPROC_EXTERNO	Grupo supervisor, corresponde con el usuario de consulta externo a la compañía, con acceso a la pantallas de consulta

Tabla 34 - Lista de grupos/roles

4.3 Configuración aplicación

La aplicación se parametriza mediante la consola del WAS en todo lo relativo a seguridad, tal y como se expone en el apartado 4.2

Para parámetros adicionales que se puedan requerir se ha creado el fichero de propiedades \opt\appdata\conf\MPRO\configs\parametros_DE.properties

Para el acceso a datos la aplicación utiliza el pool de conexiones gestionado por el WebSphere. Debe configurarse el *pool* mostrado en la figura 63:

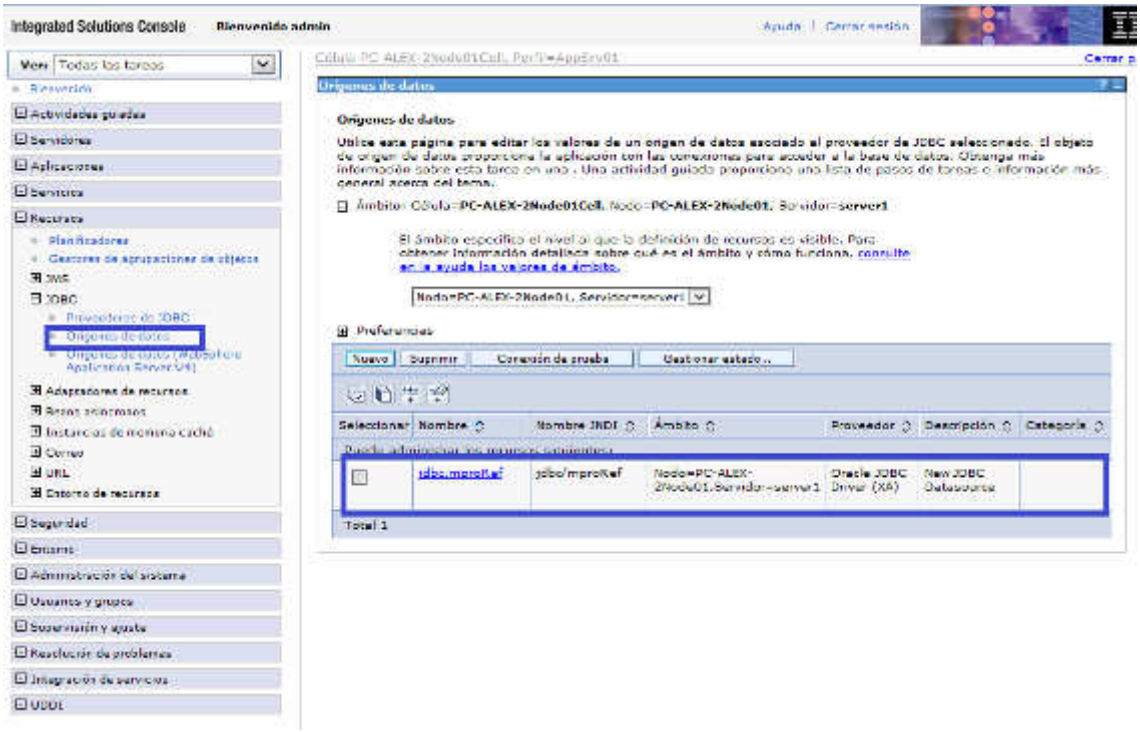


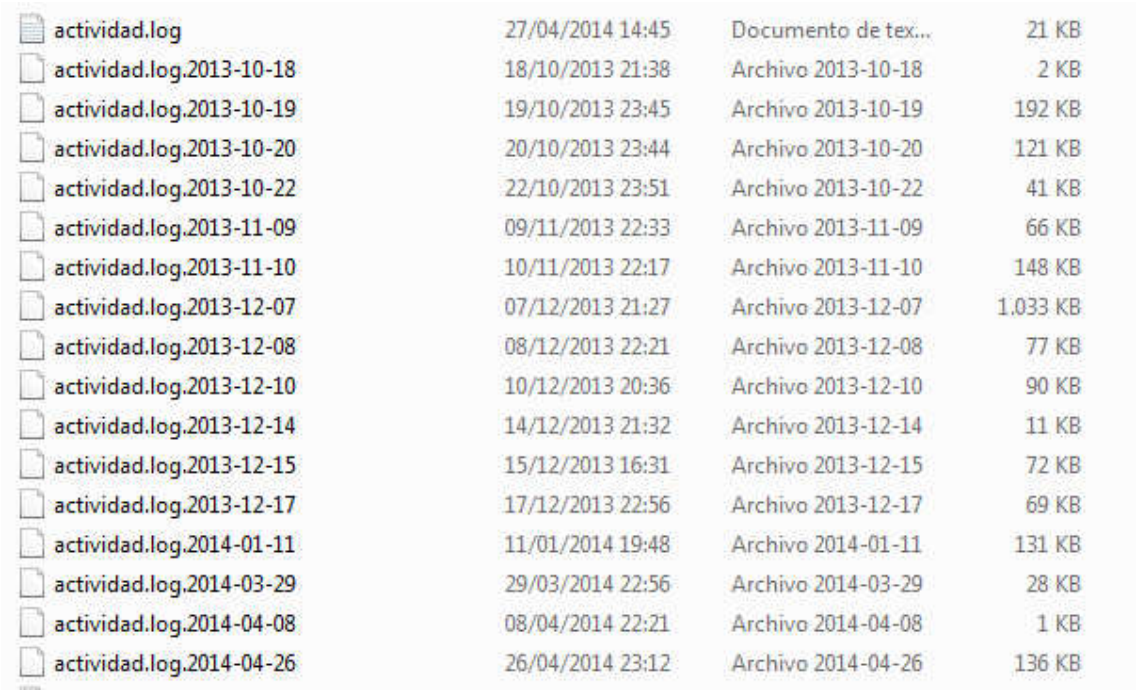
Figura 63 - pool de conexiones en el WAS

4.4 Logs

La aplicación deja en el directorio /var/log/aplicaciones/MPRO/ un conjunto de logs agrupados por:

- Actividad: logs que reflejan la actividad del sistema sin entrar en detalle
- Debug: logs detallados de la actividad del sistema
- Error: Logs de los errores que se producen en el sistema

Los logs son diarios, cada día se renombra el log del día anterior y se genera uno nuevo automáticamente, tal y como se muestra en la figura 64.



actividad.log	27/04/2014 14:45	Documento de tex...	21 KB
actividad.log.2013-10-18	18/10/2013 21:38	Archivo 2013-10-18	2 KB
actividad.log.2013-10-19	19/10/2013 23:45	Archivo 2013-10-19	192 KB
actividad.log.2013-10-20	20/10/2013 23:44	Archivo 2013-10-20	121 KB
actividad.log.2013-10-22	22/10/2013 23:51	Archivo 2013-10-22	41 KB
actividad.log.2013-11-09	09/11/2013 22:33	Archivo 2013-11-09	66 KB
actividad.log.2013-11-10	10/11/2013 22:17	Archivo 2013-11-10	148 KB
actividad.log.2013-12-07	07/12/2013 21:27	Archivo 2013-12-07	1.033 KB
actividad.log.2013-12-08	08/12/2013 22:21	Archivo 2013-12-08	77 KB
actividad.log.2013-12-10	10/12/2013 20:36	Archivo 2013-12-10	90 KB
actividad.log.2013-12-14	14/12/2013 21:32	Archivo 2013-12-14	11 KB
actividad.log.2013-12-15	15/12/2013 16:31	Archivo 2013-12-15	72 KB
actividad.log.2013-12-17	17/12/2013 22:56	Archivo 2013-12-17	69 KB
actividad.log.2014-01-11	11/01/2014 19:48	Archivo 2014-01-11	131 KB
actividad.log.2014-03-29	29/03/2014 22:56	Archivo 2014-03-29	28 KB
actividad.log.2014-04-08	08/04/2014 22:21	Archivo 2014-04-08	1 KB
actividad.log.2014-04-26	26/04/2014 23:12	Archivo 2014-04-26	136 KB

Figura 64 - Logs

En la carpeta /var/log/aplicaciones/ en el fichero eventos.log se registran los errores graves, de forma que si varias aplicaciones usan el mismo sistema de logs revisando este fichero se puede realizar una auditoría de todas las aplicaciones.

En el fichero log4j.xml que se encuentra en el módulo POJO_COMUN se puede configurar el nivel de log así como los ficheros que genera.

La aplicación utiliza la clase TrazasUtil para gestionar los logs. Para cualquier cambio en la gestión de logs basta con modificar esa clase.

4.5 Módulos

La aplicación está compuesta por los módulos mostrados en la tabla 35:

Nombre del módulo	Descripción
DAO_INTR	Implementación de las interfaces de acceso a datos.
DAO_LOCAL	Interfaces de acceso a datos
POJO_NEGOCIO	Implementación de los servicios de negocio
POJO_NEGOCIO_LOCAL	Interfaces de los servicios de negocio
POJO_COMUN	Capa transversal con elementos comunes como las entidades, las excepciones...
POJO_COMUN_DOMINIO	Capa transversal de dominio con todos los DTOs de la aplicación. Contiene los DTOs del dominio interno como del externo (EJBs remotos)
WEB_COMUN	Módulo con utilidades para otros módulos de presentación, tanto WEB como WebServices
WEB_MPRO_INTR	Módulo web con los componentes de la presentación WEB.

WS_PROCESOS	Módulo WebService con los servicios que se exponen para la recepción de datos de ejecuciones de procesos.
EJB_MPRO	Implementación de los EJBs remotos para la inserción de datos desde otras aplicaciones mediante EJBs remotos
EJB_MPRO_CLIENTE	Interfaces de los EJBs remotos.
EAR_MPRO_INTR	EAR empaquetado de toda la aplicación.

Tabla 35 - Módulos de la aplicación

La tabla 36 muestra las relaciones entre módulos:

Módulo	Relación
DAO_INTR	POJO_COMUN POJO_COMUN_DOMINIO DAO_LOCAL
DAO_LOCAL	POJO_COMUN POJO_COMUN_DOMINIO
POJO_NEGOCIO	POJO_COMUN POJO_COMUN_DOMINIO DAO_LOCAL POJO_NEGOCIO_LOCAL
POJO_NEGOCIO_LOCAL	POJO_COMUN POJO_COMUN_DOMINIO
POJO_COMUN	
POJO_COMUN_DOMINIO	
WEB_COMUN	POJO_COMUN POJO_COMUN_DOMINIO
WEB_MPRO_INTR	POJO_COMUN POJO_COMUN_DOMINIO WEB_COMUN POJO_NEGOCIO_LOCAL
WS_PROCESOS	POJO_COMUN POJO_COMUN_DOMINIO WEB_COMUN POJO_NEGOCIO_LOCAL
EJB_MPRO	POJO_COMUN POJO_COMUN_DOMINIO POJO_NEGOCIO_LOCAL EJB_MPRO_CLIENTE
EJB_MPRO_CLIENTE	POJO_COMUN POJO_COMUN_DOMINIO
EAR_MPRO_INTR	DAO_INTR DAO_LOCAL POJO_NEGOCIO POJO_NEGOCIO_LOCAL POJO_COMUN POJO_COMUN_DOMINIO WEB_COMUN WEB_MPRO_INTR WS_PROCESOS EJB_MPRO

	EJB_MPRO_CLIENTE

Tabla 36 - Relaciones entre módulos de la aplicación

4.6 Interfaces

La interface para informar de las ejecuciones de procesos se muestra en la figura 65:

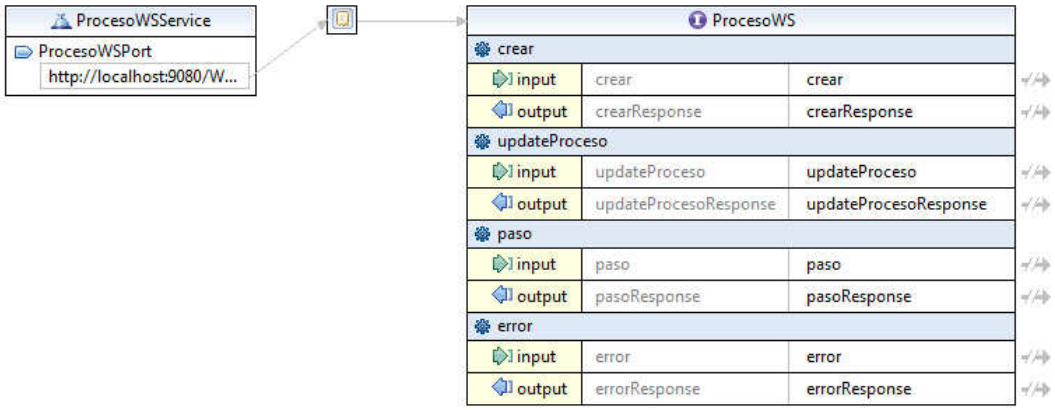


Figura 65 - Interface principal

Un servicio Web llamado ProcesoWSService consta de 4 métodos, como se puede ver en la tabla 37:

Nombre del método	Descripción
crear	Informa que un proceso se ha planificado para su ejecución
updateProceso	Permite modificar datos de la ejecución del proceso en cualquier momento. Hay algunos datos que no se van a tener hasta el final de la ejecución, como los ficheros.
paso	Registra la ejecución de un paso en el proceso. Esto es, que la aplicación está pasando por un punto de chequeo predefinido.
error	Registra la ejecución de un error en el proceso. Esto es, que la aplicación está lanzando un error predefinido.

Tabla 37 - Métodos de la interface principal

Para crear y modificar un proceso se deben facilitar los datos que muestra la tabla 38:

Parámetro	Obligatorio	Descripción
ficheroDestinoCau	No	Fichero para el CAU
ficheroDestinoUsr	No	Fichero para el usuario
msgCorrectos	No	Número de mensajes procesados correctamente
msgErroneos	No	Número de mensajes procesados con algún error
msgRecibidos	No	Número total de mensajes recibidos
aplicacion	Si	Identificador de la aplicación que realiza el proceso
canal	Si	Identificador del canal por el que se ha recibido el fichero: web, WS, FTP…
emisor	Si	Identificador del emisor del fichero
ficheroOrigen	No	Fichero original

perfilEmisor	No	Perfil del emisor (permite tener emisores diferenciados con un mismo identificador)
proceso	Si	Identificador del proceso
receptor	Si	Identificador del receptor
referencia	Si	Referencia única por emisor - proceso

Tabla 38 - Datos de ejecución de proceso

Para crear un paso, los datos que muestra la tabla 39:

Parámetro	Obligatorio	Descripción
aplicacion	Si	Identificador de la aplicación que realiza el proceso
comentario	No	Texto libre
emisor	Si	Identificador del emisor del fichero
paso	Si	Identificador del paso
proceso	Si	Identificador del proceso
referencia	Si	Referencia única por emisor - proceso

Tabla 39 - Datos de paso

Para crear un error, los datos que muestra la tabla 40:

Parámetro	Obligatorio	Descripción
aplicación	Si	Identificador de la aplicación que realiza el proceso
Comentario	No	Texto libre
Emisor	Si	Identificador del emisor del fichero
errorCode	Si	Identificador del error
errorName	No	Nombre del error dentro de la aplicación que realiza el proceso
exception	No	Si el error está relacionado con una excepción se puede registrar la excepción que se ha lanzado
proceso	Si	Identificador del proceso
referencia	Si	Referencia única por emisor - proceso

Tabla 40 - Datos de error

4.7 Base de datos

La base de datos debe tener un apartado propio en la construcción del proyecto debido al uso que se le ha dado que va más allá de ser un repositorio de datos. Se ha utilizado la potencia de los procedimientos almacenados en Pl Sql de Oracle, el lanzamiento de estos procesos mediante triggers, el Oracle Scheduler para la planificación de Jobs basados en procedimientos pl sql, el envío de mails mediante el paquete UTL_MAIL…

Se han creado los procedimientos almacenados en PL-SQL que muestra la tabla 41:

Nombre	Descripción
ALTAEXALERTA	Realiza el alta de una alerta
CHECKALERTAPROCESOCHECKPOINTS	Comprueba si un *checkpoint* tiene asociada una alerta y, si la tiene, la genera.
PROCESAALERTAS	Procesa las alertas y genera los emails
ASUNTO_MAIL	Genera el asunto de los emails
DATOSDEFINICIONPROCESO	Genera un *string* con los datos del proceso para el envío de emails.
DATOSEJECUCIONPROCESO	Genera un *string* con los datos de ejecución de un proceso para el envío de emails.

time_diff	Calcula la diferencia entre dos fechas en segundos

Tabla 41 - Procedimientos almacenados

4.8 Compilación y despliegue

Desde el propio RSA se realiza la compilación de forma automática. Para la generación del EAR (Enterprise ARchive) con toda la aplicación desde el RSA se puede generar a través del menú con la opción Archivo → Exportar. El despliegue del EAR se realiza en la consola del WAS. Para más información ver la documentación de IBM.

5 Demo de la aplicación

En este apartado se pretende realizar una demostración de la aplicación completa mostrando sus funcionalidades. Para ello disponemos de la aplicación funcionando, un navegador y las herramientas necesarias descritas en el apartado Tecnología empleada – Herramientas.

En la demo vamos a simular una empresa que tiene varias tiendas virtuales cuyo comportamiento quiere monitorizar para detectar de forma proactiva los problemas que se producen y resolverlos. Las tiendas virtuales tienen varios procesos que se podrían agrupar en dos grupos: procesos de integración entre la tienda virtual y el fabricante o vendedor y procesos de venta entre la tienda virtual y los compradores. En el primer grupo estarían procesos de integración del catálogo de venta, de precios, ofertas y de gestión de pedidos y órdenes de compra. En el segundo grupo habrían procesos de muestra del catálogo, de gestión de clientes y de venta. Con estos datos iremos dando de alta los distintos procesos para cubrir nuestras tiendas virtuales.

La pantalla inicial de la aplicación es la siguiente (figura 66):

Figura 66 - Pantalla inicial

En “Administración de procesos” vamos a gestionar todas las tareas administrativas de aplicaciones, procesos y alertas.

En “Consulta de ejecuciones de procesos” vamos a ver las ejecuciones de procesos.

En “Informes de procesos” vamos a poder exportar datos de la aplicación con fines estadísticos.

A nivel general, en toda la aplicación podemos ver en qué lugar nos encontramos viendo el hilo de Ariadna que se nos muestra en la cabecera debajo del nombre de la aplicación (marcado en azul en esta captura, figura 67):

Figura 67 - hilo de Ariadna

Posteriormente una etiqueta nos indica que estamos viendo (marcado en azul en esta captura, figura 68):

Figura 68 - Título de página

Las tablas que aparecen en toda la aplicación para mostrar listados se pueden ordenar en función del campo que se desee pulsando sobre el título de la columna. Debajo de cada tabla indica el total de registros que contiene, el número de registros que muestra, la página que está mostrando, el total de páginas y botones para avanzar, retroceder ir al inicio o ir al final de las páginas (figura 69):

Figura 69 - Listados de datos

Otro punto a mostrar a nivel global en la aplicación son los errores. Si se produce un error en la aplicación se va a mostrar al final del formulario en rojo (figura 70):

Figura 70 - Errores de la aplicación

5.1 Aplicaciones

Entrando en la primera opción del menú principal vemos la lista de aplicaciones que existen en el sistema (figura 71):

Figura 71 - Lista de aplicaciones

Mediante “Alta aplicación“ procedemos con el alta de una nueva aplicación (figura 72):

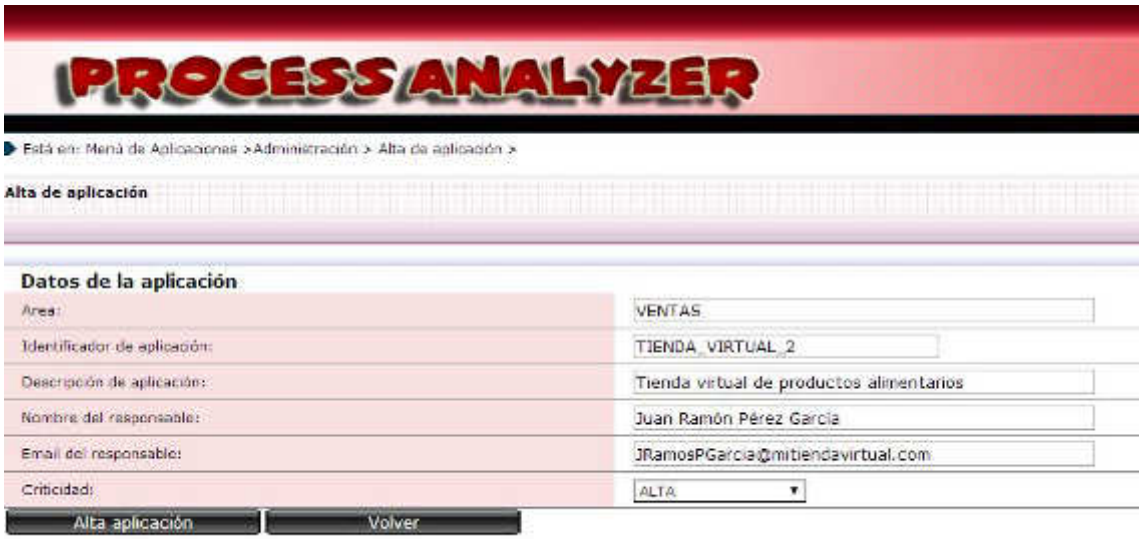


Figura 72 - Alta de aplicación nueva

Aparece la tienda nueva en el listado (encuadrada en azul, figura 73):

Figura 73 - Alta de aplicación realizada

Pulsando sobre ella vamos a ver el detalle de la aplicación (figura 74):

Figura 74 - Detalle de aplicación

Mediante “Modificar aplicación“ podemos modificar sus datos:

Figura 75 - Modificación de aplicación

El “Identificador de aplicación” es un dato mediante el cual se identifica la aplicación tanto internamente como externamente, por lo que es un dato que no puede modificarse.

Mediante “Eliminar aplicación” se eliminaría la aplicación y todos los datos relacionados.

5.2 Procesos

Para proceder con la administración de los procesos se debe ir al detalle de la aplicación cuyos procesos se desean administrar. En el ejemplo procedemos a ir al detalle de la aplicación “TIENDA_VIRTUAL_2“:

Figura 76 - Detalle de aplicación - Lista de procesos

Mediante "Añadir proceso" se añaden procesos nuevos a la aplicación. Vamos a añadir un proceso de integración de catálogo de ventas:

Figura 77 - Nuevo proceso

Aquí aparecen dos conceptos: Criticidad y urgencia. La urgencia se refiere al margen de demora para resolver una incidencia relativa al proceso. La criticidad marca la importancia del proceso en sí. Son conceptos que van muy unidos pero no tiene por qué tener el mismo valor ya que realmente se refieren a cosas distintas. En este ejemplo la integración del catálogo de ventas es crítico porque en él se van a basar las ventas, pero si falla no hay una caída del servicio, la tienda sigue en marcha sin actualizar precios, por lo que no tiene una urgencia elevada. Los problemas con esta integración son importantes, deben resolverse cuanto antes, pero no tiene por qué pasar por delante de otros problemas. Estas categorías ayudan a clasificar las incidencias dentro de los procedimientos de mantenimiento y tienen impacto en los acuerdos de nivel de servicio que se establecen ya que en algunos casos las demoras en la resolución de incidencias de carácter urgente o crítico pueden traer penalizaciones. Hay otros criterios para categorizar incidencias en procesos tales como severidad o impacto, pero estos criterios categorizan las incidencias, no el proceso en sí, por lo que no se puede asociar el proceso per se a una categoría

de este tipo. Como ejemplo, una caída de la base de datos en el sistema tiene un impacto elevado independientemente del proceso al que se asocie este problema.

Podemos ver la aplicación con el proceso nuevo (marcado en azul, figura 78):

Figura 78 - Detalle de aplicación - Lista de procesos

Pulsando sobre el proceso podemos ver su detalle (figura 79):

Figura 79 - Detalle de proceso

Es esta pantalla se muestra inicialmente un resumen de la aplicación, el detalle del proceso y los datos estadísticos, actualmente en estado inicial ya que acabamos de dar de alta el proceso.

Luego hay tres bloques: lista de puntos de chequeo, lista de errores y lista de alertas. Cada uno de estos bloques muestra los datos asociados al proceso. Pulsando sobre ellos se puede entrar en el detalle de cada uno, como se muestra a continuación.

A continuación nos muestra una lista de botones con las funciones que podemos realizar en este punto:

- Modificar proceso: muestra una pantalla donde se permite modificar los datos del proceso.
- Eliminar proceso: Elimina el proceso y todos sus datos asociados.
- Alta *checkpoint*: muestra una pantalla para el alta de un punto de chequeo asociado al proceso.
- Alta error: muestra una pantalla para el alta de un error asociado al proceso.
- Alta de alerta: muestra una pantalla para el alta de una alerta asociada al proceso.

En la descripción de los requisitos funcionales se detalla que en el alta de un proceso se dan de alta automáticamente tres puntos de chequeo (INICIO, FIN y UNK) y un error (UNK). Estos datos sirven para dar la funcionalidad básica al sistema, a partir de aquí ya puede funcionar asociando todos los puntos de paso a UNK y todos los errores a UNK. No obstante se pueden detallar más puntos de paso y errores.

En la pantalla del detalle del proceso, pulsando sobre modificar muestra una pantalla donde permite modificar los datos (figura 80):

Figura 80 - Modificación de proceso

El proceso puede activarse o desactivarse de forma que el sistema deje de admitir datos de ejecución de los procesos desactivados.

5.2.1 Chekpoints

En la pantalla del detalle del proceso se muestra una lista de los puntos de chequeo del proceso. Pulsando sobre el punto de chequeo “INICIO” podemos ver el detalle de este (figura 81):

Figura 81 - Detalle de checkpoint

Esta pantalla nos muestra un resumen de la aplicación, un resumen del proceso y los datos punto de chequeo. También nos muestra datos estadísticos sobre las ejecuciones del punto de chequeo. Posteriormente nos muestra una lista de las alertas asociadas al punto de chequeo. En este particularmente podemos ver dos cosas: no tiene alertas y no tiene ningún botón que permita su eliminación ni su modificación. Es un dato maestro de la aplicación así que ni su modificación ni su eliminación están permitidas ya que la aplicación no funcionaría correctamente sin este dato. Igual pasa con el resto de puntos de chequeo iniciales: "INICIO", "FIN" y "UNK".

Procedemos a dar de alta un nuevo punto de chequeo del proceso de integración de datos de ventas. Para ello, en la pantalla del detalle del proceso de integración pulsamos sobre el botón "Alta *checkpoint*" (figura 82):

Figura 82 - Alta de checkpoint

La integración de un catálogo de ventas tiene un paso inicial de parseado del fichero y sus datos. El paso por este punto es obligatorio ya que no se puede integrar un fichero sin haberlo parseado antes. Procedemos con el alta (figura 83):

Figura 83 - Detalle de proceso - Lista de checkpoints

Pulsando sobre el punto de chequeo (figura 84):

Figura 84 - Detalle de checkpoint

Se puede modificar el punto de chequeo (figura 85):

Figura 85 - Modificación de checkpoint

Se puede eliminar el punto de chequeo y todos sus datos asociados o asociarle alertas. Esto último se verá más adelante.

5.2.2 Errores

En la pantalla del detalle del proceso se muestra una lista de los errores del proceso. Pulsando sobre el error “UNK” podemos ver el detalle de este (figura 86):

Figura 86 - Detalle de error

Es un error especial de la aplicación que recogerá todos los errores desconocidos. No puede eliminarse pero si se le pueden asociar alertas.

Procedemos a dar de alta un nuevo error del proceso de integración de datos de ventas. Para ello, en la pantalla del detalle del proceso de integración pulsamos sobre el botón “Alta error” (figura 87):

Figura 87 - Alta de error

Si se produce un error en el parseo del fichero con el catálogo de ventas se enviará este error. Es un error con criticidad alta ya que si el catálogo no se puede leer correctamente no se va a integrar en el sistema. La lista de errores de proceso se ve incrementada con este error nuevo (figura 88):

Figura 88 - Detalle de proceso- listado de errores

Clicando sobre él podemos ver su detalle (figura 89):

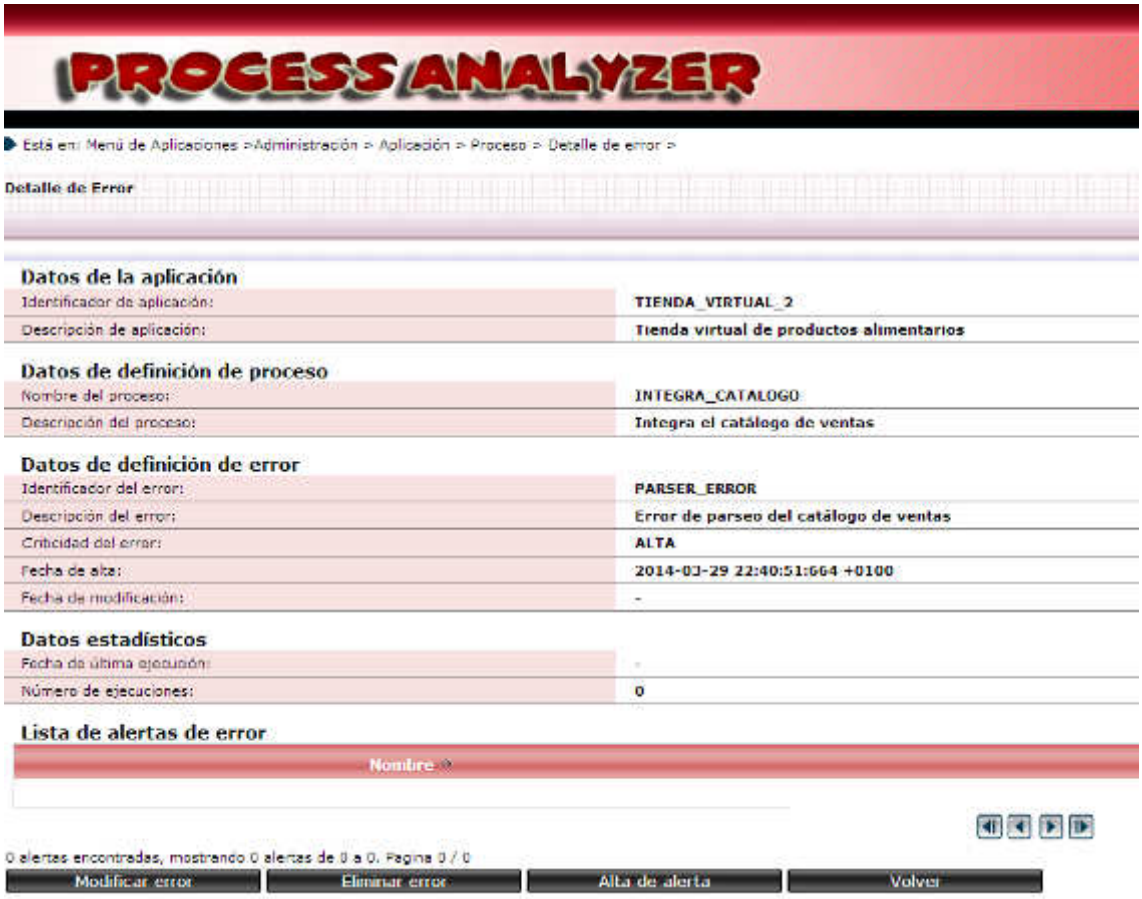


Figura 89 - Detalle de error

Está dividido en varios bloques: un resumen de la aplicación, un resumen del proceso, los datos del error y los datos estadísticos. Luego muestra una lista de las alertas asociadas a este error y el conjunto de acciones disponibles:

- Modificar error: Permite modificar los datos del error.
- Eliminar error: Elimina el error y los datos asociados a él.
- Alta de alerta: Procede con el alta de una alerta asociada a este error.

Si se edita el error permite modificar sus datos (figura 90):

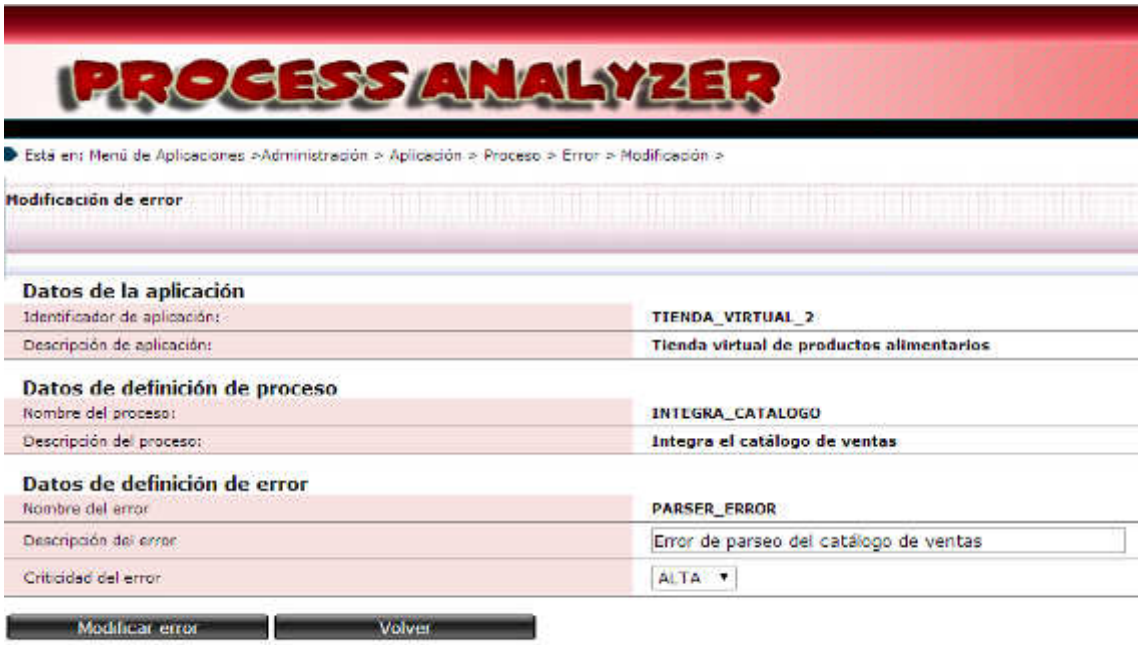


Figura 90 - Modificación de error

5.3 Alertas

5.3.1 Alertas de error

Para dar de alta una alerta asociada al error, ir al detalle del error (figura 91):

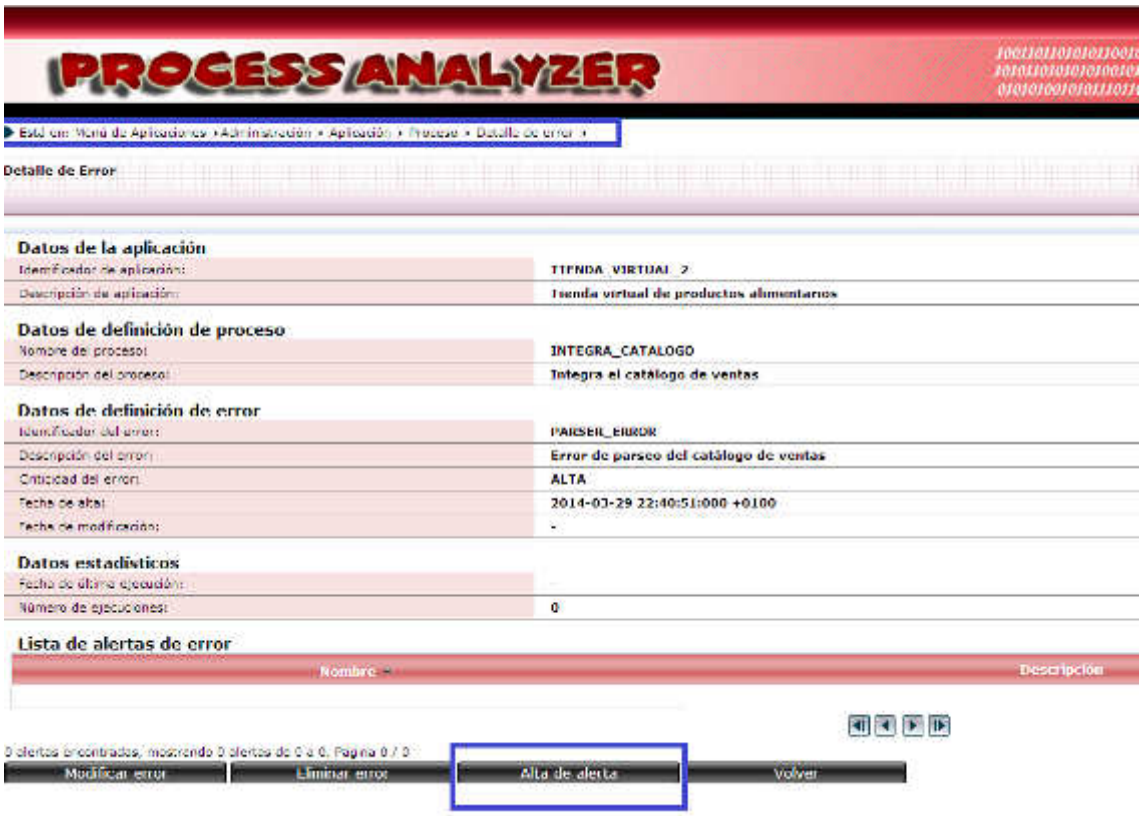


Figura 91 - Detalle de error

Y pulsar “Alta de alerta” (figura 92):

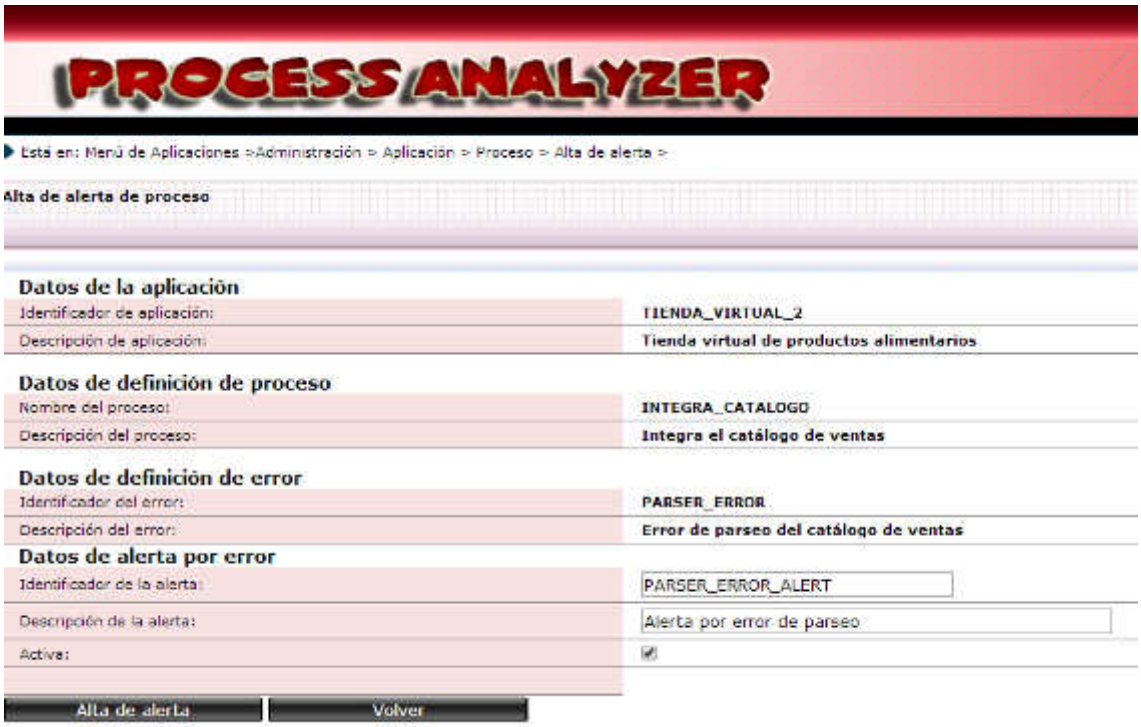


Figura 92 - Alta de alerta por error

Rellenando los datos se procede al alta de la alerta (figura 93).

Figura 93 - Alta de alerta de error realizada

Pulsando sobre la alerta se puede ver los detalles y añadirle destinatarios (figura 94):

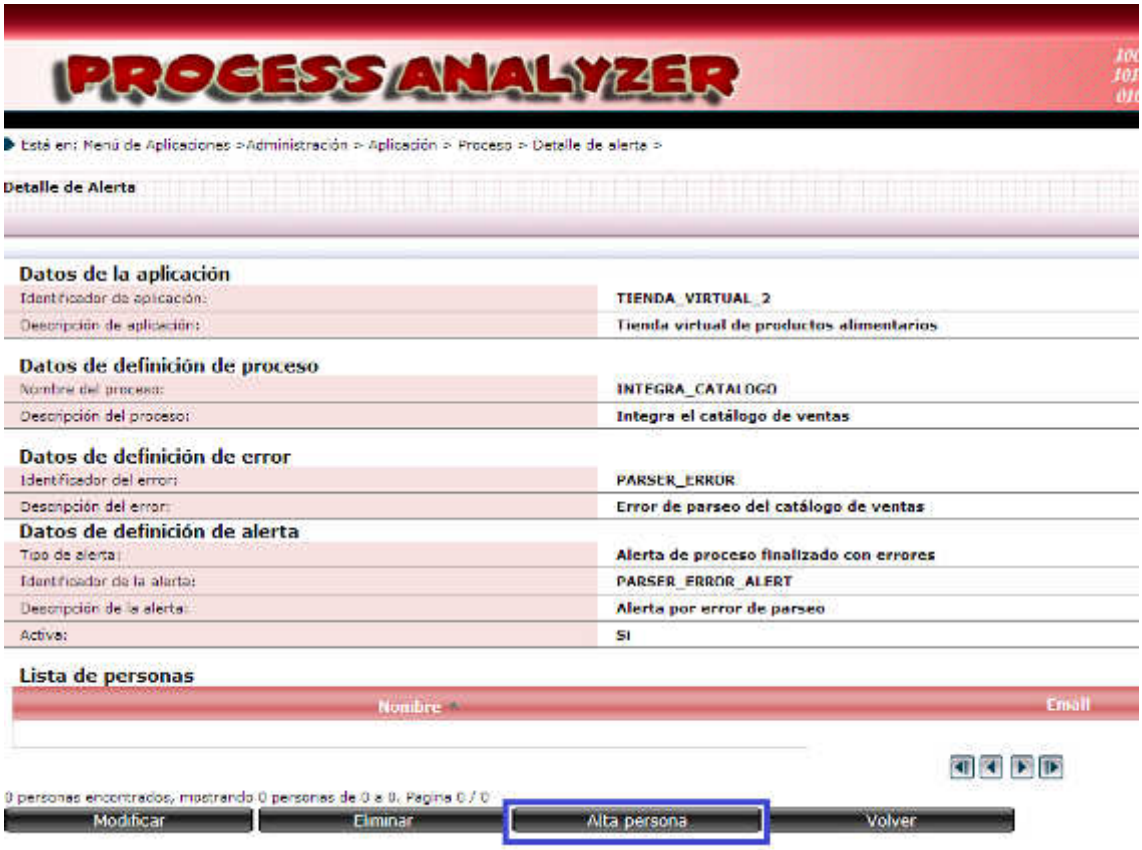


Figura 94 - Detalle de alerta de error

Pulsando en “Alta persona” se procede con el alta de destinatarios de la alerta (figuras 95 y 96):

Figura 95 - Alta de persona

Figura 96 - Alta de persona realizada

Pulsando sobre la persona se puede ver el detalle del destinatario (figura 97):

Figura 97 - Detalle de persona

Se puede modificar los datos de la persona (figura 98):

Figura 98 - Modificar persona

O eliminar la persona.

5.3.2 Alertas de *checkpoint*

Para dar de alta una alerta asociada a un *checkpoint*, ir al detalle del *checkpoint*. Para ello se ha creado un punto de chequeo "Informar proceso del catálogo" cuyo objetivo es la información mediante la generación de una alerta del proceso del catálogo. Así, mediante esta alerta se podría avisar automáticamente a la tienda virtual que su catálogo ha sido integrado.

En el detalle del punto de chequeo está la opción "Alta de alerta" (figura 99):

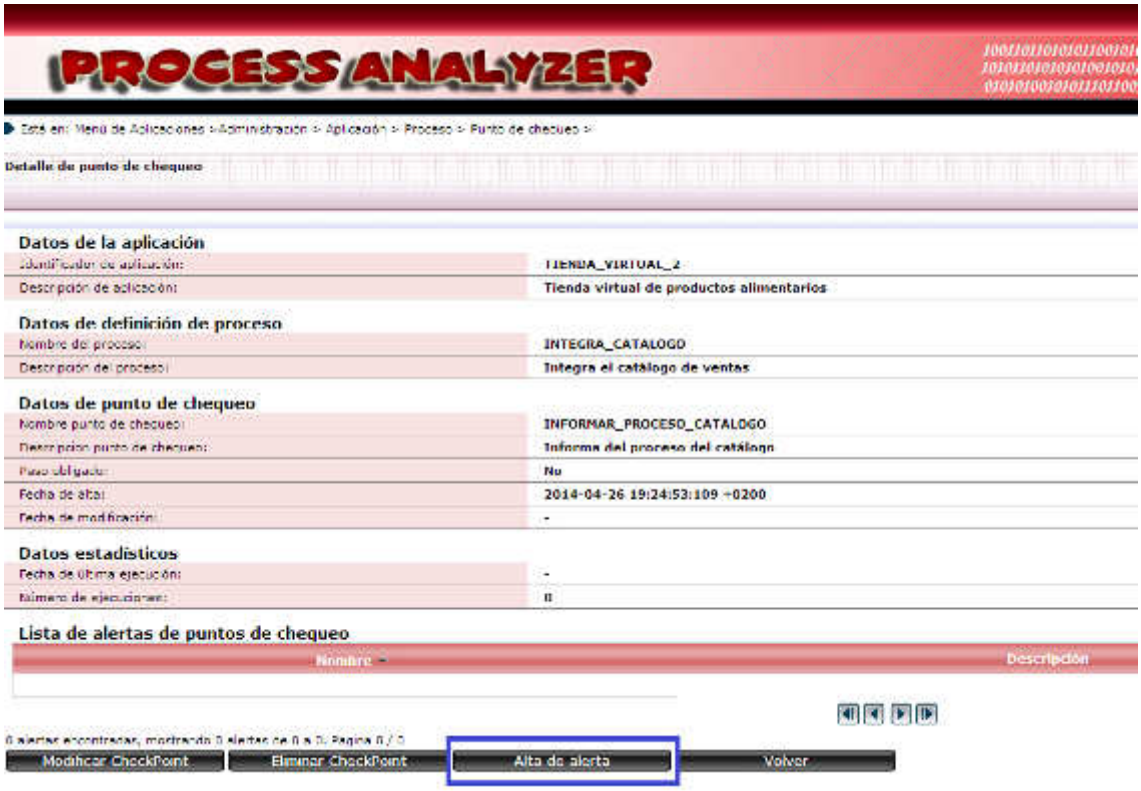


Figura 99 - Detalle punto de chequeo

Se procede con el alta de la alerta (figura 100):

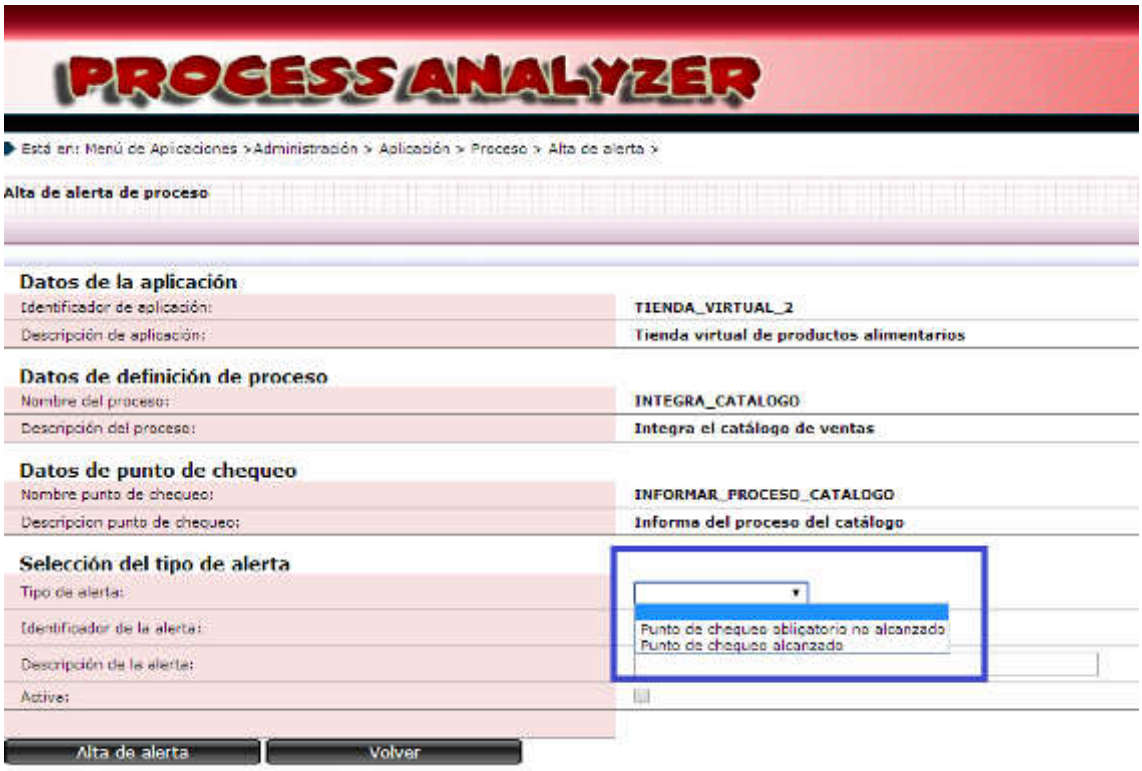


Figura 100 - Alta de alerta

Aquí existen dos opciones: “Punto de chequeo obligatorio no alcanzado” y “Punto de chequeo alcanzado”. La primera envía la alerta cuando el proceso finaliza sin pasar por el punto

de chequeo si este es obligatorio. La segunda opción genera una alerta siempre que se pase por el punto de chequeo, que es lo que se necesita en este caso. Por tanto, se selecciona la segunda opción (figuras 101 y 102):

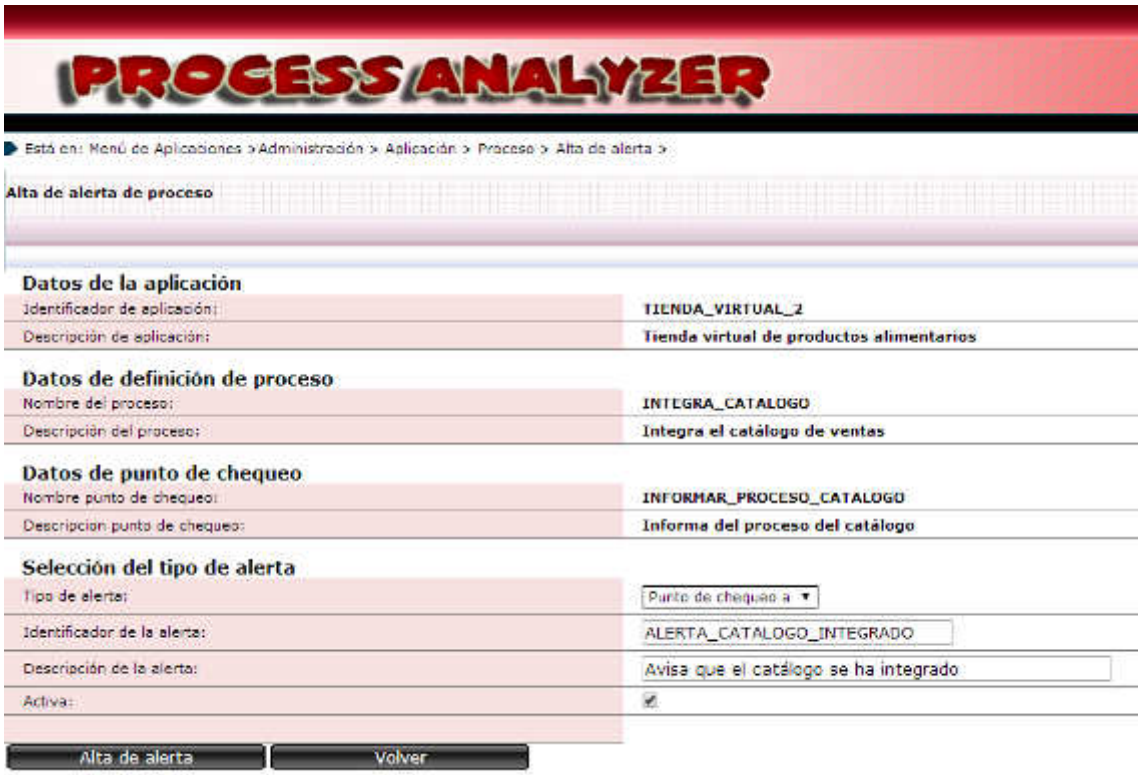


Figura 101 - Alta alerta de checkpoint

Figura 102 - Alta de alerta de checkpoint

Mostrando los detalles de la alerta se pueden añadir destinatarios como se ha mostrado en el punto anterior.

5.3.3 Alerta de procesos

Se pueden asociar alertas a los procesos directamente de forma que se avise cuando se produce un error, sea el que sea, cuando el proceso no finaliza en determinado tiempo, cuando un proceso planificado para ejecutarse diariamente a una hora no se ejecuta o cuando un proceso finaliza sin alcanzar un paso marcado como obligatorio. Para ello en el detalle del proceso pueden crearse las alertas necesarias (figura 103):

Figura 103 - Alta alerta de proceso

Aparece una lista con los distintos tipos de alertas (figura 104):

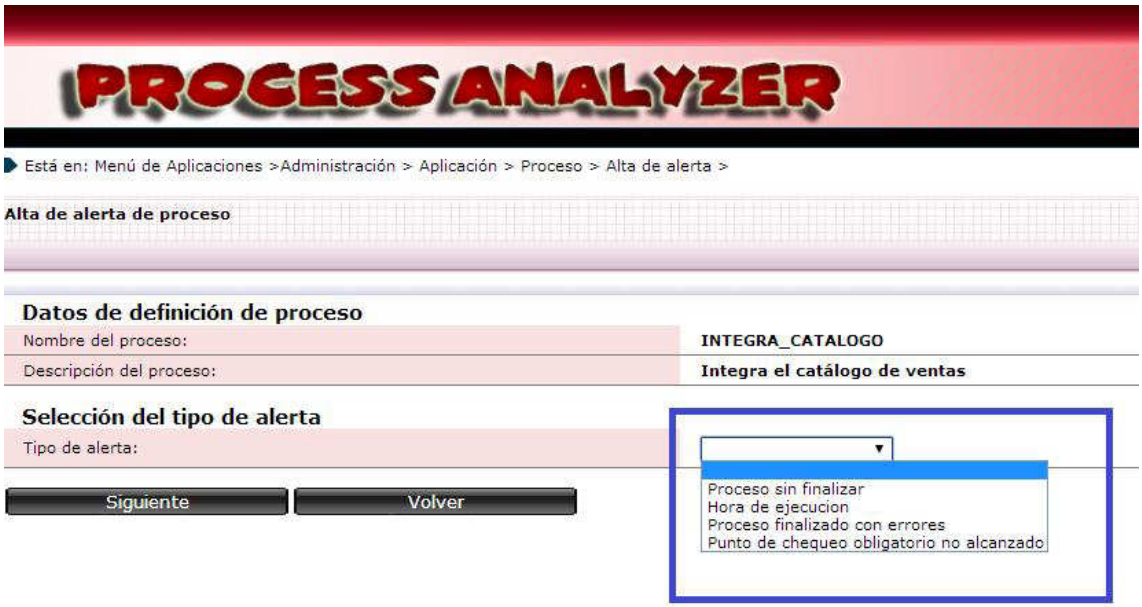


Figura 104 - Tipos de alertas de proceso

Se selecciona el tipo y se procede con el alta (figura 105):

Figura 105 - Alta alerta de proceso

En este caso el procedo debe terminar en 500 segundos o se lanzará una alerta.

Para añadir destinatarios se procede igual que en los puntos anteriores, a partir del detalle de la alerta se pueden añadir los destinatarios deseados.

En el detalle del proceso se puede ver la lista de las alertas asociadas (figura 106):

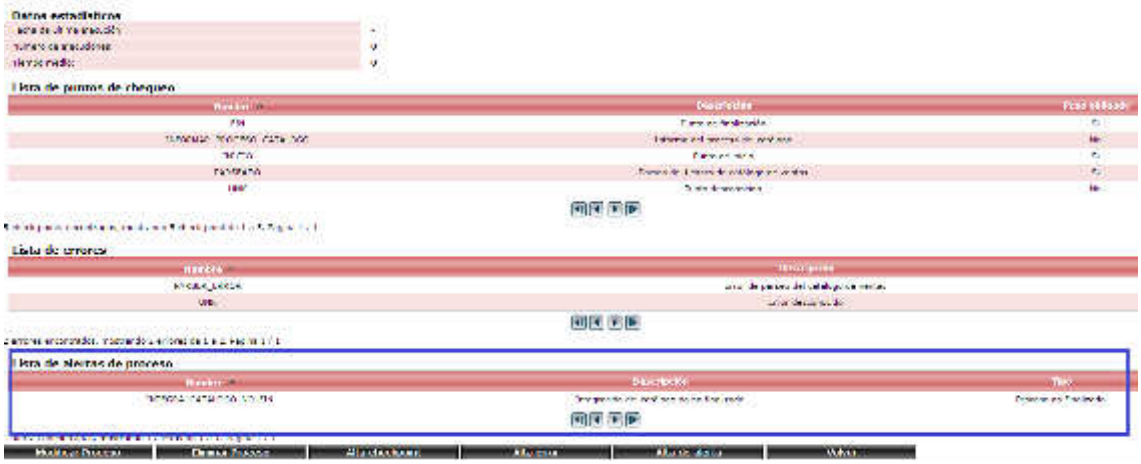


Figura 106 - Alertas asociadas a procesos

5.4 Ejecución de procesos

5.4.1 Inicio de proceso

Se informa mediante el *WebService* del inicio de la ejecución de un proceso. Se realiza el envío de los datos mediante la aplicación para testeado de *WebServices* SoapUI, como muestra la figura 107:

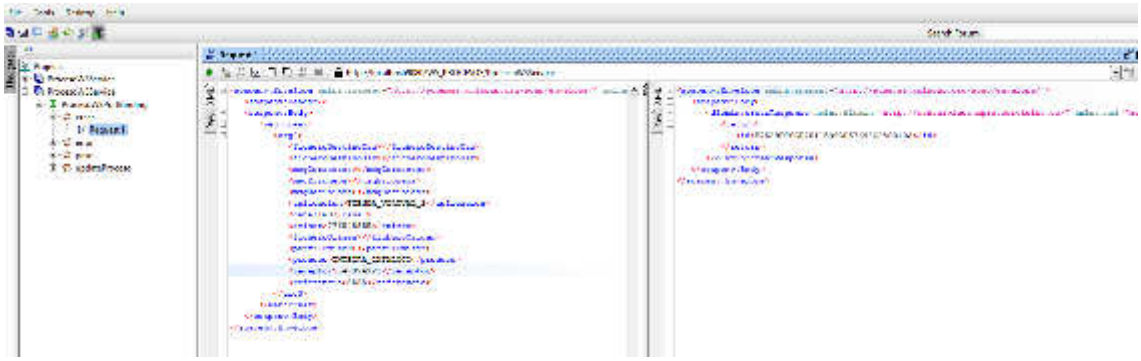


Figura 107 - Inicio de la ejecución de un proceso

En el lado izquierdo se pueden ver los parámetros de la llamada y en el lado derecho la respuesta del servidor. Para más información sobre los parámetros ver el apartado “Interfaces” en “Construcción del proyecto”. En este caso responde la aplicación con el identificador de la ejecución del proceso que le ha asignado.

Se puede ver en la base de datos el registro creado en la tabla de ejecuciones de procesos (EX_PROCESO). Se utiliza el Oracle SQLDeveloper para ver el contenido de las tablas de la base de datos (figura 108):

Figura 108 - Proceso iniciado en la base de datos

Se puede ver que el identificador de proceso “E75A9D70CD7511E39C0E7698C0A80103” coincide con la respuesta al *WebService*.

También se puede ver el proceso en la pantalla de consulta (figura 109):

Figura 109 - Proceso iniciado en la pantalla de consulta

Y pulsando sobre el registro se puede consultar el detalle (figura 110):

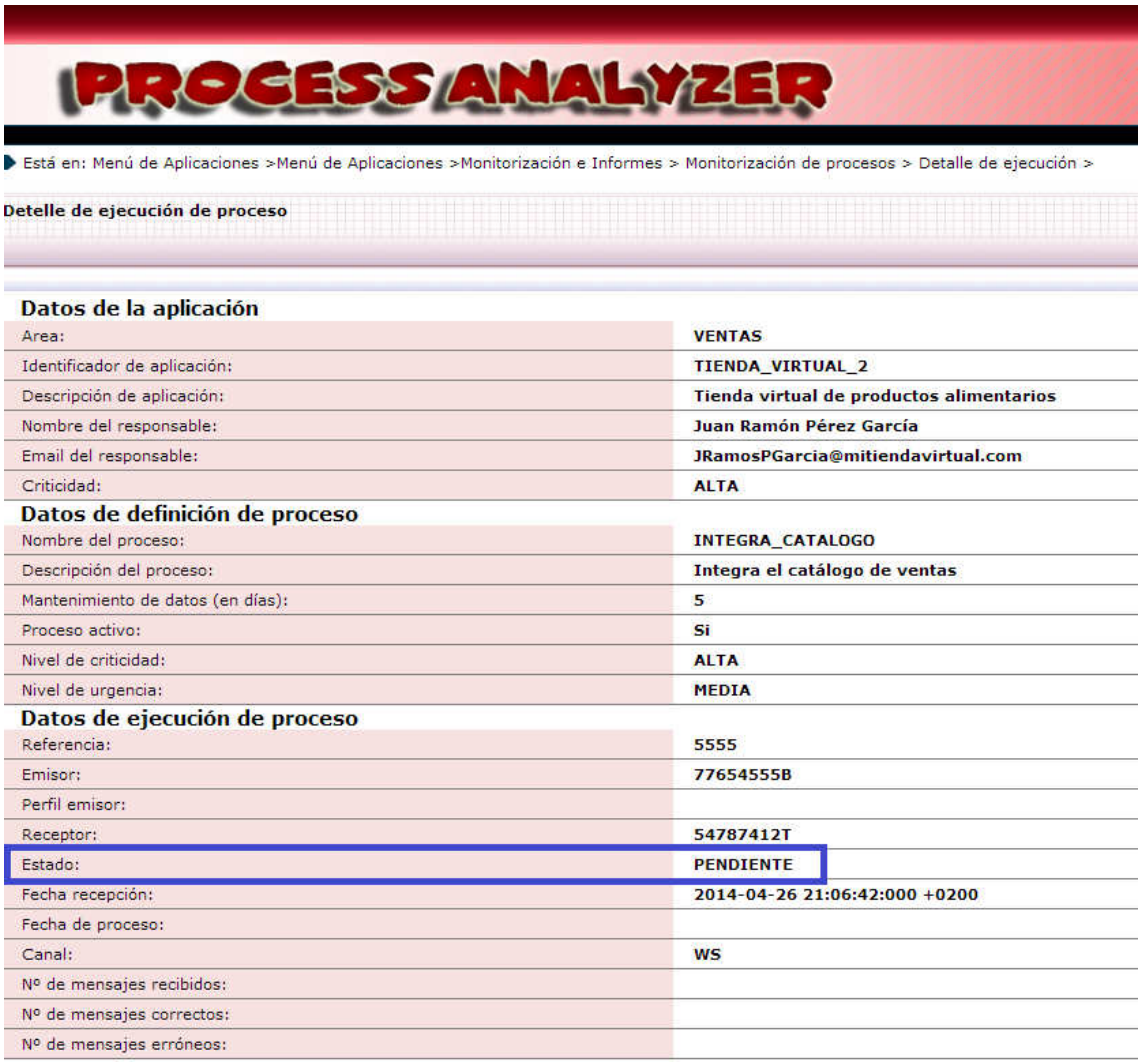


Figura 110 - Detalle de la ejecución del proceso

Mientras no se envíe el inicio de la ejecución del proceso el estado es “PENDIENTE”.

Se inicia la ejecución del proceso de integración del catálogo. Para ello se manda la información del primer punto de chequeo “INICIO” a través de *WebService* (figura 111):

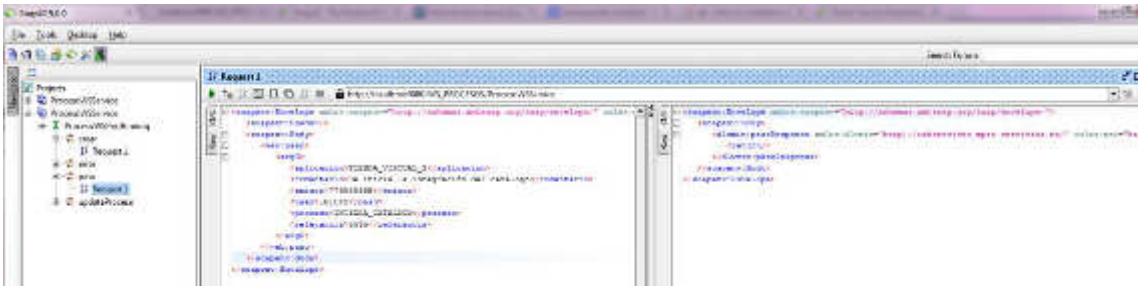


Figura 111 - Paso de inicio de proceso

La consulta del proceso ha cambiado (figura 112):

Figura 112 - lista de proceso iniciado

El icono del estado ha cambiado, si se consulta el detalle (figura 113):

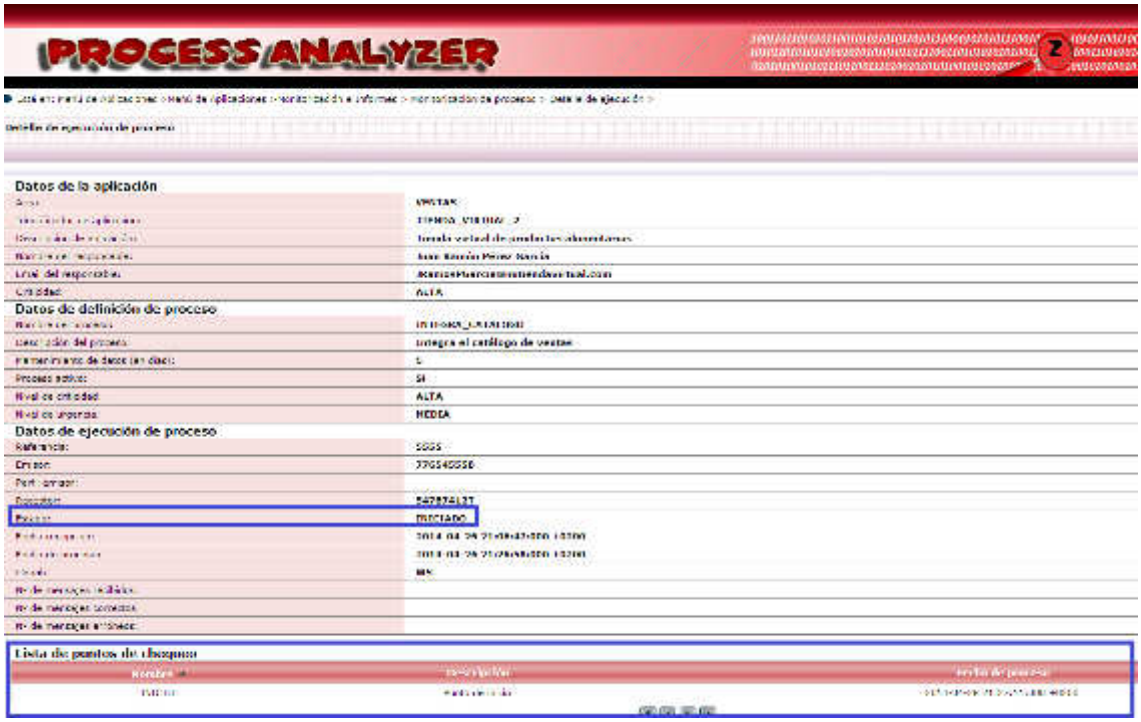


Figura 113 - Detalle de proceso iniciado

El estado ha cambiado a “INICIADO” y ya hay un punto de chequeo que corresponde con el inicio.

Para ver la respuesta que da el sistema frente a un error, se vuelve a enviar el proceso que ya ha sido enviado. Este proceso no debería registrarse en el sistema ya que la referencia del proceso ya existe para ese emisor – proceso y ese error debería ser comunicado (figura 114):

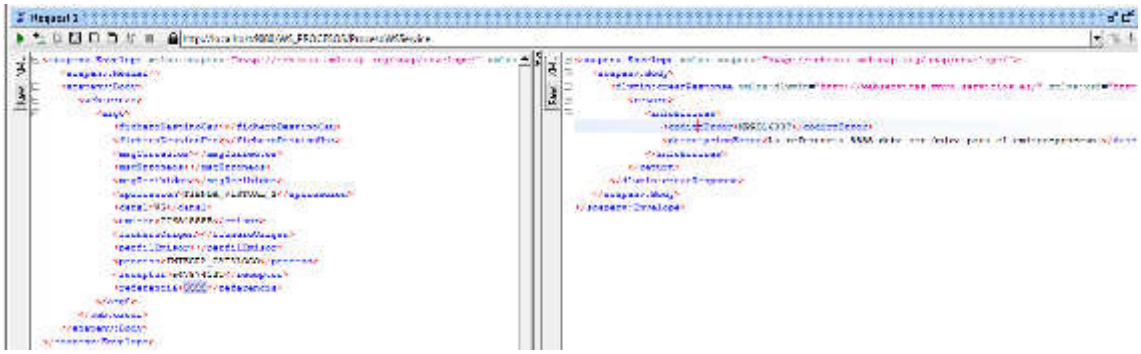


Figura 114 - Error en el registro de un proceso

Se puede ver que la respuesta “La referencia 5555 debe ser única para el emisor-proceso.” es la esperada.

Hasta el momento no se ha producido ningún evento que deba generar alertas.

5.4.2 Información de puntos de chequeo

Se va a proceder a insertar la información de un *checkpoint* creado anteriormente que tiene asociada una alerta. El paso es “INFORMAR_PROCESO_CATALOGO”. Como en el apartado anterior se realizará la inserción de datos mediante la interface *WebService* realizando la llamada con SoapUI (figura 115):

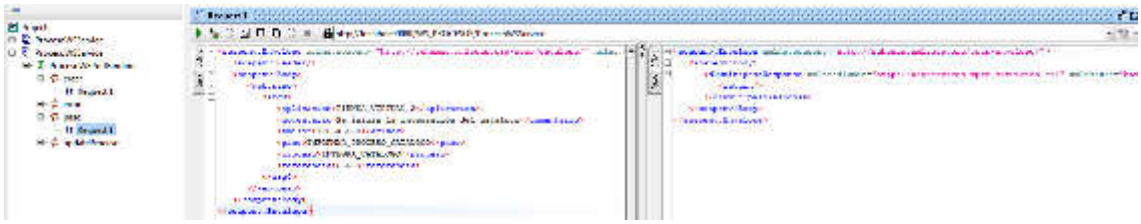


Figura 115 - Información de un paso

Una vez insertado el *checkpoint* se puede ver por la aplicación Web, mediante la consulta de ejecuciones, la ejecución del *checkpoint* (figura 116):

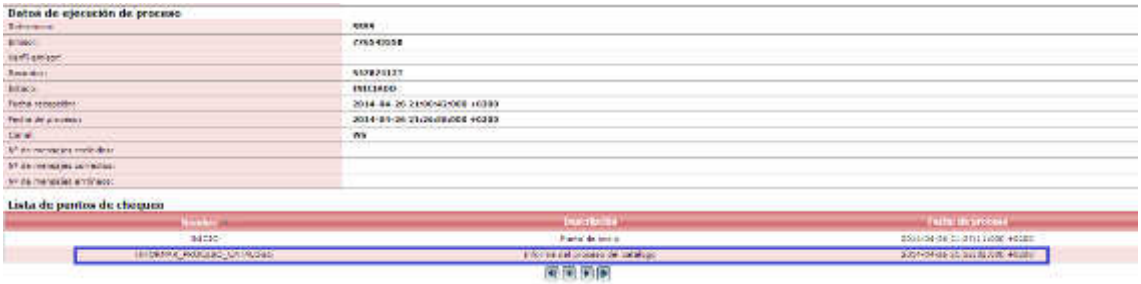


Figura 116 - Resultado de la información de un paso

Como es un *checkpoint* asociado a una alerta, se debería haber creado un registro de alerta asociado a este *checkpoint*. Esto se muestra a través de la base de datos, las alertas están en la tabla “EX_ALERTA”:

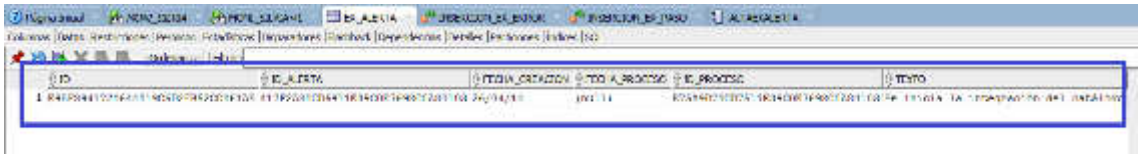
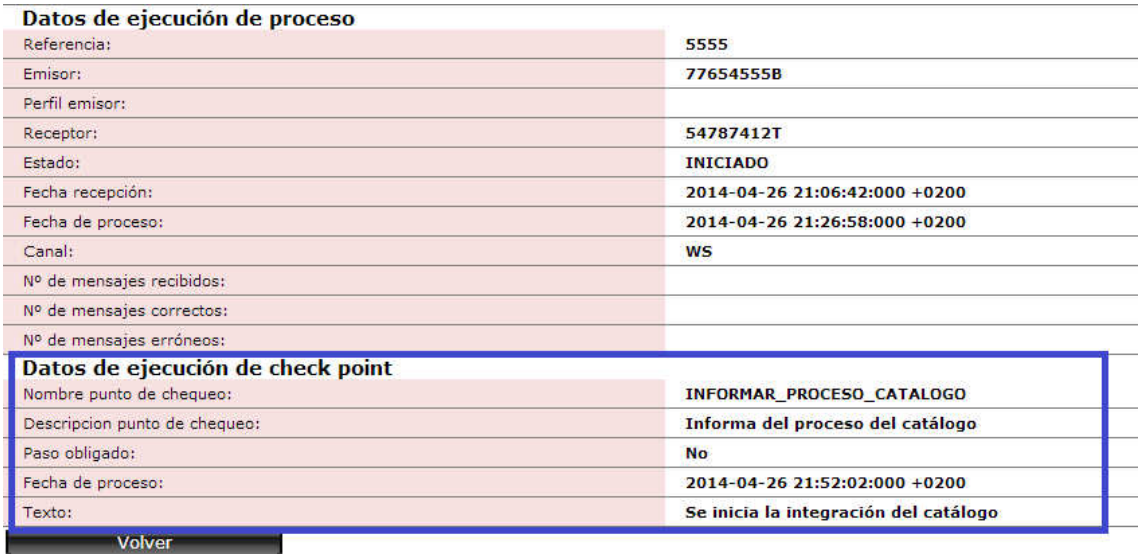


Figura 117 - Alerta asociada a un paso

La alerta será procesada por un *job* de Oracle más tarde que actualizará el campo "FECHA_PROCESO" de la alerta.

En la pantalla de detalle del proceso se puede consultar el detalle del punto de chequeo (figura 118):



Datos de ejecución de proceso	
Referencia:	5555
Emisor:	77654555B
Perfil emisor:	
Receptor:	54787412T
Estado:	INICIADO
Fecha recepción:	2014-04-26 21:06:42:000 +0200
Fecha de proceso:	2014-04-26 21:26:58:000 +0200
Canal:	WS
Nº de mensajes recibidos:	
Nº de mensajes correctos:	
Nº de mensajes erróneos:	
Datos de ejecución de check point	
Nombre punto de chequeo:	INFORMAR_PROCESO_CATALOGO
Descripcion punto de chequeo:	Informa del proceso del catálogo
Paso obligado:	No
Fecha de proceso:	2014-04-26 21:52:02:000 +0200
Texto:	Se inicia la integración del catálogo

Volver

Figura 118 - Detalle de la ejecución de un paso

Para ver la respuesta que da el sistema frente a un error, se vuelve a enviar el *checkpoint* indicando un punto de chequeo que no existe en el sistema. Este *checkpoint* no debería registrarse en el sistema ya que el punto de chequeo no se encuentra (figura 119):

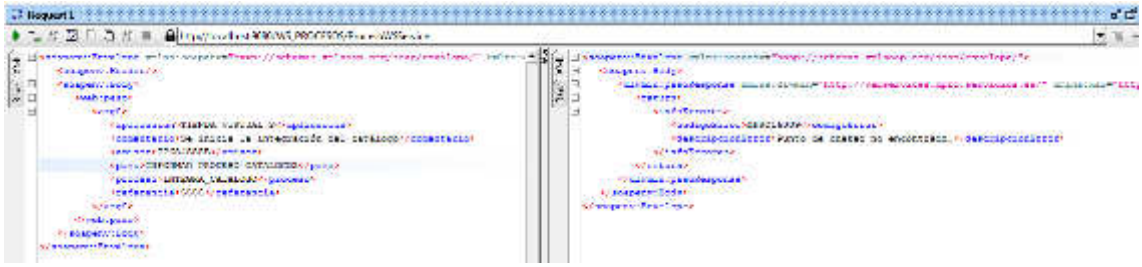


Figura 119 - Ejecución de un paso con errores

Se puede ver que la respuesta "Punto de chequeo no encontrado." es la esperada.

5.4.3 Información de errores

Se va a proceder a insertar la información de un error creado anteriormente que tiene asociada una alerta. El error es "PARSER_ERROR". Como en el apartado anterior se realizará la inserción de datos mediante la interface *WebService* realizando la llamada con SoapUI (figura 120):

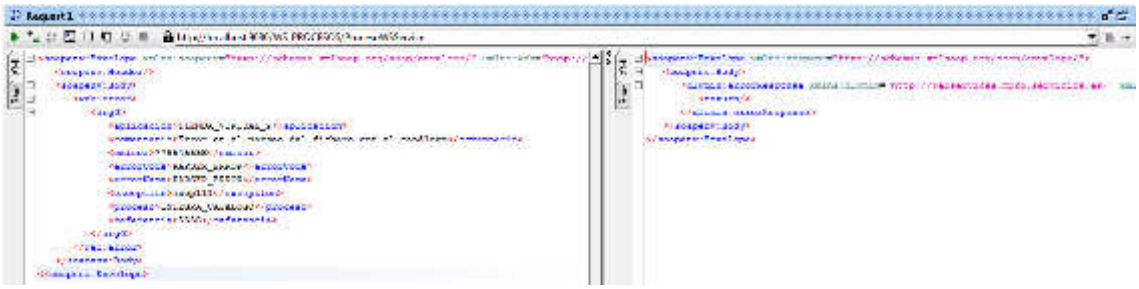


Figura 120 - Información de un error

Se puede ver en la aplicación Web, en el detalle del proceso, el error (figura 121):

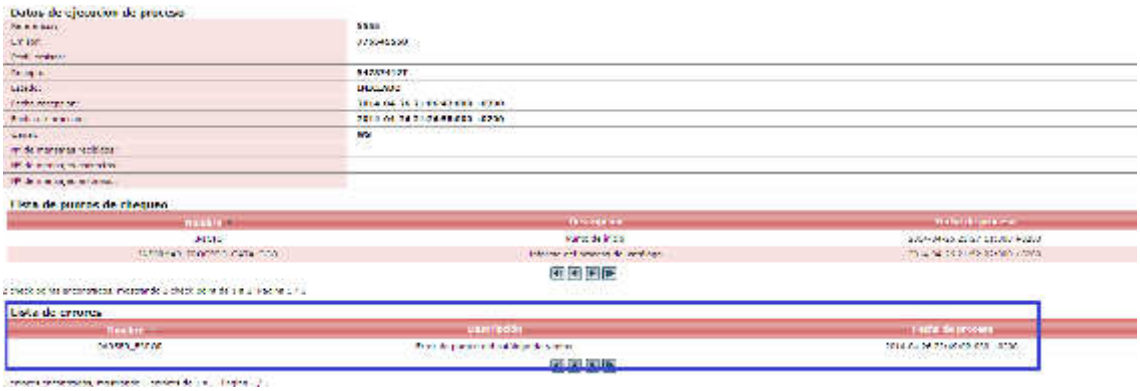


Figura 121 - Resultado de la información de un error

Este error tiene asociada una alerta. Se puede consultar en la base de datos la alerta que se ha generado (figura 122):

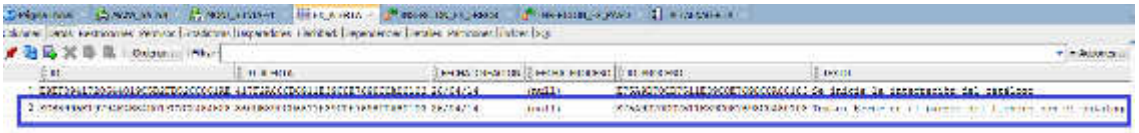


Figura 122 - Alerta de error en la base de datos

En el detalle del proceso, pulsando sobre el error se puede ver el detalle del error generado (figura 123):

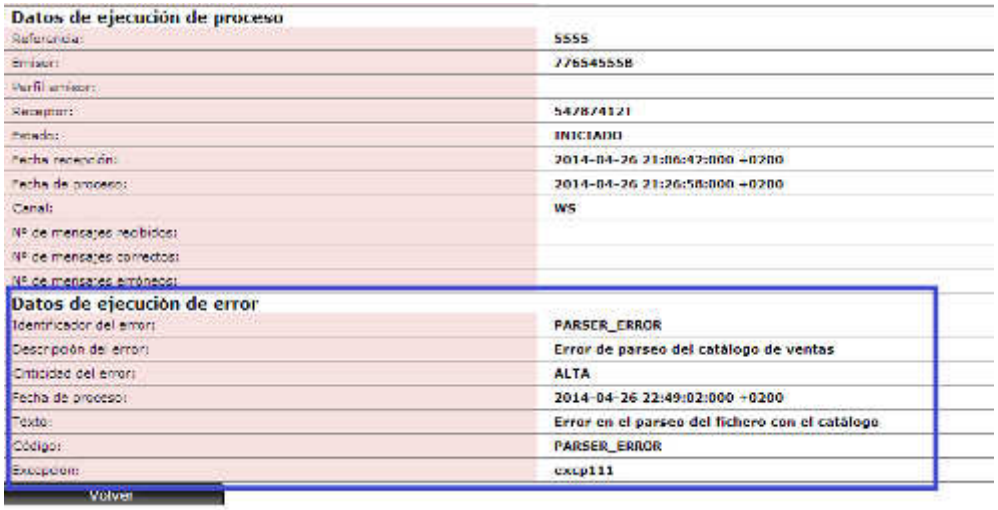


Figura 123 - Detalle del error

Para ver la respuesta que da el sistema frente a un error, se vuelve a enviar el indicando un error que no existe en el sistema. Este error no debería registrarse en el sistema ya que el error no se encuentra (figura 124):

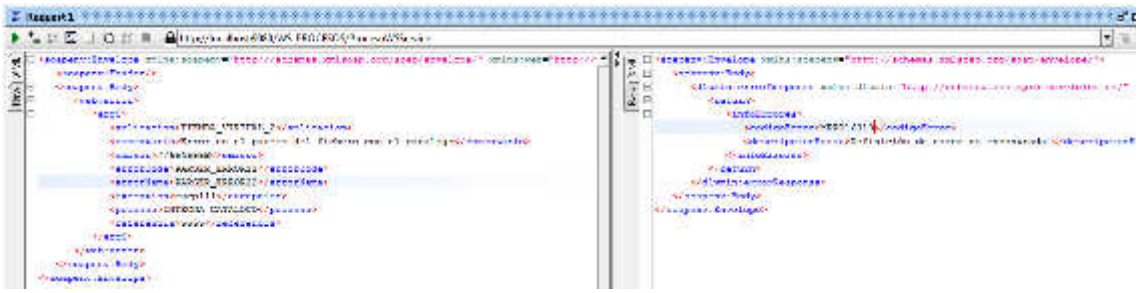


Figura 124 - información de error con errores

Se puede ver que la respuesta “Definición de error no encontrada” es la respuesta esperada.

5.4.4 Finalización de proceso

Para la finalización de un proceso se informa el paso “FIN” (figura 125):

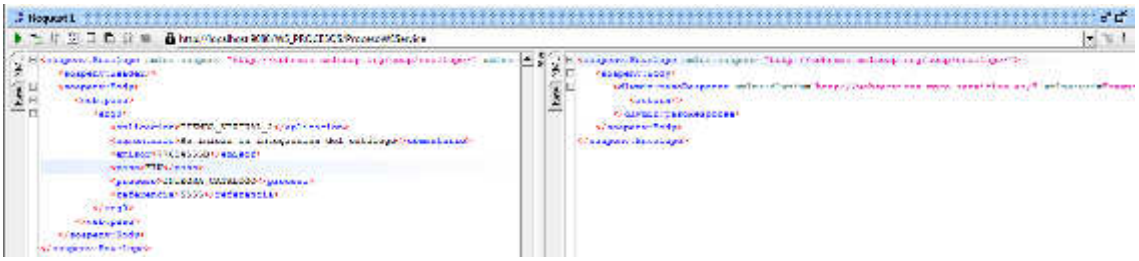


Figura 125 - Finalización de proceso

Se puede ver en la pantalla de consulta como queda la ejecución del proceso (figura 126):

Figura 126 - Proceso finalizado

El icono del estado ha cambiado pues el proceso ha finalizado con errores. Se puede consultar el detalle de a ejecución (figura 127):

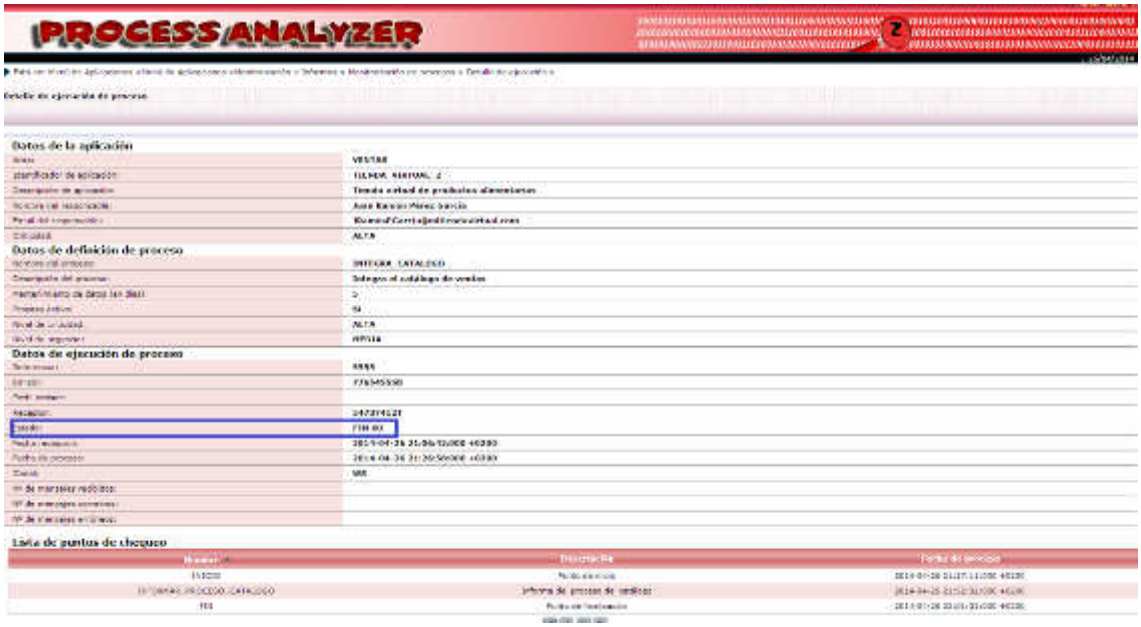


Figura 127 - Detalle del proceso finalizado

El estado ha cambiado a "FIN KO" pues ha terminado con errores. En el apartado de "Consulta de ejecuciones" se verá como quedan los procesos que termina bien, los que están pendientes de terminar y los que terminan con errores.

La ejecución de un proceso actualiza la información estadística del proceso. Esta información se puede consultar en los detalles de la definición del proceso. Se han realizado una serie de ejecuciones del proceso que se usa de referencia y así se ha actualizado esta información (figura 128):

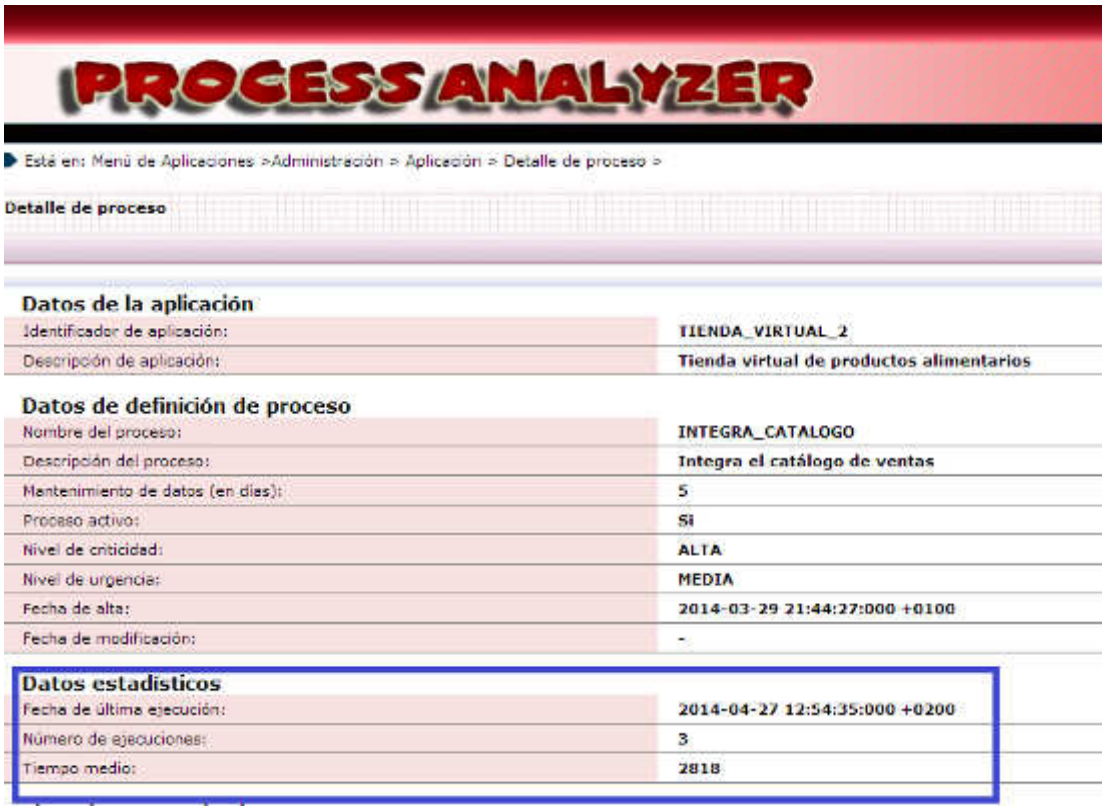


Figura 128 - Datos estadísticos de proceso

5.5 Consulta de ejecuciones

Para realizar una consulta de las ejecuciones registradas se accede a través de la segunda opción del menú principal (figura 129):

Figura 129 - Consulta de ejecuciones de procesos

A partir de esta opción se accede a una pantalla con un conjunto de filtros para realizar la consulta (figura 130):

Figura 130 - Filtro de consulta

A través de los filtros podemos buscar por referencia, aplicación, proceso, estado, fechas de envío, fechas de proceso…

Se realiza una búsqueda de los procesos que se están tratando en la demo. Se han insertado varios procesos en los distintos estados posibles: Pendiente, Iniciado, Finalizado OK, Finalizado KO (figura 131):

Figura 131 - Listado de consulta

Pueden verse los 4 estados en la tabla 42:

Estado	Icono
PENDIENTE	
INICIADO	
FINALIZADO OK	
FINALIZADO KO	

Tabla 42 - Estados

Pulsando sobre cada registro podemos ver el detalle de la ejecución (figura 132):

Figura 132 - Detalle de ejecución de proceso

Esta pantalla muestra un resumen de la aplicación, del proceso y de la ejecución. También muestra la lista de los puntos de chequeo por donde el proceso ha ido pasando, los errores que se han registrado, las horas de ejecución…

Pulsando sobre los puntos de chequeo se puede ver el detalle del paso (figura 133):

Figura 133 - Detalle ejecución de checkpoint

El detalle muestra toda la información que tiene el sistema sobre el *checkpoint* cuya ejecución se está consultando: la aplicación, el proceso, el punto de chequeo y los datos de ejecución.

En la pantalla anterior, pulsando sobre el error se muestra toda la información relativa al error (figura 134):

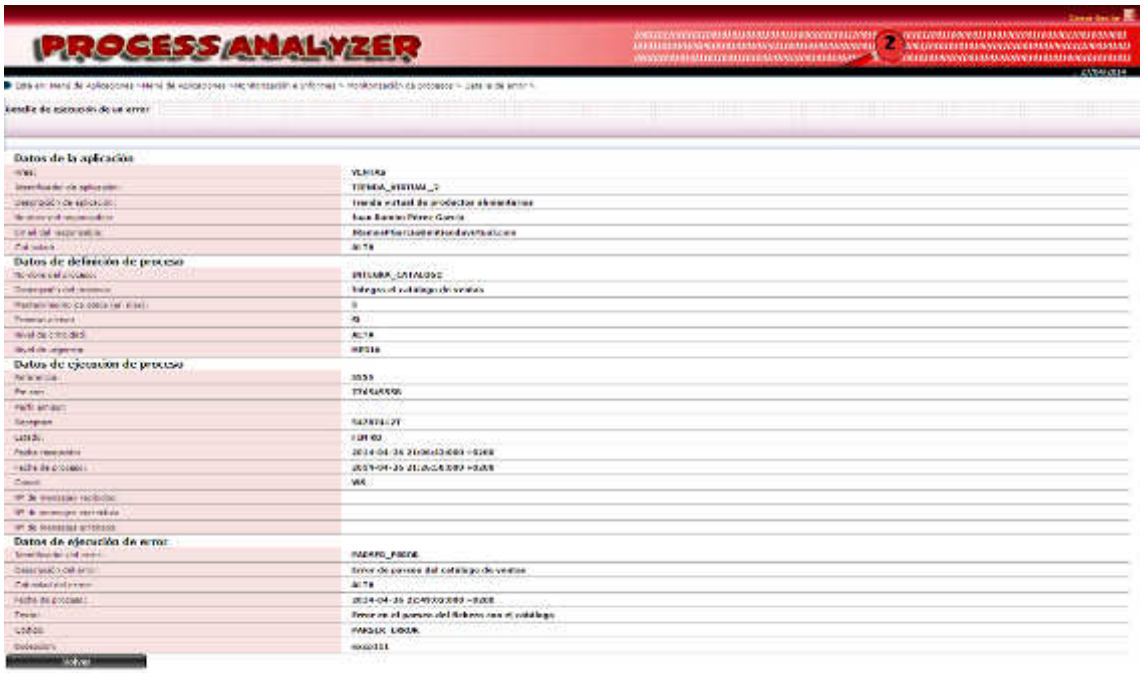


Figura 134 - Detalle ejecución de error

Igual que la pantalla del *checkpoint*, muestra toda la información sobre la aplicación, el proceso y la ejecución.

5.6 Informes

Con un funcionamiento similar al de las consultas se generan los informes. Estos informes son exportaciones de los datos en formato CSV (fichero con los campos separados por comas). Para acceder a estas funciones se debe seleccionar la tercera opción del menú principal (figura 135):

Figura 135 - Informe de procesos

A partir de la pantalla de filtros se realiza la petición del informe (figura 136):

Figura 136 - Filtros de informes

Y se genera el informe que se puede abrir con el Excel (indicándole el tipo de fichero CSV y que delimite los campos con comas) (figura 137):

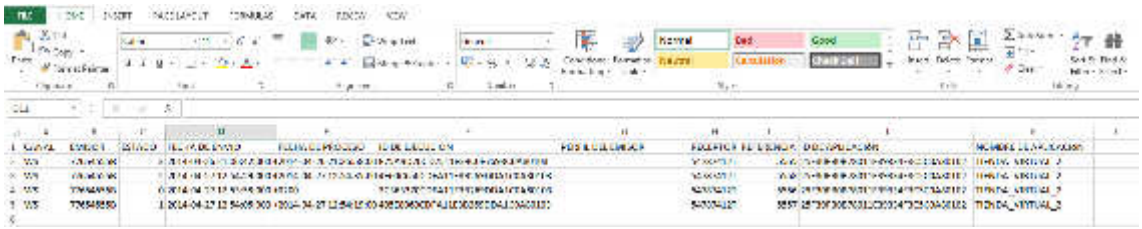


Figura 137 - Informe generado

Estos informes pueden ser explotados por herramientas estadísticas de forma que puedan obtenerse información sobre la calidad de los datos recibidos, de los procesos, que procesos son los que fallan más, que procesos tardan más, cuales se ejecutan más frecuentemente…

6 Conclusiones

6.1 Balance

El objetivo inicial se había definido como la creación de una aplicación web para la administración de procesos (gestión del catálogo de procesos) y alertas asociadas a los procesos. Este objetivo se ha cumplido sobradamente pasando por las distintas fases de desarrollo de un producto de software.

No obstante se había indicado que el sistema debía ser robusto, en alta disponibilidad, rápido, eficiente y fiable. Esto no se puede garantizar sin la realización de unas pruebas de rendimiento que garanticen que el sistema es robusto y estable en el tiempo. Por los motivos que se han especificado en el apartado "Definición del plan de pruebas" no se han podido realizar pruebas de rendimiento de forma que garanticen que el sistema es estable. Se ha realizado el diseño y la construcción de la aplicación de forma que debería mantenerse estable, no obstante si las pruebas no hay garantía. Por lo tanto este objetivo no se puede garantizar.

6.2 Ampliaciones

Se ha realizado un listado de aquellos puntos que pudieran ampliarse:

- Tipos de alertas: Se pueden incrementar el tipo de alertas que gestiona la aplicación, por ejemplo: alertas de rendimiento cuando se observe un aumento del tiempo de proceso, alertas en función de la criticidad y urgencia de un proceso, etc.
- Destinatarios de las alertas: actualmente los destinatarios de las alertas están predefinidos en la aplicación, podrían indicarse destinatarios de otra forma, por ejemplo, que la aplicación monitorizada indique posibles destinatarios en la información de las ejecuciones de procesos.
- Categorización de procesos – errores: Actualmente los procesos se encuentran categorizados con dos parámetros: urgencia y criticidad. Estos parámetros son puramente informativos, podrían añadirse otros parámetros para categorizar mejor un proceso, tales como si podría generar una pérdida de servicios, SLA que le aplica… En las guías de ITIL se definen varias formas de categorizar los procesos.
- Tipos de envíos de notificaciones: Actualmente las notificaciones de alertas se comunican mediante EMAIL, pero podrían ampliarse: SMS, Whatsapp…
- Acceso a LDAP: Podría gestionarse los usuarios mediante un LDAP corporativo.
- Lanzamiento de procesos en *batch*: hay distintos *frameworks* que se encargan de lanzar procesos en *batch* en función de una planificación: quark, spring-batch… se podría utilizar cualquiera de estos *frameworks* para adaptarlo a la aplicación de forma que se lancen los procesos que se planifiquen a través de la interface gráfica de la aplicación. Esto implicaría crear formularios para el control y la gestión de la planificación de procesos. Sería interesante obtener una solución completa para la administración de procesos *batch*.

7 Bibliografía

- RedBooks de IBM, Tabla 43

Libro	URL
Developing Web Services Applications	http://www.redbooks.ibm.com/redpapers/pdfs/redp4884.pdf
IBM WebSphere Application Server V7.0 Web Services Guide	http://www.redbooks.ibm.com/redbooks/pdfs/sg247758.pdf
WebSphere Application Server for Developers V7	http://www.redbooks.ibm.com/redbooks/pdfs/sg247913.pdf
Developing Enterprise JavaBeans Applications	http://www.redbooks.ibm.com/redpapers/pdfs/redp4885.pdf
Design and Implement Servlets, JSPs, and EJBs	http://www.redbooks.ibm.com/redbooks/pdfs/sg245754.pdf

Tabla 43 - RedBooks de IBM

- Oracle

Optimización SQL en Oracle – Javier Morales - http://www.optimizacionsqlenoracle.com/ - ISBN-13: 978-1479190249

PL/SQL User's Guide and Reference - http://docs.oracle.com/cd/B10501_01/appdev.920/a96624/toc.htm

Scheduling Jobs with Oracle Scheduler - http://docs.oracle.com/cd/B28359_01/server.111/b28310/scheduse.htm#i1033533

Envío de mails desde Oracle mediante UTL_MAIL - http://docs.oracle.com/cd/B19306_01/appdev.102/b14258/u_mail.htm

- UML

Especificaciones oficiales de UML publicadas en: http://www.omg.org/spec/

- Gestión de servicio, Tabla 44

	WebSite
ITIL	http://www.itil-officialsite.com/home/home.asp
ITIL	http://www.itilfoundation.org/
Guía sobre SLA	tp://www.service-level-agreement.net/

Tabla 44 - Gestión de servicio

- Varios, Tabla 45

Libro	URL
IEEE Recommended Practice for Software Requirements Specifications (IEEE 830)	http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf
HTML, CSS, JavaScript,…	http://www.w3schools.com/
SCRUM	https://www.scrum.org/
OpenJPA	https://openjpa.apache.org/

Tabla 45 - Referencias varias