Creación de herramienta de back-office para la gestión de devoluciones en la sección de E-business de Mango

Egoitz Ortega Rodrigo

Resumen– A día de hoy la venta online supone uno de los mayores canales de venta para las empresas, incluyendo al sector textil. Mango no es una excepción y por ello dispone un sistema que soporta todos los elementos de la actividad e-commerce a nivel internacional, entre ellos módulos de pagos y devoluciones de pedidos. Dentro del departamento de e-business de la empresa, todas las incidencias se gestionan en primera instancia por un equipo de soporte que se encarga generalmente de resolverlas hasta que es posible detectar y solucionar la causa del problema. El objetivo de este proyecto es diseñar e implementar una solución dentro del marco de este equipo de soporte, buscando la mejora del sistema de tratamiento utilizado actualmente, que implica procesos manuales en muchos casos.

Palabras clave– E-commerce, diseño de software, Test-Driven Development, Feature-Driven Development, JSP, testing unitario

Abstract– Nowadays online retailing is one of the greatest selling channels for companies, including textile industry. Mango is no exception and that is why it has a system supporting all international e-commerce business activity elements, including order payments and refunds modules. Inside the e-business area, all the problems are managed first by a support team who usually solves them until the source of the issue can be identified and corrected. On this project the objective is to design and implement a solution with the scope set on this support team, looking to improve the treatment system used at the moment, applying manual procedures on many occasions.

Keywords– E-commerce, software design, Test-Driven Development, Feature-Driven Development, JSP, unit testing

1 Introducción

En la actualidad nos encontramos en un escenario donde la venta online ya está completamente establecida como forma alternativa para la compra de bienes y servicios, incluso comienza a ganar terreno a sistemas más tradicionales, con plataformas como Amazon o Alibaba.

Prácticamente cualquier empresa orientada a la venta que quiera potenciar su negocio recurre al e-commerce. Esto se materializa mediante la creación de un portal web o app móvil junto con un sistema de back-end que permita la gestión de la actividad comercial. Una opción alternativa es la publicación de los productos a través de marketplaces como Amazon, una solución más económica para introducir el producto en el mercado.

Este proyecto surge de la estancia en prácticas que está realizando el autor del artículo, donde se observa de primera mano la manera en la que se gestionan las incidencias de devoluciones de pedidos. Al identificar que en parte del tratamiento es posible introducir mejoras, se plantea al tutor en la empresa tanto el poder diseñar una solución a aplicar como el utilizar dicho proyecto como trabajo final de grado. La posibilidad de trabajar sobre un problema real y poner en práctica las competencias adquiridas durante el grado y la mención son el impulso final para decidir llevar adelante el proyecto que este artículo describe.

Es importante primero tener una visión más concreta de parte del funcionamiento de Mango, especialmente de la sección de e-business y cómo encaja en la actividad de la empresa y la forma en la que se gestionan las peticiones e información de las devoluciones.

• E-mail de contacto: egoitz.ortega.rodrigo@gmail.com
• Mención realizada: Ingeniería del Software
• Trabajo tutorizado por: Francesc Xavier Roca Marva (Departamento de ciencias de la computación)
• Curso 2015/16

1.1. Contexto de la empresa y las devoluciones

Mango es una empresa del sector textil con venta en más de 100 países que además de disponer de una red de tiendas físicas, también utiliza un portal web y apps nativas en dispositivos Android e iOS para ofrecer sus productos y venderlos a nivel mundial. También utiliza como puntos de venta marketplaces genéricos como Amazon o específicos de moda como La Redoute. Además de pagar en efectivo, se ofrece la posibilidad de utilizar un vale con un importe disponible para utilizar en las compras.

Al ser una empresa que originalmente solo se ha dedicado a la venta en tienda física, en la creación del área de e-business se hizo un diseño de la arquitectura para que se trabajase de forma paralela al sistema de *Enterprise Resource Planning* (ERP) que utiliza la empresa. Dentro de esta arquitectura se incluye un portal web que actua de back-office, donde se puede consultar información relativa a la actividad del departamento y realizar gestiones.

Debido a la estructura de la empresa, cuando un cliente realiza una compra online su pedido queda registrado en la base de datos de e-business, pero la gestión logística y la gestión de los datos de facturación se hace de forma externa. La comunicación de los pedidos se hace mediante el envío de un fichero con la información necesaria, registrándose en el ERP utilizado por Mango. Una vez hecho el registro, se ponen en marcha los procesos de preparación y expedición del pedido.

De forma similar, cuando un cliente hace la devolución de un pedido, se genera un fichero XML desde el sistema ERP de Mango y se envía al área de e-business. Este fichero sigue un circuito de procesado (ver figura 1), con dos fases que se ejecutan de forma asíncrona.

- **Fase de aceptación:** Se valida que el contenido del fichero sea coherente, revisando que por ejemplo el importe informado en la devolución se ajuste a la suma de los importes parciales de los artículos. También se hace un cotejado de los datos con la información almacenada en base de datos, parte necesaria para aceptar finalmente la devolución y registrarla.
- **Fase de procesado** Se recupera de base de datos la solicitud de la devolución y se prepara la información, generalmente en formato JSON o XML, para enviar vía API a la pasarela de pagos utilizada originalmente para el pago del pedido.

Si en alguna parte de este circuito se produce algún error, este queda registrado en la base de datos.

El protocolo a seguir por el área de e-business cuando se detecta una incidencia es una primera actuación por parte del equipo de soporte e incidencias. Este equipo se encarga de analizarlas, investigar las causas y solucionarlas si la causa es conocida, siempre que no pueda interferir con proyectos activos o artefactos críticos, como por ejemplo el módulo de pagos.

Además de poder realizar consultas en la base de datos, en la back-office del área de e-business existe una herramienta de consulta para visualizar los detalles de un pedido y sus devoluciones. A través de esta herramienta también se pueden ver los errores producidos al procesar las devoluciones, siendo en muchas ocasiones utilizada para obtener parte de la información relevante para localizar incidencias.

Las incidencias sobre las que se quiere influir con la elaboración de este proyecto suelen ser de causas conocidas a las que se aplica una solución, las que afectan a devoluciones.

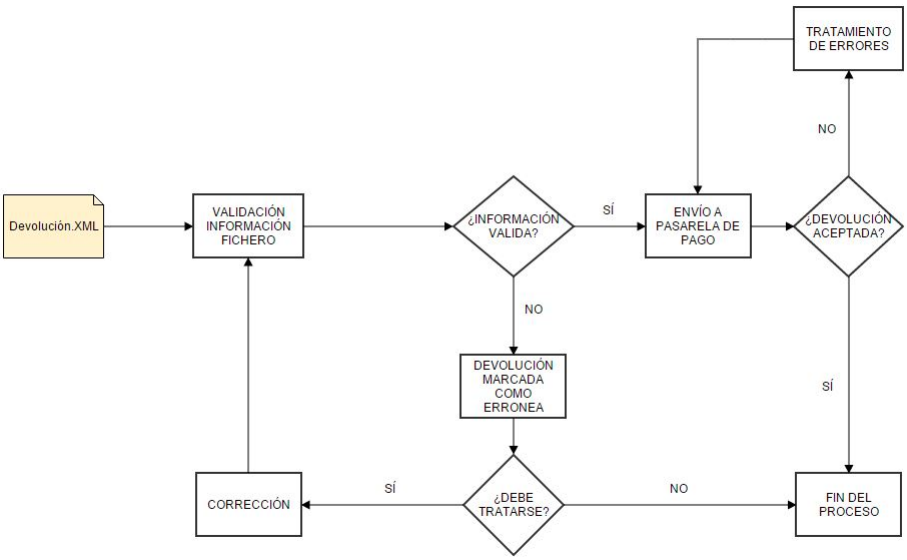


Fig. 1: Circuito de procesado para un fichero de devolución

1.2. Problemas del sistema

El trabajar con un sistema que depende un traspaso de ficheros, que además se genera con una base de datos que no siempre coincide, da lugar a que el proceso no sea completamente infalible. Aunque el volumen de incidencias no es elevado respecto al total, generalmente inferior al 3 % de las peticiones procesadas, estas incidencias deben tratarse siempre lo antes posible por afectar a pagos.

La mayor parte de incidencias registradas en relación a las devoluciones suelen ser errores en la información de la devolución, ya sea a nivel de fichero o en la información

generada para enviar vía API. También se puedan llegar a producir, aunque con un número bastante inferior, errores de conexión a la hora de realizar las peticiones a las distintas APIs o incluso detectar ficheros duplicados de devolución.

La mayor parte de incidencias de devoluciones suelen ser errores en la información que llega. Para tratarlas los procedimientos habituales suelen ser realizar modificaciones en la información almacenada en base de datos, sobre el fichero o solicitar el reenvío de la devolución al equipo responsable. Cabe destacar que el acceso a estos ficheros implica la extracción de información de base de datos y formatearlos correctamente o hacer una conexión remota para recuperarlos directamente.

Los errores no siempre responden a la misma casuística, pudiendo tener dos incidencias con el mismo error registrado pero con causas distintas, cada una con un tratamiento diferente. Ésto implica que es frecuente revisar caso a caso, para determinar cómo se debe resolver. Actualmente se puede consultar desde la back-office si un pedido tiene alguna devolución y su estado dentro del sistema, pero no se dispone de un histórico de los estados de una devolución, que puede derivar en tratar varias veces un mismo error sin conseguir resolverlo. En base de datos tampoco se puede visualizar esta información de ninguna forma.

La inversión de tiempo necesaria, unido a que la intervención de los miembros del equipo implica modificaciones manuales en parte de los casos, deja claro que es un proceso con margen de mejora.

La solución ideal implicaría una resolución de la causa raíz de cada uno de los errores detectados, pero debido a la manera de trabajar en la empresa estas soluciones se retrasan de forma frecuente. El hecho de que el equipo de soporte tenga bastantes funciones asociadas, que suponen un trabajo diario constante, dificultan aún más esta vía.

El autor de este artículo, al estar realizando un convenio de prácticas con la empresa dentro del equipo de soporte del área de e-business, observó lo explicado en los párrafos anteriores. Se hizo la propuesta de diseñar una solución que permitiese agilizar la gestión de las incidencias en las devoluciones al jefe del equipo y a los responsables de desarrollo del departamento. Con su aceptación, nace este trabajo final de grado, que se detalla en las siguientes secciones.

2 OBJETIVOS

El principal objetivo es la creación de una solución que permita agilizar los procesos de gestión en las incidencias de devoluciones, enfocado a la manera de trabajar desde el equipo de soporte e incidencias del área de e-business.

Para la creación de esta solución se contempla seguir las fases fundamentales en el desarrollo de un proyecto software, que incluyen la captura de requisitos, diseño de una solución, implementación del diseño y testing.

Originalmente se pretendía generar una herramienta en la back-office donde poder centralizar la visualización de las devoluciones de un pedido, el histórico de cada devolución y además poder aplicar soluciones a los errores utilizando una pantalla específica para este fin. En esta primera propuesta se quería además definir accesos a la aplicación según perfiles de usuario, para impedir que usuarios no autorizados tuviesen acceso a parte de la información o a manipular datos.

Finalmente en la fase de captura de requisitos inicial se acabó dividiendo el proyecto entre funcionalidades a implementar en aplicaciones de la back-office y la creación de un proceso automático para el tratamiento de errores en devoluciones.

Expuesto esto, el estado final de los objetivos marcados para el TFG es el siguiente:

2.1. Generación de ficheros de devolución

Desarrollar una aplicación en la back-office del área de e-business que a partir de los datos de un pedido se pueda generar un fichero XML con el formato de las devoluciones procesadas en entorno de producción. Se fija este objetivo para dar soporte a la resolución de errores sin solución automática y facilitar los test de sistema del circuito de devoluciones, no solo útil para el trabajo desarrollado en el TFG sino que es también aplicable a otros proyectos de la empresa.

2.2. Procesado automático de errores de devolución aplicando reglas

Desarrollo de un proceso ejecutado periódicamente por un scheduler que recupere errores de devolución registrados en base de datos, utilizando una serie de reglas para determinar la causa concreta del error y aplicar el tratamiento correspondiente. Una regla constituye un proceso de análisis que determina si la causa del error corresponde a la de la regla y posteriormente una fase de tratamiento para solucionar la causa. Este proceso debe tener en cuenta también si se ha tratado ese error, más de una vez para una devolución o si no se ha podido aplicar tratamiento alguno, notificándose vía mail en cualquiera de estos dos casos para revisarlo.

Como la creación de reglas específicas para cubrir cada error supone invertir más tiempo que el disponible para la realización del trabajo final de grado, se plantea desarrollar e implementar la arquitectura base del proceso e implementar alguna regla de tratamiento.

2.3. Mantener un histórico de los errores de devolución

Estudiar e implementar una forma de incluir en el sistema un histórico de los errores registrados por una devolución. Para el ámbito de este trabajo se define que debe servir como soporte al proceso automático de tratamiento de errores, habilitando que se pueda validar si una regla procesado no está funcionando correctamente.

2.4. Visualización de devoluciones en vale

En la captura de requisitos inicial se propuso desde el equipo de negocio del departamento añadir en la aplicación de consulta de devoluciones una visualización directa de las recargas hechas en un vale. Se justifica como algo útil al registrar periódicamente incidencias con inconsistencias en la recarga de estos vales que no se detectan fácilmente hasta que se produce una queja.

2.5. Prioridad de los objetivos

De estos objetivos fijados, el más crítico y que debe poder completarse dentro del tiempo fijado en el proyecto es el procesado automático de errores mediante reglas. Considerando que el desarrollo de la herramienta de creación de ficheros de devolución sirve de soporte a los test de validación del proceso y que por lo tanto es interesante cumplir ese objetivo primero, el orden de prioridad para los objetivos es el siguiente.

1. Generación de ficheros de devolución
2. Mantener un histórico de los errores de devolución
3. Procesado automático de errores de devolución aplicando reglas
4. Visualización de devoluciones en vale

En las fases iniciales del proyecto se decidió integrar el histórico de los errores de devolución como parte del proceso automático de errores, hecho que se verá reflejado tanto en la planificación de las etapas del proyecto como en los resultados.

3 Estado del arte

Las empresas que se dedican a la venta online necesitan, además de un canal de venta por el que ofrecer sus productos, un sistema para mantener un control de sus recursos y de la información generada por el negocio.

Una de las soluciones más populares es el uso de un sistema de *Enterprise Resource Planning* o ERP, integrando mediante módulos toda la lógica del negocio que pueda necesitar la empresa, como logística, control de stock en almacenes, contabilidad, etc. Dentro de la oferta de soluciones ERP, de las más conocidas y utilizadas son Microsoft Dynamics, Oracle JD Edwards EnterpriseOne o SAP Business [1].

Cuando incluimos el e-commerce como parte del negocio, el mercado también dispone de módulos integrables en sistemas ERP, como K-ecommerce, Dynamics eShop, Oracle Commerce, etc. Estas soluciones también incluyen las herramientas para desarrollar la arquitectura de una tienda online, unificando la creación del canal de venta junto centralizar todos los aspectos de la gestión, evitando posibles problemas derivados de la comunicación al tratar con sistemas independientes entre sí.

De todas formas, cuando hablamos de grandes empresas, especialmente con el alto nivel de exigencia dentro del sector de la moda, es habitual que se prefiera una solución a medida que no es fácil de satisfacer con este tipo de soluciones. Las grandes empresas del sector de la moda, como el grupo Inditex, el Corte Inglés o la propia Mango apuestan por este tipo de arquitectura para su sistema.

La solución escogida para la integración del e-commerce influye en la forma de procesar los pedidos y sus devoluciones a nivel interno, pero el ciclo de vida de los pagos mantiene un denominador común, que es la integración con los canales de pago.

Esta integración requiere el tratamiento de la información generada del negocio y el consumo del webservice específico del método de pago. Estos webservices se basan en arquitecturas SOA o REST, utilizando XML o JSON como estándar para el envío de la información.

Para el consumo de estos servicios se debe crear un cliente que realice las distintas peticiones, implementable en lenguajes de programación multipropósito como Java o C#. Hoy día es frecuente que se utilice Spring, un framework opensource basado en Java, para desarrollar este cliente por la simplicidad que ofrece usando su módulo Spring MVC o su implementación para Webservices.

Debido al contexto de este trabajo final no podemos explorar la inclusión de una solución nueva, sino que es necesario trabajar sobre la arquitectura ya existente en la empresa.

4 Metodología seguida

A continuación se expone la metodología de desarrollo software elegida para llevar a cabo los objetivos marcados en el trabajo y las herramientas utilizadas.

4.1. Metodología de desarrollo

Dentro del área de e-business los equipos de desarrollo trabajan siguiendo una metodología agile, Scrum. La intención inicial a la hora de definir la metodología a seguir era utilizar una agile, pero se optó por escoger una alternativa en vez de seguir la línea de la empresa. En este caso, el autor se decantó por adoptar al máximo las fases de la metodología Feature Driven Development o FDD [2] para la realización del proyecto.

Esta metodología agile define un proceso de iteraciones cortas cuyo objetivo es proporcionar resultados periódicos y tangibles. El énfasis del FDD es en las características, funcionalidades del valor para el cliente que se concretan en una relación acción-resultado-objeto.

Siguiendo los procesos del FDD (ver figura 2), el desarrollo del proyecto incluye las siguientes etapas.

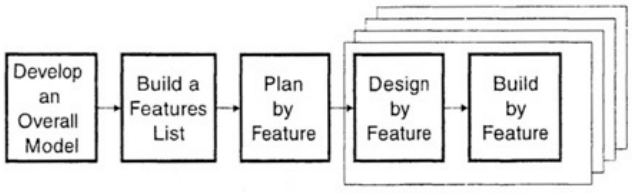


Fig. 2: Fases de un proyecto según la metodología FDD.

1. **Desarrollo de un modelo global:** se diseña un modelo del sistema teniendo en cuenta el contexto, la visión del proyecto y los requisitos capturados. En la metodología base se realiza un diseño de clases por cada área identificada, pero para trabajo se ha optado por solo plantear un diagrama de la arquitectura (ver figura 3) del área de e-business para esta fase, con el encaje de cada una de las funcionalidades dentro de la misma. Esta decisión se toma por considerar que trabajando sobre una arquitectura ya creada y con posibles cambios externos no tiene sentido definir una estructura de clases completa desde el inicio.
2. **Construcción de una lista de características:** construcción de un listado resumen de las funcionalidades

del sistema, creando funcionalidades más pequeñas y concretas a partir de las generales. Este proceso debe realizarse de conjuntamente con el cliente para que toda característica incluida tenga su conformidad.

Para este proyecto se ha optado por generar una primera versión del documento e ir ampliándolo con las funcionalidades más específicas al comenzar cada fase posterior a la inicial del proyecto. Esta decisión también viene motivada por la naturaleza del proyecto, permitiendo mayor flexibilidad a la hora de encarar posibles modificaciones en el sistema durante el transcurso del trabajo.

3. **Planificación por característica:** ordenación y priorización de las funcionalidades incluidas en la lista de características. Como este proyecto se lleva a cabo por una única persona, no se hace una asignación a distintos programadores de las funcionalidades, solo se priorizan cada una de las características.

4. **Diseño por característica:** con una o varias funcionalidades de la lista se fijan los objetivos de un ciclo de iteración. Se realiza el diseño de la funcionalidades de esa iteración, manteniendo siempre presente el foco solo en las características de la iteración actual.

5. **Construir por característica:** con el diseño de la iteración como base se implementa la funcionalidad en el sistema. Para esta parte del proceso se decide por seguir parte de la metodología de desarrollo Test-Driven development o TDD.

   Con esta metodología se opta primero por el desarrollo de test unitarios en primer lugar y después ir implementado código que supere estos test. Es un proceso incremental donde la cobertura de los test unitarios va aumentando y el código crece en consecuencia. Se incluye también refactorización de los métodos para eliminar código redundante, simplificar los métodos, etc.

   Para facilitar este tipo de test se han utilizado mock-up objects, que simulan la interacción con otras clases complejas de instanciar. Con esta técnica podemos imitar, por ejemplo, el comportamiento de una clase que realiza transacciones con una base de datos y devuelve un objeto que la clase sometida a test debe procesar.

   También se han realizado tests de integración de los componentes de cada una de las principales funcionalidades (ver sección 2).

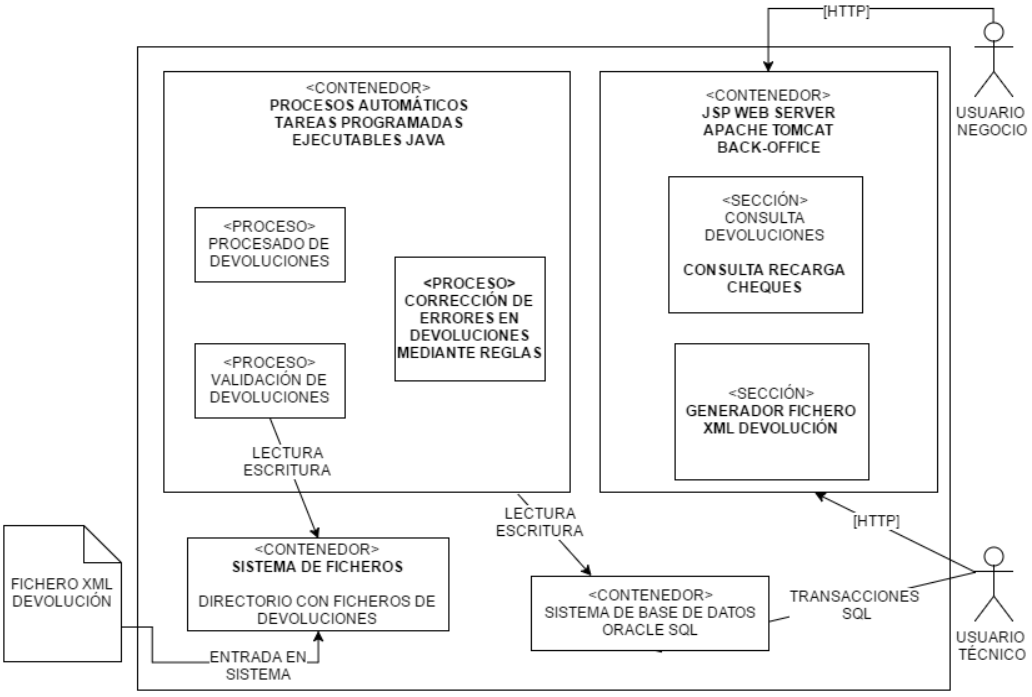


Fig. 3: Diagrama de la arquitectura del sistema con los elementos relevantes para el proyecto

Se ha escogido utilizar la metodología FDD con los cambios explicados arriba al considerar que la división de las fases del proyecto según las funcionalidades era más natural que trabajar con una metodología basada en sprints cortos, como Scrum.

La apuesta por el TDD como forma de desarrollo al construir las características tiene la intención de poder asegurar la calidad del producto generado.

El desarrollo de esta forma implica una inversión de tiempo adicional para comprender en detalle cuales son los casos base y los necesarios para garantizar la corrección de la solución.

4.2. Herramientas escogidas

Parte de las tecnologías que se han utilizado en el desarrollo del proyecto se han escogido por ser las que se están utilizando en la empresa y así facilitar la integración de los entregables generados al finalizar el trabajo.

Para la parte de desarrollo que se realiza sobre la back-office se ha utilizado como tecnología base Java Server Pages o JSP [3], basado en servlets de Java, programas ejecu-

tables a nivel de servidor. La implementación de la parte del cliente se ha hecho utilizando lenguajes XHTML y JQuery, este último por facilitar la creación de scripts que mediante el uso de Javascript requeriría de mayor esfuerzo de implementación.

Para el back-end se ha utilizado Java EE en la implementación de la lógica, elección natural teniendo en cuenta el uso de JSP, y Spring. Spring es un framework para desarrollo de aplicaciones basado en Java que ofrece una característica denominada *Dependency Injection* [4], que consiste en la definición de dependencias entre objetos a través de un constructor, argumentos de un método factoría o propiedades definidas en un fichero que se aplican tras la construcción del objeto.

Esta característica implica que no es necesario instanciar en código de forma explícita los objetos de los que se tiene alguna dependencia, sino que al generarse una instancia del objeto, éste se encarga de instanciar a su vez el resto de objetos de los que depende.

Esto es útil para sistemas donde se busca poco acoplamiento de clase. Se ha decidido aplicar esta característica en el proyecto como forma de facilitar el testing, jugando con inyección de dependencias para utilizar una clase u otra en las pruebas. Además, se aprovechará la implementación de JDBC Template [5] para gestionar las transacciones con la base de datos a traves de Data Access Objects o DAOs.

Precisamente para facilitar el test unitario se ha escogido incluir como tecnologías a utilizar Mockito [6], un framework de testing basado en Java que facilita la generación de objetos mock-up.

Finalmente, para la serialización y deserialización de datos en formato XML, necesario para parte de las características que se quieren introducir en el sistema, se ha escogido el uso de Single XML Framework [7]. Este framework también está basado en Java y su principal ventaja es que no requiere configuraciones complejas, realizando anotaciones en las clases que contengan la información a serializar en formato XML es suficiente.

4.3. Planificación del proyecto

A raíz de los objetivos planteados, la metodología de desarrollo a seguir y la necesidad de adquirir conocimientos en tecnologías que no se han utilizado en el ámbito personal o durante la carrera, definimos las siguientes etapas para la elaboración del proyecto.

1. Fase inicial del proyecto.

   Desarrollo del modelo global

   Captura de características y requisitos general

2. Primera iteración: Generación de ficheros de devolución

   Captura de requisitos y características

   Diseño de la funcionalidad

   Implementación y testing

3. Segunda iteración: Procesado automático de errores de devolución aplicando reglas

   Captura de requisitos y características

   Diseño de la funcionalidad

   Implementación y testing

4. Tercera iteración: Visualización de devoluciones en vale

   Captura de requisitos y características

   Diseño de la funcionalidad

   Implementación y testing

5. Integración de las funcionalidades en el sistema

5 RESULTADOS

A continuación se exponen los resultados obtenidos al seguir la planificación, explicando las particularidades de cada etapa y mostrando en la medida de posibles evidencias de su implementación. Al ser un proyecto desarrollado en empresa hay limitaciones a la hora mostrar detalles internos, en este caso referentes a la arquitectura de clases final o a código específico.

5.1. Generación de ficheros de devolución

5.1.1. Requisitos finales

En la fase de concreción de las funcionalidades a implementar en esta etapa se definen los siguientes puntos:

- Se debe poder buscar un pedido existente en base de datos a partir de su identificador único. Los datos a mostrar deben ser los mismos que los proporcionados por la herramienta de consulta de devoluciones.
- Debe darse la opción de seleccionar un tipo de devolución completa del pedido, que seleccione todos los artículos.
- Debe poder seleccionarse cada producto del pedido junto con un motivo de devolución (condición necesaria para procesados posteriores).
- Los posibles motivos de devolución deben cargarse desde base de datos, pues es la única información de una devolución que puede verse sujeta a cambios con facilidad.
- Debe poder seleccionarse el transporte de expedición del pedido y también otros conceptos para incluirlos en la devolución.
- Debe incluirse un botón para a partir de los datos seleccionados descargar un fichero XML que siga el formato utilizado en producción.

De forma autónoma se han incluido dentro del listado de características para esta herramienta la inclusión de alertas en caso de no cumplir las condiciones necesarias para usar cada opción. Por ejemplo, querer generar una devolución sin haber seleccionado productos.

5.1.2. Diseño

Se genera primero un mock-up de la interfaz gráfica de la aplicación, basada en la de la herramienta de consulta de devoluciones. Se toma esta decisión por estar los futuros usuarios familiariazados con ella y así facilitar la primera toma de contacto.

Para poder cumplir el requisito de parametrizar desde base de datos la carga de motivos de devolución se genera una nueva entidad en base de datos. Esta entidad incluye datos como el código de cada motivo, su descripción y si corresponden a una devolución completa o parcial del pedido.

El diseño de la funcionalidad sigue el patrón MVC aplicado para JSP [8]. Solo se incluye una vista, que debe mostrar elementos estáticos y datos obtenidos de dos beans, una que actua como formulario que almacenará los datos a mostrar y un controlador. Éste último gestionara la actualización de la vista a través de modificaciones de la bean del formulario y la comunicación con un DAO específico de esta aplicación.

Para poder representar la información recuperada de base de datos se definen entidades para representar a un pedido y los artículos.

Finalmente, para poder generar el fichero de devolución se crean una serie de clases que actúan de entidades. Estas entidades implementaran el sistema de anotaciones del Simple XML Framework para serializar los datos recogidos de la aplicación. El proceso de serialización y la generación de la descarga se gestionaran desde la clase controladora.

5.1.3. Implementación y testing

Durante la fase de implementación se ha ido construyendo la aplicación realizando pequeñas fases iterativas. Partiendo de replicar las características de la herramienta de consulta, se han ido ampliando las funcionalidades del lado del cliente, como la selección de productos (ver figura 4) para generar la devolución, la devolución completa, etc.

En cuanto al testing, principalmente se han realizado pruebas unitarias sobre el controlador de la aplicación, utilizando la inyección de dependencias de Spring y Mockito para emular los datos devueltos por el DAO, consiguiendo cobertura de todos los métodos del controlador.

Además de pruebas de integración validando que la aplicación se ajusta a los requisitos definidos, se pasa un test de validación para cerrar la iteración subiendo los cambios de la back-office a un servidor de test y permitiendo que el cliente dé su aprobación.

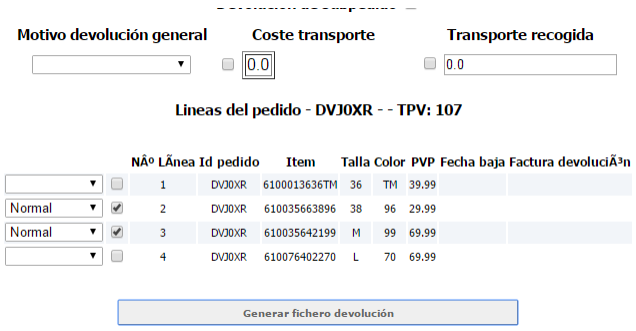


Fig. 4: Selección de productos en la herramienta de generación de ficheros de devolución

5.2. Procesado automático de errores de devolución aplicando reglas

5.2.1. Requisitos finales

Al ser la fase más importante del proyecto y también la más compleja debido a la gran cantidad de diseño e implementación necesaria para una cobertura total de errores, se acabaron definiendo las siguientes características a incluir en esta iteración:

- Las reglas de tratamiento deben cargarse desde la base de datos, incluyendo un parámetro que indique si están activas o no.
- El proceso debe cargar los errores registrados desde la ultima ejecución del proceso.
- El proceso debe mantener un histórico de errores de devoluciones tratados.
- El proceso debe seguir una serie de pasos (ver figura 5) que incluyen la consulta en un histórico de errores, el procesado de los errores aplicando las reglas y la notificación en caso de no poder tratar el error por alguna causa.
- Una restricción para la arquitectura del proceso es que debe tener como base la arquitectura de procesos definida en el área de e-business.

Con estos requisitos en mente, el objetivo de la iteración de esta funcionalidad es diseñar e implementar la base del proceso, además de implementar una regla para poder validar la arquitectura del mismo.

5.2.2. Diseño

Para poder cumplir los requisitos de la carga de reglas desde base de datos se empieza primero incluyendo dos nuevas entidades en base de datos.

1. Entidad con la información de las reglas. Contiene el nombre de una regla, si trata errores de la fase de validación o procesado del circuito de devoluciones, una identificación del error y un campo que refleje si la regla está activa.
2. Entidad con la información de errores de devolución tratados. Contiene el identificador único de la devolución, identificador único del pedido, el error generado, la fase del circuito a la que pertenece ese error y la fecha en la que se inserta el registro.

Una vez definidas las entidades, se prepara el diseño de la arquitectura del proceso. Partiendo de la restricción sobre la arquitectura, la clase principal del proceso implementa una interfaz que define la ejecución de tres métodos.

Dos de ellos son métodos para realizar pre y post ejecución de la lógica del proceso. En el primero se realiza la apertura de conexiones con la base de datos y la carga de las configuraciones de contexto, que nos sirve para definir si estamos realizando una ejecución en un entorno de producción o uno de testing. En el segundo se realiza el cierre de las conexiones abiertas anteriormente y gestiona la creación y envío del mail de aviso si se encontrará algún caso que justificase el generarlo.

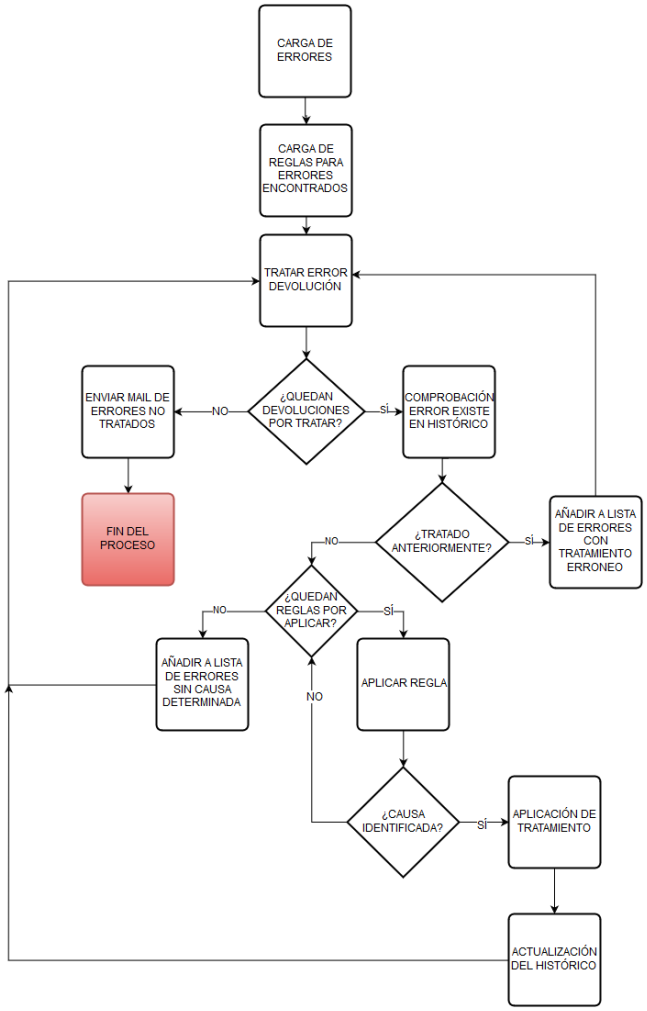


Fig. 5: Diagrama de flujo del proceso de tratamiento mediante reglas de errores.

El método de procesado de la clase principal se encarga de gestionar la lógica que refleja el diagrama de flujo (ver figura 5) del proceso. Esto incluye las llamadas a un DAO que recupera los errores de base de datos, las reglas, realiza la consulta del histórico de devoluciones de base.

Para poder procesar un error es necesario incluir en el diseño una entidad que represente una devolución fallida, incluyendo información de los productos del pedido original.

Como explicábamos en la sección 1.2, también es frecuente tener que generar ficheros nuevos o modificar el fallido, así que en esta clase entidad se incluye un atributo que representa la información almacenada en el fichero XML de la devolución fallida.

Para la implementación de las reglas se definen tres interfaces base.

- **Rule:** clase base de una regla, gestiona la lógica de ejecución de un regla.
- **Analysis:** clase que gestiona la lógica de evaluación de las causas de un error.
- **Treatment:** clase que gestiona el tratamiento a aplicar para una devolución.

Cada regla deberá implementar la interfaz base, extendiendola si fuese necesario e instanciando las implementaciones correspondientes de clases de análisis y tratamiento.

Para evitar que la instanciación suponga un incremento de código excesivo en la clase principal del proceso se incluye en el diseño la aplicación del patrón de diseño orientado a objetos Factory [9], que nos permite delegar la instanciación a un método factoría y no necesitar conocer la clase concreta a instanciar.

Por ello, definimos clases factoría (ver figura 4) de **Rule**, **Analysis** y **Treatment**. La factoría de reglas se instancia desde el proceso principal, instanciando una de las implementaciones de **Rule** a partir del nombre de la regla recuperado en base de datos.

La instancia de **Rule** se encargará en su constructor de usar el resto de factorias para obtener las instancias concretas de **Analysis** y **Treatment** correspondientes a la regla.

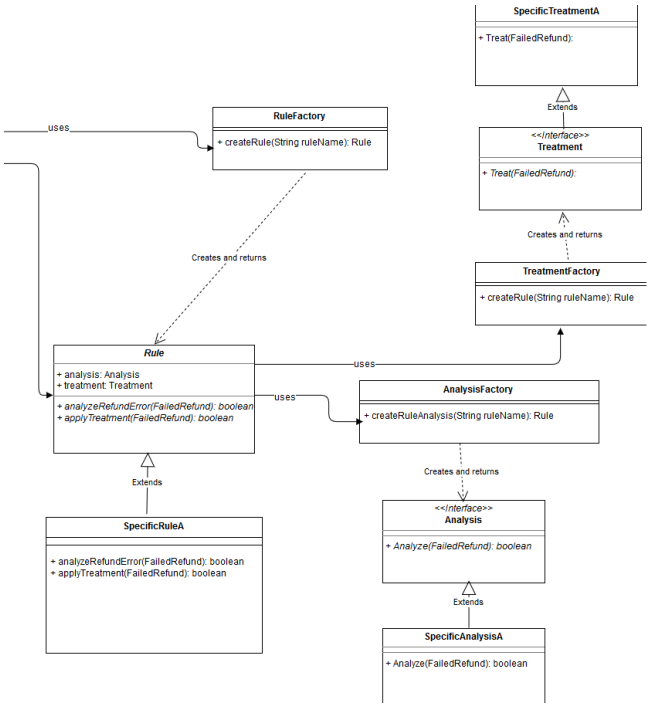


Fig. 6: Parte del diagrama de clases del proceso implementando el patrón Factory

5.2.3. Implementación y testing

De forma similar a lo explicado en el apartado 5.1.3, la implementación del diseño ha ido ligada a la definición de los casos de test a superar y de la creación de código en los métodos que los supere.

También se ha utilizado la inyección de dependencias de Spring junto con Mockito para emular el comportamiento del DAO y poder garantizar la cobertura del método principal.

Para poder superar los test de validación de esta funcionalidad era necesario implementar un regla concreta, sino estaríamos simplemente ofreciendo una base del proceso pero sin ninguna aplicación real.

Se ha escogido en este caso una regla que trata un error generado cuando en la fase de validación se detecta que se intenta insertar una devolución que ya ha sido completamente reembolsada.

Una de las causas más frecuentes suele ser que desde e-business se genera una devolución por no recibir el fichero correctamente en el sistema. Esta devolución se registra como una devolución normal y, en el caso de generar desde el sistema ERP el fichero de nuevo, provoca que al detectar una incogruencia salte el fallo.

Si bien esto no es una incidencia que afecta a cliente final, sí que influye en la gestión contable, pues los datos de facturación que se generan no coinciden.

Para esta regla en concreto el tratamiento consistiría en la actualización en base de datos de un campo definido a nivel de pedido.

La elección de este tipo de regla se hace para poder montar una prueba de integración del proceso funcionando con cada uno de sus elementos. También facilita la prueba de validación final, que se supera.

5.3. Visualización de devoluciones en vale

5.3.1. Requisitos finales

Para esta funcionalidad se nos solicita que la incorporemos como parte de la lógica de la herramienta de consulta de devoluciones, cumpliendo los siguientes requisitos.

- Solo debe mostrarse la información referente a vales cuando exista una devolución en el pedido que lo recargue.
- Debe mostrarse el detalle completo de movimientos del vale, diferenciando entre pagos y devoluciones.
- Debe mostrarse también el importe original del vale y el total restante.

5.3.2. Diseño

En esta etapa también se empieza con la elaboración de un mock-up de la interfaz gráfica, cogiendo la base de la herramienta e incluyendo la nueva sección.

A nivel de diseño de clases no se ha tenido que modificar nada de la arquitectura actual para poder realizar la implementación de esta funcionalidad.

5.3.3. Implementación y testing

También se ha seguido la metodología de diseñar primero los casos de test e ir desarrollando el nuevo código a partir de los mismos.

El resultado final de esta fase se puede ver en la figura 7. Los test de validación para esta fase han consistido en subir el artefacto de la back-office con los cambios a un entorno de test para su revisión por parte del cliente.

Fig. 7: Vista de la información de devoluciones en cheque

6 Conclusiones

En primer lugar debo destacar que se han conseguido cumplir las etapas definidas en la planificación, exceptuando una última fase de integración con el sistema para su subida a producción. Teniendo en cuenta que verano es un periodo crítico para la actividad de la empresa y que no es frecuente la subida de nuevas funcionalidades considero que tampoco es un factor negativo.

También me gustaría realizar una reflexión sobre la motivación original del trabajo, que era de forma autónoma plantear una solución que mejorase la gestión de las incidencias, especialmente en una parte tan importante como es la post-venta dentro del e-commerce.

La experiencia de llevar adelante un proyecto de estas características, en un entorno multinacional y contando con apoyo por parte de sus miembros ha sido sumamente interesante.

De las características que se introducen en el sistema, hay dos en particular que son de especial utilidad. La generación de ficheros de devolución permite agilizar tanto resoluciones de incidencias como testing de nuevas funcionalidades y el procesado automático de devoluciones puede permitir liberar parte de la carga de trabajo del equipo de soporte y también llegar a una visión más clara de posibles soluciones al problema raiz a la hora de plantear solución temporal.

Si bien este último comentario puede resultar algo extraño, me gustaría recordar que el proyecto se hace desde un equipo que no siempre tiene la posibilidad de intervenir sobre la causa raiz, al menos en este momento.

También he tenido la oportunidad de entrar en contacto con tecnologías nuevas para mi y que tienen aplicación en el mercado actual, como Spring, que tiene múltiples aplicaciones gracias a la amplia cantidad de módulos de los que dispone, o Mockito como herramienta de testing.

Desde luego, el trabajo desarrollado en este proyecto no es algo final, teniendo unas lineas futuras a seguir bastante claras.

6.1. Lineas de trabajo futuras

La tarea más evidente con la que continuar sería la implementación de más reglas para el proceso de tratamiento automático de errores.

El seguir por esta vía permite la inclusión de reglas hasta cierto punto, pues la escalabilidad del proceso no óptima. Para la implementación que se ha realizado no se ha añadi-

do un procesado concurrente de errores, principalmente por restricciones respecto a la base de datos.

Otras posibilidades incluyen la eliminación del uso de los DAO, que en función del crecimiento del proceso pueden suponer una explosión de clases. Como sustituto, se podría implementar uno o varios web services desarrollados con Spring e Hibernate, que gestionasen las transacciones con la base de datos. Como resultado de una petición REST al servicio web se podría gestionar la comunicación a traves de mensajes en formato JSON, más simple de gestionar y testear a la hora de realizar nuevos desarrollos.

AGRADECIMIENTOS

Para acabar este artículo quiere expresar mi más sincero agradecimiento a Sergio Herrero, el tutor en la empresa de este proyecto, por su compromiso con el trabajo y el apoyo prestado a lo largo del proyecto.

También quiero agradecerle a Xavier Roca Marva su consejo y guía en los aspectos más formales del trabajo, especialmente con el formato y contenido de los distintos entregables.

Finalmente, quiero agradecerle a Lucía por ser un apoyo externo al proyecto y una pequeña fuente de inspiración en momentos de atasco, fundamentales para superar pequeños escollos personales.

BIBLIOGRAFÍA

[1] SelectHub, Ranking on 2016 top ERP software systems", 01 Jun.2016; https://selecthub.com/news/selecthub-announces-top-10-erp-systems-for-2016/

[2] R. Mahdavi-Hezave and R. Ramsin, "FDMD: Feature-Driven Methodology Development" Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on. pp. 229–237, 2015.

[3] Oracle documentation on Java Standard Library Tags (JSTL), 30 Abr.2016; http://docs.oracle.com/javaee/5/tutorial/doc/bnakh.html

[4] Pivotal Software Inc., "The IoC Container", 26 Feb.2016; http://docs.spring.io/autorepo/docs/spring/3.2.x/spring framework- reference/html/beans.html

[5] Pivotal Software Inc., "Data access with JDBC", 26 Feb.2016;http://docs.spring.io/spring/docs/current /springframework- reference/html/jdbc.html

[6] Mockito project Wiki, "Mockito gitHub Wiki", 25 Mar.2016; https://github.com/mockito/mockito/wiki

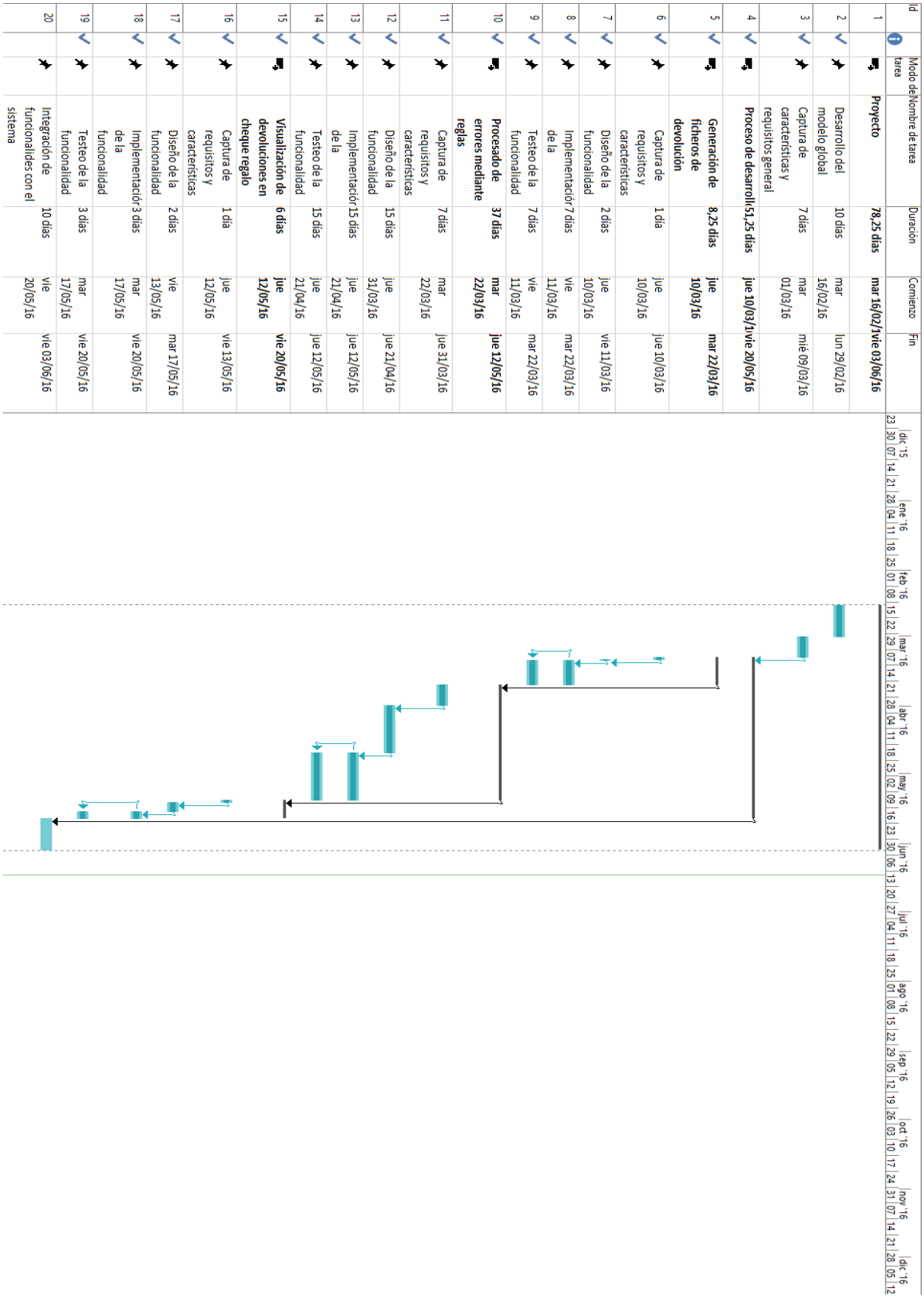
[7] Simple XML Framework Main Page, 23 Mar.2016; http://simple.sourceforge.net/

[8] javaTPoint, "MVC in JSP", 23 Mar.2016; http://www.javatpoint.com/MVC-in-jsp

[9] Object Oriented Design section on Factory Design Pattern, 4 May.2016; http://www.oodesign.com/factory-pattern.html

APÉNDICE

A.1. Diagrama de Gantt del proyecto



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**Proyecto**	**78,25 días**	**mar 16/02/1**	**vie 03/06/16**
2		Desarrollo del modelo global	10 días	mar 16/02/16	lun 29/02/16
3		Captura de características y requisitos general	7 días	mar 01/03/16	mié 09/03/16
4		**Proceso de desarroll**	**51,25 días**	**jue 10/03/1**	**vie 20/05/16**
5		**Generación de ficheros de devolución**	**8,25 días**	**jue 10/03/16**	**mar 22/03/16**
6		Captura de requisitos y características	1 día	jue 10/03/16	jue 10/03/16
7		Diseño de la funcionalidad	2 días	jue 10/03/16	vie 11/03/16
8		Implementación de la	7 días	vie 11/03/16	mar 22/03/16
9		Testeo de la funcionalidad	7 días	vie 11/03/16	mar 22/03/16
10		**Procesado de errores mediante reglas**	**37 días**	**mar 22/03/16**	**jue 12/05/16**
11		Captura de requisitos y características	7 días	mar 22/03/16	jue 31/03/16
12		Diseño de la funcionalidad	15 días	jue 31/03/16	jue 21/04/16
13		Implementación de la	15 días	jue 21/04/16	jue 12/05/16
14		Testeo de la funcionalidad	15 días	jue 21/04/16	jue 12/05/16
15		**Visualización de devoluciones en cheque regalo**	**6 días**	**jue 12/05/16**	**vie 20/05/16**
16		Captura de requisitos y características	1 día	jue 12/05/16	vie 13/05/16
17		Diseño de la funcionalidad	2 días	vie 13/05/16	mar 17/05/16
18		Implementación de la funcionalidad	3 días	mar 17/05/16	vie 20/05/16
19		Testeo de la funcionalidad	3 días	mar 17/05/16	vie 20/05/16
20		Integración de funcionalidades con el sistema	10 días	vie 20/05/16	vie 03/06/16