Gestión de muestras estándar

Alberto García Balaguer
ETIG / ETIS

Vicenç Font Sagrista

15/06/2015

Índice

Introducción

Desde hace años que trabajo en el mundo de la informática a través de una empresa farmacéutica.

Observando durante el tiempo la gestión que hacían del laboratorio, me sorprendió la cantidad de tiempo perdido en el mantenimiento de datos de inventariado de bajo nivel que había en el laboratorio. Al ser de pequeño ámbito, no se gestiona el stock de muestras en el laboratorio con un sistema logístico, si no con hojas Excel, y plantillas Word. Esto resulta muy ineficiente ya que es difícil de que más de una persona lo mantenga de una manera eficiente, y los datos pueden ser inconsistentes, además correr riesgos de errores humanos en la gestión de la hoja.

Lógicamente, dadas estas circunstancias, la idea de un gestor de base de datos que almacene esta información y la muestre a los usuarios, con búsquedas rápidas y efectivas, así como su transformación en informes que sustituyan a las plantillas Word, es una idea que aparece de manera natural. Así es como nace la motivación de este proyecto.

En concreto, este proyecto se centra en un tipo de muestra clave para un laboratorio: las muestras estándar. Estas muestras son la clave para comparar el resto de muestras. Es decir, toda muestra del producto que entre al proyecto, debe cumplir unos límites para ser aceptada como válida, y en caso de necesitarse un producto que se sabe que lo cumple, se debe usar la muestra estándar que esta almacenada “in-situ” en el laboratorio. Como el laboratorio puede gestionar muchos productos, es importante saber de dónde procede cada muestra y dónde está almacenada. Esta es la funcionalidad que resuelve esta aplicación. Además, cada muestra necesita presentar sus datos clave al analista, que, posiblemente, no tenga acceso a un ordenador para consultarlos, por los que necesitara las fichas, o informes oportunos para estas gestiones en formato papel.

Los requerimientos eran claros y concisos, por lo que, prácticamente se han podido cubrir todas estas funcionalidades.
Para el desarrollo de la aplicación, hemos explotado el producto Liferay Portal. Herramienta opensource, referente en el mercado de portales, y que nos ha servido para desarrollar con tecnologías estándar como los son Spring, o Hibernate. Así como desarrollar con la base de datos Oracle, explotando el lenguaje SQL a través de los reports, desarrollados con la herramienta, también opensource Jasper Reports. Con lo que he recorrido varios frentes dentro del mundo J2EE y he explorado herramientas como Eclipse para el desarrollo, o Tomcat para el despliegue.

Objetivo de este proyecto

Este proyecto tiene como objetivo la creación de una aplicación web que sirva como gestor e inventario de un laboratorio ficticio. En dicho laboratorio, se realizan análisis, los cuales necesitan una muestra estándar para comparar el resto de análisis. La aplicación a desarrollar se centrará en la gestión de esas muestras.

Básicamente, esta aplicación se centrará en el inventariado de estas muestras, representadas por el bote que las contiene, siendo este bote perteneciente a un lote en concreto, y este lote siendo de un único producto (definiendo así la estructura base de la aplicación). Además de la introducción y mantenimiento de estos elementos, la aplicación permitirá la explotación de estos datos, a través de búsquedas, listados, informes y un módulo de estadísticas, y permitiendo la exportación de estos datos a formatos externos (como PDF o Excel).

El proyecto estará basado en tecnologías J2EE, desplegado sobre el gestor de portales Liferay, funcionando con una base de datos Oracle.

Nos focalizaremos en el desarrollo de:

- Diseño y estructura de la base de datos
- Desarrollo de la capa de abstracción de datos
- Desarrollo de la capa controladora de la aplicación
- Desarrollo de la capa visual de la misma
- Implementación de librerías externas para la exportación de formatos

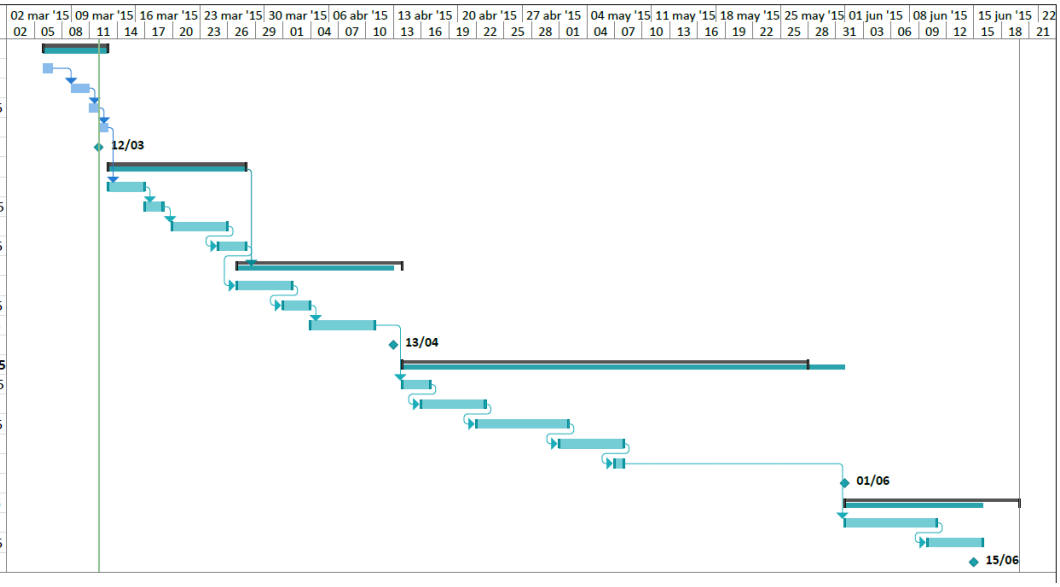
Enfoque y planificación

El desarrollo de este proyecto ha seguido diferentes etapas, en la que se ha dividido las tareas de desarrollo por las diferentes funcionalidades que la aplicación iba a resolver. Cada fase pretendía que al finalizarla, hubiese un pequeño producto con una funcionalidad completa (a bajo nivel, pero algo parecido a lo que pretende la metodología *scrum*). Al tratarse de un proyecto pequeño e individual, ha sido difícil seguir este tipo de metodología, pero ha servido de orientación para seguir un orden y una planificación lógica.

Inicialmente se desarrollarón las fáses básicas y lógica de cualquier proyecto de desarrollo de software: plan de trabajo, requerimientos, análisis, diseño, desarrollo y testing, para llegar al “goLive” final.

Se realizó una planificación a alto nivel, fácilmente que se podía seguir dada la dedicación que se puede dedicar a un proyecto desarrollado en un semestre (que ni mucho menos se le puede dedicar un recurso apenas unas horas semanales). Es por este motivo por lo que no se entró en planificaciones diarias, si no en periodos.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		**Plan de trabajo**	**5 días**	**vie 06/03/15**
2		Especificación del	1 día	vie 06/03/15
3		Toma de	2 días	lun 09/03/15
4		Generación de tareas	1 día	mié 11/03/15
5		Planificación	1 día	jue 12/03/15
6		Entrega PAC1	0 días	jue 12/03/15
7		**Analisis**	**11 días**	**vie 13/03/15**
8		Especificar	2 días	vie 13/03/15
9		Decidir alcance del	2 días	mar 17/03/15
10		Casos de uso	4 días	vie 20/03/15
11		Entidades necesarias	3 días	mié 25/03/15
12		**Diseño**	**12 días**	**vie 27/03/15**
13		Esquemas de casos de	4 días	vie 27/03/15
14		Flujo de visualización	3 días	mié 01/04/15
15		Diseño de intefaces de	6 días	sáb 04/04/15
16		Entrega PAC2	0 días	lun 13/04/15
17		**Desarrollo**	**32 días**	**mar 14/04/15**
18		Instalación de entornos	3 días	mar 14/04/15
19		Desarrollo de la lógica	5 días	jue 16/04/15
20		Desarrollo de la lógica	8 días	mié 22/04/15
21		Desarrollo de las	5 días	vie 01/05/15
22		Desarrollo del módulo	1 día	jue 07/05/15
23		Entrega PAC3	0 días	lun 01/06/15
24		**Entrega**	**15 días**	**lun 01/06/15**
25		Redacción de la	8 días	lun 01/06/15
26		Presentación	4 días	mié 10/06/15
27		Entrega final	0 días	lun 15/06/15



Aunque esta ha sido la planificación inicial, que a groso modo, se ha seguido, en la fase de desarrollo nos hemos centrado en dividirla en:

- Desarrollo de la capa de abstracción y el modelo de datos
- Desarrollo de la entrada de datos (alta/baja/modificación) de las entidades principales.
- Desarrollo de las búsquedas simples (sin diseño de la tabla principal).
- Desarrollo de las búsquedas complejas (sin diseño de la tabla principal).
- Desarrollo e investigación de la integración con los informes
- Diseño de los informes
- Diseño de la parte visual de la tabla principal

Siguiendo estos pasos, se ha intentado tener pequeños módulos funcionales en cada iteración.

Productos obtenidos

Como resultado final obtendremos un portal 100% operacional, que nos dará “out-of-the-box” muchas funciones (como publicación de contenido web, gestión documental, wiki, blogs, etc) siendo relevantes para este proyecto la gestión de usuarios (que luego nos entrega una API para manejarla desde nuestros desarrollos) y un aspecto visual base. Esto nos permite tener algunas partes básicas desarrolladas, y centrarnos en desarrollar la lógica de negocio, que es la que realmente va a aportar el valor añadido a este proyecto.

A nivel técnico, dentro de este producto de Portal, destacaremos el despliegue de los datos del mismo, y de los de nuestra aplicación sobre una base de datos Oracle. Referente en las bases de datos relacionales y estándar en el mundo empresarial. Se ha escogido el modelo XE (Xpress Edition) de distribución gratuita, y que proporciona todas las funciones básicas de Oracle (gestión de tablas, procedures, PL/SQL y schedules).

Para subir el portal, que al final es un conjunto de funciones J2EE, se ha elegido el servidor de aplicaciones Java de la fundación Apache Tomcat. Tomcat es capaz de soportar todo el backend desarrollado en nuestra aplicación y los JSP que presentan la parte visual. Tomcat es un referente en servidores de aplicación, dada su robustez y estabilidad, y su compatibilidad para hacer balanceo de carga y alta disponibilidad. Con esta instalación obtenemos un servidor completo, donde, además del portal, se pueden desplegar otras aplicaciones Java.

Además, hemos obtenido una serie de informes, desarrollados con la tecnología Jasper Reports. Estos informes, para este caso, están integrados dentro del despliegue de la aplicación, y se llaman desde la aplicación que los sirve a través de la web, pero están preparados para ser desplegados en un servidor Jasper y que este los sirvieran utilizando otros métodos.

Y por último, como uno, el producto principal seria el *portlet* desplegado que da toda la funcionalidad a la aplicación. Un usuario administrador del portal podría desplegar y colocar el *portlet* en cualquier página, y asignar los permisos que crea convenientes sobre estas páginas.

Contenido de la memoria

Esta memoria es un resumen y recopilación de la documentación generada en un proyecto de desarrollo de software.

En ella encontraremos los puntos relativos a:

- Los requerimientos iniciales con los que se basó la definición del proyecto
- La documentación le que se basó el análisis y diseño de la aplicación
- Ell mapa de casos de uso y definición de actores
- La definición de las tecnologías que se han utilizado para realizar el proyecto
- Un aporte sobre los métodos de desarrollo utilizados
- Las conclusiones a las que nos ha llevado este proyecto
- La documentación consultada para su desarrollo

Requerimientos generales

La aplicación, al tratarse de una aplicación para un entorno controlado de un laboratorio, debe cumplir una serie de requerimientos mínimos a nivel de seguridad y trazabilidad:

- Los usuarios accederán de manera personal, a través de un usuario y password.
- La aplicación almacenará, en todo momento, quién y qué se ha modificado en la aplicación (trazabilidad de datos).

Solución propuesta

Se propone la creación de un aplicativo, en entorno Web, que nos permita agilizar y mejorar la experiencia del usuario, quedando todo el proceso digitalizado y trazado en todo momento.

Requerimientos básicos

- La aplicación debe permitir el alta/baja/modificación de todos los datos que se definan, estando estos identificados de manera única, y almacenados en una estructura de tabla en una base de datos relacional.
- Se definirá una matriz de permisos y características para controlar el comportamiento y la visibilidad de los elementos.
- El sistema proporcionará búsquedas para hallar elementos concretos almacenados. Dichas búsquedas o listados, serán exportables a formatos externos (PDF o Excel).
- Existirán informes en formato PDF que proporcionaran formularios para el uso en el laboratorio.
- Un informe especial, será una etiqueta, para identificar el bote, proporcionada con un código de barras.

Requerimientos específicos

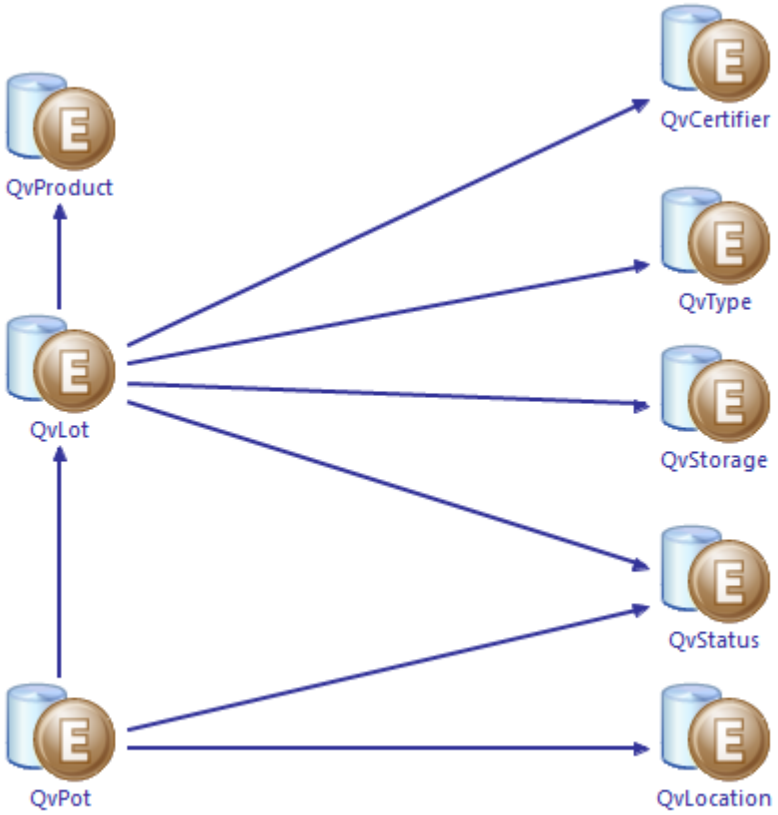
Existen una serie de requerimientos que son clave en el desarrollo de la aplicación, y que se deben valorar con especial énfasis:

- Los roles que habrá en el sistema serán tres:
    - Visualizador
    - Editor
    - Administrador
- Cada bote podrá pertenecerá a 1 lote, mientras que 1 lote pertenece a 1 producto.

- La visualización de los elementos seguirá esta estructura de árbol, se visualizará un producto del que colgarán lotes, de los que colgarán los botes.
- Tanto los lotes como los botes tendrán un estado, que será:
    - En uso
    - Caducado
    - Fuera de uso
    - Agotado
    - Reserva
    - Entregado
- Habrá un método para duplicar, o añadir clones, de botes ya existentes en el sistema.
- Todos los datos auxiliares (que aparecerán en los campos desplegables) deberán poder ser editados por un usuario con rol de edición.
- Los informes, según el tipo, irán ligados a un producto, a un lote, a un bote, o se accederán desde el módulo de estadísticas.

Esquema de entidades de la aplicación

Este sería un esquema aproximado de como quedarían las entidades, contando las posibles tablas auxiliares que puede tener el sistema.



Requerimientos técnicos

Arquitectura

Se propone una arquitectura basada en la tecnología “portlet”, desarrollada con J2EE, desplegada sobre el gestor de portales Liferay, utilizando las tecnologías Spring para la capa controladora y Hibernate para la abstracción de datos.

Entornos

Dentro del alcance del proyecto, se incluye la instalación y configuración de un servidor del producto **Liferay 6.2. Community Edition,** que será desplegado sobre un servidor de aplicaciones Java Tomcat 7.0, todo ello instalado en un servidor virtual sobre sistemas **CentOs 7.0.1** y funcionando como base de datos **Oracle XE 11g.**

Tecnologías de desarrollo

El proyecto estará basado, en todo momento, en el desarrollo con las diferentes variantes que nos da J2EE, y que implementa el producto Liferay, y que pone a disposición del programador:

- J2EE (programación directamente en clases java)
- Spring
- JavaScript
- Hibernate
- Webscripts
- Entornos de visualización web (CSS, UI, JS, YUI, AJAX)
- Así como la explotación de la API propia de Liferay

Además usaremos API externas para extender algunas funcionalidades:

- POI
- Jasper Reports

Todas estas tecnologías se utilizarán mediante el IDE Eclipse.

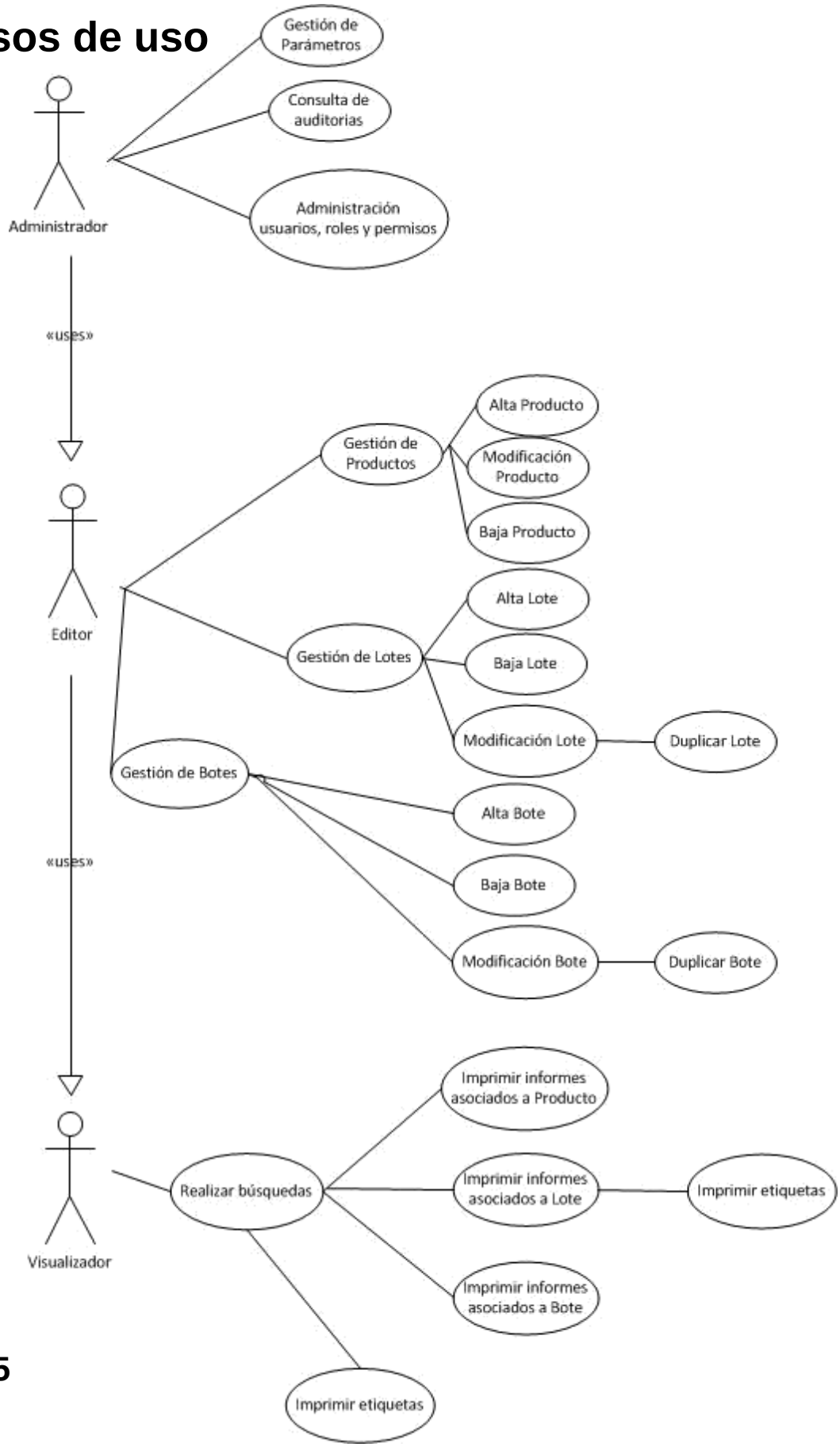
Funcionalidades

FUNCIONALIDAD DE LA APLICACIÓN	MÉTODO DE DESARROLLO
Alta/baja/modificación de elementos	Uso de la tecnología MVC Spring y Service Builder de Liferay
Gestión de usuarios, roles y permisos	Se usará la gestión de usuarios y roles que usa Liferay por defecto, implementando en el portlet las restricciones a uso y contenido pertinentes.
Auditoría de los elementos	Se utilizará una librería de Liferay para realizar el Audit, llamándola desde cada vez que se realice una acción que se quiera auditar.
Búsquedas de elementos	Se implementará métodos utilizando MVC Spring, Hibernate, o, incluso, para las más generalizadas, consultas Lucent (usando la indexación de elementos de Liferay).
Exportación de datos	Se implementará sobre la aplicación métodos de exportación a diferentes formatos, usando la API *POI*.
Reporting	Los informes estándar para impresión del sistema, se desarrollarán con la tecnología Jasper Reports.
Impresión de etiquetas	Las etiquetas se imprimirán por dos métodos: usando la tecnología Jasper, implementando fuentes de código de barras, y usando la integración de un servicio de impresión automatizada de etiquetas, externo a la aplicación, llamado MAEWIN.

Actores

- **Visualizador:** Ve la información, puede hacer
- **Editor:** Sube el documento, realiza las “working copy” y lanza el workflow cuando la edición finaliza.
- **Administrador:** tiene total control del repositorio, los roles y tiene capacidad para modificar cualquier fichero no aprobado.

Casos de uso


Gestión de Parámetros
Consulta de auditorias
Administración usuarios, roles y permisos
Administrador
«uses»
Gestión de Productos
Alta Producto
Modificación Producto
Baja Producto
Editor
Gestión de Lotes
Alta Lote
Baja Lote
Modificación Lote
Duplicar Lote
Gestión de Botes
Alta Bote
Baja Bote
Modificación Bote
Duplicar Bote
«uses»
Realizar búsquedas
Imprimir informes asociados a Producto
Imprimir informes asociados a Lote
Imprimir etiquetas
Imprimir informes asociados a Bote
Visualizador
Imprimir etiquetas

Desglose de los casos de uso

Añadir producto	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema
Postcondición	Hay un producto nuevo en el sistema
Casos de uso relacionados	Modificación de producto, baja de producto, añadir lote, imprimir informes de producto
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario pulsa el botón “Añadir Contenido”	
El usuario rellena los datos del producto	
El usuario pulsa “guardar”	

Añadir lote	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema y debe tener de alta un producto para asociarlo.
Postcondición	Hay un nuevo lote en el sistema
Casos de uso relacionados	Modificación de lote, baja de lote, imprimir informes de lote, añadir producto, añadir bote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario pulsa sobre el botón “Añadir lote” sobre ese producto	

El usuario rellena los datos del lote
El usuario pulsa “guardar”
Escenario alternativo
El usuario se conecta al sistema y hace log-in
El usuario pulsa el botón “Añadir Contenido”
El usuario añade un producto nuevo (caso de uso “añadir producto”)
El usuario accede (desde la misma pantalla anterior) a añadir X lotes para ese producto.

Añadir bote	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema y debe tener de alta un lote y un producto para asociarlo.
Postcondición	Hay un nuevo bote en el sistema
Casos de uso relacionados	Modificación de bote, baja de bote, imprimir informes de bote, añadir producto, añadir lote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario pulsa sobre el botón “Añadir bote” sobre ese lote	
El usuario rellena los datos del bote	
El usuario pulsa “guardar”	
Escenario alternativo	
El usuario se conecta al sistema y hace log-in	
El usuario pulsa el botón “Añadir Contenido”	
El usuario añade un producto nuevo (caso de uso “añadir producto”)	

El usuario añade un nuevo lote asociado a ese producto (escenario alternativo del caso de uso “añadir lote”)
El usuario accede (desde la misma pantalla anterior) a añadir X botes para ese lote.

Duplicar lote	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema y debe tener de alta un producto para asociarlo, y un lote como modelo.
Postcondición	Hay uno (o varios) nuevos lote en el sistema
Casos de uso relacionados	Modificación de lote, baja de lote, imprimir informes de lote, añadir producto, añadir lote, añadir bote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario pulsa sobre el botón “Duplicar lote” sobre ese lote	
El sistema muestra un pop-up preguntando al usuario cuantos el número de lotes duplicados que desea	
El usuario introduce el número y pulsa “aceptar”	
El sistema crea los lotes con los mismo datos que el lote origen	
El usuario puede editar esos lotes (si le interesa)	

Duplicar bote	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema y debe tener de alta un producto y un lote para asociarlo, y un bote como modelo.
Postcondición	Hay uno (o varios) nuevos bote en el sistema
Casos de uso relacionados	Modificación de bote, baja de bote, imprimir informes de bote, añadir producto, añadir lote, añadir bote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario selecciona un bote asociado a ese lote	
El usuario pulsa sobre el botón “Duplicar bote” sobre ese lote	
El sistema muestra un pop-up preguntando al usuario cuantos el número de botes duplicados que desea	
El usuario introduce el número y pulsa “aceptar”	
El sistema crea los botes con los mismo datos que el bote origen	
El usuario puede editar esos botes (si le interesa)	

Modificar producto	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema, y el producto debe de estar dado de alta en el sistema
Postcondición	El producto cambia en el sistema

Casos de uso relacionados	Alta de producto, baja de producto, añadir lote, imprimir informes de producto
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca el producto	
El usuario pulsa el botón “Modificar producto”	
El usuario modifica los datos del producto	
El usuario pulsa “guardar”	

Modificar lote	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema y debe tener de alta un producto para asociarlo y el lote debe estar dado de alta en el sistema
Postcondición	Cambia el lote en el sistema
Casos de uso relacionados	Alta de lote, baja de lote, imprimir informes de lote, añadir producto, añadir bote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario pulsa sobre el botón “Modificar lote” sobre ese lote	
El usuario modifica los datos del lote	
El usuario pulsa “guardar”	

Modificar bote	
Actor principal	Editor, Administrador
Precondición	Debe haber elementos emergentes dados de alta en el sistema y debe tener de alta un lote y un producto para asociarlo y el bote debe estar dado de alta en el sistema
Postcondición	El bote cambia en el sistema
Casos de uso relacionados	Alta de bote, baja de bote, imprimir informes de bote, añadir producto, añadir lote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario selecciona un bote asociado a ese lote	
El usuario pulsa sobre el botón “Modificar bote” sobre ese bote	
El usuario modifica los datos del bote	
El usuario pulsa “guardar”	

Baja de producto	
Actor principal	Editor, Administrador
Precondición	Debe estar el producto dado de alta en el sistema
Postcondición	Deja de haber un producto en el sistema
Restricciones	El sistema NO permitirá el borrado de un producto si existe un lote asociado
Casos de uso relacionados	Modificación de producto, alta de producto, añadir lote, imprimir informes de producto
Escenario principal	
El usuario se conecta al sistema y hace log-in	

El usuario busca en un producto
El usuario pulsa el botón “Dar de baja producto”
El sistema pregunta al usuario si está seguro de la acción
El usuario pulsa “Sí”

Baja de lote	
Actor principal	Editor, Administrador
Precondición	El lote debe estar dado de alta
Postcondición	Deja de haber un lote en el sistema
Reestricciones	El sistema NO permitirá el borrado de un lote si tiene botes asociados
Casos de uso relacionados	Alta de lote, modificación de lote, imprimir informes de lote, añadir bote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario pulsa sobre el botón “dar de baja lote” sobre ese lote	
El sistema pregunta al usuario si está seguro de la acción	
El usuario pulsa “Sí”	

Baja de bote	
Actor principal	Editor, Administrador
Precondición	El bote debe estar dado de alta
Postcondición	Deja de haber un bote en el sistema
Reestricciones	El sistema NO permitirá el borrado de un lote si tiene

	botes asociados
Casos de uso relacionados	Alta de lote, modificación de lote, imprimir informes de lote, añadir bote
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado a ese producto	
El usuario pulsa sobre el botón “dar de baja lote” sobre ese lote	
El sistema pregunta al usuario si está seguro de la acción	
El usuario pulsa “Sí”	

Realizar búsquedas	
Actor principal	Visualizador, Editor, Administrador
Precondición	-
Postcondición	-
Casos de uso relacionados	-
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto, por su descripción, o código	

Imprimir informes asociados a producto	
Actor principal	Visualizador, Editor, Administrador
Precondición	Debe haber un producto dado de alta en el sistema
Postcondición	Se generará un documento PDF con el informe
Casos de uso relacionados	Modificación de producto, alta de producto, búsqueda simple o avanzada
Escenario principal	

El usuario se conecta al sistema y hace log-in
El usuario busca un producto
El usuario pulsa en el botón “Imprimir informe de producto” del producto
El sistema sirve al usuario el PDF correspondiente

Imprimir informes asociados a lote	
Actor principal	Visualizador, Editor, Administrador
Precondición	Debe haber un producto y un lote dado de alta en el sistema
Postcondición	Se generará un documento PDF con el informe
Casos de uso relacionados	Modificación de lote, alta de lote, búsqueda simple o avanzada
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado	
El usuario escoge uno de los informes de lote disponibles	
El sistema sirve al usuario el PDF correspondiente	

Imprimir informes asociados a bote	
Actor principal	Visualizador, Editor, Administrador
Precondición	Debe haber un producto, un lote y un bote dado de alta en el sistema
Postcondición	Se generará un documento PDF con el informe
Casos de uso relacionados	Modificación de lote, alta de lote, búsqueda simple o avanzada
Escenario principal	

El usuario se conecta al sistema y hace log-in
El usuario busca un producto
El usuario selecciona un lote asociado, y un bote asociado al mismo
El usuario escoge uno de los informes de bote disponibles
El sistema sirve al usuario el PDF correspondiente

Imprimir etiquetas	
Actor principal	Visualizador, Editor, Administrador
Precondición	Debe haber un producto y un lote dado de alta en el sistema
Postcondición	Se generará una (o varias) etiquetas
Casos de uso relacionados	Modificación de lote, alta de lote, búsqueda simple o avanzada
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario busca un producto	
El usuario selecciona un lote asociado	
El usuario escoge la opción “impresión de etiquetas”	
El sistema muestra un *pop-up* preguntando el número de etiquetas a imprimir	
El sistema enviará a la impresora el número de etiquetas solicitado	

Exportar resultados	
Actor principal	Visualizador, Editor, Administrador
Precondición	Se debe haber realizad una búsqueda que arroje algún resultado
Postcondición	Se generará una hoja de calculo
Casos de uso relacionados	Búsqueda simple o avanzada

Escenario principal
El usuario se conecta al sistema y hace log-in
El usuario realiza una búsqueda
El usuario selecciona el botón “exportar resultados”
El sistema sirve una hoja de cálculo con los resultados

Consultar auditorias	
Actor principal	Administrador
Precondición	-
Postcondición	-
Casos de uso relacionados	-
Escenario principal	
El usuario se conecta al sistema y hace log-in	
El usuario realiza una búsqueda y accede a un elemento	
El usuario selecciona el botón “ver auditorias” del elemento	
El sistema sirve una tabla con las modificaciones realizadas sobre el elemento	

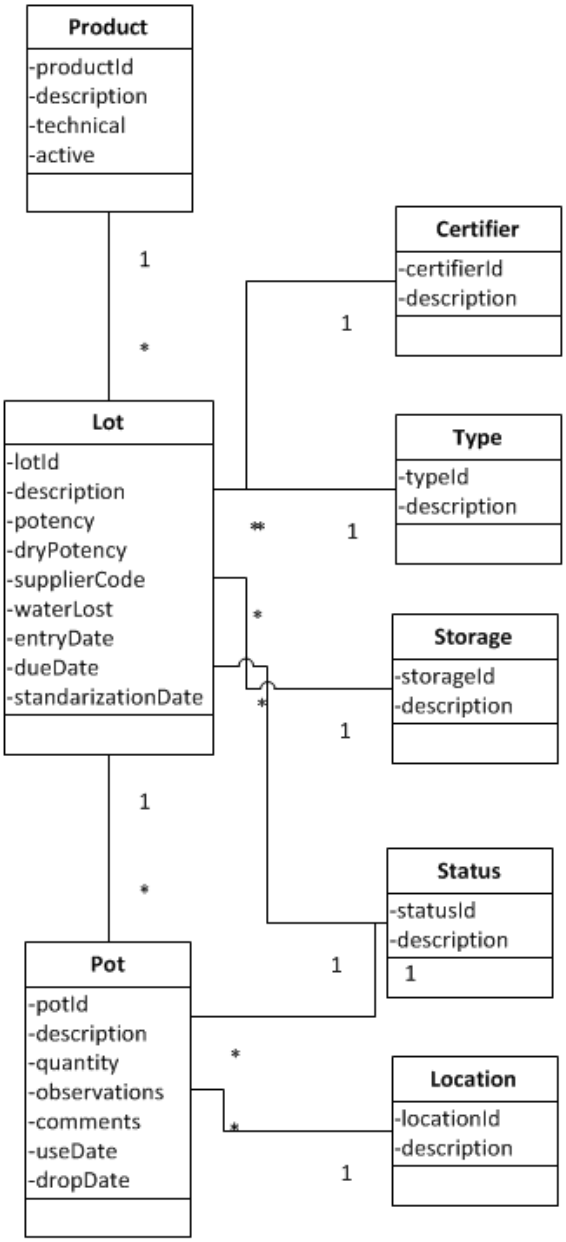
Casos de uso especiales (actor: administrador)

- **Gestión de parámetros:** El administrador tiene opción para cambiar los parámetros (elementos de los desplegables).
- **Administración de usuarios, roles y permisos:** El administrador tiene opción a asignar roles, determinar actores, y dar privilegios sobre carpetas o partes de repositorio a usuarios o grupos de usuario determinados.

Modelo de datos

Las 3 entidades principales serán: Producto, Lote y Bote. Las 3 entidades están relacionadas entre si, de manera que el producto tiene N Lotes, y el Lote N Botes. Es por ello que el nucleo de la aplicación es el producto, y todo estará relacionado con esta entidad.

Existirán entidades auxiliares que permitirán reaprovechar datos introducidos por el usuario, a modo de valores prefijados en un desplegable. Estos datos serán referenciados a las tablas principales mediante *foreign keys*.

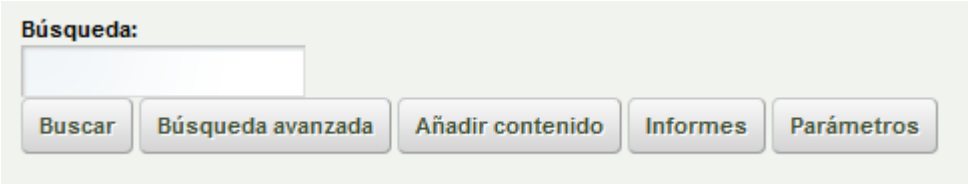


Diseño de las pantallas

El diseño a nivel visual de la aplicación, se basará en un diseño HTML limpio, y muy poco recargado para el usuario. La funcionalidad no es muy compleja, por lo que no hace falta grandes complejidades a la hora de mostrar los datos.

Pantalla inicial

La primera de las pantallas será absolutamente básica, dejando al usuario solo la opción de buscar un producto, o un código, y acceder a las opciones auxiliares de la aplicación:

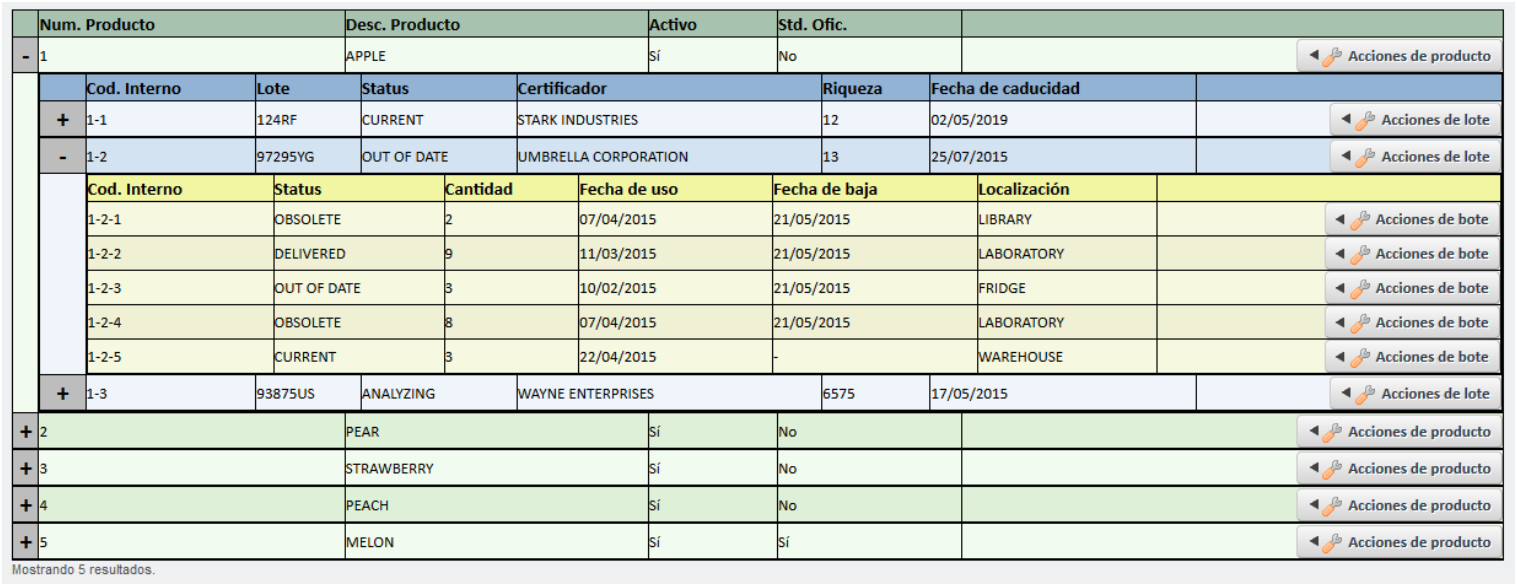


Búsqueda genérica

Si el usuario realiza la búsqueda desde el cuadro de la pantalla inicial, y pulsa en el botón “Buscar”, el sistema buscará esa cadena en una serie de campos, tanto campo completo, como fragmentos de este. En concreto se buscará en:

- Código de producto
- Código de lote
- Código de bote
- Descripción de producto

Cuando el sistema recuperé los resultados, estos se mostrarán en una tabla desplegable, de manera jerárquica, de manera que al usuario le sea fácil escalar entre los 3 niveles de datos. Una imagen de la tabla sería así:



	Num. Producto	Desc. Producto	Activo	Std. Ofic.	
-	1	APPLE	Sí	No	Acciones de producto

	Cod. Interno	Lote	Status	Certificador	Riqueza	Fecha de caducidad	
+	1-1	124RF	CURRENT	STARK INDUSTRIES	12	02/05/2019	Acciones de lote
-	1-2	97295YG	OUT OF DATE	UMBRELLA CORPORATION	13	25/07/2015	Acciones de lote

Cod. Interno	Status	Cantidad	Fecha de uso	Fecha de baja	Localización	
1-2-1	OBSOLETE	2	07/04/2015	21/05/2015	LIBRARY	Acciones de bote
1-2-2	DELIVERED	9	11/03/2015	21/05/2015	LABORATORY	Acciones de bote
1-2-3	OUT OF DATE	3	10/02/2015	21/05/2015	FRIDGE	Acciones de bote
1-2-4	OBSOLETE	8	07/04/2015	21/05/2015	LABORATORY	Acciones de bote
1-2-5	CURRENT	3	22/04/2015	-	WAREHOUSE	Acciones de bote

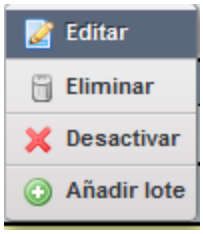
+	1-3	93875US	ANALYZING	WAYNE ENTERPRISES	6575	17/05/2015	Acciones de lote

+	2	PEAR	Sí	No	Acciones de producto
+	3	STRAWBERRY	Sí	No	Acciones de producto
+	4	PEACH	Sí	No	Acciones de producto
+	5	MELON	Sí	Sí	Acciones de producto

Mostrando 5 resultados.

Acciones de producto

Desde el nivel de la tabla de producto, se podrá acceder a un menú contextual dónde se podrán acceder a las opciones que tendremos disponibles referentes a producto:



- Editar: permite modificar los datos introducidos del producto.
- Eliminar: permite eliminar el producto.
- Desactivar: marca el producto como “no activo”.
- Añadir Lote: Permite añadir un nuevo lote sobre ese Producto

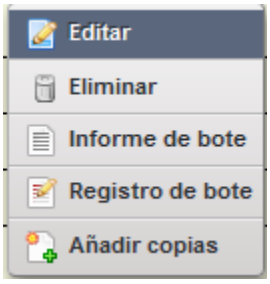
Acciones de lote

Desde el nivel de la tabla de producto, se podrá acceder a un menú contextual dónde se podrán acceder a las opciones que tendremos disponibles referentes a lote:

- Editar: permite modificar los datos introducidos del lote.
- Eliminar: permite eliminar el lote.
- Añadir bote: Permite añadir un nuevo bote sobre este lote.
- Informe de lote: el sistema sirve un PDF con un resumen del lote.
- Informe de todos los botes: el sistema sirve un PDF con todas las fichas de bote asociadas a este lote.
- Informe de todos los registros de botes: el sistema sirve un PDF con todos los registros de botes asociados a este lote.
- Imprimir etiquetas: el sistema preguntará al usuario cuantas etiquetas desea imprimir, y comunicará con la impresora para imprimirlas.

Acciones de bote

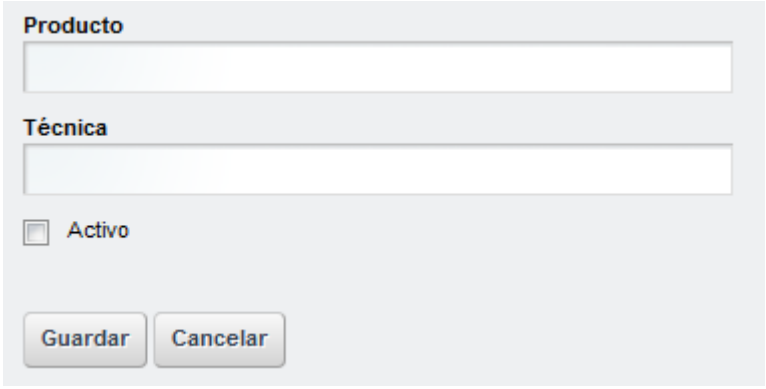
Desde el nivel de la tabla de producto, se podrá acceder a un menú contextual dónde se podrán acceder a las opciones que tendremos disponibles referentes a lote:



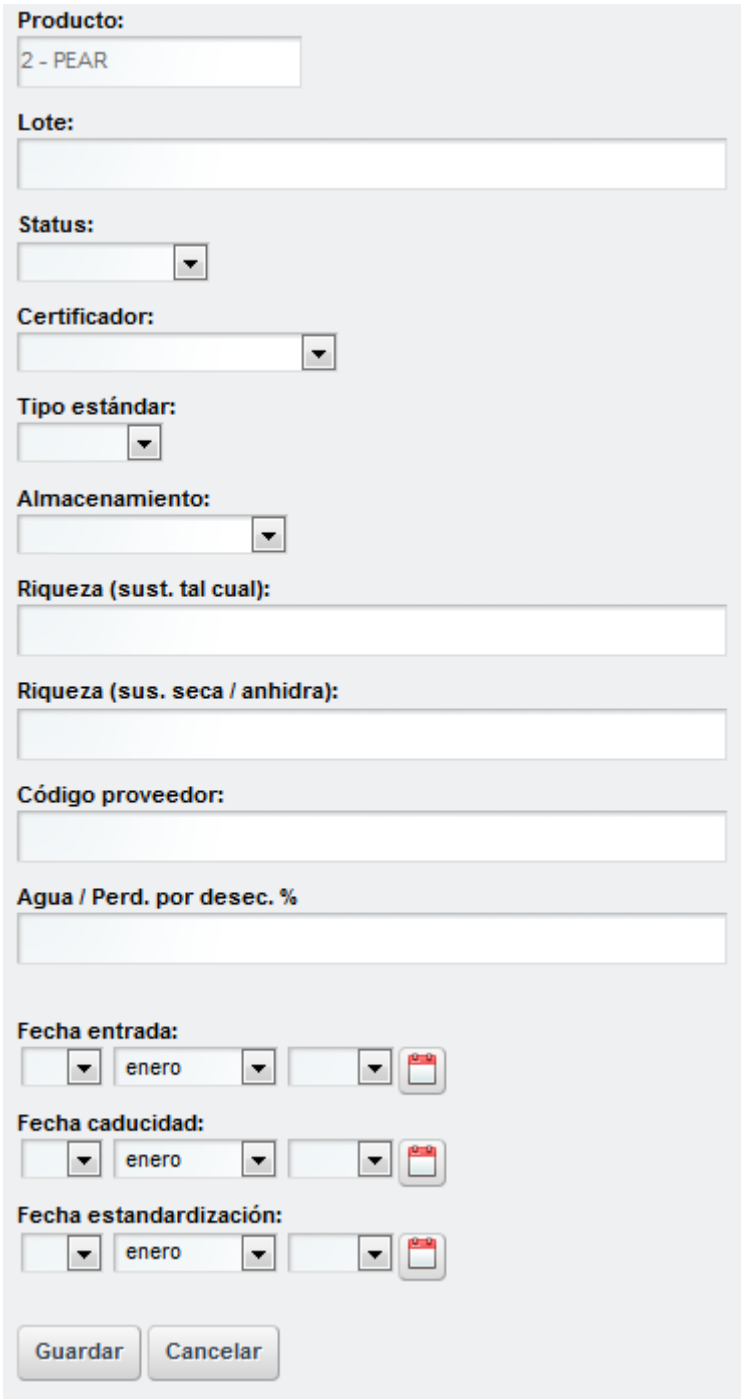
- Editar: permite modificar los datos introducidos del bote.
- Eliminar: permite eliminar el bote.
- Informe de bote: el sistema sirve un PDF con el resumen del bote.
- Registro de bote: el sistema sirve un PDF con el registro de bote.
- Añadir copias: el sistema preguntará al usuario el número de copias a crear de ese bote, y creará replicas, siempre haciendo avanzar la numeración interna del sistema.

Edición de elementos

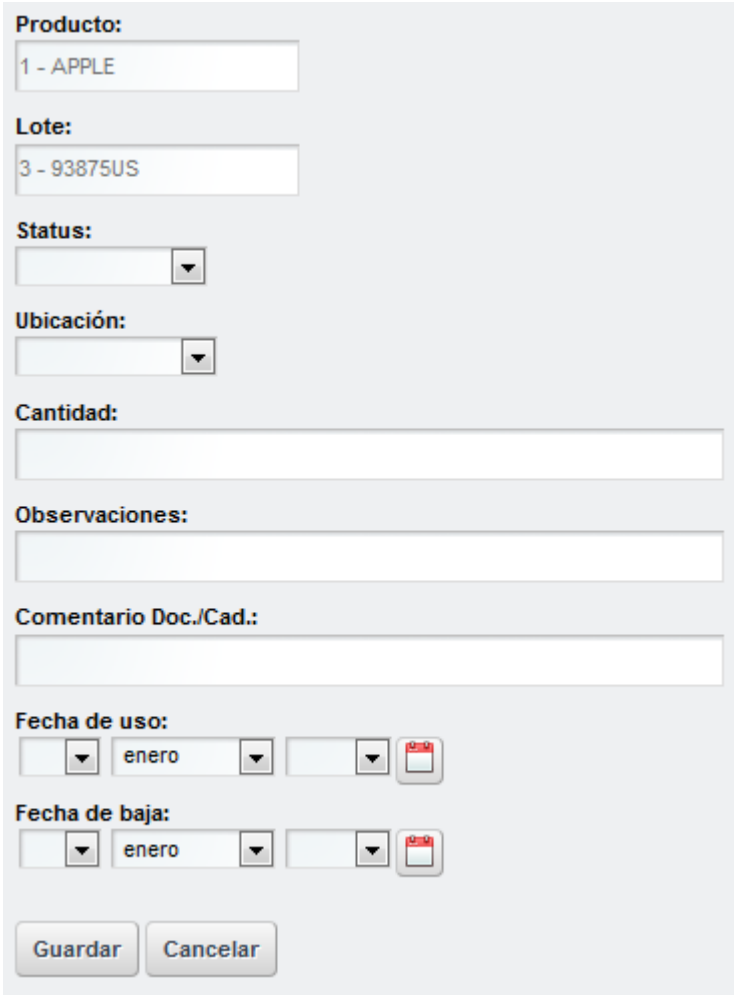
Los productos se editarán desde el siguiente formulario:



Así como este será el formulario de edición de lotes:

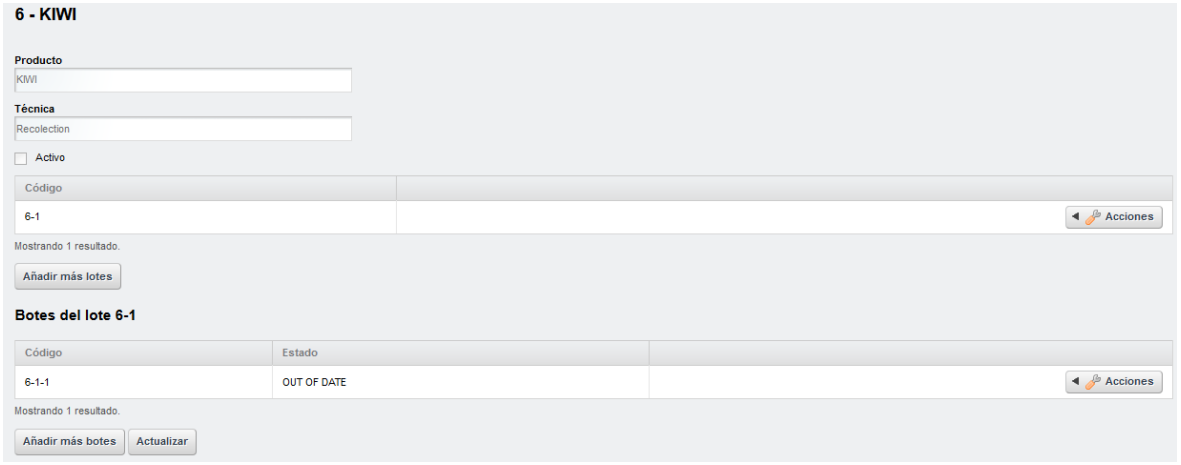


Y este el de botes:



Añadir contenido

A través del botón del añadir contenido, el usuario podrá añadir, desde el mismo sitio, un producto, X lotes, y X botes. El sistema irá almacenando pequeñas tablas donde el usuario podrá añadir y quitar elementos.



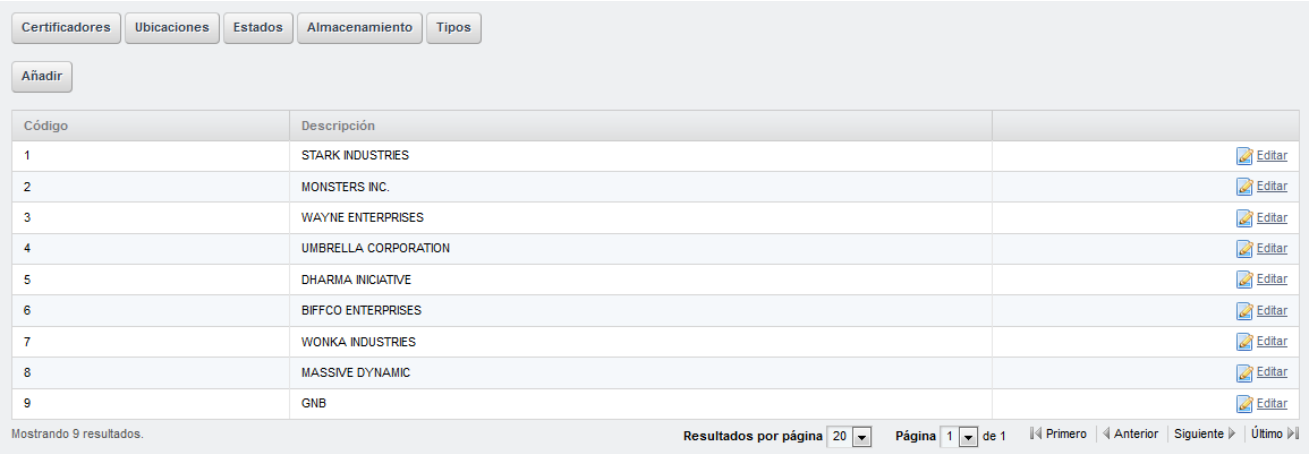
Informes

Los informes que no encajan en los menús anteriores, se situarán en esta pantalla:

- Informe de lotes a caducar: se mostrará un informe con los estándares que están próximos a la fecha de caducidad, dados 5 rangos de fechas.
- Informes a revisar: dados los parámetros de fecha de revisión, y los tipos escogidos, se mostrará un listado ordenado.
- Informe de productos: Se mostrará un resumen de los productos filtrando por su almacenamiento.

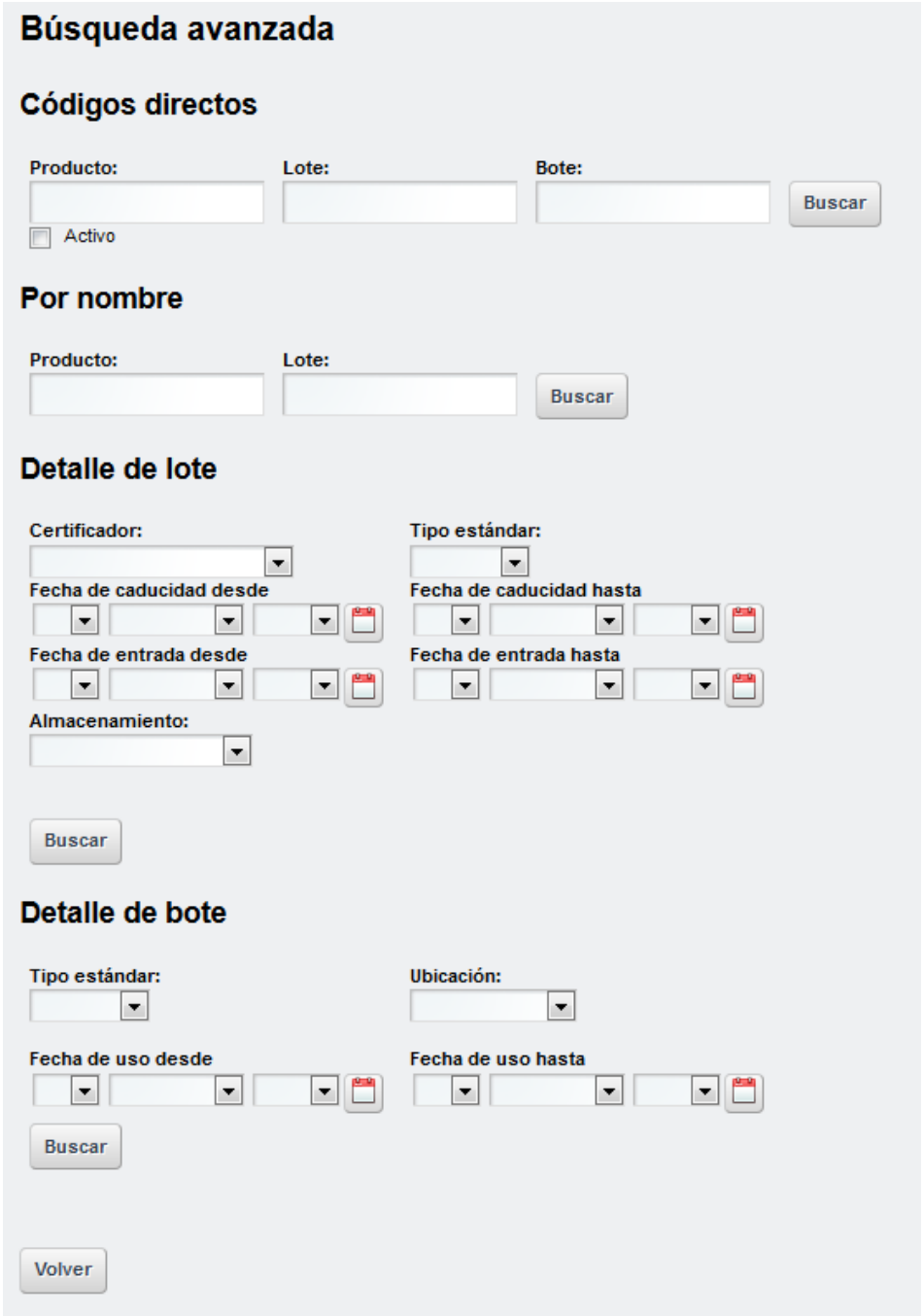
Parámetros

El mantenimiento de los parámetros se podrá realizar de manera que podremos cambiar entre los diferentes parámetros, y editar los existentes, o crear nuevos:



Búsqueda avanzada

La búsqueda avanzada permitirá al usuario buscar por casi cualquiera de los parámetros que existen en la aplicación, de manera que el se podrá hacer su filtro personalizado, y luego visualizarlo en la tabla/árbol vista anteriormente:



Aporte técnico

Arquitectura

Se propone una arquitectura basada en el gestor de contenidos Liferay. Utilizando sus tecnologías para el desarrollo de aplicaciones web, basadas en tecnología *portlet*, y tecnologías basadas en J2EE clásico, se desplegará en el portal la aplicación customizada, de manera que se combinarán los servicios propios de Liferay (como permisos, gestión de usuarios y autentificación, y taxonomía de la web) y desarrollos hechos a medida (lógica de negocio, visualización del *front-end* y consultas de los datos).

Entornos

Dentro del alcance del proyecto, se incluye la instalación y configuración de un servidor del producto **Liferay 6.2. Community Edition,** que será desplegado sobre un servidor de aplicaciones Java Tomcat 7.0, todo ello instalado en un servidor virtual sobre sistemas **CentOs 7.0.1 (plataforma Windows para el entorno de desarrollo)** y funcionando como base de datos **Oracle 11g Express Edition.**

Tecnologías de desarrollo

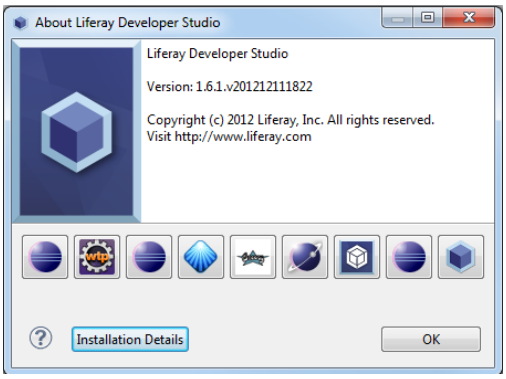
El proyecto estará basado, en todo momento, en el desarrollo con las diferentes variantes que nos da J2EE, y que implementa el producto Liferay, y que pone a disposición del programador:

- J2EE (programación directamente en clases java)
- Jasper Reports
- Spring
- JavaScript
- SQL
- Hibernate
- Entornos de visualización web (CSS, UI, JS, YUI, AJAX)
- Así como la explotación de la API propia de Liferay

Todas estas tecnologías se utilizarán mediante el IDE Eclipse.

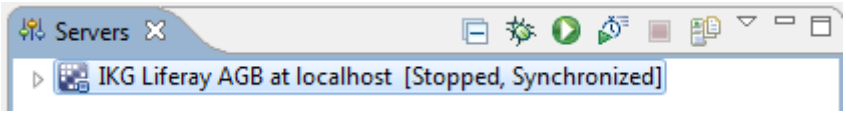
Elementos de desarrollo

Para desarrollar este proyecto hemos utilizado el IDE Eclipse, en su versión customizada denominada “Liferay Developer Studio”.

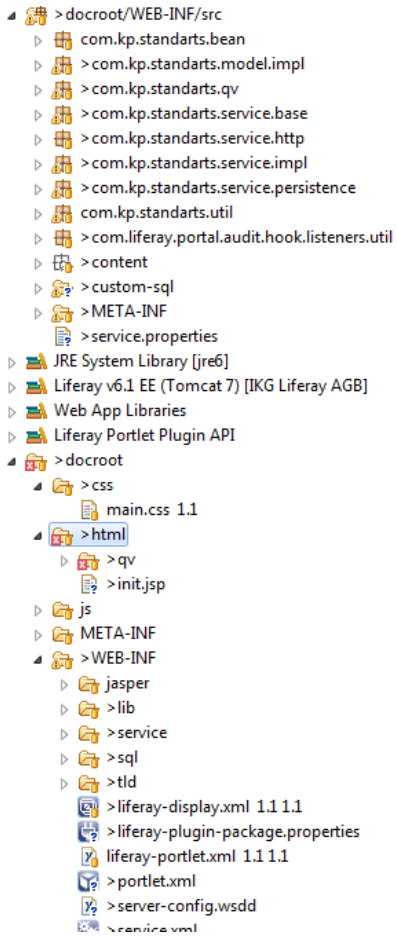


El funcionamiento de esta suitte nos permite tener todas las ventajas de Eclipse, pero teniendo algunas funciones de Liferay en añadido.

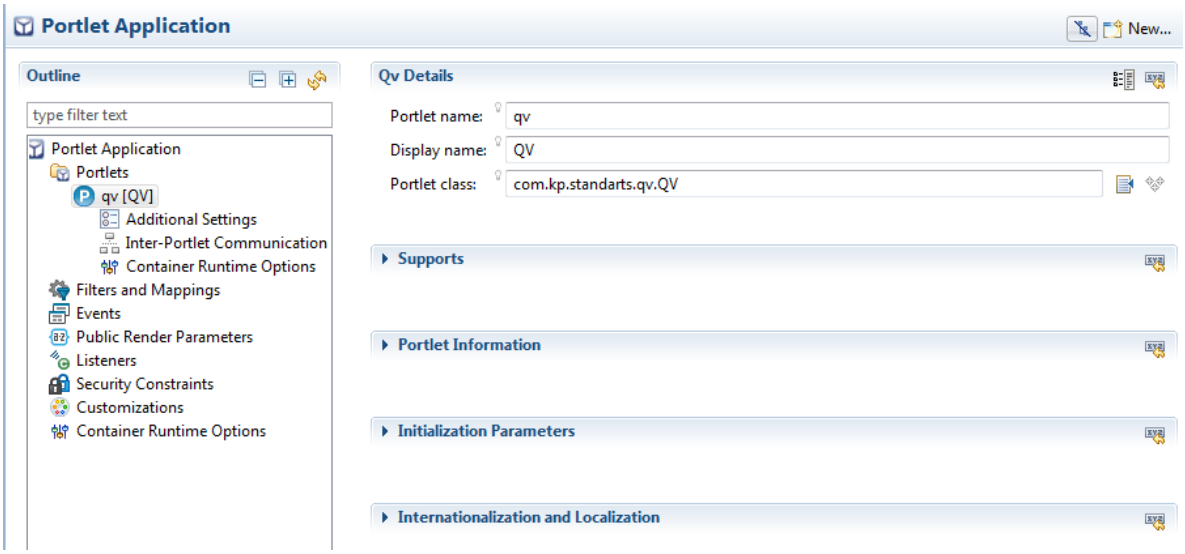
Por ejemplo, la instalación de Tomcat sobre el eclipse (que se realiza para tener las funciones de debugging durante el testing), la hacemos directamente dando por hecho que lo desplegado sobre este Tomcat es Liferay.



Además, con los plug-ins de Liferay el IDE nos permite la creación de los proyectos con la estructura que toca, por ejemplo nuestro Portlet:



Además, tenemos una manera visual e intuitivita de visualizar y rellenar los ficheros XML de configuración:



Elementos externos utilizados

Para implementar algunas de las partes se han desarrollado métodos utilizando APIs externas. Para el caso del elemento de audittrail, hemos utilizado una API que nos da unos métodos que guardarán los movimientos generados en una tabla del sistema. Para ello haremos una llamada del paquete “AuditRouter” en los métodos que impliquen alta, baja o modificación:

```
if (QVValidator.validateQvProduct(product, errors)) {
    product.setLastMod(new Date());
    product.setUserMod(themeDisplay.getUser().getScreenName());

    QvProduct productResult = QvProductLocalServiceUtil.addQvProduct(product);

    SessionMessages.add(request, "added-successfully");

    AttributesBuilder attributesBuilder = new AttributesBuilder(product, productResult);
       attributesBuilder.add("productId");
       AuditMessage auditMessage = AuditMessageBuilder.buildAuditMessage(
          "product.add", QvProduct.class.getName()/*"App. QV Standarts"*/, productResult.getProductId(), attributesBuilder.getAttributes());

       try {
             AuditRouterUtil.route(auditMessage);
          } catch (AuditException e) {

          }

        PortalUtil.copyRequestParameters(request, response);
```

Así cada vez que se haga esta llamada se hará una inserción con esos datos en la base de datos, que luego se podrá consultar:

5275	model.resource.com.kp.standarts.model.QvPot	action.pot.add	Ninguno	11/06/15 15:29
5275	model.resource.com.kp.standarts.model.QvLot	action.lot.add	Ninguno	11/06/15 15:28

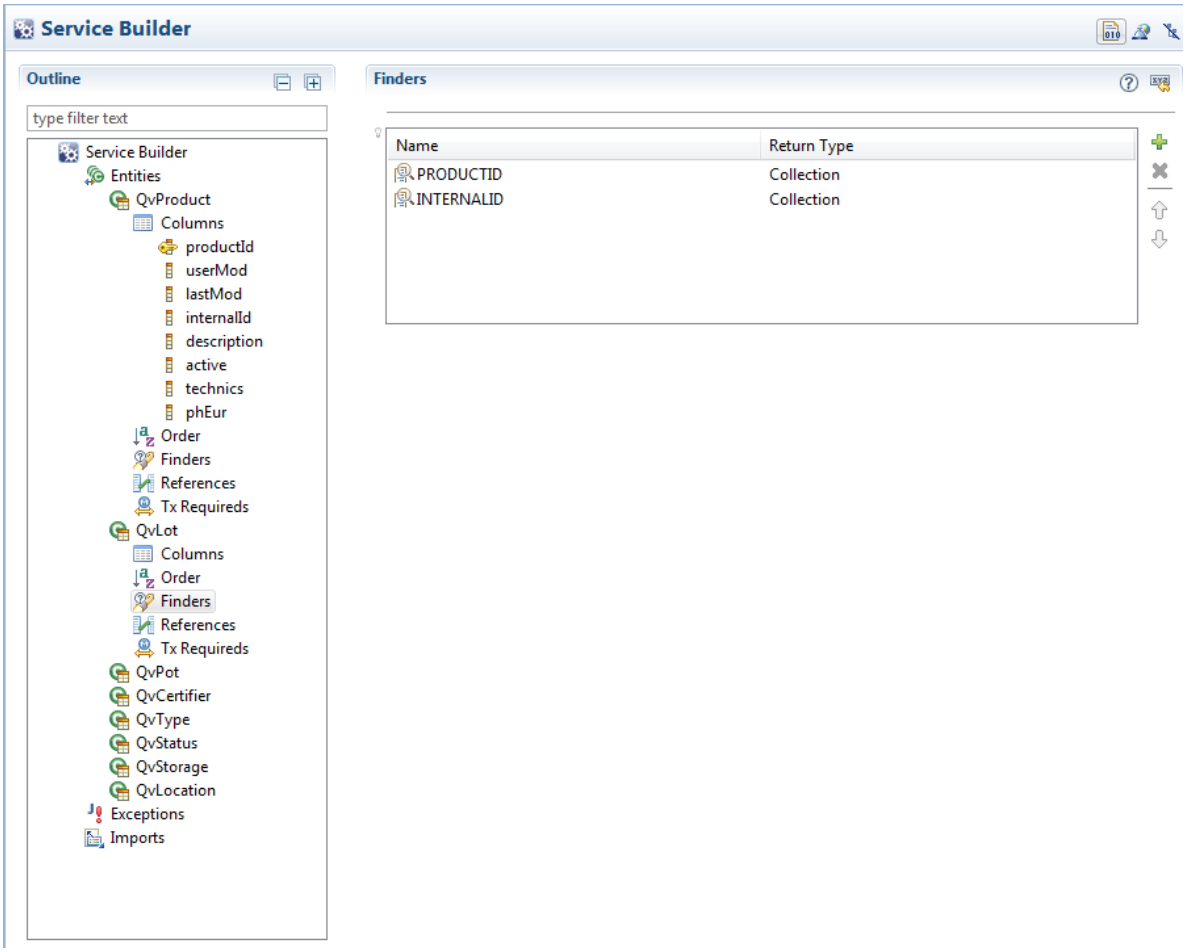
De manera parecida, se ha usado para la llamada a Jasper Reports. En la clase principal del portlet (QV.java). Hacemos llamadas (con un case para cada report) que nos permiten la exportación entre formatos, recogiendo los datos de la base de datos (a través de una sentencia SQL definida en el XML del report, y su output en pantalla. Por defecto lo exportaremos a PDF, aunque se pueden utilizar otros formatos dependiendo de la naturaleza del informe:

```
        //Creamos un hashMap de valores
        HashMap<String, Object> mapValues = new HashMap<String, Object>();
        String s = ParamUtil.getString(request,"dueDateDay");
        List<ElementoHash> parametros = locateParameters(request);
        //Recorremos la lista de parametros para crear un HashMap de parï¿½metros
        for(int i = 0; i < parametros.size(); i++){
            ElementoHash elem = parametros.get(i);
            mapValues.put(elem.getName(), elem.getValue());
        }

        //Indicamos el tipo de contenido que regresaremos ("application/pdf").
        response.setContentType("application/pdf");

        //Indicamos al usuario si desea guardar el archivo o abrirlo con alguna aplicaciï¿½n
        response.setProperty(HttpHeaders.CONTENT_DISPOSITION,"attachement;filename="+"file1.pdf");
        ////Para evitar que nuestro navegador guarde el archivo en cache, indicamos las siguientes cabeceras
//      response.setProperty("Cache-Control","no-cache");
//      response.setProperty("Pragma","no-cache");
//      response.setProperty("Expires", "0");

        //ServletOutputStream out = null;
        OutputStream out = null;
        Connection conn = null;

        try{
            //Recuepramos el fichero "*.jasper"
            JasperReport reporte = (JasperReport) JRLoader.loadObject(getPortletContext().getRealPath(locateReport(request)));
            //Construimos el pdf con los parï¿½metros enviados como parï¿½metros
            try {

                try {
                    Class.forName("oracle.jdbc.driver.OracleDriver");
//                  Class.forName("com.mysql.jdbc.Driver");

                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                }
```

Diseño de la arquitectura

La arquitectura de la aplicación se ha basado en el modelo MVC (Modelo vista-controlador). Para la implementación nos hemos ayudado de Service Builder. Elemento que nos proporciona una interfaz pseudo-grafica para definir los modelos, y sus relaciones, y luego componer toda la capa de servicios de una manera dinámica, a través de las *interfaces* necesarias, dando la consistencia necesaria al código.

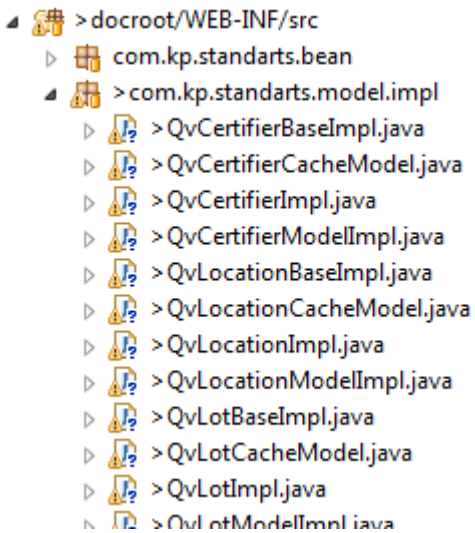
Service Builder funciona de la siguiente manera, al definir todas las entidades en el fichero XML del servicio, y ejecutando un script *ANT*, este propaga los cambios hacia todas las clases definidas anteriormente, o, si no existen porque hemos definido la entidad de nuevo, las crea.



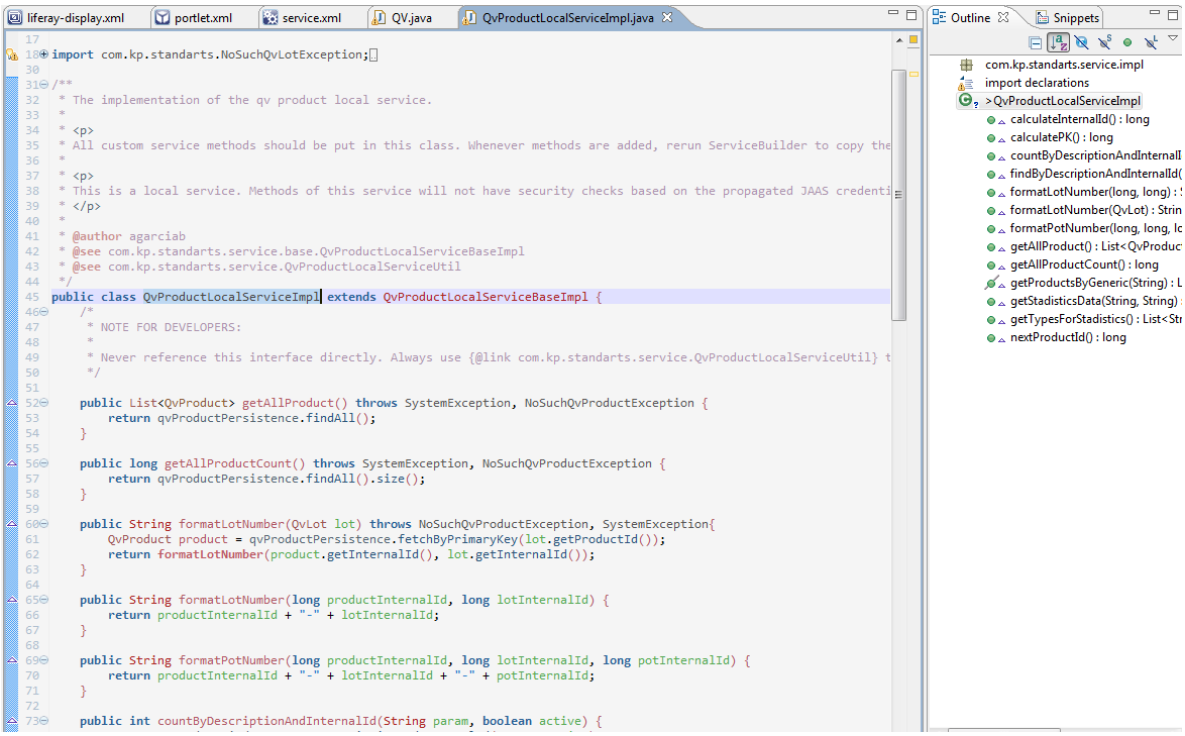
De esta manera definimos toda la estructura de entidades, y sus propiedades, y ejecutando el script ANT:

```
[Console output redirected to file:C:\Liferay\workspace\.metadata\.plugins\com.liferay.ide.eclipse.sdk\sdk.log]
Buildfile: C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\build.xml
build-service:
     [jar] Building MANIFEST-only jar: C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\build-service-classpath.jar
  [delete] Deleting: C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\build-service-classpath.jar.manifest
  [delete] Deleting: C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\build-service-classpath.jar
    [echo] Loading jar:file:/C:/PROYECTOS_LIFERAY/bundles/tomcat-7.0.40/webapps/ROOT/WEB-INF/lib/portal-impl.jar!/system.properties
    [echo] Loading jar:file:/C:/PROYECTOS_LIFERAY/bundles/tomcat-7.0.40/webapps/ROOT/WEB-INF/lib/portal-impl.jar!/portal.properties
    [echo] Loading jar:file:/C:/PROYECTOS_LIFERAY/bundles/tomcat-7.0.40/webapps/ROOT/WEB-INF/lib/portal-impl.jar!/com/liferay/portal/tools/dependencies/portal-tools.pr
    [echo] 18:09:38,202 WARN  [main][PDFProcessorImpl:235] Liferay is not configured to use ImageMagick. For better quality document and image previews, install ImageM
    [echo] Building QvCertifier
    [echo] Building QvLocation
    [echo] Building QvLot
    [echo] Building QvPot
    [echo] Building QvProduct
    [echo] Building QvStatus
    [echo] Building QvStorage
    [echo] Building QvType
    [mkdir] Created dir: C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\docroot\WEB-INF\service-classes
compile-java:
     [copy] Copied 7 empty directories to 7 empty directories under C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\docroot\WEB-INF\service-classes
    [javac] Compiling 177 source files to C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\docroot\WEB-INF\service-classes
      [zip] Building zip: C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\docroot\WEB-INF\lib\kp-standarts-portlet-service.jar
  [delete] Deleting directory C:\Liferay\liferay-plugins-sdk-6.1.20\portlets\kp-standarts-portlet\docroot\WEB-INF\service-classes
BUILD SUCCESSFUL
Total time: 15 seconds
```

En este punto el script nos genera todos los ficheros necesarios para ejecutar la aplicación, con nuestras propiedades definidas, y las propaga por todos las dependencias necesarias:



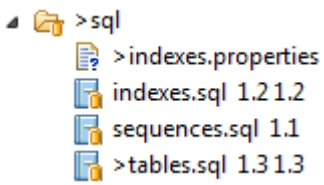
De esta manera Service Builder nos crea la estructura necesaria para comenzar a trabajar. Dentro del encapsulamiento de código que nos proporciona, todas las llamadas a los servicios se harán a través de interificies, por lo que nosotros desarrollaremos la lógica dentro de un fichero “Impl” donde quedará nuestro código personalizado:



Y para propagar estos cambios al resto de clases, será tan fácil como volver a ejecutar nuestro script *ANT* para que los cambios se vean reflejados en las clases necesarias.

De esta manera nos es muy fácil cambiar una entidad, o la funcionalidad de un servicio, sin tener que revisar a mano un puñado de ficheros de definiciones.

No solo esto. Service Builder, cuando nos crea estos servicios, nos crea una serie de scripts SQL con la estructura de entidades definidas, que se ejecutarán al desplegarse la aplicación en el servidor. Si estos objetos no existen en la base de datos, se crearán automáticamente. Esto da la potencia necesaria nuestra aplicación para ser desplegada en cualquier servidor adaptado sin necesidad de ninguna adaptación (porque además, los scripts SQL no están orientados a un SGBD en concreto, si no que se ejecutaran en cualquiera).



Conclusiones

Este proyecto me ha servido para realizar el desarrollo de un ciclo completo de un proyecto de software. Desde la sintetización de la idea original, la toma de requerimientos, la planificación y asignación del recurso, el análisis y diseño, desarrollo, test y despliegue. Sobre todo, a algo a lo que estoy poco acostumbrado, a generar una documentación de apoyo.

Sin duda la parte del análisis, y diseño previo me ha hecho ver la importancia de estos pasos. El desarrollo ha sido relativamente sencillo apoyándome en este material, ya que me podía basar en esta idea global, y no "improvisar cada parte según viene" que es como había planteado el mundo de la programación hasta ahora. He sido consciente del valor añadido que esto da a la hora del desarrollo posterior, sobre todo si nos pusiéramos en el caso en el que este proyecto no hubiese sido desarrollado por una sola persona, si no, por un equipo, sin este trabajo previo, posteriormente "las piezas no encajarían". Sin duda, es algo que he adquirido y que me será tremendamente útil en futuros proyectos de este tipo en los que pueda participar en el futuro. En este sentido querría destacar lo importante que ha sido, y que no le daba la importancia necesaria el hecho de trabajar con esquemas y gráficos, ya que es lo más útil a la hora de consultar rápidamente que debía hacer, y que no, la aplicación.

Muy enriquecedora ha sido la parte de investigar probar y elegir diferentes tecnologías en pos de resolver una necesidad que tiene el proyecto. Siempre se ha dicho que no hay que "reeinventar la rueda", y en el mundo del desarrollo de software podemos estar casi seguros que si queremos hacer algo, alguien habrá intentado hacerlo antes, por lo que hay que preguntar e informarse en que tecnologías hay existentes que nos resuelvan un punto en concreto. Muchas veces encontraremos muchas opciones, el escoger la más eficaz será una gran responsabilidad, ya que hay que tener en cuenta muchas factores (escalabilidad, factor económico, y, como no, las tendencias de mercado) para que esta herramienta escogida nos dé un valor añadido, y no un problema al proyecto. Una elección errónea, o un producto desactualizado, inseguro, o que no resuelva la necesidad puede hacer fracasar un proyecto, aunque el resto de la aplicación este bien desarrollada.

Glosario

- **Gestor de portales:** Es un software que despliega un servidor de aplicaciones, y que es capaz de gestionar aplicaciones de diferentes tipos y diferentes propósitos, administrados y visualizados desde un mismo entorno (normalmente web) de manera que pueda parecer una única aplicación para el usuario final.
- **Scrum:** metodología de desarrollo de software, denominada "ágil" ya que propone dividir la aplicación en pequeñas tareas que se deberán ir resolviendo rápidamente, y si no es posible, pasarlas a un compañero para que las resuelva, y solucionar otra, con el objetivo de tener pequeñas entregas con funcionalidades completas (denominadas "sprints") que serán probadas y aceptadas, y repitiéndose estas fases hasta que el producto sea completo.
- **Service Builder:** Se trata de una herramienta de desarrollo que permite la creación de la capa de persistencia de manera dinámica, y evitando las tareas repetitivas en la hora de dar de alta entidades.
- **Portlet:** Elemento Java que es desplegado en un servidor de aplicaciones, y que se puede ubicar dentro de una web, como un trozo de la misma, y dar una lógica de negocio completa, independizándose del resto de la web.
- **Lucent:** Sistema de indexación de contenidos, que permite la consulta rápida de elementos.
- **Spring MVC:** Framework de desarrollo orientado a la tecnología Portlet, que permite la creación de modelos de datos y su gestión, así como su capa de presentación.
- **Hibernate:** Framework de desarrollo que permite interectactuar con el sistema de gestión y acceso a Base de Datos, desde una interfaz Java.
- **POI:** Framework que permite, desde una interfaz Java, la creación e interactuación de elementos ofimáticos de tipo MS Office, o PDF.
- **MAEWIN:** Sistema automatizado de impresión de etiquetas.
- **Liferay Audit:** Librería que permite la trazabilidad de elementos desde una llamada API realizada desde código Java, almacenada tablas de base de datos.
- **Jasper Reports:** Herramienta de desarrollo que permite generar informes, en varios formatos, dado un fichero de definición XML.

Bibligrafia

- Piensa en Java. Bruce Eckel. 4a Edición. Pearson Prentice Hall.
- SCJP. Study Guide. Kathy Sierra, Bert Bates. McGrawHill.
- Liferay Development: https://dev.liferay.com/
- Service Builder: https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-1/what-is-service-builder
- Auditing: http://www.liferay.com/es/community/wiki/-/wiki/Main/Adding+Auditing+Functionality+to+Portlets
- POI: https://poi.apache.org/
- Jasper Reports: http://community.jaspersoft.com/project/jasperreports-library