Universidad Carlos III

Proyecto de Fin de Carrera

Proyecto de Trazabilidad Alimentaria

Autor:
Fabriciano García

Supervisor:
Miguel Ángel Patricio

2 de julio de 2013

"A mi familia y a toda la gente que ha hecho posible este proyecto gracias a sus aportaciones"

Índice

Índice de figuras

Índice de código

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación y antecedentes

A la gran variedad de empresas que hoy en día se dedican a la alimentación, la Agencia Española de Seguridad Alimentaria y Nutrición les exige llevar la trazabilidad de sus productos, esto es, encontrar y seguir el rastro a través de todas las etapas de producción, transformación y distribución, de un alimento, un pienso o un ingrediente.

Con esta herramienta de trazabilidad se contribuye a facilitar la retirada de los alimentos en los que se haya detectado algún problema y permite que los consumidores reciban información específica y exacta sobre los productos en cuestión.

1.1.1. Marco legal

El 28 de enero de 2002 el Parlamento y el Consejo Europeo crean la Autoridad Europea de Seguridad Alimentaria y fijan procedimientos relativos a la seguridad alimentaria, así como, establecen los principios y requisitos generales de la legislación alimentaria. En el artículo 18 se establece por primera vez, con carácter horizontal, para todas las empresas alimentarias y de piensos que forman parte de la cadena alimentaria la obligación de poner en marcha, aplicar y mantener un sistema de trazabilidad. Vemos a continuación el citado artículo: Artículo 18. Trazabilidad

- En todas las etapas de producción, la transformación y la distribución deberá asegurarse la trazabilidad de los alimentos, los piensos, los animales destinados a la producción de alimentos y de cualquier otra sustancia destinada a ser incorporada en un alimento o un pienso, o con probabilidad de serlo.

- Los operadores económicos de empresas alimentarias y de empresas de piensos deberán poder identificar a cualquier persona que les haya suministrado un alimento, un pienso, un animal destinado a la producción de alimentos, o cualquier sustancia destinada a ser incorporada en un alimento o un pienso, o con probabilidad de serlo. Para tal fin, dichos operadores económicos pondrán en práctica sistemas y procedimientos que permitan poner esta información a disposición de las autoridades competentes si éstas así lo solicitan.

- Los operadores económicos de empresas alimentarias y de empresas de piensos deberán poner en práctica sistemas y procedimientos para identificar a las empresas a las que hayan suministrado sus productos. Pondrán esta información a disposición de las autoridades comptetentes si éstas así lo solicitan.

- Los alimentos o los piensos comercializados o con la probabilidad de comercializarse en la Comunidad deberán estar adecuadamente etiquetados o identificados para facilitar su trazabilidad mediante documentación o información pertinentes, de acuerdo con los requisitos pertinentes de disposiciones más específicas.

- Podrán adoptarse disposiciones para la aplicación de lo dispuesto en el presente artículo en relación con sectores específicos de acuerdo con el procedimiento contemplado en el apartado 2 del artículo 58.

1.1.2. Herramientas de mercado

Existen productos comerciales específicos para cubrir esta necesidad de llevar la trazabilidad alimentaria. El que se presenta en este documento ha sido realizado específicamente para el Grupo Presto a Mangiare, conteniendo todos los requisitos demandados por éste.

La necesidad de un nuevo software y el no uso de uno de los ya existentes en el mercado estriba en los diferentes problemas que el cliente encuentra en cada uno de ellos. Siendo el principal problema, el complejo funcionamiento y la curva de aprendizaje a causa de sus extensas funcionalidades.

A continuación se enumeran algunos de estos productos existentes para finalmente obtener el conjunto de ventajas y desventajas que aportan. Sobre este pequeño estudio inicial se apoyará finalmente el objetivo de este proyecto fin de carrera, que pretenderá cubrir las desventajas de estos productos, basándose en las exigencias del cliente.

Trazagest	
Ventajas	software como servicio no requiere instalación múltiples bases de datos seguridad en los datos mantenimiendo y actualizacion permanente
Desventajas	no incluye generación de código de barras precio 600 mantenimiendo 45 mensuales complejidad

Cuadro 1.1: Trazagest

ilean	
Ventajas	integrabilidad con otras aplicaciones migración del software anterior
Desventajas	muchas funcionalidades no necesarias complejidad

Cuadro 1.2: ilean

ioalimentacion	
Ventajas	integrable apps. microsoft office
Desventajas	integrado en ERP Microsoft Dynamics NAV muchas funcionalidades no necesarias complejidad

Cuadro 1.3: ioalimentacion

1.2. Objetivos

El objetivo final de este proyecto es el de desarrollar un programa para la gestión de la trazabilidad que mantenga las ventajas de los anteriores productos y los mejore. Tras la finalización del proyecto se dispondrá de un software que permitirá la gestión de la trazabilidad alimentaria, que incluye:

- Ingreso de nuevos tipos de producto
- Ingreso de productos fabricados

- Generación de número de lote
- Generación de código de barras
- Generación de etiquetas
- Modificar productos
- Buscar productos
- Generación de informes
- Administración de perfiles y usuarios...

estas y el resto de funcionalidades se detallarán mas adelante.

1.2.1. Visión general

Para la gestión de la trazabilidad, el programa, que es una aplicación web, usará una sola página, comunicándose con el servidor por medio de AJAX, permitiendo así una interfaz siempre igual y un servicio rápido para el usuario.

Al iniciar la aplicación el usuario deberá ingresar los datos con los que quiere acceder, estos son nombre de usuario y contraseña. Existen dos tipos de usuario, el normal y el de administrador que contiene todas las funciones del usuario normal mas alguna con peligro de seguridad para los datos recogidos.

Una vez en la página principal, el usuario, se encontrará con un menú horizontal en la parte superior del navegador, desde el cual con un solo click, tendrá acceso directo a todas las funcionalidades del programa.

A continuación se muestra un esquema del funcionamiento general de las principales funcionalidades a desarrollar.

1.2.2. Definición de un nuevo tipo de producto

Antes que nada el administrador deberá definir que tipos de productos se fabrican en la empresa, que serán los mismos que todos los usuarios puedan ingresar como productos fabricados, sin excepción. De este nuevo tipo de producto deberá indicar:

1. Nombre.
2. Nombre de cada uno de los ingredientes que lo componen.

3. Tipo de formatos en los que se puede fabricar.
4. Caducidad indicada en días, meses o años.
5. Conservación.
6. Modo de uso.

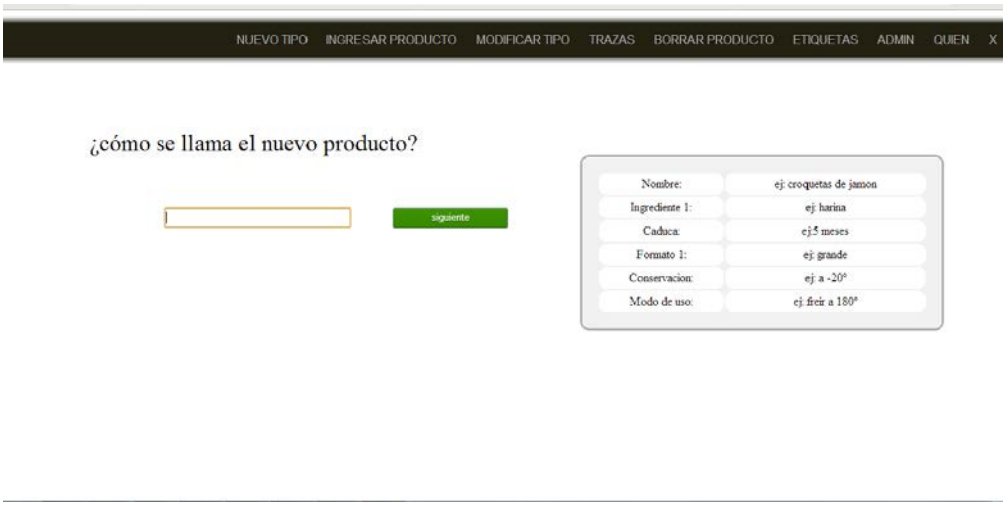


FIGURA 1.1: Pantalla ingresar nuevo tipo de producto

1.2.3. Ingresar producto fabricado

A continuación tanto el usuario como el administrador tienen la capacidad de ingresar los datos de la fabricación de un producto. Siempre que éste haya sido ya ingresado en ingresar un nuevo tipo. Los datos que tiene que ingresar para registrar el producto fabricado son:

1. Nombre. Esta función es un autocompletar realizado con JQuery, en el que al pulsar dos de las letras del producto te muestra todas las opciones posibles.
2. Lote de los ingredientes. Aquí deberá introducir al menos un lote de cada ingrediente con los que se fabrica el producto. Puede haber varios lotes de cada ingrediente.
3. Formato fabricado.
4. Cajas fabricadas.

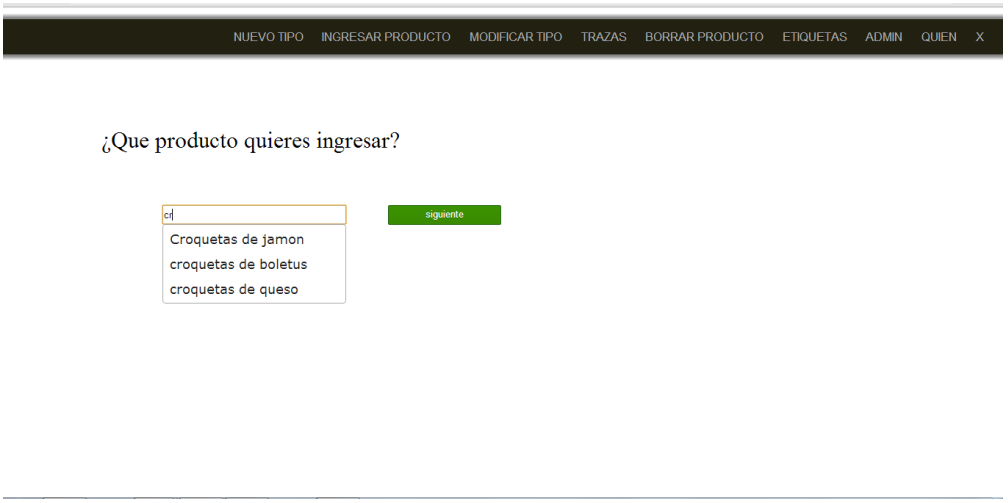


FIGURA 1.2: Pantalla ingresar producto fabricado

1.2.4. Modificar tipo de producto

En esta sección el administrador podrá cambiar todo lo que introdujo en el apartado de ingresar tipo de producto, seleccionando el tipo de producto a modificar.

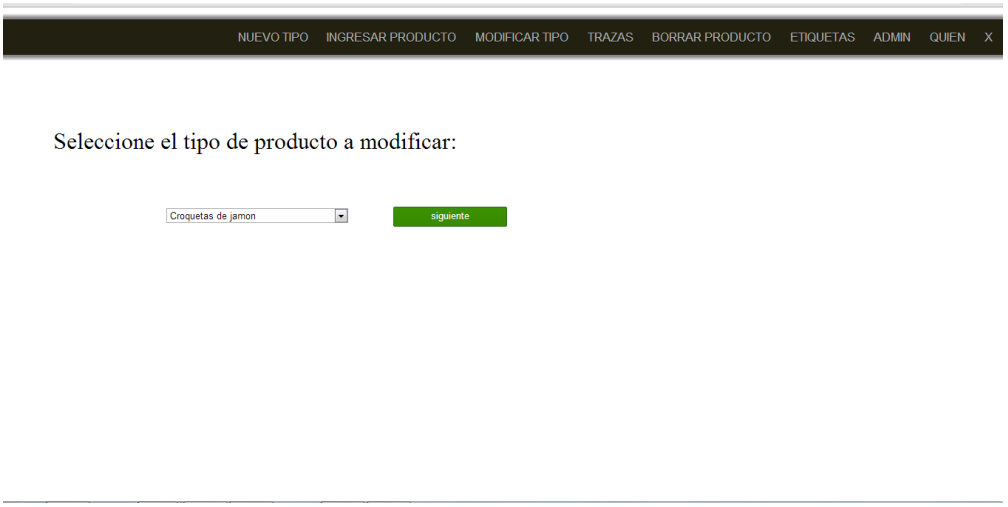


FIGURA 1.3: Pantalla modificar tipo de producto

1.2.5. Buscar trazas

Aquí se podrá buscar la trazabilidad de un producto fabricado. Esta búsqueda se puede hacer por diferentes parámetros:

1. Todos.
2. Lote de producto.

3. Fecha de fabricación.
4. Nombre del producto.
5. Fecha de caducidad.
6. Intervalo de fechas.
7. Resumen ean.

Dependiendo del parámetro elegido el administrador deberá introducir datos o no. Y la salida será diferente, mostrando a veces todos los datos de un producto, como a veces algún dato específico solicitado. También mostrará, en algunos casos, un botón para descargar los datos mostrados en una hoja de cálculo.

FIGURA 1.4: Pantalla buscar trazas

1.2.6. Borrar producto hecho

Si se ha introducido en el sistema un producto que no debería haber sido introducido, porque esté mal introducido alguno de sus datos. O simplemente como una prueba. El administrador podrá borrarlo en esta sección.

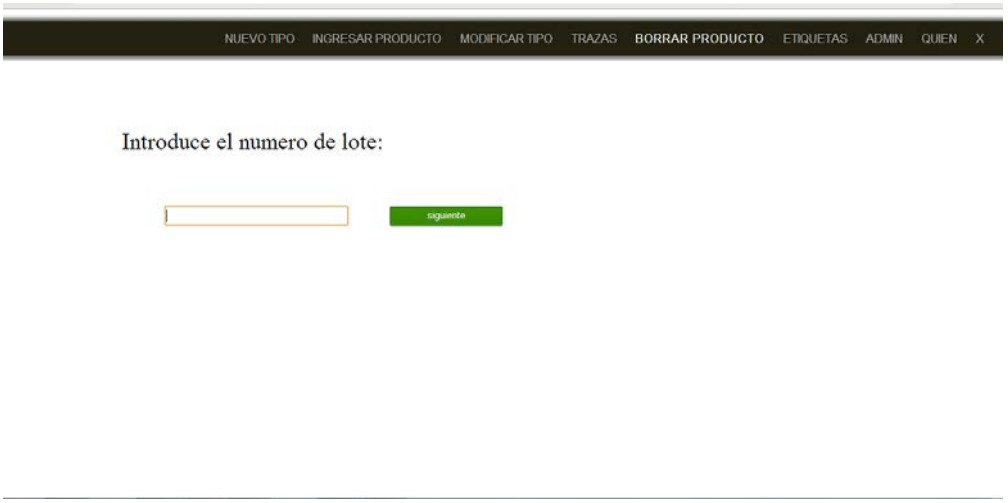


FIGURA 1.5: Pantalla borrar producto hecho

1.2.7. Imprimir etiquetas

A la hora de imprimir etiquetas ha podido haber un error, ya sea con la impresora, que al usuario se le olvide imprimirlas, o quien sabe... En esta pantalla se podrán imprimir las etiquetas de los productos fabricados introduciendo su número de lote.

FIGURA 1.6: Pantalla imprimir etiquetas

1.2.8. Admin

Esta pantalla es, evidentemente, solamente para el administrador. En ella podrá:

- Crear un usuario, asignándole un nombre, una contraseña y un perfil, usuario o administrador.

- Borrar un usuario.
- Datos de la empresa. Aquí deberá introducir los datos de la empresa que luego serán mostrados en las etiquetas: nombre, dirección, email, teléfono, fax, código ean y número de registro sanitario.

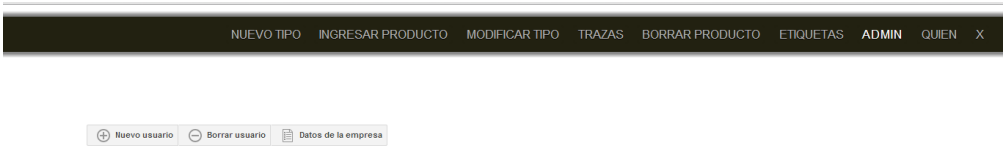


FIGURA 1.7: Pantalla administrador.

1.2.9. Quién

El administrador podrá consultar aquí el usuario que fabricó cierto producto, introduciendo su número de lote. El usuario al hacer log-in introdujo su nombre y contraseña. El sistema guarda la hora de conexión y así se puede comparar la fecha y hora en la que se fabricó un producto y compararlo con la tabla de conexión de usuarios.

1.3. Estructura de la memoria

Este documento está dividido en 8 partes:

- Una primera parte de introducción al proyecto desarrollado. En este punto entraremos a valorar la justificación del proyecto y se establecen los objetivos del mismo.
- En la segunda parte donde se detalla cada uno de los requerimientos que finalmente deberá cumplir la implementación final del proyecto.
- En la tercera parte donde se explica la tecnología utilizada, además del modelo de datos y las distintas funciones y entidades utilizadas.

- En el cuarto capítulo se muestra un manual de instalación.
- En el quinto capítulo se muestra un manual de usuario de la aplicación.
- En la sexta parte haremos un presupuesto del proyecto.
- En el séptimo capítulo podemos ver la planificación.
- En el último capítulo se expondrá una conclusión y futuros proyectos.

Además de los distintos capítulos, encontraremos un índice, una lista de tablas, una lista de figuras y una lista de códigos de programación.

1.4. Definiciones, acrónimos y abreviaturas

- AJAX: Asynchronous JavaScript And XML.
- BBDD: Bases de datos.
- CLASSPATH: se trata de una variable empleada por la máquina virtual java que ejecuta cualquier aplicación con tecnología java. Esta variable almacena una relación de diferentes ubicaciones y ficheros de la máquina donde se encuentra instalado. El objetivo es que la máquina virtual conozca el conjunto de librerías y ficheros sobre los que se debe apoyar para su ejecución.
- Driver JDBC: Software Java que permite desde aplicaciones java acceder a diferentes tipos de BBDD de forma uniforme y transparente. Esta tecnología permite aislar la implementación de las aplicaciones java de la BBDD que finalmente se empleará.
- HTML: HyperText Markup Language.
- CSS: lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas.
- J2EE: conjunto de especificaciones de API's Java para la construcción de aplicaciones empresariales en varias capas.
- JDK (Java Development Kit): Kit de desarrollo java.
- SQL: se trata del lenguaje estándar empleando para elaborar las sentencias (Query) de acceso a las BBDD.
- MYSQL: sistema de gestión de bases de datos relacional, multihilo y multiusuario.

- XML (eXtensible Markup Language): se trata de un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes.
- XSL: lenguaje empleado para transformar documentos XML en otros formatos empleando reglas.
- Javascript: lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
- JQuery: librería JavaScript.
- JQueryUI: librería JavaScript para la interfaz de usuario.
- Apache FOP: Apache FOP (Formatting Objects Processor) is a print formatter driven by XSL formatting objects (XSL-FO) and an output independent formatter.
- Barcode4j: extensión de Apache Fop para la generación de etiquetas.

Capítulo 2

Requerimientos del proyecto

2.1. Requisitos funcionales

A continuación se enumeran todos los requisitos funcionales que el programa de trazabilidad debe cumplir.

- **Código:** RF001
- **Título:** Ingresar nuevo tipo de producto.
- **Prioridad:** E-Esencial
- **Descripción:** El administrador deberá ser capaz de introducir un nuevo tipo de producto.

- **Código:** RF002
- **Título:** Nombre del nuevo tipo de producto.
- **Prioridad:** E-Esencial
- **Descripción:** No puede haber dos productos con el mismo nombre. El nombre no puede superar los 60 caracteres.

- **Código:** RF003
- **Título:** Ingredientes del nuevo tipo de producto.
- **Prioridad:** E-Esencial

- **Descripción:** Un producto puede tener tantos ingredientes como sean necesarios. Dos ingredientes no se pueden llamar igual.

- **Código:** RF004
- **Título:** Vida útil del nuevo tipo de producto.
- **Prioridad:** E-Esencial
- **Descripción:** La vida útil de un producto debe ser, al menos, un día. Ésta debe ser indicada con una cifra y luego una unidad de tiempo días, meses o años.

- **Código:** RF005
- **Título:** Formato del nuevo tipo de producto.
- **Prioridad:** E-Esencial
- **Descripción:** Un producto puede tener tantos formatos de fabricación como sean necesarios, por lo menos tendrá uno.

- **Código:** RF006
- **Título:** Conservación y modo de uso del nuevo tipo de producto.
- **Prioridad:** E-Esencial
- **Descripción:** Se debe poder indicar la conservación, así como, el modo de uso del nuevo tipo de producto. Como mucho podrán ocupar 150 caracteres.

- **Código:** RF007
- **Título:** Introducir un producto fabricado.
- **Prioridad:** E-Esencial
- **Descripción:** Tanto el usuario como el administrador deberán ser capaces de introducir un producto fabricado indicando el nombre del mismo. Produciendo el sistema, al final, una etiqueta para ese producto.

- **Código:** RF008
- **Título:** Ingredientes producto fabricado.
- **Prioridad:** E-Esencial

- **Descripción:** Se permite introducir tantos números de lotes de cada ingrediente del producto como requiera la situación.

- **Código:** RF009
- **Título:** Formato, cajas y número de etiquetas del producto fabricado.
- **Prioridad:** E-Esencial
- **Descripción:** Una vez ingresados todos los datos del producto fabricado, el sistema deberá generar tantas etiquetas, para ese producto, como número de cajas haya introducido el usuario. Salvo que, el usuario haya introducido el formato unidades, generando tan solo una etiqueta con el número de unidades en ella.

- **Código:** RF010
- **Título:** Etiquetado de los productos.
- **Prioridad:** E-Esencial
- **Descripción:** El sistema generará una etiqueta con los siguientes datos del producto fabricado: nombre de la empresa, dirección, teléfonos, email, número de registro sanitario, nombre del producto, formato, ingredientes, conservación, modo de uso, número de lote, consumo preferente y código de barras.

- **Código:** RF011
- **Título:** Modificar los tipos de producto.
- **Prioridad:** E-Esencial
- **Descripción:** El administrador podrá cambiar cualquier dato de los tipo de productos, estos son: nombre, ingredientes, formatos, caducidad, modo de uso y conservación.

- **Código:** RF012
- **Título:** Buscar la trazabilidad de los productos.
- **Prioridad:** E-Esencial
- **Descripción:** Deberá haber una herramienta para buscar los tipos de producto disponibles para la fabricación, así como los productos previamente fabricados. Ésta búsqueda deberá poderse hacer según diversos parámetros: todos, nombre, lote de producto, fecha de fabricación, fecha de fabricación, intervalo de fechas y resumen con código ean.

- **Código:** RF013
- **Título:** Generación de informes.
- **Prioridad:** E-Esencial
- **Descripción:** Para algunos de los datos salientes en el apartado de buscar trazabilidad, el sistema generará un fichero EXCEL, con todos estos datos a modo de resumen o detallado según indicación del usuario.

- **Código:** RF014
- **Título:** Administrar usuarios.
- **Prioridad:** E-Esencial
- **Descripción:** El administrador deberá ser capaz de crear y borrar los usuarios pertinentes.

- **Código:** RF015
- **Título:** Modificar los datos de la empresa.
- **Prioridad:** E-Esencial
- **Descripción:** El administrador podrá cambiar los datos de la empresa.

- **Código:** RF016
- **Título:** Conocer el usuario que introdujo un ingrediente.
- **Prioridad:** E-Esencial
- **Descripción:** Introduciendo el número de lote, el administrador conocerá el usuario que introdujo ese ingrediente en el sistema.

- **Código:** RF017
- **Título:** Cursor de escribir en el campo al aparecer en la página.
- **Prioridad:** E-Deseable
- **Descripción:** Al cargar un campo de texto para introducir un valor en la página, el cursor deberá estar en el mismo para facilitar y agilizar la usabilidad.

- **Código:** RF018

- **Título:** Autocompletar en el nombre del producto a ingresar.
- **Prioridad:** E-Deseable
- **Descripción:** Para agilizar la introducción de productos fabricados el sistema será capaz de utilizar la función de autocompletar, permitiendo así al usuario introducir un producto con tan solo escribir dos letras del mismo.

2.2. Requisitos no funcionales

2.2.1. Requisitos de Operación

- **Código:** RO001
- **Título:** Facilidad de uso.
- **Prioridad:** E-Esencial
- **Descripción:** El sistema debe ser fácil de usar tanto para los usuarios como para los administradores, ofreciendo una interfaz que permita acceder a la información y ejecutar las operaciones más habituales sin necesidad de recorrer múltiples ventanas y menús.

2.2.2. Requisitos de Prestaciones

- **Código:** RP001
- **Título:** Usabilidad a través de navegador web.
- **Prioridad:** E-Esencial
- **Descripción:** El sistema debe poder usarse a través de un navegador web. Descartando así aplicaciones de escritorio u otras que no correspondan con esta descripción.

- **Código:** RP002
- **Título:** Uso de herramientas open source.
- **Prioridad:** E-Esencial
- **Descripción:** El sistema deberá realizarse enteramente usando material de código abierto. No podrá pagarse ninguna licencia para le realización del mismo.

2.2.3. Requisitos de Seguridad

- **Código:** RS001
- **Título:** Seguridad en acceso a la aplicación.
- **Prioridad:** E-Esencial
- **Descripción:** El sistema deberá preguntar el nombre y contraseña del usuario antes de que éste pueda realizar ninguna acción en el mismo.

- **Código:** RS002
- **Título:** Dos tipos de usuarios.
- **Prioridad:** E-Esencial
- **Descripción:** El sistema deberá mostrar distinto contenido dependiendo de quien se haya introducido un usuario o un administrador.

2.2.4. Otros requisitos

- **Código:** RO001
- **Título:** Cumplimiento Metodología J2EE
- **Prioridad:** E-Esencial
- **Descripción:** Para todo el diseño y desarrollo del sistema se deberá seguir la metodología J2EE en 3 capas que actualmente se exige por la gran mayoría de clientes en todos los desarrollos.

  Además, este tipo de metodología, al separar la lógica de presentación de la lógica de negocio, facilitará el desarrollo de un sistema capaz de ser accesible desde otros dispositivos diferentes al PC habitual (PDA's, Pocket's PC's, etc.).

  Por otro lado, el cumplimiento de esta metodología facilitará posteriormente el mantenimiento del sistema y su evolución.

- **Código:** RO002
- **Título:** Escalabilidad y fácil ampliación de componentes y atributos
- **Prioridad:** E-Esencial

- **Descripción:** Se deberá diseñar y construir el sistema de tal manera que la adaptación y mejora de sus funcionalidades así como la inclusión de nuevas funciones sea lo más rápida posible sin que el resto de funcionalidades ni elementos software comunes se vean afectados.

Capítulo 3

Diseño

3.1. Tecnología

A continuación se muestra un esquema general de la arquitectura del programa de trazabilidad.

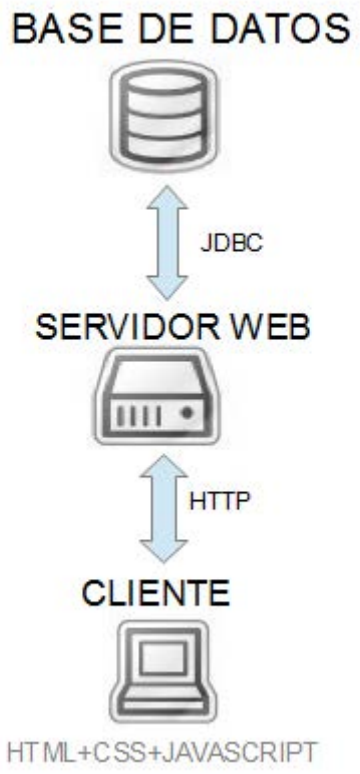


Figura 3.1: Arquitectura del sistema

Los usuarios acceden a la aplicación mediante un navegador web. El navegador envía una petición al servidor web y este es el encargado de ejecutar el programa de trazabilidad. El servidor, en caso de que lo necesite, accederá a la base de datos MYSQL. El servidor web y la base de datos no es necesario que estén en la misma máquina, dependiendo de la cantidad de datos a manejar será recomendable una cosa u otra.

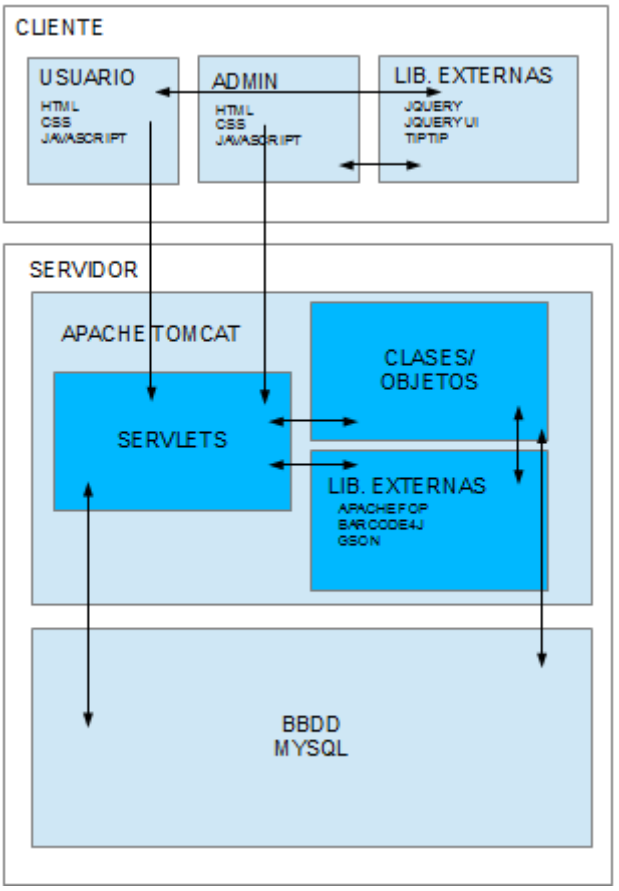


FIGURA 3.2: Arquitectura y tecnologia del programa

La tecnología empleada en el desarrollo y comportamiento del Programa de Trazabilidad se recoge igualmente en la figura anterior y se detalla a continuación cada una de sus partes:

1. El cliente se comunica con el servidor web mediante un navegador web, mandando una petición por medio de AJAX a los diferentes servlets.

2. En el lado del cliente podemos encontrar también librerías externas, utilizadas para agilizar el desarrollo de la aplicación. Algunas de las librerías utilizadas son JQUERY, JQUERY UI y TIPTIP.

3. Los servlets están alojados en un servidor Tomcat, que es un servidor web con soporte para servlets y JSPs, en este caso también trabaja con un servidor Apache.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

4. Los servlets consultando en la base de datos y usando las clases le envían una respuesta en formato HTML,CSS Y JAVASCRIPT.

5. En el servidor también se encuentran las clases desarrolladas para el perfecto funcionamiento del programa. Cómo es un programa orientado a objetos podemos contar con muchos y diferentes tipos de objetos, como por ejemplo producto, empresa, usuario, informe...

6. En ocasiones los servlets necesitan el uso de librerías externas para elaborar la respuestas. Algunas de las librerías utilizadas en este proyecto son:

   - Apache FOP: esta librería es utilizada para convertir archivos XML(con los datos) Y XSL(documento XML en el que se especifica cómo se van a formatear unos datos para presentarlos en pantalla, papel u otros medios) en formato pdf, para la generación de etiquetas.
   - Barcode4j: esta librería se utilizada para la generación de código de barras. Pasándole un archivo XML con los diferentes datos y argumentos generará un código de barras. En esta ocasión barcode4j trabaja con Apache FOP pasándole el código de barras para su integración en el pdf de la etiqueta.
   - Gson: es una biblioteca de código abierto, desarrollada por Google, para el lenguaje de programación Java que permite la serialización y deserialización entre objetos Java y su representación en notación JSON. Esta librería es la encargada de pasarle los datos como los quiere recibir al módulo Autocomplete de JQUERY.

7. El servidor se comunica con la base de datos por medio del driver JDBC para java. Éste permite enviar consultas SQL a nuestra base de datos MYSQL.

8. En la BBDD se guardan todos los datos necesitados por el programa, esto incluye, los tipos de productos, los productos fabricados, los datos de la empresa y los datos de los usuarios. En esta ocasión se ha utilizado MYSQL, por el hecho de ser gratuita, open source y contar con una gran comunidad para resolver dudas y problemas que puedan surgir.

3.2. Modelo de datos

A continuación se incluye la figura con el modelo lógico de las entidades de BBDD en las que se basa el programa de trazabilidad.

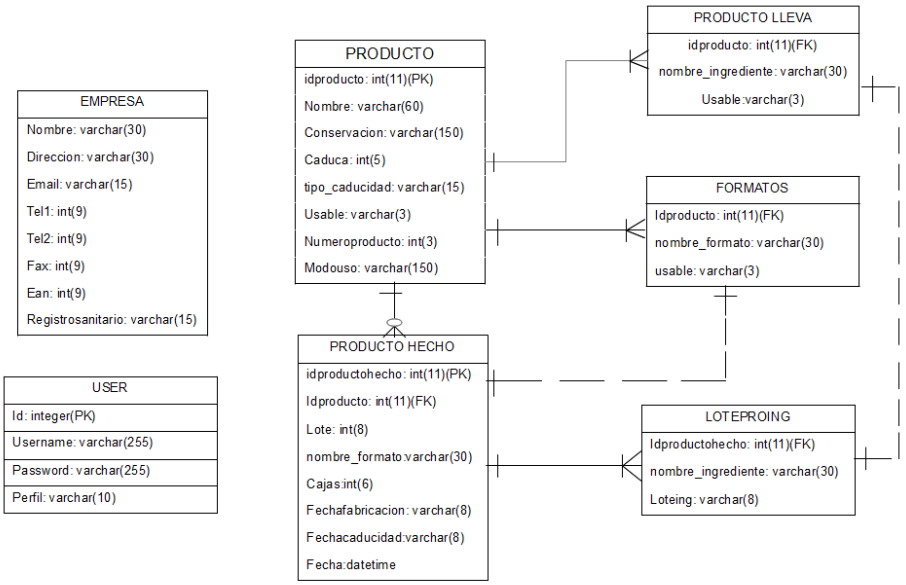


FIGURA 3.3: Modelo de datos

3.2.1. Entidad Empresa

En esta entidad se almacenarán los datos de la empresa.

Código campo	Tipo	Observaciones
Nombre	varchar(30)	Nombre de la empresa
Dirección	varchar(30)	Dirección de la empresa
Email	varchar(15)	Email de la empresa
Tel1	int(9)	Primer teléfono
Tel2	int(9)	Segundo teléfono
Fax	int(9)	Fax de la empresa
Ean	int(9)	Código EAN de la empresa
Registro sanitario	varchar(15)	Nº del registro sanitario

CUADRO 3.1: Entidad Empresa

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE ‘empresa‘ (
  ‘nombre‘ varchar(30) collate utf8_spanish_ci NOT NULL,
  ‘direccion‘ varchar(30) collate utf8_spanish_ci NOT NULL,
  ‘email‘ varchar(15) collate utf8_spanish_ci NOT NULL,
```

```
  `tel1` int(9) NOT NULL,
  `tel2` int(9) NOT NULL,
  `fax` int(9) NOT NULL,
  `ean` int(9) NOT NULL,
  `registrosanitario` varchar(15) collate utf8_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

Codigo 3.1: Script crear tabla empresa

3.2.2. Entidad User

En esta entidad se almacenarán los nombres de los usuarios, su contraseña y su perfil, que puede ser admin o usuario.

Código campo	Tipo	Observaciones
Id	Integer	Identificador del usuario
User	varchar(255)	Nombre del usuario
Password	varchar(255)	Contraseña del usuario
Perfil	varchar(10)	Perfil: admin o usuario

Cuadro 3.2: Entidad Login

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE user
(
   id integer primary key,
   username varchar(255) unique,
   password varchar(255),
   perfil varchar(10)
);
```

Codigo 3.2: Script crear tabla login

3.2.3. Entidad Producto

En esta entidad se almacenan los datos del producto de los que solo pueda haber una ocurrencia.

Código campo	Tipo	Observaciones
Idproducto(PK)	int(11)	Identificador del producto
Nombre	varchar(60)	Nombre del producto
Conservacion	varchar(150)	Método de conservación
Caduca	int(5)	Cantidad de días, meses o años de vida útil
Tipo$_c$*aducidad*	varchar(15)	Días, meses o años
Usable	varchar(3)	Se puede usar. Si o no
Numeroproducto	int(3)	Se utiliza para la generación del código de barras
Modouso	varchar(150)	Explicación de la utilización del producto.

Cuadro 3.3: Entidad Producto

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE ‘producto‘ (
  ‘idproducto‘ int(11) NOT NULL auto_increment,
  ‘nombre‘ varchar(60) collate utf8_spanish_ci NOT NULL,
  ‘conservacion‘ varchar(150) collate utf8_spanish_ci NOT NULL,
  ‘caduca‘ int(5) NOT NULL,
  ‘tipo_caducidad‘ varchar(15) collate utf8_spanish_ci NOT NULL,
  ‘usable‘ varchar(3) collate utf8_spanish_ci NOT NULL,
  ‘numeroproducto‘ int(3) NOT NULL,
  ‘modouso‘ varchar(150) collate utf8_spanish_ci NOT NULL,
  PRIMARY KEY  (‘idproducto‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci AUTO_INCREMENT=1 ;
```

Codigo 3.3: Script crear tabla producto

3.2.4. Entidad Producto Lleva

En esta entidad se almacenan los diferentes ingredientes de cada producto.

Código campo	Tipo	Observaciones
Idproducto(FK)	int(11)	Identificador del producto
Nombre$_{i}ngrediente$	varchar(30)	Nombre del ingrediente
Usable	varchar(3)	Si o no

Cuadro 3.4: Entidad Producto Lleva

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE ‘producto_lleva‘ (
  ‘idproducto‘ int(11) NOT NULL,
  ‘nombre_ingrediente‘ varchar(30) character set ucs2 NOT NULL,
  ‘usable‘ varchar(3) collate utf8_spanish_ci NOT NULL default ’si’,
  KEY ‘idproducto‘ (‘idproducto‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
ALTER TABLE ‘producto_lleva‘
  ADD CONSTRAINT ‘producto_lleva_ibfk_1‘ FOREIGN KEY (‘idproducto‘) REFERENCES
  ‘producto‘ (‘idproducto‘) ON DELETE CASCADE ON UPDATE CASCADE;
```

Codigo 3.4: Script crear tabla producto lleva

3.2.5. Entidad Formatos

En esta entidad se almacenan los diferentes formatos en los que se puede fabricar un producto.

Código campo	Tipo	Observaciones
Idproducto(FK)	int(11)	Identificador del producto
Nombre$_{f}ormato$	varchar(30)	Nombre del formato
Usable	varchar(3)	Si o no

Cuadro 3.5: Entidad Formatos

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE ‘formatos‘ (
  ‘idproducto‘ int(11) NOT NULL,
  ‘nombre_formato‘ varchar(30) collate utf8_spanish_ci NOT NULL,
```

```
  ‘usable‘ varchar(3) collate utf8_spanish_ci NOT NULL default ’si’,
  KEY ‘idproducto‘ (‘idproducto‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
ALTER TABLE ‘formatos‘
  ADD CONSTRAINT ‘formatos_ibfk_1‘ FOREIGN KEY (‘idproducto‘) REFERENCES
  ‘producto‘ (‘idproducto‘) ON DELETE CASCADE ON UPDATE CASCADE;
```

Codigo 3.5: Script crear tabla formatos

3.2.6. Entidad Producto Hecho

En esta entidad se almacenan los datos del producto de los que solo pueda haber una ocurrencia.

Código campo	Tipo	Observaciones
Idproductohecho	int(11)(PK)	Identificador del producto fabricado
Idproducto(pk)	int(11)(FK)	Identificador del producto
Nombre	varchar(60)	Nombre del producto
Conservacion	varchar(150)	Método de conservación
Caduca	int(5)	Cantidad de días, meses o años de vida útil
Tipo$_c$*aducidad*	varchar(15)	Días, meses o años
Usable	varchar(3)	Se puede usar. Si o no
Numeroproducto	int(3)	Se utiliza para la generación del código de barras
Modouso	varchar(150)	Explicación de la utilización del producto.

Cuadro 3.6: Entidad Producto Hecho

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE ‘producto_hecho‘ (
  ‘idproductohecho‘ int(11) NOT NULL auto_increment,
  ‘idproducto‘ int(11) NOT NULL,
  ‘lote‘ int(8) NOT NULL,
  ‘nombre_formato‘ varchar(30) character set ucs2 NOT NULL,
  ‘cajas‘ int(6) NOT NULL,
  ‘fechafabricacion‘ varchar(8) collate utf8_spanish_ci NOT NULL,
  ‘fechacaducidad‘ varchar(8) collate utf8_spanish_ci NOT NULL,
  ‘fecha‘ datetime NOT NULL,
```

```
  PRIMARY KEY  (‘idproductohecho‘),
  KEY ‘idproducto‘ (‘idproducto‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci AUTO_INCREMENT=1 ;
ALTER TABLE ‘producto_hecho‘
  ADD CONSTRAINT ‘producto_hecho_ibfk_1‘ FOREIGN KEY (‘idproducto‘) REFERENCES
  ‘producto‘ (‘idproducto‘) ON DELETE CASCADE ON UPDATE CASCADE;
```

Codigo 3.6: Script crear tabla producto hecho

3.2.7. Entidad Lote ProIng

En esta entidad se almacenan los números de lote de los diferentes ingredientes usados para la fabricación de un producto.

Código campo	Tipo	Observaciones
Idproducto(pk)	int(11)(FK)	Identificador del producto
Nombre$_{i}ngrediente$	varchar(30)	Nombre del ingrediente
Loteing	varchar(8)	Número de lote del ingrediente

Cuadro 3.7: Entidad Lote-Proing

El script de generación de esta entidad se incluye a continuación. Este script se corresponde con una BBDD MySql.

```
CREATE TABLE ‘loteproing‘ (
  ‘idproductohecho‘ int(11) NOT NULL,
  ‘nombre_ingrediente‘ varchar(30) character set ucs2 NOT NULL,
  ‘loteing‘ varchar(8) collate utf8_spanish_ci NOT NULL,
  KEY ‘idproductohecho‘ (‘idproductohecho‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
ALTER TABLE ‘loteproing‘
  ADD CONSTRAINT ‘loteproing_ibfk_1‘ FOREIGN KEY (‘idproductohecho‘) REFERENCES
  ‘producto_hecho‘ (‘idproductohecho‘) ON DELETE CASCADE ON UPDATE CASCADE;
```

Codigo 3.7: Script crear tabla lote proing

3.3. Servlets

Los servlets se encargan de generar el contenido de forma dinámica. Reciben una HttpServletRequest y según lo convenido devuelven una HttpServletResponse. En el programa de trazabilidad encontramos los siguientes servlets:

- ServletBorrarProducto: se encarga de borrar un producto fabricado. Recibiendo un número de lote.
- ServletBorrarTipoProducto: se encarga de borrar un tipo de producto. Recibiendo el nombre del producto.
- ServletBorrarUsuario: se encarga de borrar un usuario. Recibiendo su nombre.
- ServletDatosEmpresa: se encarga de guardar los datos de la empresa, ya sea porque se han modificado o porque se han introducido nuevos datos.
- ServletImprimirEtiquetas: se encarga de generar una etiqueta. Recibe un número de lote y devuelve un pdf.
- ServletInformes: genera los informes en formato EXCEL. Recibe los parámetros introducidos por el administrador, devuelve el fichero para su descarga, no para su visualización en el navegador.
- ServletIngresarProducto: se encarga de ingresar un producto fabricado en la base de datos. Recibe todos los datos del producto y recoge la fecha del sistema para introducirla en la base de datos.
- ServletLogin: se encarga del acceso de los usuarios. Recibe un nombre y una contraseña, si es correcta dejará seguir si no devolverá un mensaje de error y redireccionará a la página de login.
- ServletLogOut: se encarga de finalizar la conexión de los usuarios.
- ServletModificarTipoProducto: se encarga de modificar los atributos de un producto. Recibe los datos del tipo de producto y modificar aquellos que sean distintos. Si el administrador ha borrado o modificado un ingrediente o un formato el servlet actualizará la base de datos poniendo el valor de "no.[en] el campo usable de ese ingrediente o formato.
- ServletNuevoTipoProducto: se encarga de introducir en el sistema un nuevo tipo de producto.
- ServletNuevoUsuario: guarda los datos de los nuevos usuarios introducidos por el administrador.

- ServletValidación: este servlet se encarga de la validación de diferentes campos introducidos en los inputs por parte del usuario.

Todos estos servlets se encuentran almacenados en el paquete "servlets".

3.4. Clases/Objetos

En la programación orientada a objetos, una clase es una construcción que se utiliza como un modelo (o plantilla) para crear objetos de ese tipo. El modelo describe el estado y el comportamiento que todos los objetos de la clase comparten. Un objeto de una determinada clase se denomina una instancia de la clase. Acontinuación se muestran las clases utilizadas en el programa:

- ConexionBD: esta clase se encarga de carga el driver para la conexión con la base de datos, así como de conectarse y desconectarse de la misma.
- GestionBD: contiene las instrucciones sql necesarias para realizar consultas e introducir datos.
- Empresa: define todos los atributos que puede tener la entidad empresa.
- EmpresaDO: define los métodos necesarios para interactuar con la entidad empresa, ya sea consultar datos o introducirlos en la base de datos. En la siguiente tabla se muestran los métodos que contiene.

Nombre	Recibe	Devuelve
guardarDatosEmpresa	Empresa	void
actualizarDatosEmpresa	Empresa	void
listarDatosEmpresa		Empresa
hayDatosEmpresa		boolean

Cuadro 3.8: EmpresaDO

- Etiqueta: define los atributos de la etiqueta: Empresa y ProductoHecho.
- EtiquetaDO: define los métodos necesarios para interactuar con la entidad Etiqueta. En la siguiente tabla se muestran los métodos que contiene.

Nombre	Recibe	Devuelve
generarXML	Empresa,ProductoHecho,String(path)	void
generarEtiqueta	String(path)	void

Cuadro 3.9: EtiquedaDO

- InformeDO: define los métodos necesarios para generar un informe. En la siguiente tabla se muestran estos métodos.

Nombre	Recibe	Devuelve
generarInformeIntervaloFechas	ArrayList<ProductoHecho>	void
generarInformeFechaFabricacion	ArrayList<ProductoHecho>	void
generarInformeLoteProducto	ProductoHecho	void
generarInformeResumenEan	ArrayList<Producto>	void

Cuadro 3.10: InformeDO

- Producto: define todos los atributos de un tipo de producto.
- ProductoDO: define todos los métodos para interactuar con la entidad Producto. En la siguiente tabla se muestran estos métodos.

Nombre	Recibe	Devuelve
GuardarNuevoTipoProducto	Producto	void
listarTiposProducto		ArrayList<String[]>
listarIngredientesProducto	String	ArrayList<String[]>
numeroIngredientes	String	int
listarFormatosProducto	String	ArrayList<String[]>
numeroFormatos	String	int
buscarTipoProducto	String	Producto
buscarTipoProducto	int	Producto
BuscarNombreProductoPorId	Integer	String
BuscarIdProductoPorNombre	String	Integer
BorrarTipoProducto	int	void
ModificarTipoProducto	Producto	void
listarConservacion	String	String
listarModouso	String	String
listarCaducidad	String	String[]
listarNumeroProducto	String	String
siguienteNumeroProducto		int
ultimoNumeroProducto		int
existeNombreProducto	String	boolean
listarTiposProductoNombreEanCaducidad		ArrayList<Producto>

Cuadro 3.11: ProductoDO

- ProductoHecho: define los atributos de un producto fabricado.

- ProductoHechoDO: define los métodos para interactuar con un producto fabricado. En la siguiente tabla se muestran estos métodos.

Nombre	Recibe	Devuelve
guardarProducatoHecho	ProductoHecho , String	String
buscarProductoHechoLote	Integer	ProductoHecho
hayProductoHechoFechaFabricacion	String	boolean
hayProductoHechoFechaCaducidad	String	boolean
hayProductoHechoLoteProducto	String	boolean
buscarProductoHechoFabricacion	String	ArrayList<ProductoHecho>
buscarProductoHechoNombre	String	ArrayList<ProductoHecho>
buscarProductoHechoCaducidad	String	ArrayList<ProductoHecho>
buscarProductoHechoTodos		ArrayList<ProductoHecho>
buscarProductoHecho	Integer	ProductoHecho
siguienteLote		String
ultimoLote		void
fechaCaducidad	String[],Calendar	String
numeroProductosHechosPorNombre	String	int
numeroProductosHechosTodos		int
buscarProductoHecho10Todos	int	ArrayList<ProductoHecho>
buscarProductoHecho10Nombre	int , String	ArrayList<ProductoHecho>
numeroProductosHechosIntervaloFechas	String , String	int
buscarProductosHechosIntervaloFechas	String, String	ArrayList<ProductoHecho>
borrarProductoHechoLote	String	void
borrarProductosPasados		void
fechaProductoHechoLote	String	String

Cuadro 3.12: ProductoHechoDO

- Usuario: define los atributos de un usuario.
- UsuarioDO: define los métodos necesarios para interactuar con la entidad usuario, se muestran en la siguiente tabla.

Nombre	Recibe	Devuelve
comprobarDisponibilidadNombre	Usuario	boolean
guarduarUsuario	Usuario	void
loginPerfil	Usuario	String
listarUsuarios		ArrayList<String>
borrarUsuario	String	void
conectarUsuarioSesiones	String , String	void
quienUsuarioSesiones	String	ArrayList<Usuario>

Cuadro 3.13: UsuarioDO

- SecureDigester: transforma la clave aplicándole un cifrado SHA.

Nombre	Recibe	Devuelve
digest	String	String

Cuadro 3.14: SecureDigester

3.5. Estilos CSS

El programa de trazabilidad al ser una aplicación orientada al usuario necesita de estilos tanto para que se vea bien como para su usabilidad. Todos los estilos utilizados se guardan en el archivo estilos.css. A continuación se muestran algunos de estos estilos. Lo primero que aplica esta hoja de estilos es un reseteador de css. Este reseteador se aplica debido a que cada navegador puede aplicar ciertos valores predeterminados para ciertos elementos lo cual hace que ese elemento no se vea siempre igual. Esto es un problema porque los desarrolladores solemos usar el mismo navegador para hacer las pruebas, pero luego lo pruebas en otro y se puede ver completamente distinto y no como deseabas. El reseteador CSS lo hemos descargado de meyerweb.com y es así:

```
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
```

```
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
        margin: 0;
        padding: 0;
        border: 0;
        font-size: 100%;
        font: inherit;
        vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
        display: block;
}
body {
        line-height: 1;
}
ol, ul {
        list-style: none;
}
blockquote, q {
        quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
        content: '';
        content: none;}

table {
        border-collapse: collapse;
        border-spacing: 0;
}
```

Codigo 3.8: Reseteador CSS meyerweb

Una vez hecho el reset, procedemos a definir los distintos elementos de la página. Como hemos explicado antes la aplicación solo cuenta con una página. Ésta está dividida en 3 zonas, un menú horizontal arriba del todo, una zona en la izquierda para introducir datos y otra en la derecha para visualizarlos. A continuación vamos a explicar el estilo utilizado en cada una de ellas.

3.5.1. Estilos Menu Principal

El menú principal consta de una lista con diferentes botones, uno para cada funcionalidad de la aplicación.

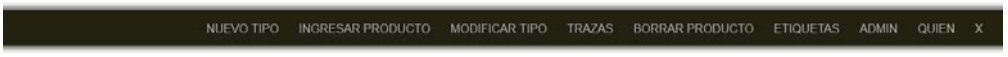


Figura 3.4: Menu principal

A continuación se muestra el estilo utilizado:

```
        /*** caja contenedor del Menu ***/
#menuarriba{
        box-shadow: 0 0 5px 5px #888;
        width:100%;
        height:3em;
        position:relative;
        background-color:#222111;
}

/*** items del menu ***/
#menuarriba li{
        margin-right:0.3em;
        margin-left:0.3em;
        margin-top:0.75em;
        padding-top:auto;
        padding-bottom:auto;
        float:right;
        position:relative;
        height:1.5em;
        display: inline-block;
```

```
        text-align: center;
}

/*** contenedor de items del menu ***/
#menuarriba ul{
        right:1%;
        position:relative;
}

/*** clase de los botones del menu ***/
 .botonesarriba{
        margin-top:auto;
        margin-bottom:auto;
        position:relative;
        height: 1.5em;
        display: block;
        text-align: center;
        font-size: 1em;
        font-weight:lighter;
        color: rgb(180, 180, 180);
        font-family: 'Helvetica Neue','Helvetica',Arial,sans-serif;
        background-color:#222111;
    text-decoration: none;
    text-shadow: 0px 1px 0px rgb(0, 0, 0);
    border-radius: 3px 3px 3px 3px;
    letter-spacing: 0px;
    border-color: #222111;
    border: none;
}

/*** acciones a realizar cuando se produce un hover en los botones ***/
 .botonesarriba:hover{
        margin-top:auto;
        margin-bottom:auto;
        position:relative;
        height: 1.5em;
        display: block;
        text-align: center;
        font-size: 1em;
```

```
        color: rgb(255, 255, 255);
        font-family: 'Helvetica Neue','Helvetica',Arial,sans-serif;
        background-color:#222111;
        text-decoration: none;
        text-shadow: 0px 1px 0px rgb(0, 0, 0);
        border-radius: 3px 3px 3px 3px;
        letter-spacing: 1px;
        border-color: rgb(150,150,150);
        border-width: 1px;
}
```

Codigo 3.9: Estilos CSS menu

El efecto producido por el hover consiste en la tranformación de la fuente, pasa a ser color blanco y aumenta su tamaño y la separación entre las letras. Como se muestra en las siguientes figuras:

Figura 3.5: Boton del menu sin hover

Figura 3.6: Boton del menu con hover

3.5.2. Ingresar datos

Éste módulo se encuentra a la izquierda. Consta de un contenedor y dentro de éste ya encomtramos un div cuyo id es "ingresarDatos". Siempre que el usuario interacciona con la aplicación y tenga que introducir datos tendrá presente en el centro de este módulo "visualizarDatos" un input de tipo text cuya clase es "ingresando", por lo tanto para el diseño de este módulo se ha tenido siempre en cuenta la posición del input.

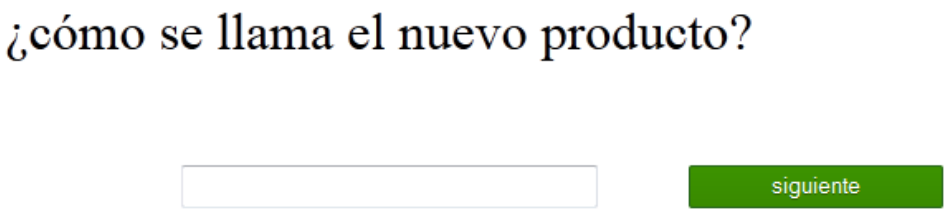


FIGURA 3.7: Div ingresar datos

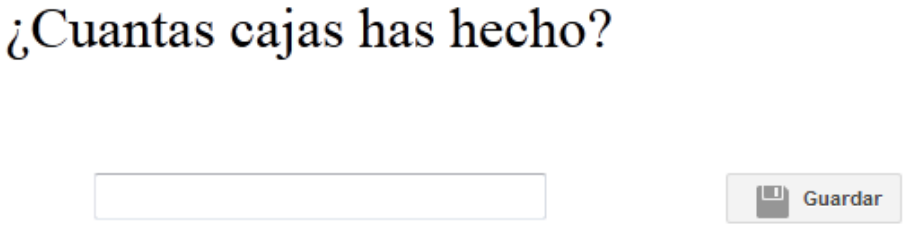


FIGURA 3.8: Div ingresar datos guardar

Como vemos en las figuras anteriores, cada vez que el usuario tenga que introducir un dato se le hará una pregunta y habrá un botón o bien de siguiente o bien de guardar. Es posible que si la situación lo requiere haya un segundo botón para introducir otro valor del mismo campo como se muestra a continuación.

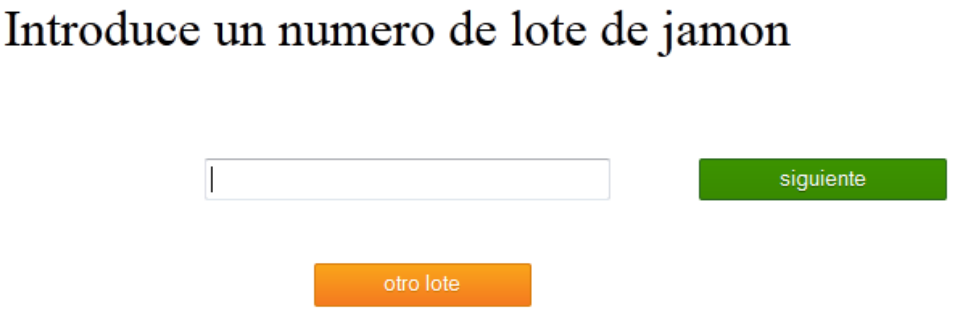


FIGURA 3.9: Div ingresar datos otro boton

Para resumir a continuación se muestra un esquema de los distintos nombres y estructuras en este módulo, y su código.

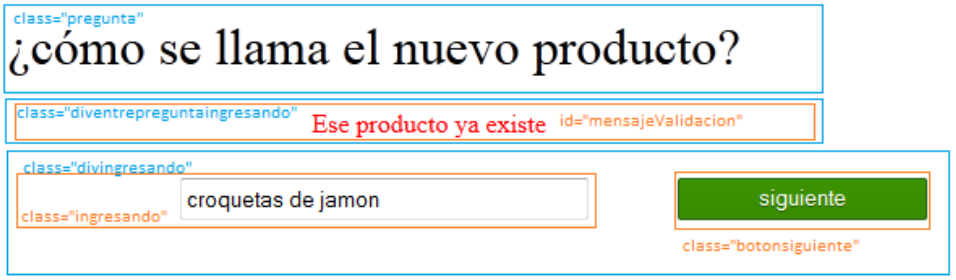


FIGURA 3.10: Esquema ingresar datos

```
#wrapperizq{
        top:10%;
        width:50%;
        height:80%;
        position:relative;
        float:left;
}
#ingresardatos{
        position:relative;
        padding:auto;
        width:100%;
        height:90%;
        float:left;
}
.divingresando{
        position:relative;
        height:30%;
        width:90%;
        margin-right: 0%;
        margin-left:10%;
}
.diventrepreguntaingresando{
        position:relative;
        height:15%;
        width:90%;
```

```
        margin-left:10%;
}
.pregunta{
        display:block;
        text-align:center;
        margin-top:5%;
        margin-left:auto;
        margin-right:auto;
        font-size: 2em;
}
.diventreingresandobotonotro{
        position:relative;
        height:30%;
        width:100%;
}
.botonsiguiente{
        font:bold;
        font:Arial;
        color: #FFF !important;
        text-shadow: 0 1px 0 #2D6200 !important;
        border: 1px solid #29691D !important;
        border-radius: 2px;
        -webkit-border-radius: 2px;
        -moz-border-radius: 2px;
        background: #3A8E00;
        background: -webkit-linear-gradient(top, #3C9300, #398A00);
        background: -moz-linear-gradient(top, #3C9300, #398A00);
        background: -ms-linear-gradient(top, #3C9300, #398A00);
        background: -o-linear-gradient(top, #3C9300, #398A00);
        -webkit-transition: border .20s;
        -moz-transition: border .20s;
        -o-transition: border .20s;
        transition: border .20s;
        vertical-align:middle;
        position:relative;
        width:25%;
        display:inline-block;
        height:2em;
        float:right;
```

```
        text-align:center;
        padding-bottom:0.3em;
}
.botonsiguiente:hover{
        font:bold;
        color: #FFF !important;
        text-shadow: 0 1px 0 #2D6200 !important;
        border: 1px solid #2D6200 !important;
        border-radius: 2px;
        -webkit-border-radius: 2px;
        -moz-border-radius: 2px;
        background: #3F83F1;
        background: -webkit-linear-gradient(top, #3C9300, #368200);
        background: -moz-linear-gradient(top, #3C9300, #368200);
        background: -ms-linear-gradient(top, #3C9300, #368200);
        background: -o-linear-gradient(top, #3C9300, #368200);
}
.botonotro{
        font:bold;
        font:Arial;
        color: #FFF !important;
        text-shadow: 0 1px 0 #da7c0c !important;
        border: 1px solid #da7c0c !important;
        border-radius: 2px;
        -webkit-border-radius: 2px;
        -moz-border-radius: 2px;
        background: #f78d1d;
        background: -webkit-gradient(linear, left top, left bottom, from(#faa51a), to
        background: -moz-linear-gradient(top,  #faa51a,  #f47a20);
        -webkit-transition: border .20s;
        -moz-transition: border .20s;
        -o-transition: border .20s;
        transition: border .20s;
        height:2em;
        padding-bottom:0.3em;
        text-align:center;
        position:relative;
        margin-left:36%;
        bottom: 0px;
```

```
        display:block;
        width:22%;
}
.ingresando{
        position:relative;
        margin-left:25%;
        display:inline-block;
        width:40%;
        height:1.5em;

}
```

Codigo 3.10: Estilos CSS ingresar datos

3.5.3. Visualizar datos

Ya que los distintos campos de los formularios son cargados dinámicamente y solo es posible ver un campo al mismo tiempo en la parte de ingresar datos es necesario poder ver durante todo el proceso de ingresar datos los campos ya introducidos. Para esto tenemos en la parte derecha de la página un div con id visualizardatos", que a su vez contiene otro con clase divdetabla", que a su vez contiene una tabla donde se mostrarán los datos con id vistabla".

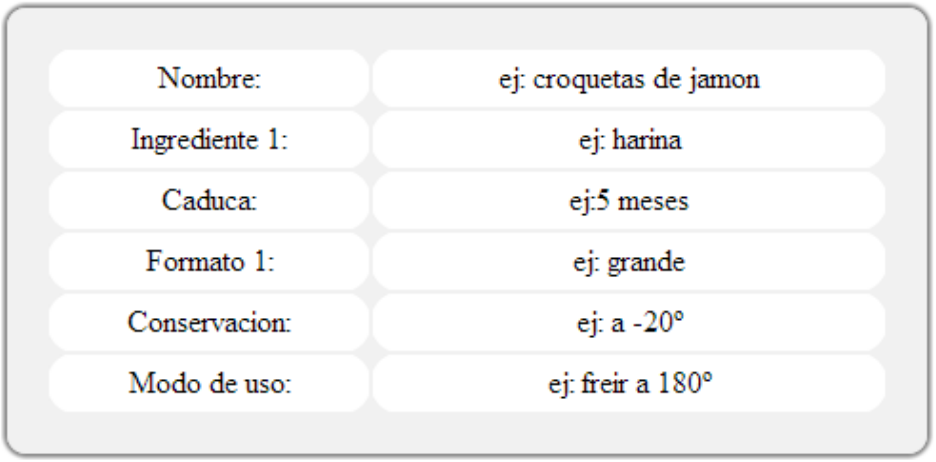


Figura 3.11: Div visualizar datos

En la figura anterior podemos ver como sería este div de visualizar datos en la pantalla ingresar nuevo tipo de producto. A continuación mostramos el código de los estilos css:

```
#vistabla{
        position:relative;
        height:100%;
        width:100%;
        border-collapse:collapse;
        max-height:100%;
        max-width:100%;
        table-layout: auto;
        min-width:0px;
        background:#F1F1F1;
}

#vistabla tr{
        background-color:white;
        border-radius:50px;
        width:100%;
}

#vistabla tr th{
        border-style:solid;
        border-width:2px;
        border-radius:50px;
        border-color:#F1F1F1;
        height:30px;
        padding-top: auto;
        padding-bottom: auto;
        vertical-align:middle;
        max-width:10em;
        word-wrap:break-word;
        min-width:50px;
}

#vistabla tr td{
        text-align:center;
        border-style:solid;
        border-width:2px;
```

```
        border-radius:50px;
        border-color:#F1F1F1;
        height:30px;
        padding-top: auto;
        padding-bottom: auto;
        vertical-align:middle;
        max-width:10em;
        word-wrap:break-word;
        min-width:110px;
        width:110px;
}

.divdetabla{
        overflow-y:auto;
        position:relative;
        margin-top:10%;
        display:block;
        margin-right:auto;
        margin-left:auto;
        box-shadow: 0 0 2px 2px #888;
        border-style:solid;
        border-width:20px;
        border-radius:10px;
        border-radius-left:15px;
        -moz-border-radius:10px;
        border-color:#F1F1F1;
        border-collapse: separate;
        max-height: 70%;
        max-width:65%;
}
```

Codigo 3.11: Estilos CSS visualizar datos

3.6. Javascript

El javascript es una parte básica de la aplicación. Gracias a el cargamos contenido dinámicamente por medio de AJAX, mejoramos la usabilidad y validamos los formularios entre otras cosas. Tenemos dos archivos javascript: funciones.js y myjquery.js. El primero

contiene solo código javascript y en el segundo nos ayudamos de la librería JQUERY. A continuación vamos a mostrar algunas de las funcionalidades hechas con javascript.

3.6.1. Enviar formulario

Esta función sirve para enviar los formularios a los servlets y así poder guardar los datos introducidos por el usuario, o mostrar los datos solicitados.

```
function enviarFormularioFormid(url, formid){
    var Formulario = document.getElementById(formid);
    var cadenaFormulario = "";
    var sepCampos;
    sepCampos = "";
    for (var i=0; i <= Formulario.elements.length-1;i++) {
        cadenaFormulario += sepCampos+Formulario.elements[i].name+'=
                '+encodeURI(Formulario.elements[i].value);
        sepCampos="&";
    }
        ajax=getAjax();
        ajax.open("POST", url, true);
        ajax.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded;
        charset=ISO-8859-1');
        ajax.send(cadenaFormulario);
        ajax.onreadystatechange = procesar;
}
```

Codigo 3.12: Javascript enviar formulario

En el código anterior vemos como la función recibe dos parámetros:

1. url: este parámetro indica la dirección a la que tiene que ser enviada la request.
2. formid: este parámetro indica el id del formulario que queremos enviar.

Lo primero que hace la función es coger los datos del form y guardarlos en la variable Formulario con la ayuda de la función javascript document.getElementById. A continuación genera una url con los parámetros dispuestos correctamente. Por último usa las funciones escritas antes mediante las cuales hace una petición AJAX al servidor, con un método POST. En el siguiente código se muestran estas funciones para la utilización del AJAX.

```
function getAjax(){
        if (window.XMLHttpRequest){
                return new XMLHttpRequest();
        }
        else if (window.ActiveXObject){
                return new ActiveXObject("Microsoft.XMLHTTP");
        }
}

function procesar(){
        if(ajax.readyState==4){
                if(ajax.status==200){
                        document.getElementById("ingresardatos").
                        innerHTML=ajax.responseText;
                }
        }
}
```

Codigo 3.13: Javascript AJAX

3.6.2. Cursor en el input al cargar la página

A continuación vamos a explicar una de las funcionalidades que mejoran la usabilidad del programa y también que ha sido solicitada por el cliente. Al cargarse una página con un input text, el cursor debe estar ya en el input para ahorrar al usuario tener que pinchar en el cuando quiera escribir.

```
$(document).ready(function(){
        $('#ingresardatos').on('DOMNodeInserted ', '#nuevoTipoNombre', function(){
            $(".ingresando").focus();
        });
}
```

Codigo 3.14: Javascript cursor en input al cargar

Vemos primero que usa la función ready() de jQuery. Ready es un método propio de jQuery, que revisa si el DOM está listo para usarse. Es más util que el window.onload, pues este debe esperar a que todos los elementos de la pagina esten cargados (como scripts e imágenes) para ejecutar. El ready", en cambio, espera solo a la estructura.

Para asignar una función a un elemento que no está en la página cuando es cargado el archivo javascript usamos la función on() de jQuery. En este caso le decimos que cuando se inserte un nuevo nodo, en este caso el elemento con id nuevoTipoNombre", ponga el cursor en el input con clase ingresando" mediante la función focus().

3.6.3. Validación de formularios

La validación de los datos introducidos por el usuario es una función básica, sin esta el programa podría tener muchos problemas e incongruencias.

```
$("#ingresardatos").on({
        keypress:function(e){
                $("#mensajeValidacion").html("");
                var code = (e.keyCode ? e.keyCode : e.which);
                if(code == 13) {
                        $(".botonsiguiente").click();
                }
        }
}, "#nuevoTipoNombre :text");
```

Codigo 3.15: Submit para validación

Usando de nuevo el método on() de jQuery hacemos que al pulsar la tecla intro/enter se pulse el botón con clase botonsiguiente" en el cual está introducida una función que es la que hace la validación. En este caso validacionNombreTipoProducto() que contiene el siguiente código:

```
function validacionEstaVacio(){
        var valorInputText=$(".ingresando").val();
        var longitud=valorInputText.length;
        if(longitud==0){
                $("#mensajeValidacion").text("El campo esta vac\u00edo");

                return true;
        }else{

                return false;
        }
};
```

```
function validacionLongitudNombreTipoProducto(){
        var nombre=$(".ingresando").val();
        var longitud=nombre.length;
        if(longitud<45){
                return true;
        }else{
                $("#mensajeValidacion").text("El nombre es demasiado largo");
                return false;
        }
};

function validacionExisteNombreTipoProducto(){
        var vuelve=false;
        //Cogemos el dato del input text ".ingresando"
        dataString=$(".ingresando").val();
        //Fijamos quien est haciendo la peticin al servletValidacin
    var soy="NombreTipoProducto";
    $.ajax({
        //Que mtodo usamos
        type: "POST",
        //Url del servlet al que le hacemos la request
        url: "ServletValidacion",
        //Fijamos async en false porque queremos que sea sncrono
        //para que el usuario no pueda continuar sin validar
        async:false,
        data: {nombre : dataString,quien: soy},
        dataType: "json",
        //if received a response from the server
        success: function( data, textStatus, jqXHR) {
             if(data.succes==true){
                 vuelve=true;
             }
        },
        //Si no hay respuesta del servidor
        error: function(jqXHR, textStatus, errorThrown){
             console.log("Something really bad happened " + textStatus);
              $("#ajaxResponse").html(jqXHR.responseText);
        },
        //capturamos la request antes de enviarla al servidor
```

```
        beforeSend: function(jqXHR, settings){
        },
        complete: function(jqXHR, textStatus){
            $('#myButton').attr("disabled", false);
        }
    });
    return vuelve;
};

function validacionNombreTipoProducto(){
        if((!validacionEstaVacio())&&(validacionLongitudNombreTipoProducto())){
                if(validacionExisteNombreTipoProducto()){
                        return true;
                }else{
                        $("#mensajeValidacion").html("Ese producto ya existe");
                        return false;
                }
        }else{
                return false;
        }
};
```

Codigo 3.16: Validacion nombreTipoProducto

Veamos lo que hace la funcion validacionNombreTipoProducto():

1. validacionEstaVacio(): esta función comprueba que el campo no esté vacío, imprimiendo un mensaje de error en el caso que lo esté.
2. validacionLongitudNombreTipoProducto(): comprueba que la longitud del dato introducido sea menor a 45 caracteres.
3. validacionExisteNombreTipoProducto(): hace una petición al servidor, en el que le pregunta si ese nombre de producto ha sido ya usado. En el caso de que haya sido usado muestra un mensaje de error en el que indica que ese producto ya existe, en el otro caso deja continuar.

3.7. Google+ UI Buttons

En algunos sitios de la página se pueden ver unos bonitos botones que cambian de forma y color al pasar el ratón sobre ellos. Esto es gracias a Google+, que nos deja usar estos botones con una licencia Creative Commons Attribution 3.0 Unported License, esto quiere decir que somos libres de: copiar, compartir, adaptar e incluso otorgarnos la autoría!

3.7.1. Instalación

Para instalar estos botones lo primero que tenemos que hacer es descargárnoslos, lo tenemos en la página http://code.brucegalpin.com/google-plus-ui-buttons/ . Una vez descargados sólo tenemos que agregar las imágenes y los archivos css a nuestro servidor y poner un link hacia "css/css3-buttons.css" en la cabecera de nuestra web. Además debemos añadir otro link hacia Jquery Tip Tip Plugin en la cabecera y subir los respectivos archivos al servidor, "jquery.tiptip.js" y "tiptip.css".

3.7.2. Modo de uso

Las imágenes de los botones vienen todas dentro de un "sprite". De hecho vemos como al descargarnos los archivos en la página antes mencionada nos vienen dos "sprites" uno en blanco y negro y otro en color. Así se conseguirá el efecto de cambio de color cuando pasamos el ratón por encima de alguno de estos botones. El uso del "sprite" aparte de facilitar su uso al no tener que incluir muchas imágenes, será bueno para el funcionamiento de la página al reducir drásticamente las peticiones al servidor.

FIGURA 3.12: Google+ ui sprite gris

FIGURA 3.13: Google+ ui sprite color

Para poner un botón tenemos dos formas de hacerlo:

Creando un botón:

```
<button><span class="icon icon108"></span></button>
```

CODIGO 3.17: Crear botón usando button

O creando un hipervínculo:

```
<a href="#" class="button"><span class="icon icon108"></span></a>
```

CODIGO 3.18: Crear botón usando a

Dependiendo del número de icono que elijamos mostraremos cada uno de los botones.

Capítulo 4

Manual de instalación

4.1. Servidor web

Como el programa de trazabilidad es una aplicación web, será necesario la instalación de un servidor web. En este caso vamos a usar Tomcat 7. Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

A continuación vamos a dar una explicación detallada de como instalar Tomcat 7.

4.1.1. Insalar Java

Tomcat requiere java para ser instalado, al final y al cabo es un contenedor de Java Servlets. Desde Tomcat 5.5 la dependencia del JDK para la compilación de JSPs ha desaparecido. Esto quiere decir que solo necesitamos instalar JRE. Lo primero que haremos será ir a la página oficial de oracle y descargar el último JRE, en este caso JRE 7. Recuerda elegir la versión de 64bits si el sistema operativo que estás corriendo también lo es. Ahora debemos fijar la variable de entorno para JAVA_HOME. Para esto vamos a propiedades de sistema, pinchamos en variables de entorno, y añadimos una

nueva variable del sitema con nombre JAVA_HOME y valor la ruta hacia el directorio de instalación de java.

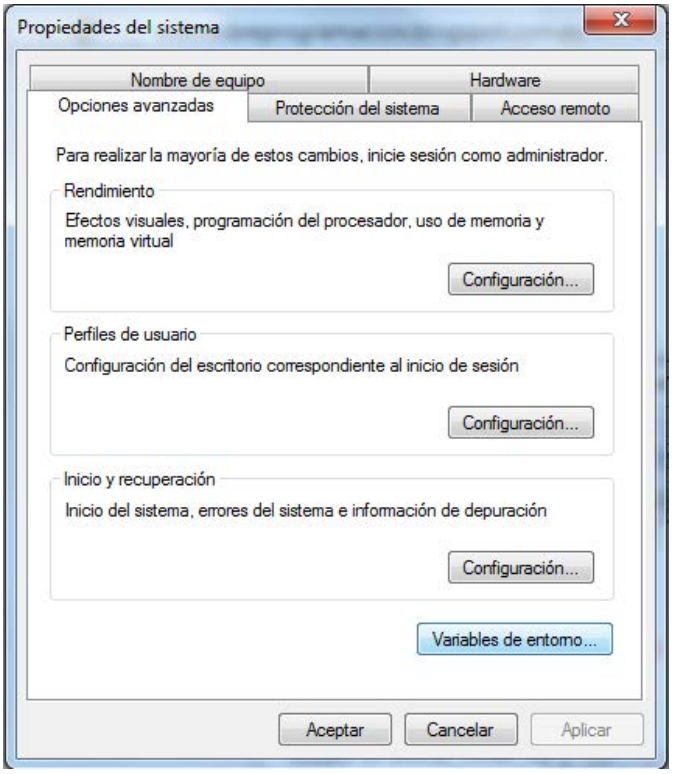


FIGURA 4.1: Propiedades del sistema

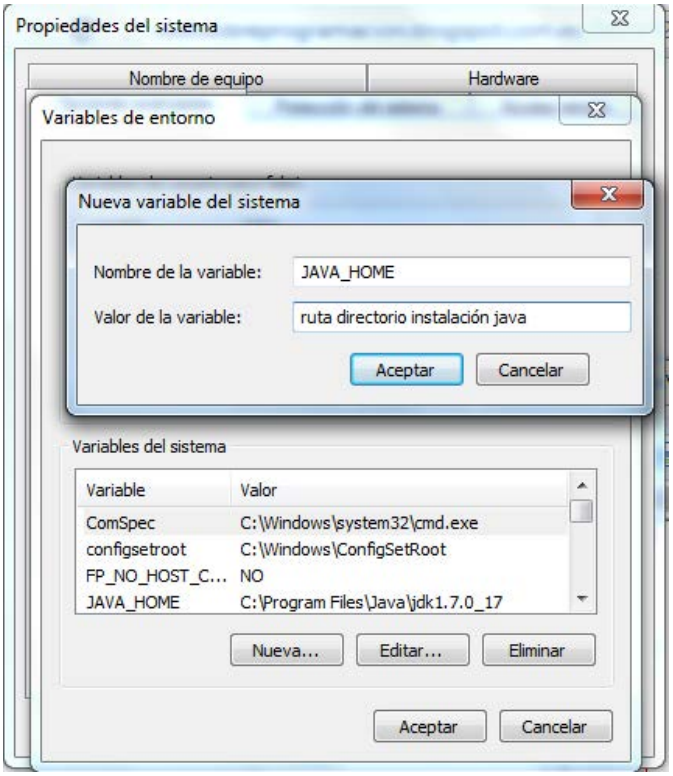


FIGURA 4.2: Variables de entorno del sistema

4.1.2. Descargar Tomcat 7

Para ello vamos a la pagina oficial de tomcat http://www.tomcat.apache.org y bajamos dentro de los binarios la opción de Windows Service Installer

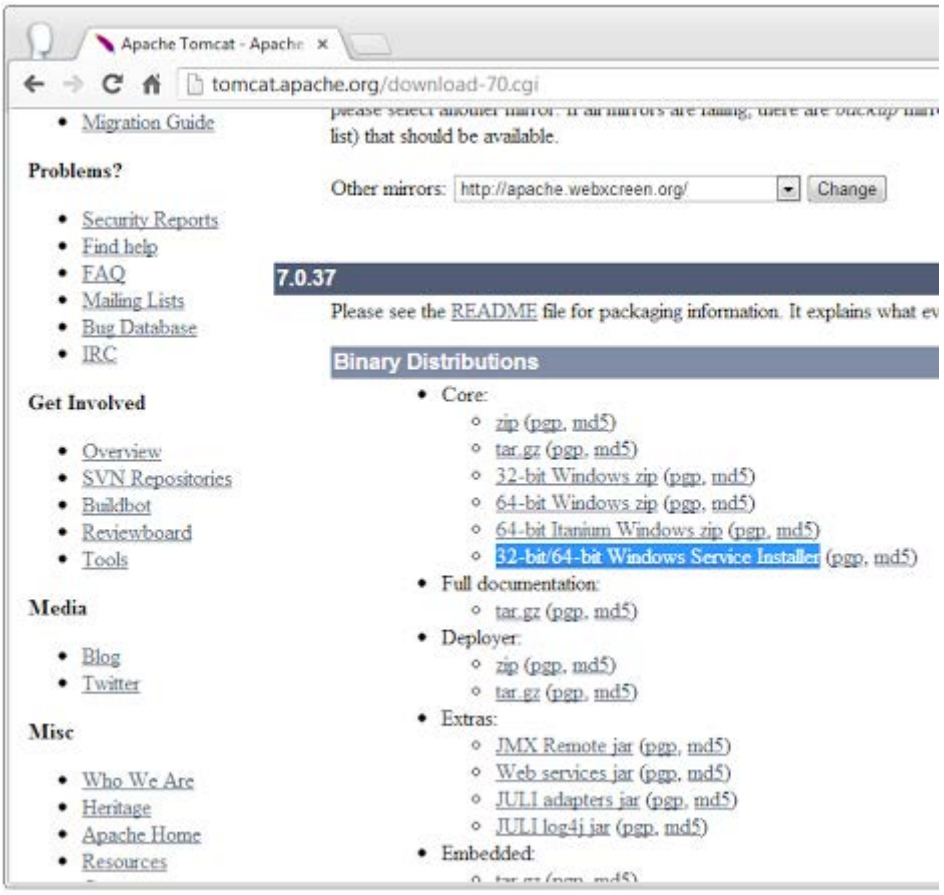


FIGURA 4.3: Descargar Tomcat

4.1.3. Asistente de instalación

Primero se nos mostrará un mensaje de bienvenida, le damos a next. A continuación nos pedirá aceptar los acuerdos de licencia, aceptamos pulsando I agree. Lo siguiente será elegir los componentes, elegimos todos menos examples, como se muestra en la siguiente figura.

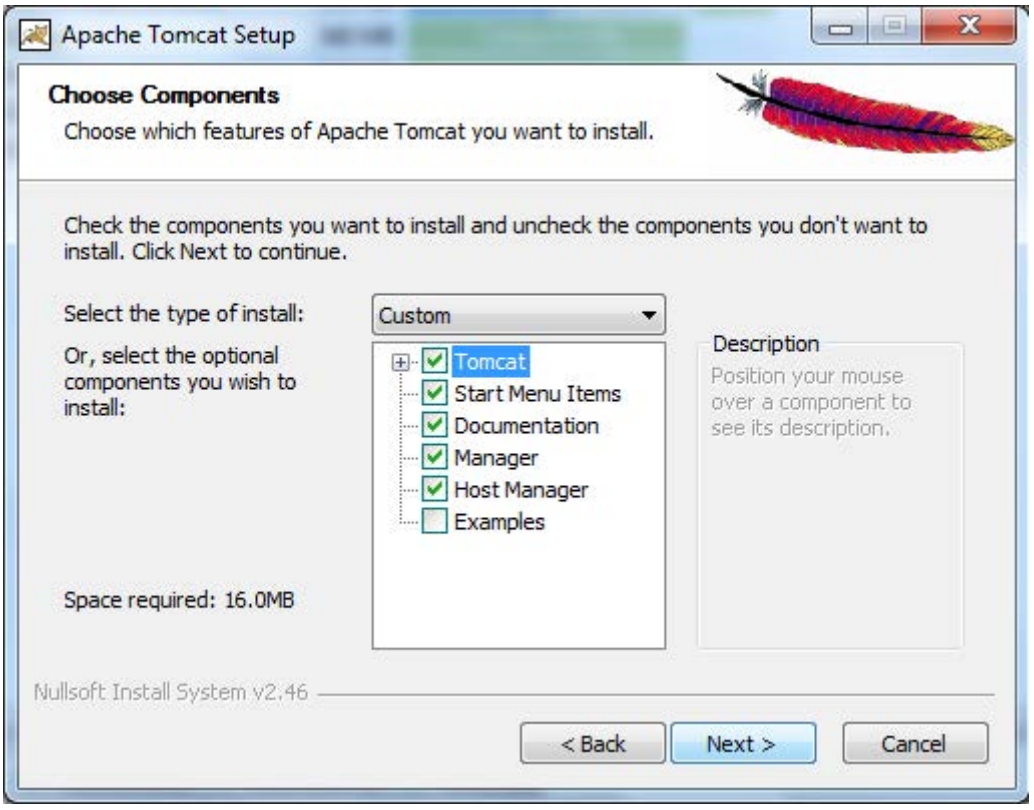


FIGURA 4.4: Tomcat elegir componentes

Después habrá que elegir el nombre del servidor, así como los puertos que utilizará y el nombre, contraseña y los roles del administrador.

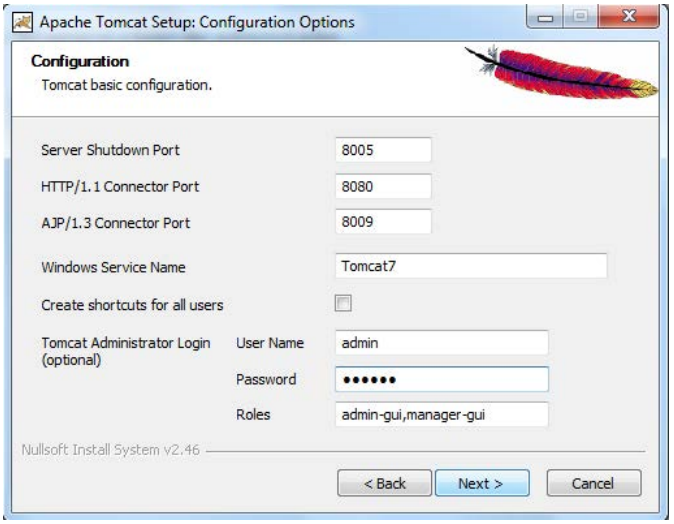


FIGURA 4.5: Tomcat nombres

A continuación nos pedirá introducir la ruta hacia la Java Virtual Machine. Elegimos la ruta en la que anteriormente habíamos instalado java y le damos a next. Por último nos pedirá que introduzcamos la ruta en la que queremos instalar Tomcat, la introducimos

y pinchamos en install. Después de seguir estos datos, tendremos instalado el Tomcat y se nos mostrará un icono en windows de esta forma:

FIGURA 4.6: Icono Tomcat

4.1.4. Puesta a punto

Una vez finalizada la instalación hace falta hacer una última gestión para que el servidor se encienda al arrancar windows, osea sea un servicio de windows. Para esto vamos al icono le damos a Configure, ficha general, y escogemos la opción Startup type. Para verificar la instalación vamos a un navegador web y escribimos http://localhost:8080 y nos deberá mostrar una pantalla como esta:

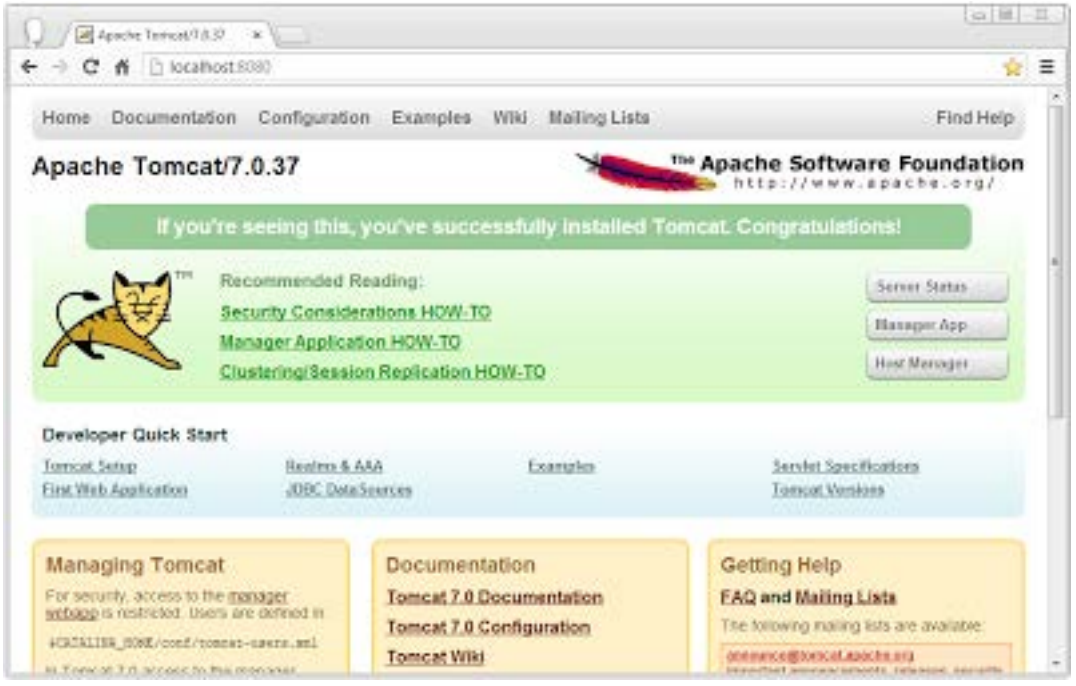


FIGURA 4.7: Tomcat pantalla inicial

4.2. Base de datos MYSQL

MYSQL es un sistema de gestion de base de datos, es desarrollado, distribuido y soportado por Oracle Corporation.

¿Porqué usar MYSQL?

- Es gratuito.
- Es open source.
- Es rápido, seguro y fácil de usar.
- Gran comunidad.

4.2.1. Instalar MYSQL server

Lo primero que tenemos que hacer es descargar MYSQL server desde la página oficial: http://dev.mysql.com/downloads/mysql/ . Elegimos la última versión y la que mas nos convenga para nuestro sistema operativo 64 o 32 bits. Nos descargará un archivo MSI Installer. Una vez se haya descargado este archivo lo ejecutamos. Y seguimos con el instalador que nos mostrará los siguientes pasos:

1. Escogemos install MYSQL products.
2. Aceptamos la licencia y pinchamos en next.
3. Nos pregunta si queremos comprobar, antes de la instalación, si hay nuevas actualizaciones disponibles, le damos a execute.
4. Elegimos el tipo de instalación y la ruta hacia el directorio de instalación. En nuestro caso custom como se muestra a continuación.

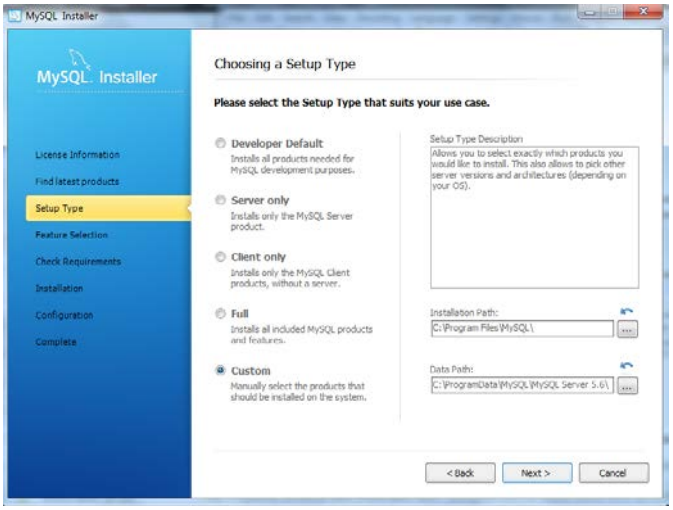


FIGURA 4.8: Instalacion MYSQL custom

5. En esta pantalla nos indican que elijamos los componentes que queremos instalar, elegimos MYSQL Server y los MYSQL Connector como se muestra en la siguiente imagen:

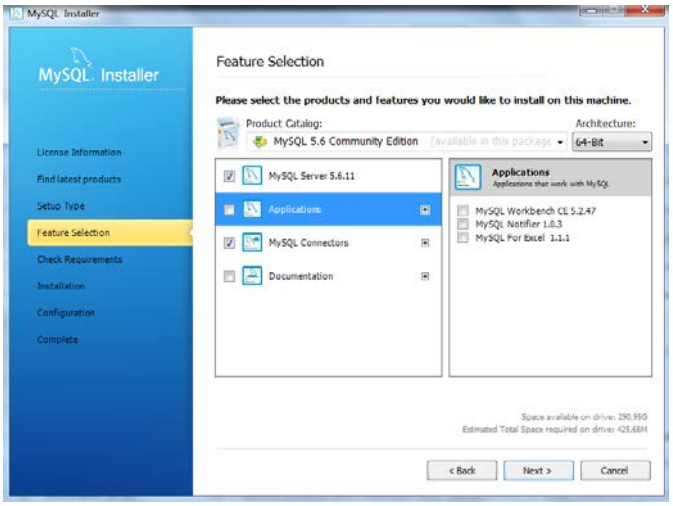


FIGURA 4.9: Instalacion MYSQL elegir componentes

6. El instalador procede a la instalación personalizada, como le hemos indicado.

7. Configuración: elegimos el tipo de configuración de MYSQL Server y el tipo de puerto.

8. Después nos solicita la contraseña del root y nos deja crear otros usuarios. Elegimos la misma contraseña que hayamos usado en nuestras conexiones desde JAVA.

9. Por último nos pregunta el nombre que le queremos dar al servicio en Windows, le damos un nombre y fijamos la opción de iniciar cuando inicie windows.

Con esto ya tendríamos el MYSQL Server instalado y funcionando.

4.3. Configuración de seguridad

Para el perfecto funcionamiento del programa y debido a que queremos usar una conexión segura debemos aplicar algunos cambios a nuestros archivos de configuración del servidor. Lo primero que vamos a hacer es generar un certificado de seguridad, podríamos contratar uno pero eso nos costaría dinero. Lo vamos a generar haciendo uso de la herramienta de java keytool, introduciendo el siguiente código en un cmd:

```
keytool -genkey ^
    -keystore keystore ^
    -alias tomcat ^
    -keyalg RSA ^
    -keysize 2048 ^
    -dname CN=localhost ^
```

```
    -storepass changeit ^
    -keypass changeit
```

Codigo 4.1: Generar certificado de seguridad

Una vez generado el certificado vamos a la carpeta de Tomcat y modificamos el fichero server.xml que está dentro de la carpeta "conf" . En este fichero debemos decirle al Tomcat que queremos que se conecte por del SSL, descomentando el siguiente código o añadiéndolo en su defecto:

```
<Connector port="8443"
   maxThreads="200"
   minSpareThreads="5"
   maxSpareThreads="75"
   enableLookups="true"
   disableUploadTimeout="true"
   acceptCount="100"
   scheme="https"
   secure="true"
   SSLEnabled="true"
   clientAuth="false"
   sslProtocol="TLS"
   keystorePass="changeit"
   keystoreFile="conf/keystore" />
```

Codigo 4.2: Tomcat server.xml

4.4. Despliegue de la aplicación

Continuando con la instalación vamos a desplegar el programa en Tomcat y la base de datos en el MYSQL Server.

4.4.1. Despliegue de la aplicación en Tomcat

Para esto vamos a necesitar exportar nuestro programa como un fichero .war en el IDE que hayamos utilizado. Una vez tengamos el archivo .war vamos al navegador e

introducimos la direccion http://localhost:8080 donde se nos mostrará una imagen como la siguiente.

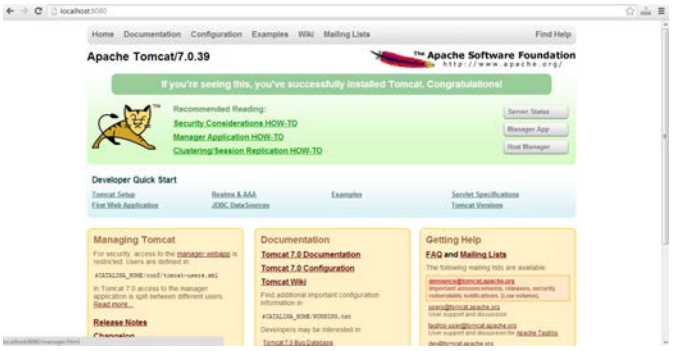


FIGURA 4.10: localhost:8080

Desde aquí vamos a Manager App, en la parte superior derecha. Nos pedirá la contraseña del Tomcat que previamente habíamos introducido, la introducimos y le damos a aceptar. En esta nueva página http://localhost:8080/manager/html en la parte inferior, en la sección Desplegar, en la opción Archivo War a desplegar seleccionamos nuestro archivo .war y le damos a desplegar como se muestr a continuación.

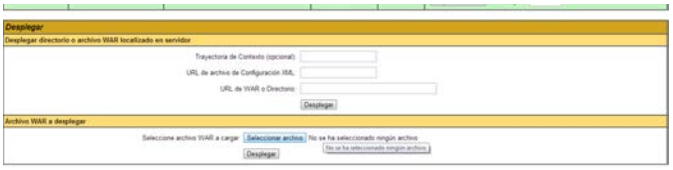


FIGURA 4.11: Desplegar arhivo WAR

Una vez acabado esto, si nos dirigimos a la dirección http://localhost:8080/trazabilidad/ nos deberá aparecer la pantalla de log-in de la aplicación de trazabilidad.

4.4.2. Despliegue de la base de datos

Para este paso vamos a utilizar el MYSQL Command Line Client que se nos ha facilitado en la aplicación. Los buscamos y lo ejecutamos. Lo primero que nos va a pedir es la contraseña que hemos introducido anteriormente. Una vez dentro de la consola seguimos los siguientes pasos.

1. Introducimos CREATE DATABASE trazabilidad; y nos debe salir el siguiente mensaje: Query OK, 1 row affected.

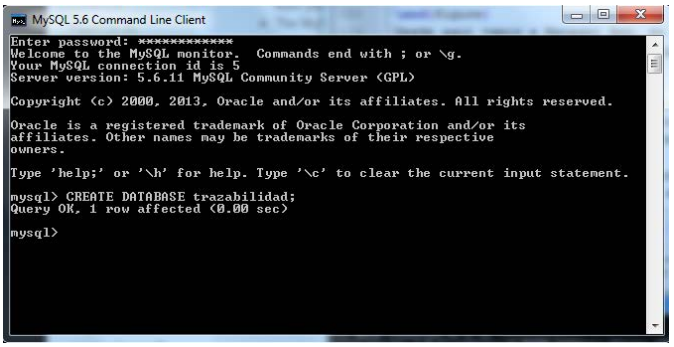


FIGURA 4.12: Crear base de datos

2. A continuación abrimos un cmd y nos dirigimos a la ruta donde tengamos el archivo .sql con la base de datos. Y escribimos el siguiente comando: mysql -p -u username database-name < file.sql, en nuestro caso mysql -p -u root trazabilidad < trazabilidad.sql

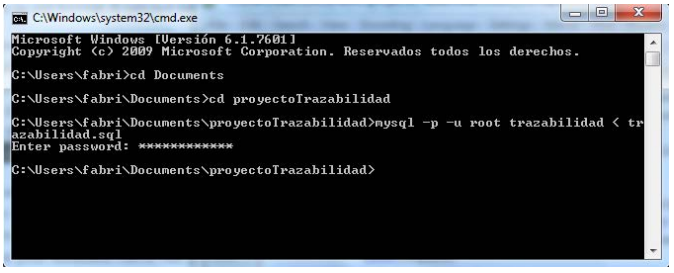


FIGURA 4.13: Importar base de datos

3. Para comprobar que se ha instalado con éxito, nos metemos en mysql, luego en la base de datos y comprobamos que estén todas las tablas que necesitamos.

FIGURA 4.14: Comprobacion base de datos

Después de esto ya tenemos nuestra aplicación en marcha.

Capítulo 5

Manual de usuario

5.1. Introducción

En este capítulo sirve como guía para el usario. Se presuponen unos conocimientos básicos de informática por parte del usuario, sobre todo en el entorno de internet. Vamos a ver como cada pantalla de la aplicación realiza una función y tendremos un punto detallando todo lo relativo a cada una de estas funciones. Vamos a seguir el manual con un ejemplo en el que introduciremos como tipo de producto croquetas de queso".

5.2. ADMIN

Lo primero que deberá hacer el administrador para empezar a usar el programa será fijar los datos de la empresa. Deberá rellenar el siguiente formulario.

FIGURA 5.1: Manual datos de la empresa

Estos datos son básicos porque sin ellos no se podrá generar las etiquetas, funcionalidad básica del programa de trazabilidad.

5.3. Nuevo tipo de producto

A continuación el usuario deberá introducir tantos tipos de producto en el sistema como la empresa pueda fabricar. Si no ha introducido los datos de algún producto previamente no podrá introducirlo como producto fabricado. A medida que valla introduciendo los datos, el usuario será capaz de verlos en la tabla a la derecha de la pantalla para comprobar que son correctos. Veamos que campos debe rellenar:

- Nombre. Debe introducir el nombre del tipo de producto.

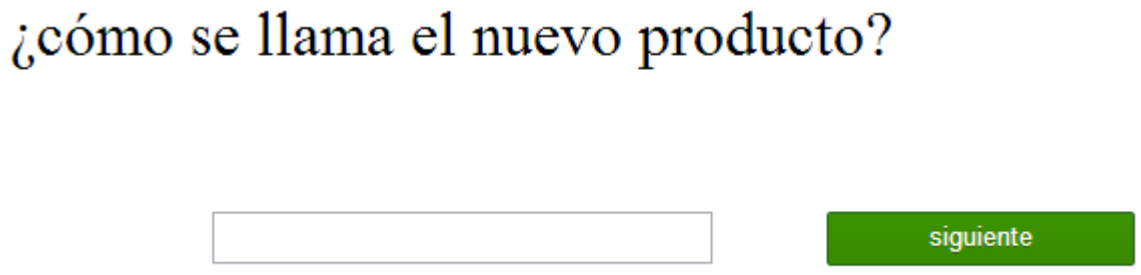


FIGURA 5.2: Manual nuevo tipo de producto - Nombre

- Ingrediente. Debe introducir el nombre de un ingrediente. Si quiere introducir otro ingrediente después, debe pulsar nuevo ingrediente" sino deberá pulsar siguiente".

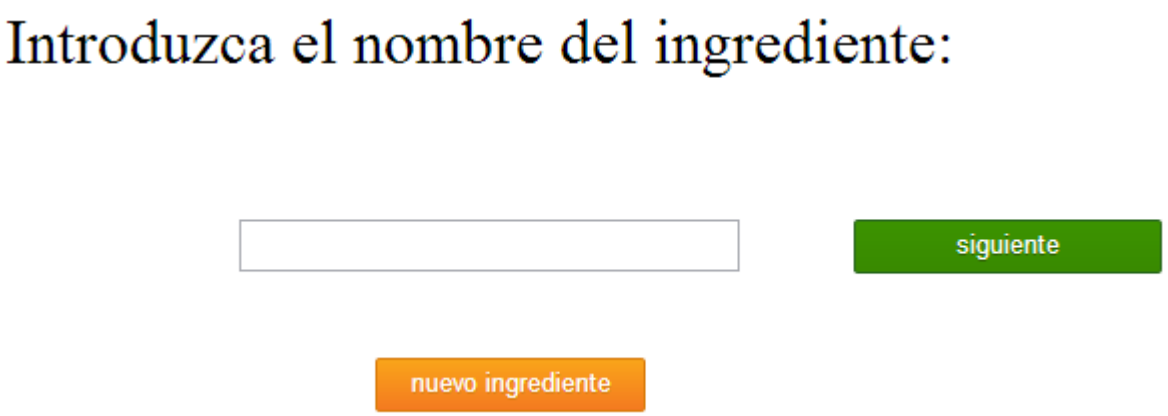


FIGURA 5.3: Manual nuevo tipo de producto - Ingrediente

- Caducidad. Debe introducir un número y luego elegir el tipo de número que ha introducido, pudiendo ser días, meses o años. Indica la vida útil del tipo de producto.

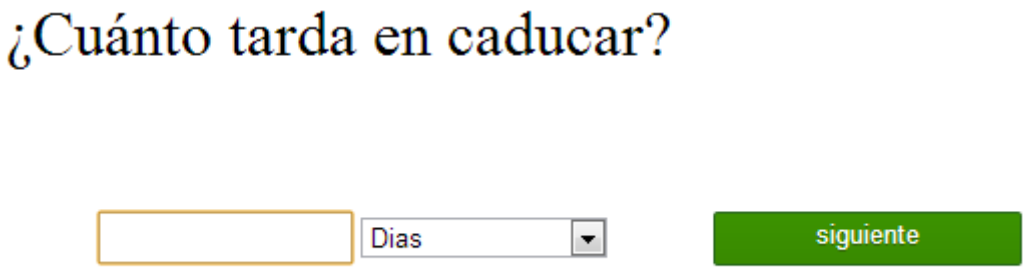


FIGURA 5.4: Manual nuevo tipo de producto - Caducidad

- Formato. Debe introducir el formato en el que se puede fabricar el tipo de producto. Si quiere introducir otro formato después, debe pulsar nuevo ingrediente" sino deberá pulsar siguiente".

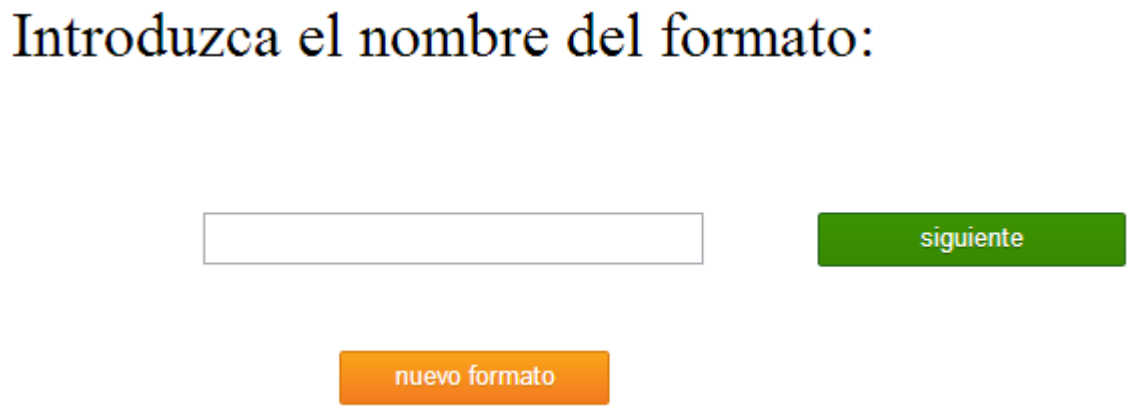


FIGURA 5.5: Manual nuevo tipo de producto - Formato

- Conservación. Debe introducir el método de conservación. Siempre que el producto necesite unas condiciones específicas de ambiénte o temperatura debe indicarlo aquí.

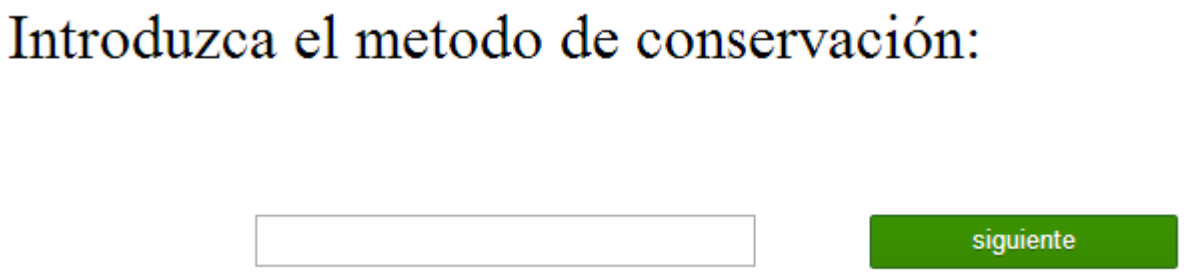


FIGURA 5.6: Manual nuevo tipo de producto - Conservacion

- Modo de uso. Debe introducir como se ha de utilizar el producto. Si hay que descongelarlo, congelarlo, freirlo, hornearlo...

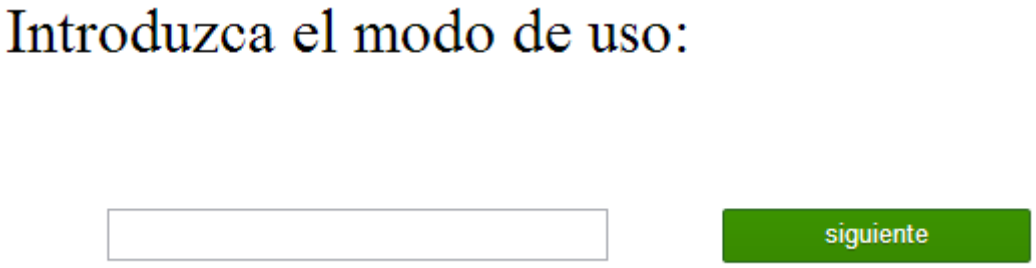
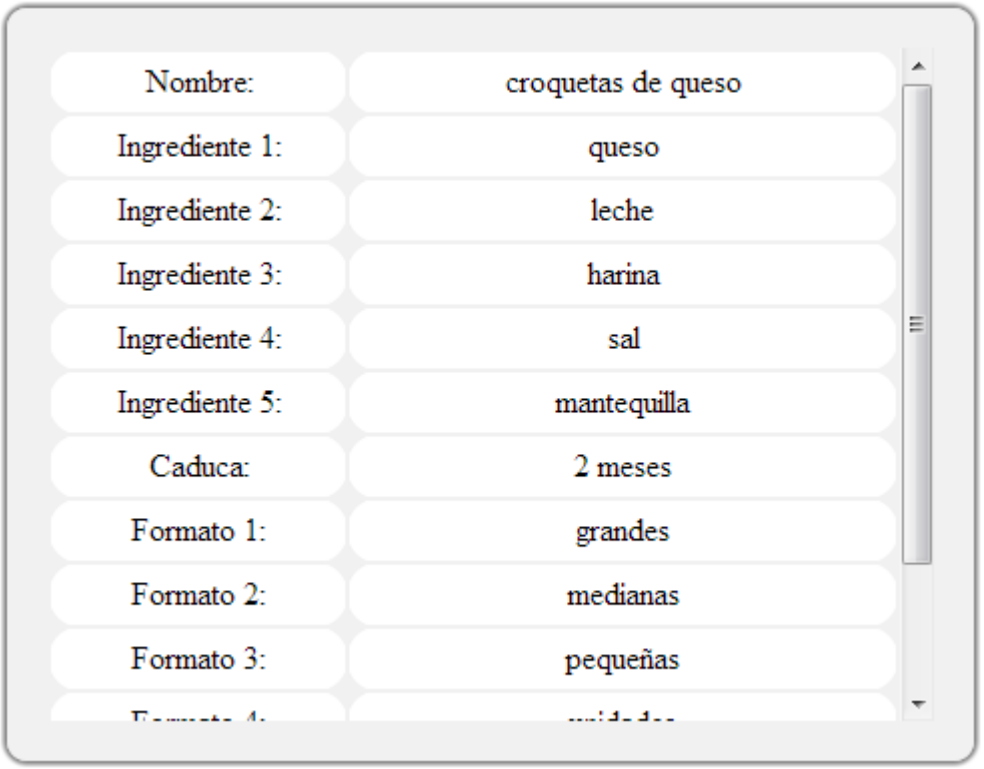


FIGURA 5.7: Manual nuevo tipo de producto - Modo de uso

- Guardar tipo de producto. Una vez introducidos todos los datos nos preguntará si queremos guardarlo. Para verificar si son correctos los datos introducidos podemos repasarlo en la tabla de la derecha que será como se muestra en la siguiente figura.



5.4. Ingresar producto fabricado

Después de haber introducido los tipos de producto, el usuario será capaz de introducir los productos fabricados. Siguiendo el ejemplo vamos a introducir la fabricación de unas croquetas de queso siguiendo las preguntas que nos realiza el programa, como vemos a continuación.

- Nombre. Con el módulo de autocompletar solo será necesario que se introduzcan dos de las letras del producto, pues el sistema mostrará las opciones disponibles.

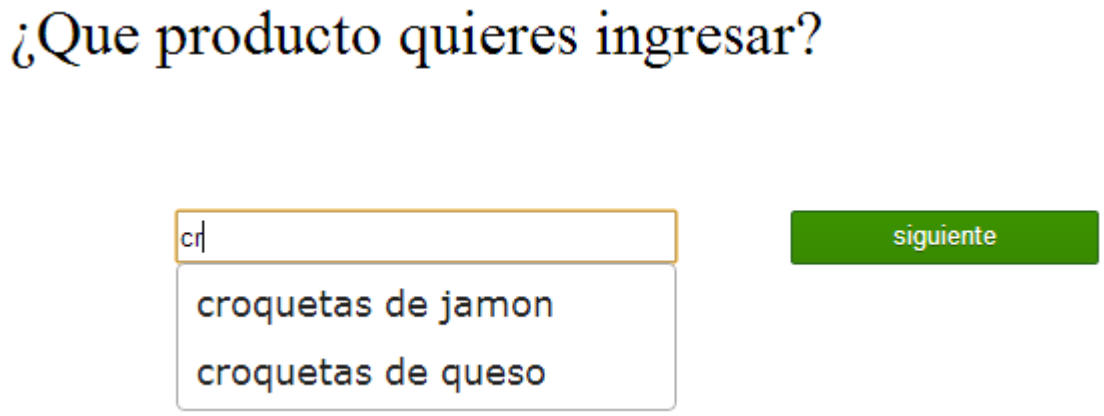


FIGURA 5.8: Manual ingresar producto - Nombre

- Lotes de los ingredientes. Esta es la parte mas importante de la aplicación pues gracias a ella se podrá conocer con que lotes de ingredientes se ha fabricado un producto. Y así determinar su trazabilidad hacia atrás. Nos pedirá que introduzcamos por lo menos un lote de cada ingrediente, pudiendo introducir uno o varios de cada uno de ellos.

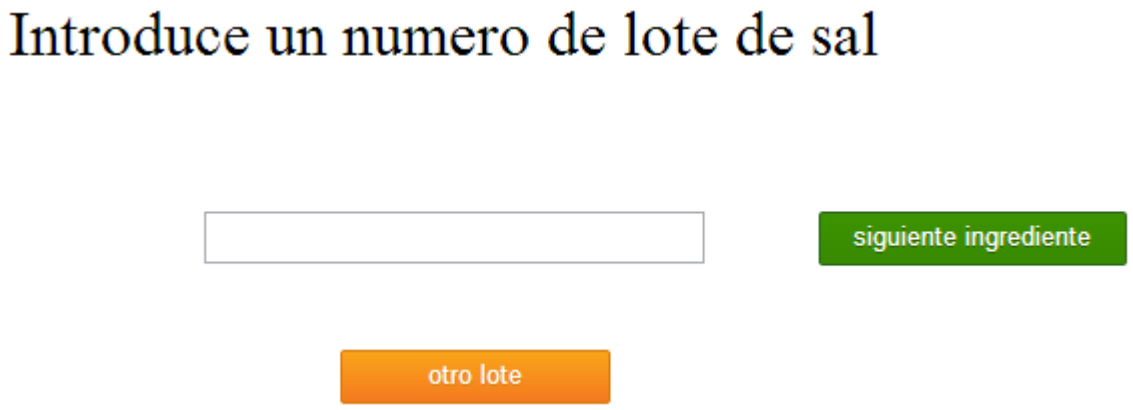


FIGURA 5.9: Manual ingresar producto - Ingredientes

- Formato. Seleccionar el tipo de formato que se ha utilizado.

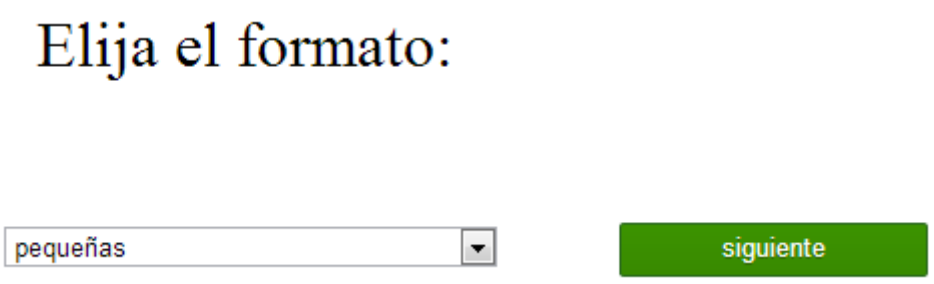


FIGURA 5.10: Manual ingresar producto - Formato

- Cajas. Se debe introducir le número de cajas fabricadas. En el caso de que se haya elegido el formato unidades, la pregunta será cuantas unidades se han fabricado. Ésto es importante porque si se ha elegido cajas solo se generará una etiqueta mientras que si es otro formato el sistema generará una etiqueta por cada caja.

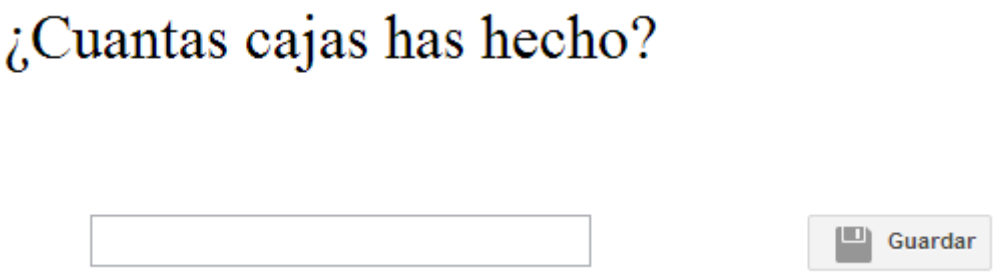


FIGURA 5.11: Manual ingresar producto - Cajas

- Imprimir etiquetas. Una vez guardado el producto, el sistema generará las etiquetas mostrándolas en pantalla. Para imprimirlas se debe hacer click con el botón derecho y darle a imprimir. Aunque el usuario solo pueda ver una etiqueta por pantalla se imprimirán tantas como sean necesarias.

FIGURA 5.12: Manual ingresar producto - Etiqueta

5.5. Modificar tipo de producto

Al cabo del tiempo es posible que sea necesario modificar un producto. Para esto el usuario deberá seleccionar el tipo de producto que quiere modificar en una pantalla como la siguiente:

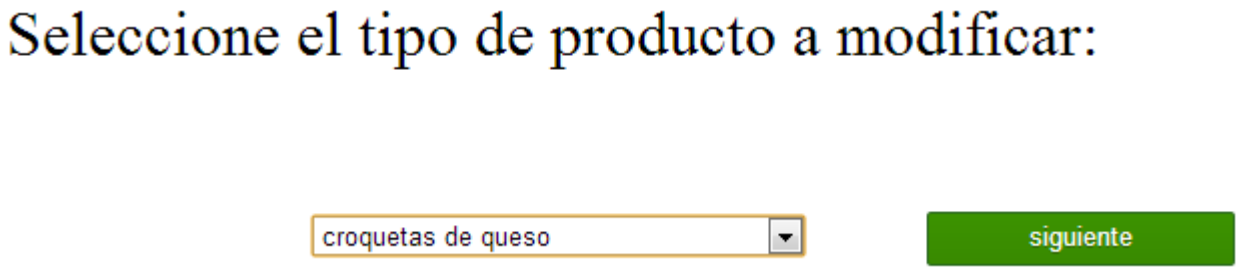


FIGURA 5.13: Manual modificar producto

Una vez seleccionado el tipo de producto el sistema mostrará una tabla, a la derecha de la pantalla, con todas las opciones disponibles. Podrá:

- Crear un nuevo ingrediente.
- Crear un nuevo formato.
- Borrar producto.
- Modificar el nombre.
- Borrar o modificar ingredientes.
- Borrar o modificar formatos.
- Modificar la fecha de caducidad.
- Modificar el modo de uso.
- Modificar la conservación.

La tabla que generada por el sistema, será similar a la que se muestra a continuación. En el caso de que el usuario quiera modificar deberá pulsar el lápiz y para borrar el cubo de basura.

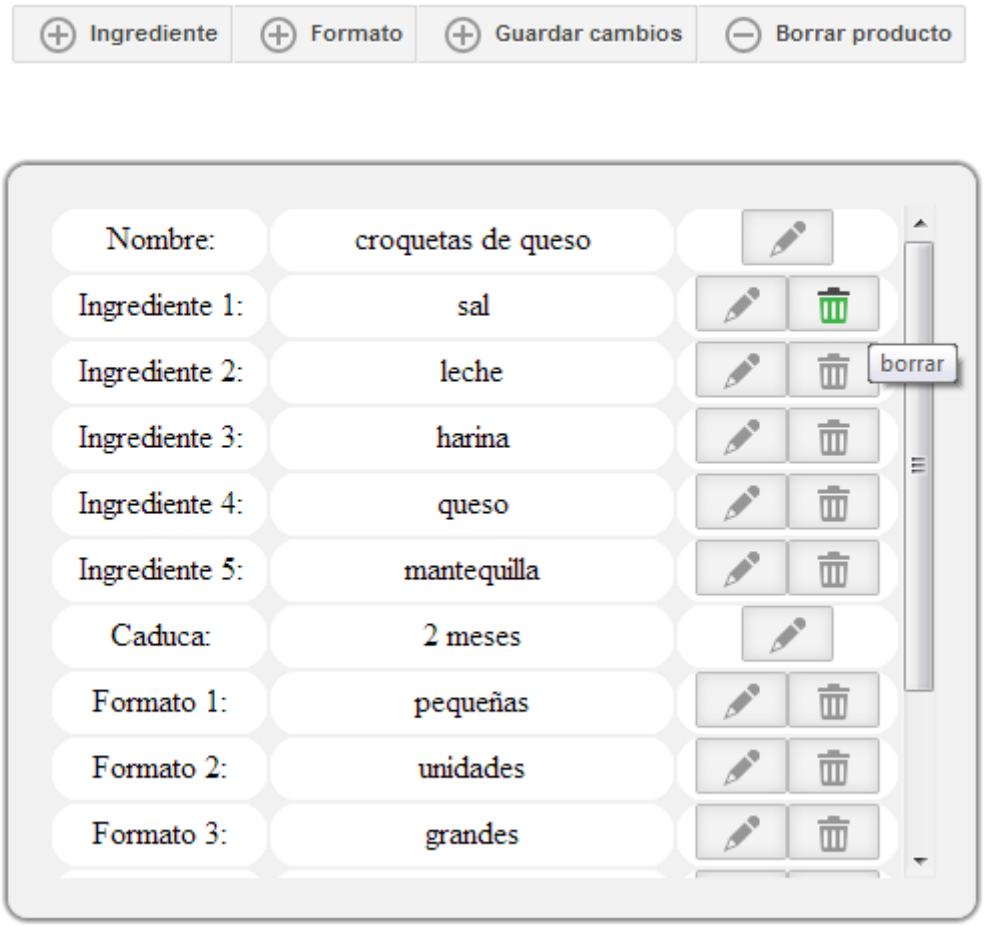


FIGURA 5.14: Manual modificar producto tabla

5.6. Buscador

Esta utilidad sirve para tener conocimiento de cualquier dato introducido en el sistema.

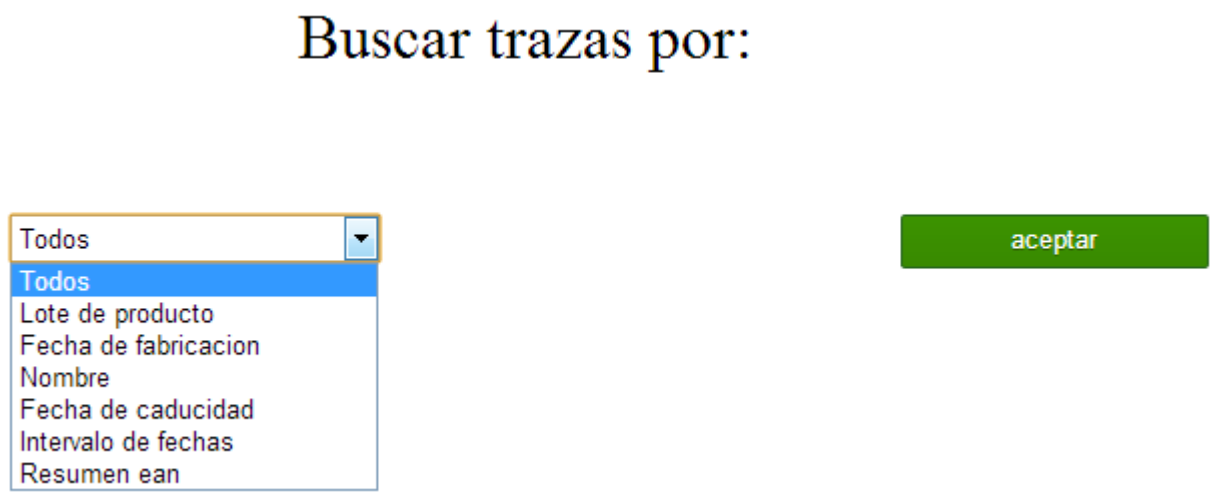


FIGURA 5.15: Manual buscador - Tipo de busqueda

Dependiendo de lo que seleccionemos en primera instancia se pedirá que se introduzcan datos o se mostrarán los datos solicitados directamente. Tipos de dato:

- Todos: si se selecciona este campo el sistema mostrará una tabla con todos los datos de los productos fabricados. Sólo mostrará diez productos a la vez ofreciendo la posibilidad de navegar hacia los restantes con un botón de siguientes y otro de anteriores.
- Lote de producto: introduciendo un número de lote de un producto fabricado el sistema mostrará todos los datos relativos a ese producto.
- Fecha de fabricación: el sistema pedirá que se introduzca un número de fecha con el formato "dd/mm/aa" y mostrará todos los datos de los productos fabricados ese día.
- Fecha de caducidad: igual que la fecha de fabricación pero ésta vez se debe introducir el día que expira la vida útil del producto.
- Nombre: introducimos el nombre del producto y el sistema mostrará todos los productos fabricados con ese nombre.
- Intervalo de fechas: el sistema solicitará dos fechas y mostrará todos los productos fabricados entre estas dos fechas.
- Resumen ean: no habrá que introducir ningún dato. El sistema mostrará el nombre, el código ean y la caducidad de todos los tipos de producto y te dejará descargarte un informe en formato EXCEL con estos mismos datos.

5.7. Borrar producto fabricado - Generar etiquetas

Tanto para borrar un producto fabricado como para generar etiquetas el sistema solicitará un número de lote del producto como se muestra en la siguiente figura.

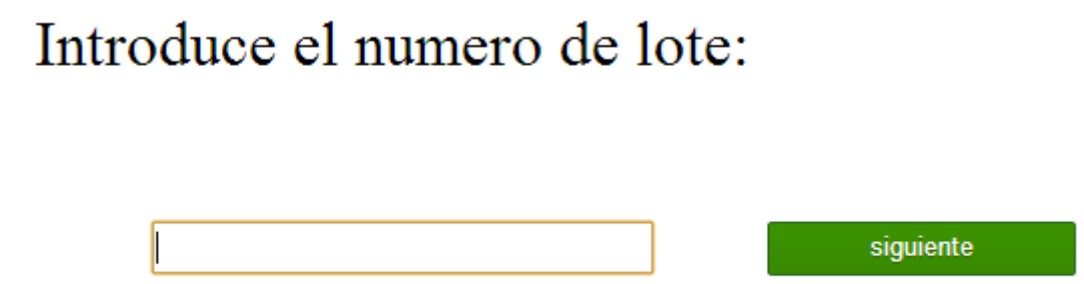


FIGURA 5.16: Manual borrar producto - Imprimir etiquetas

Una vez introducido el número de lote el sistema comprobará que existe. Si es así nos mostará una mensaje de confirmación en la opción de borrar producto o nos mostrará la etiqueta solicitada.

Capítulo 6

Presupuesto

El coste de desarrollo de este proyecto, es especificado en base al coste del hardware y software necesario para su creación, así como al generado por las personas que han trabajado en el mismo.

El material empleado en el desarrollo ha sido el siguiente:

6.1. Material hardware

El hardware utilizado se describe a continuación:

- 1 ordenador con el sistema operativo Windows 7, Intel Core i-5 2410 2.3 GHz
- Impresora térmica Zebra lp 2824
- Scanner lector código de barras CCD ZEBEX Z-3220 USB Negro

El coste del material hardware se describe a continuación:

Producto	Precio (€)
Ordenador Intel Core i-5 2410 2.3 GHz	495,85
Impresora térmica Zebra lp 2824	235,93
Scanner lector código de barras	29,00
TOTAL	**760,78**

Cuadro 6.1: Presupuesto hardware

6.2. Material software

El material software utilizado se describe a continuación:

- Microsoft Windows 7 Home Premium
- Eclipse IDE Indigo for Java EE Developers
- Notepad++
- Tomcat 7
- MYSQL Server

El coste del material software se describe a continuación:

Producto	Precio Licencia(€)
Microsoft Windows 7 Home Premium	70,24
Eclipse IDE	0
Notepad++	0
Tomcat 7	0
MYSQL Server	0
TOTAL	**70,24**

Cuadro 6.2: Presupuesto software

6.3. Recursos humanos

Como recurso humano ha intervenido en el desarrollo del proyecto un ingeniero junior.

- Requisitos: 40 horas
- Analisis: 40 horas
- Implementacion: 300 horas
- Pruebas: 40 horas
- Documentación: 60 horas
- Seguimiento: 15 horas

El precio por hora del ingeniero junior es de 15 €.

Tarea	Horas	Precio
Requisitos	40	600
Análisis	40	600
Implementación	300	4.500
Pruebas	40	600
Documentación	60	900
Seguimiento	15	225
TOTAL	495	**7.425**

Cuadro 6.3: Presupuesto mano de obra

6.4. TOTAL

Concepto	Precio (€)	IVA(21 %)	Precio Total (€)
Material Hardware	760,78	159,76	920,54
Material Software	70,24	17,85	85
Recursos humanos	7.425	1.559,25	8.984,25
TOTAL	**8.256,02**	**1.733,76**	**9989,78**

Cuadro 6.4: Presupuesto total

El presupuesto del proyecto asciende a NUEVE MIL NOVECIENTOS OCHENTA Y NUEVE CON SETENTA Y OCHO euros.

Capítulo 7

Planificación

En este apartado se definirán los roles necesarios para llevar a cabo el proyecto. Debido a que este proyecto es personal todas las tareas serán resueltas por una sola persona. Roles:

- Analista. Estará encargado de extraer la información del sistema al cliente. También podrá participar en la fase de diseño y deberá trabajar conjuntamente con la persona encargada de él.
- Diseñador. Deberá encargarse del diseño detallado del sistema. Estará presente en la fase de análisis y trabajar al mismo tiempo con el analista. A su vez será quién defina las pruebas a las que será sometido el sistema para validar y verificar su funcionamiento.
- Programador. Codificará la información generada en la fase de diseño.
- Técnico de pruebas. Con motivo de verificar que todo funciona correctamente y validar que la aplicación reproduce adecuadamente las audiodescripciones, éste ejecutará las pruebas establecidas por el diseñador.

7.1. Diagrama de GRANTT

					Sep 2012	Oct 2012	Nov 2012	Dic 2012	Ene 2013	Feb 2013	Mar 2013	Abr 2013	May 2013	Jun 2013
Nº	Tarea	Comienzo	Fin	Duración										
1	Requisitos	03/09/12	15/09/12	40										
2	Análisis	17/09/12	28/09/12	40										
3	Implementación	01/10/12	11/01/13	300										
4	Pruebas	14/01/13	25/01/13	40										
5	Documentación	28/01/13	15/02/13	60										
6	Seguimiento	18/02/13	18/06/13	15										
7	Memoria	18/02/13	12/04/13	160										

FIGURA 7.1: Diagrama de Grantt

Capítulo 8

Conclusión

8.1. Conclusiones

Tras la finalización del proyecto es momento de valorar los resultados y sacar conclusiones. Tal como se establecía en el primer capítulo de este documento el objetivo era crear un programa para la gestión de la trazabilidad alimentaria.

El principal objetivo era que fuese fácil de usar, debido a que los usuarios van a ser gente sin experiencia en informática, con ligeros conocimientos en navegación web. Este objetivo se ha cumplido con creces. Todas las funcionalidades del programa son disparadas a través de la página principal, no hay que pasar por ninguna otra ventana para llegar a lo que se quiera hacer. El programa, aparte de llevar el control sobre la trazabilidad alimentaria, genera una etiqueta por cada producto fabricado, lo que facilita el seguimiento de la trazabilidad hacia adelante.

En este documento se explica como instalar el proyecto en un entorno con Windows dentro de la empresa pero este programa, gracias a que está escrito en Java, se puede instalar en un servidor Linux, por ejemplo, consiguiendo bajar considerablemente el precio. También se propone al cliente instalarlo en un servidor externo para facilitar tareas de mantenimiento y disponibilidad, pero el cliente deniega de esta opción por los costes y por la necesidad de estar conectado a internet para su uso.

8.2. Futuros trabajos

Gracias a que hemos elegido un patrón de arquitectura de software basado en Modelo-Vista-Controlador la aplicación es facilmente ampliable e integrable en los distintos

sistemas de la empresa. A continuación se proponen una serie de futuros trabajos que se podrían realizar para mejorar esta aplicación y otras en la empresa.

- Aplicación para el repartidor: actualmente el repartidor de los productos alimentarios de la empresa va al lugar de destino y hace firmar al cliente en un papel el acuerdo de que ha recibido el producto. Se propone hacer una aplicación para smarthphone en la cual el cliente firmará y se guardarán los datos de cada cliente y cada pedido.
- Aplicación resumen de ventas: gracias a los datos guardados por la aplicación propuesta anteriormente se pueden conseguir todo tipo de datos y resúmenes de ventas. Facilitando la parte comercial y de gestión de la empresa.
- Sistema ERP: se propone ampliar las funciones del programa de trazabilidad alimentaria para llevarla a los diferentes áreas de la empresa, gestionando todos los recursos como pueden ser: recursos humanos, stock de productos y materias primas, control de calidad, contabilidad, etc.
- Sistema de ayuda: siempre que se tenga una duda utilizando la aplicación, sería de gran ayuda tener un "botón.[en] el que se pueda buscar cualquier consulta referente a dudas de utilización.

Bibliografía

[1] Definiciones: www.wikipedia.org

[2] Búsqueda de información: www.google.es

[3] Documentación MySQL: www.mysql.com

[4] Turner, David y Chae, Jinseok 2007-2009. Seguridad en aplicaciones web Java: web

[5] Botones Google+: http://code.brucegalpin.com/google-plus-ui-buttons/

[6] Información sobre la librería de javascript JQuery: www.jquery.com

[7] Información sobre JQueryUI, interfaz de usuario: www.jqueryui.com

[8] Latex, sistema de composición de textos: www.latex-project.org

[9] Documentación de Java: Java docs

[10] Documentación Apache FOP: Apache FOP

[11] Recursos gratuitos sobre HTML Y CSS: HTML CSS

[12] Página oficial Eclipse IDE: www.eclipse.org

[13] Información sobre trazabilidad alimentaria: Agencia Española de Seguridad Alimentaria y Nutrición

[14] Página oficial generador de código de barras Barcode4j: Barcode4j

[15] Información plugin de JQueryUI autocompletar: JqueryUI Autocomplete

[16] Germán Galeano Gil, Pablo Díaz Márquez, José Carlos Sánchez Alonso. Manual Imprescindible de HTML 4. Editorial Anaya, 2001.

[17] Baron Shuwartz, Meter Zaitsev, Vadim Tkachenko, Jeremy D. Zawodny, Arjen Lentz, Derek J. Balling. MySQL Avanzado (Segunda Edición). Editorial Anaya, 2009.

[18] Luis Joyanes Aguilar, Ignacio Zahonero Martínez. Programación en JAVA 2: Algoritmos, estructura de datos y programación orientada a objetos. Editorial MCGRAW-HILL / INTERAMERICANA DE ESPAÑA, 2002.