1395: Aplicación para la gestión de un sistema de incidencias de un almacén robotizado

Memoria del Proyecto
Ingeniería Informática
realizado por:
Jaume Mir Armada
y dirigido por :
Josep Maria Ganyet Cirera
Bellaterra, Septiembre de 2009

Agradecimientos

Antes comenzar quisiera dar las gracias a todas aquellas personas que han ayudado a lo largo de estos duros años de carrera.

Esta memoria va dedicada a mi abuelo, por la difícil enfermedad que le queda por afrontar a sus 80 años, y quizás también por el interminable número de veces que me ha preguntado que es un Ingeniero Informático, y que yo al parecer jamás se lo he sabido explicar para que él lo pudiera comprender, pese a eso se sentía orgulloso.

Debo empezar agradeciendo a mis padres Jaime e Inés toda la ayuda prestada durante estos años, sin ellos no hubiera sido capaz de lograrlo. A mi padre por todos los ánimos y buenos consejos que me ha dado para afrontar todos mis problemas. A mi madre tengo que darle un agradecimiento especial por todos los años perdidos persiguiéndome en mi adolescencia, para que finalmente cursara unos estudios, sin ella nada de esto hubiera sido posible ¡ gracias !

Seguidamente continúo por mi novia Nuria, por la paciencia que ha tenido durante estos años en mis momentos de mal humor y por toda la ayuda prestada en todo lo que ha podido, para facilitarme las cosas para llegar a conseguir esto.

No me puedo olvidar del resto de mi familia y especialmente de mi tío Jesús que siempre confió en mí y me dio ánimos.

Gran merito tienen mis compañeros de clase, mejor dicho amigos, especialmente Iván por todas las horas de estudio, de clase, de insomnio y risas. No me olvido de Josep, Arnau, Sojo, Pastís y a todos los de la banda del Min (Min, David, Sergio, Guillem) y tantos otros compañeros que siempre me han apoyado y ayudado, haciendo estos años de facultad más llevaderos e inolvidables.

Me gustaría también dar las gracias a mi compañero de trabajo Óscar por toda la ayuda prestada sin él este proyecto no hubiera sido lo que es.

Por último como no, me gustaría agradecer a Josep María Ganyet, toda la ayuda y facilidades que me ha brindado para hacer este proyecto, hoy en día es difícil encontrar profesores así.

Índice

1. INTRODUCCIÓN 2

1.1. MOTIVACIÓN DEL PROYECTO 2
1.2. OBJETIVOS DEL PROYECTO 3
1.3. ESTADO DEL ARTE 5
1.4. ¿QUÉ ES UN SISTEMA DE INCIDENCIAS? 5
1.4.1. El sistema incidental 5
1.4.2. Las ventajas de un sistema de incidencias son: 6
1.4.3. Elementos que garantizan la efectividad de un Sistema de incidencias: 7
1.4.4. La gestión de incidencias como elemento de motivación: 8
1.5. ESTRUCTURA DE LA MEMORIA 8
1.6. PLANIFICACIÓN DEL PROYECTO 9

2. ANÁLISIS DE REQUERIMIENTOS 11

2.1. INTRODUCCIÓN 11
2.1.1. Propósito 11
2.2. DESCRIPCIÓN GENERAL 11
2.2.1. Descripción del problema 11
2.3. FUNCIONES 13
2.4. REQUERIMIENTOS ESPECÍFICOS 13

3. DISEÑOS 16

3.1. DIAGRAMAS DE FLUJO 16
3.1.1. Identificación usuarios 16
3.1.2. Módulo Incidencias 17
3.1.3. Módulo Estadísticas 18
3.1.4. Módulo Materiales 19
3.1.5. Módulo usuarios 20
3.1.6. Facturación 21
3.1.7. Gestión Cuenta 22
3.1.8. Ciclo de vida de una incidencia 23
3.2. DISEÑO DE LA BASE DE DATOS. 24
3.2.1. Diseño conceptual 24
3.2.2. Diseño Físico 31

4. TECNOLOGÍAS UTILIZADAS 32

4.1. ARQUITECTURA DISTRIBUIDA 33
4.1.1. Java 2 Enterprise Edition 33
4.1.2. El modelo de desarrollo de J2EE 33
4.2. APACHE ANT 35
4.3. MCV (MODELO VISTA CONTROLADOR) 35
4.3.1. ¿Qué es Java Server Faces? 35
4.3.2. ¿Cómo funciona JSF? 37
4.3.3. Elementos principales que constituyen una aplicación JSF 37
4.3.4. Los backbeans 37
4.3.5. Estructura de las páginas 37
4.3.6. Richfaces 39
4.4. HTML 41
4.5. XML 41
4.6. ECLIPSE 41
4.7. BASE DATOS. 42

4.7.1. ¿Qué es PostGreSQL? ... *42*

5. IMPLEMENTACIÓN ... **43**

5.1. PÁGINAS DE LA APLICACIÓN ... 43
5.1.1. Index.jsp ... *43*
5.1.2. Inicio.jsp ... *43*
5.1.3. moduloCreacionIncidencias.jsp ... *44*
5.1.4. moduloConsultaIncidencias.jsp ... *45*
5.1.5. moduloEdicionIncidencias.jsp ... *46*
5.1.6. moduloEstadisticas.jsp ... *47*
5.1.7. moduloMateriales.jsp ... *48*
5.1.8. admonUsuarios.jsp ... *50*
5.1.9. admonPerfiles.jsp ... *51*
5.1.10. moduloFacturacion.jsp ... *53*
5.1.11. miCuenta.jsp ... *55*

6. TEST DE LA APLICACIÓN ... **56**

6.1. TEST FUNCIONAL ... 57
6.1.1. Test funcional perfiles ... *57*
6.2. TEST DE CARGA ... 59
6.2.1. Test de carga 1 ... *59*
6.2.2. Test de carga 2 ... *61*
6.2.3. Comparativa entre ambas pruebas ... *62*

7. CONCLUSIONES ... **63**

7.1. MEJORAS FUTURAS ... 64

8. ANEXO 1 ... **65**

8.1. CÓDIGO FUENTE ... 65
8.1.1. Un ejemplo de código ... *65*
8.1.2. Envio de correo para notificar incidencias ... *70*

9. ANEXO 2 ... **72**

9.1. EJEMPLOS DE QUERYS UTILIZADAS ... 72
9.1.1. Query para Insertar datos de la persona nueva en tabla personas ... *72*
9.1.2. Query para devuelve los datos de la persona que coincida con el id correspondiente ... *72*
9.1.3. Query para modificar un material ya existente en nuestra base de datos ... *72*
9.1.4. Query para el cálculo del numero de incidencias para mostrar las estadísticas ... *72*
9.1.5. Query para eliminar un material de la base de datos ... *73*
9.1.6. Query para eliminar un material de la base de datos ... *73*
9.2. CONEXIÓN CON LA BBDD ... 73
9.2.1. DBConfig.properties ... *74*

10. BIBLIOGRAFIA ... **75**

11. LINKS EXTERNOS ... **75**

Índice Figuras

Figura 1 - El sistema incidental 6
Figura 2 – Efectividad del sistema 7
Figura 3– Planificación 10
Figura 4 – Identificación Usuarios 16
Figura 5 – Módulo de incidencias 17
Figura 6– Módulo estadísticas 18
Figura 7 – Módulo Materiales 19
Figura 8 – Módulo Usuarios 20
Figura 9 – Módulo Facturación 21
Figura 10 – Módulo Gestión Cuentas 22
Figura 11– Ciclo de vida de una incidencia 23
Figura 12 – Diagrama E-R 30
Figura 13 - Diagrama Físico 31
Figura 14 – Patrón MCV 36
Figura 15 – Módulo Flujo JSF 36
Figura 16– Etapas procesamiento JSF 38
Figura 17– Etapas procesamiento Richfaces 40
Figura 18 –Componentes Richfaces 41
Figura 19 –Index 43
Figura 20- Inicio 44
Figura 21- Crear Incidencias 45
Figura 22- Consulta incidencias 46
Figura 23- Modificación Incidencias 47
Figura 24- Estadísticas 48
Figura 25- Modificar Material 49
Figura 26- Crear Material 49
Figura -27 Eliminar Material 50
Figura 28- Operaciones usuarios 51
Figura 29- Operaciones perfiles 52
Figura 30- Crear perfiles 52
Figura 31- Consulta facturas 53
Figura 32- Crear facturas 54
Figura 33- Gestión cuenta 55
Figura 34- Test 56
Figura 35 - fases testing 56
Figura 36- Tmap 57

1. Introducción

Actualmente, en nuestro día a día experimentamos un gran número de cambios debidos a la gran evolución que están teniendo las nuevas tecnologías, y por ello cada vez más las utilizamos para facilitar nuestra vida cotidiana y mejorar los procesos que gestionamos.

De la misma manera esto también afecta de forma muy rápida a los procesos del mundo laboral, donde se utiliza la Informática como herramienta de solución para poder automatizar aquellos procesos que anteriormente se realizaban manualmente, aumentando la eficiencia del trabajo, y facilitando las funciones desempeñadas a los usuarios, ahorrando costes, y optimizando los procesos para obtener los mejores beneficios.

El presente documento pretende esbozar a grandes rasgos el propósito de demostrar que automatizando un trabajo que se realizaba de forma manual aumentamos la eficiencia del proceso, la velocidad, y a su vez abaratamos costes y evitamos cuellos de botella innecesarios.

1.1. Motivación del proyecto

La motivación que me ha llevado a la elección de este proyecto viene determinada por diversos factores:

- En primer lugar, he considerado muy enriquecedor para concluir esta etapa de mi formación, el hecho de poder realizar un proyecto final de carrera en una empresa, ya que gracias a ello se me permitía poder ver el **ciclo vital del proyecto**, desde el estudio de los requerimientos hasta la puesta en producción.

- **A nivel de Ingeniería de Software** me permite poder dar una solución a un problema que nos plantea un cliente: Debemos realizar un estudio de la situación actual;
  Una vez realizado nos solicitan crear una aplicación que automatice todo el proceso.

- **Técnicamente** me ha servido como motivación para poder aprender nuevos lenguajes, ya que por parte de la consultora me solicitaban realizar el proyecto con Java Server Faces (JSF) con el entorno eclipse, -lenguaje de programación que desconocía y que me servirá mucho en mi futuro laboral a corto plazo-. También creo que servirá de mucho tratar con las herramientas que he necesitado para hacerlo y los diferentes entornos de programación como PotGresSQL, Eclipse, JSF, etc.

- Para afianzarme en mi puesto de trabajo y poder dejar a un lado el marco teórico de las asignaturas y por tanto trabajar más la parte **práctica**.

1.2. Objetivos del Proyecto

La pretensión de este proyecto es crear una aplicación que sirva como solución para el control incidental y de los costes que generan la resolución de estas. Se me plantea un doble reto para realizar el proyecto:

Desde mi empresa consultora se me pide un proyecto modular que pueda ser reutilizable para futuros clientes y por otro lado que simultáneamente satisfaga cada una de las necesidades que nos plantee el cliente.

Los principales objetivos son:

1. La creación de una aplicación modular que pueda ser reutilizable para futuros cliente de la empresa consultora.

2. Esta aplicación debe registrar todas las incidencias, y de esta manera sirva como nexo de unión comunicativa entre los departamentos de operadores de almacén y los mecánicos. Esta aplicación es un encargo del Directivo de la empresa, así que funcionalmente deberá satisfacer todas las necesidades que nos indique el cliente.

3. Esta aplicación solo debe ser accesible dentro de la empresa, no es necesario tener un acceso desde el exterior. Por tanto el servidor que contendrá la aplicación, solo será accesible desde la red interna de la empresa.

4. La aplicación debe permitir de forma segura la conexión de diferentes usuarios. Cada uno de estos usuarios tendrá diferentes permisos según el trabajo que desempeñen. Se deben conectar con un usuario y password.

5. Solo el usuario Administrador tendrá capacidad para poder crear usuarios nuevos.

6. La creación de diferentes perfiles con adjudicación de su permiso de forma concreta.

7. Debe poseer un módulo donde se puedan dar de alta materiales, con los que se solucionarán las incidencias, almacenando el valor de su coste y la cantidad de elementos de que disponemos, para poder tener un control óptimo de Stock

8. Dar a los técnicos de mantenimiento, una herramienta necesaria para poder indicar que elementos han utilizado para resolver las incidencias.

9. Se debe implementar un apartado desde el que se podrán generar informes con valor estadístico de las incidencias, así como de las soluciones para poder realizar por parte de los directivos estudios, sobre cómo mejorar el sistema de trabajo, y de cómo obtener información sobre los problemas que se tienen diariamente, como evitarlos, y que no vuelvan a producirse para una mejor calidad del trabajo

10. El cliente nos solicita una funcionalidad que solo podrá gestionar el Administrador del Sistema con la que una vez se hayan solucionado las incidencias, se puedan generar las facturas, y calcular los costes de la operación dependiendo de los materiales utilizados por los técnicos Nivel.

11. Se debe tener todo almacenado en una Base de datos donde se relacionen todos los elementos que forman parte del proyecto: usuarios, incidencias, materiales, facturas, etc…

12. Se deben poder escalar las incidencias y asignarlas de unos usuarios a otros, cambiar el estado y la severidad de las mismas.

13. Se debe notificar a los diferentes usuarios relacionados con la misma incidencia, tanto la apertura, como las posibles modificaciones hasta el cierre de la misma por correo electrónico. Este procedimiento es importante ya que los mecánicos llevarán una PDA con conexión a internet y acceso a su cuenta de correo y de esta manera se les notifica la incidencia instantáneamente y así podrá tomar la decisión de si es prioritario o no.

14. Hay que tener en cuenta que los usuarios de este tipo de aplicaciones no están muy familiarizados con las herramientas informáticas por lo que hay que crear una aplicación amigable, intuitiva, ágil y fácilmente utilizable.

15. A petición del director del proyecto, se nos propone que la aplicación sea visualmente lo más atractiva posible.

Por último, me gustaría resaltar que el objetivo principal del proyecto es su funcionalidad como solución para que, de una forma estructurada se realice el trabajo y le facilite la tarea tanto a los mecánicos, como a los operadores de almacén, para poder realizar los pedidos de los clientes en el menor tiempo posible, evitando las pérdidas de incidencias con el consecuente retraso que ocasiona si se trata de un elemento indispensable para poder concluir un pedido.

Se debe satisfacer la necesidad de la empresa consultora para la creación de la aplicación modular, con el fin de ser reutilizable para futuros clientes con el menor número de cambios posibles, es decir ha de ser una aplicación reutilizable.

1.3. Estado del arte

En los últimos años Internet, se ha convertido en la fuente principal de información, tanto de las empresas, como entre los propios particulares.
Todo esto es debido al fácil acceso y a la cantidad de información de la que se dispone para poder cumplir nuestros objetivos.

La aplicación solicitada será diseñada para ayudar y asegurar la calidad de la reparación de las incidencias de nuestro almacén robotizado y evitar que se produzcan cuellos de botella en el suministro de los pedidos de los clientes.

Tras la búsqueda de información, concluimos que actualmente hay herramientas que sirven para la gestión de incidencias y que son muy utilizadas en los procesos de testing para registrar los diferentes defectos en el Sotfware en las aplicaciones, Bug Tracking System (BTS)

Podemos destacar entre ellos el **sistema de Mantis**, uno de los más utilizados con una gran cantidad de funcionalidades y de software libre, aunque cabe destacar alguna de las deficiencias de este sistema que lleva a una falta de satisfacción del cliente ya que no ve cubiertas las necesidades solicitadas, puesto que el sistema Mantis no sirve como un elemento de control de los materiales utilizados para la solución de incidencias.

1.4. ¿Qué es un sistema de incidencias?

Resumiendo podemos decir que un Sistema de Incidencias Operacionales, está concebido para el registro de los problemas que ocurren durante el desarrollo del trabajo para mantener una comunicación directa entre los diferentes departamentos para llegar a su solución. También es utilizada por los altos directivos de la empresa con vistas a la determinación de las causas que motivan su presencia con el objetivo de establecer acciones concretas que contribuyan a su erradicación.

1.4.1. El sistema incidental

La organización es el proceso de combinar los medios e instrumentos: tecnológicos, lógicos, informativos, técnicos, materiales y humanos, todo ello con la finalidad de lograr un objetivo: solucionar un problema.

La organización en esencia es un modelo del sistema de dirección construido sobre la base de los principios y técnicas del enfoque sistémico, cuyo campo de trabajo dentro

de los sistemas socioeconómicos es muy amplio y debe estar regulado por los procedimientos correspondientes.

Cabe destacar la importancia de la actividad comercial internacional, en cuyo desarrollo y expansión, la calidad y la atención sistemática juegan un papel fundamental y prioritario, señalando la relevancia de disponer de un sistema de registro y análisis de cualquier situación que pueda afectar a una operación comercial, es decir **controlar las incidencias.**

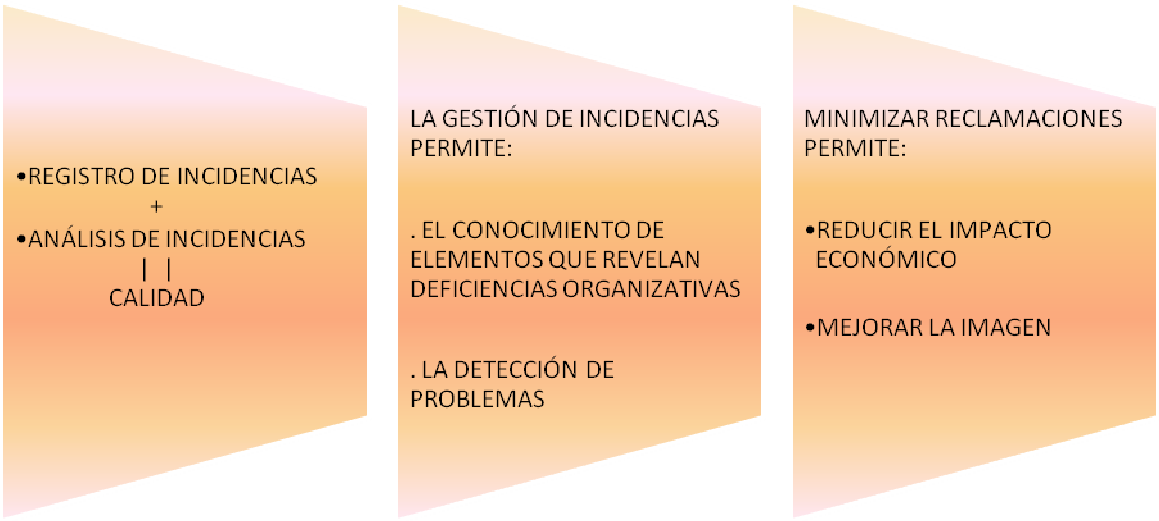

Figura 1 - El sistema incidental

El objetivo de un sistema de gestión de incidencias es facilitar y estandarizar el acceso de forma fiable a este tipo de información, ofreciendo al primer nivel de dirección de esta manera, facilidad para analizar y tomar decisiones que perfeccionarán el funcionamiento de la empresa.

1.4.2. Las ventajas de un sistema de incidencias son:

- Determinar los fallos de las operaciones de la empresa y las causas que los motivan, así como conocer el origen de la incidencia que permite erradicarla y con ello incrementar la eficiencia

- Conocer óptimamente la calidad del producto (a nivel funcional y económico) facilita la elaboración de un plan de negocio, así como la venta al cliente externo del mismo.

1.4.3. Elementos que garantizan la efectividad de un Sistema de incidencias:

Garantizar la sencillez de la información primaria y su comunicación:

- Recursos óptimos en cuanto al personal que gestiona la incidencia, medio de transmisión y lugar de emisión
- Definición diferenciada de áreas de responsabilidad
- Contenido de la incidencia (uniformidad de los conceptos que permita su tratamiento estadística y funcional)

Garantizar la diseminación selectiva de la información y el control en la gestión:

- Controlar el coste de la no calidad
- Identificar las caudas que originan las incidencias y la toma de decisiones para gestionarlas

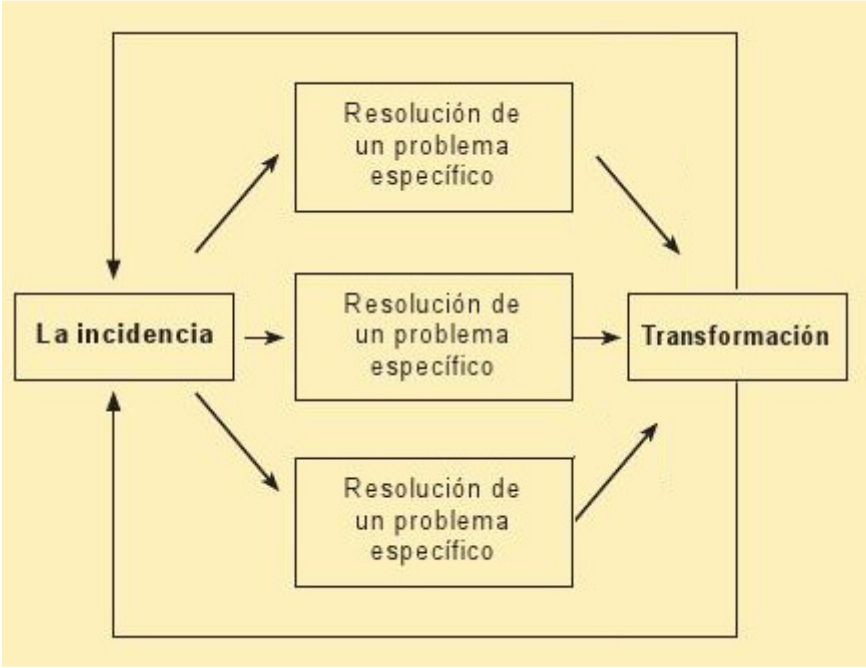

Figura 2 – Efectividad del sistema

1.4.4. La gestión de incidencias como elemento de motivación:

La gestión de incidencias contribuye a incrementar la disciplina a nivel organizativo ya que permite que los empleados tengan la percepción de controlar y supervisar su trabajo. Por otro lado permite formar al nuevo personal en base a la experiencia que conlleva el conocimiento en la gestión.
Finalmente destacaré que la percepción de un empleado ante el compromiso empresarial y ante la gestión de incidencias contribuye a incrementar el sentimiento de pertenencia a la organización.

1.5. Estructura de la memoria

Esta memoria está dividida en seis secciones independientemente de los apéndices, donde se muestran elementos de interés dentro de nuestro proyecto.

Descripción de las secciones:

Sección 2 Análisis de requerimientos: En esta sección se describen cuales son los requerimientos que nuestra aplicación debe cumplir, que necesidades debe satisfacer para poder llegar a la solución final.

Sección 3 Diseños: Una vez estudiados los requerimientos, se realizan los diseños de las funcionalidades del aplicativo. También en esta sección se detalla el diseño conceptual de la base de datos, las tablas y el modelo de datos.

Sección 4 Tecnologías utilizadas: En este apartado se tratarán las diferentes tecnologías utilizadas para poder realizar nuestro proyecto. Se hará una descripción de cada una y de cómo funciona. También se explica porqué se optó por cada una de las tecnologías utilizadas.

Sección 5 Implementación: En este apartado vemos el funcionamiento de la aplicación, así como que parte del código la representa. Se pondrán imágenes de la aplicación con el fin de entender mejor el funcionamiento de esta.

Sección 6 Test: En esta sección se describen las pruebas que se han realizado para comprobar que nuestra aplicación funciona correctamente, se han realizado tanto pruebas funcionales como pruebas de carga para asegurar el correcto funcionamiento del sistema de incidencias.

Sección 7 Conclusiones: En este punto se realizará una valoración final de la aplicación, haciendo una reflexión sobre los objetivos logrados en el proyecto así como de que aspectos nos han quedado por cumplir. Por último se explicarán posibles vías de mejora.

1.6. Planificación del proyecto

Antes de iniciar el proyecto es necesario que se realice un estudio que nos permita de una forma aproximada, poder planificar el tiempo necesario para realizar el trabajo, y así poder ofrecer al cliente una fecha de entrega del producto que en un futuro no tenga muchas desviaciones.

Para realizar esta planificación es necesario tener claras cuales son las diferentes fases por las que ha de pasar un proyecto y que se debe realizar en cada una de ellas. De esta forma, teniendo claro el marco teórico del procedimiento que se ha de realizar, se puede llegar a hacer una planificación mucho más exacta y real, evitando demoras los entregables del producto y cumpliendo con los plazos estipulados con el cliente.

Las fases de las que se compone el Proyecto que deseamos desarrollar se pueden dividir en 4 grandes fases.

1. **Arranque proyecto :** Corresponde con las primeras semanas del proyecto los esfuerzos estarán enfocados básicamente a mantener las reuniones necesarias con el cliente para poder, recoger todos los requerimientos que este necesite para satisfacer sus necesidades.
   Una vez recogidos los requerimientos que deseará nuestro cliente, necesitaremos realizar el estudio de las herramientas tecnológicas que podrán satisfacer sus necesidades y si es preciso comienzo de la formación del personal en estas.

2. **Implementación y pre test:** Después de haber definido el funcional de nuestra aplicación y teniendo claros los requerimientos que nos solicitó el cliente, comienza la fase de implementación de la solución. Esta será la fase que más esfuerzo requiere en todo el proyecto.

   Paralelamente con el funcional se deben definir los test scripts que deberá pasar la aplicación cuando ya se tenga una versión estable, para asegurar que se cumplen todos los requisitos especificados en el funcional.

3. **Preparación del prototipo y test de integración:** Una vez se tiene una versión estable comienza el test de la aplicación. Se han de pasar los test scripts funcionales para validar nuestro producto. También se irán solucionando las incidencias hasta llegar a nuestra versión final.

   Una vez tenemos la versión final se realiza la presentación al cliente del prototipo y se fija un día para su puesta en producción.

4. **Puesta en producción:** En este parte se realizan las formaciones a los usuarios de la aplicación y por último se instala el producto y se pone en funcionamiento en las instalaciones del cliente.

La gran mayoría de las veces se producen desviaciones de esta planificación teórica por los diferentes problemas que surgen sobre todo durante la fase dos del proyecto. Por eso es más que necesario que en otras realizar una buena estimación del tiempo, siendo consciente de los recursos que se posee y teniendo claro cuándo debe comenzar el testing para poder corregir los errores antes de la puesta en producción.
Cuanto antes se empiece el testing de la aplicación, más fácil será completar los objetivos marcados para llegar a realizar los entregables y minimizar los retrasos que se puedan introducir.

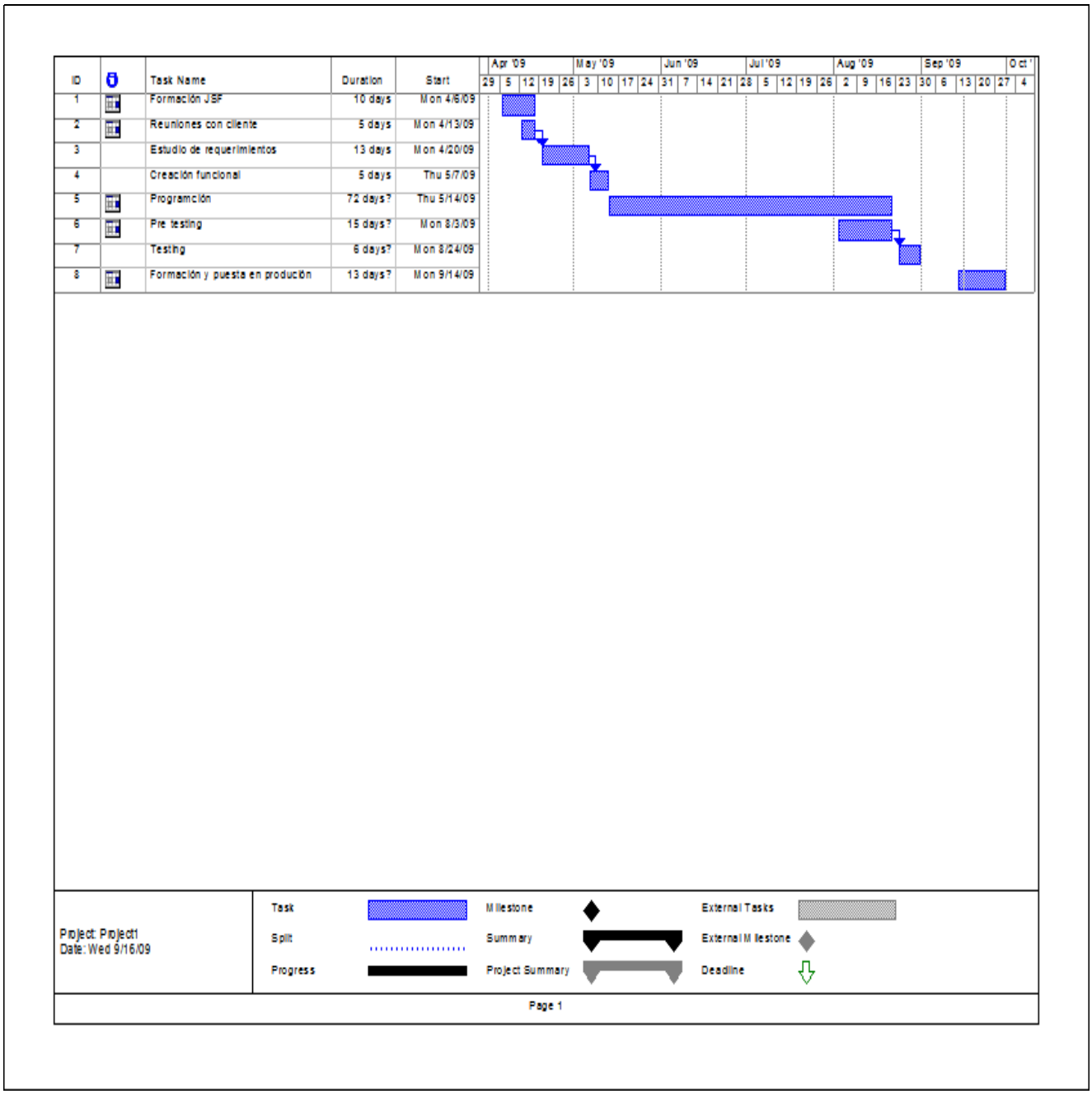

Figura 3– Planificación

2. Análisis de requerimientos

2.1. Introducción

2.1.1. Propósito

En el siguiente análisis de requerimientos, el principal objetivo es, poder definir todos los aspectos que serán necesarios para la creación de la aplicación que se quiere desarrollar, cubriendo todas las necesidades solicitadas por el cliente.

2.2. Descripción General

2.2.1. Descripción del problema

El contexto del problema se sitúa en una empresa. que dedica su actividad al almacenamiento y distribución de tornillería inoxidable.
Los Stocks se distribuyen desde su central, que es a su vez es el almacén, Desde allí se sirven los pedidos a los clientes.

Una vez se ha realizado un pedido por parte de un cliente, el departamento de ventas pasa la orden a los operadores del almacén para que entreguen el pedido en el menor tiempo posible.

El almacén está compuesto de los elementos genéricos (cintas transportadoras, grúas mecánicas, toros mecánicos…) y de los específicos del cliente. Son los que marcan la distinción del cliente permitiéndoles ser líderes en su sector por su rapidez en el servicio (tres almacenes robotizados , controlados por un sistema informático)

Lógicamente este sistema es complejo y obliga a la empresa a disponer de un alto número de maquinaria para poder satisfacer los pedidos, en consecuencia tienen un alto índice de averías diarias:

- Estas averías son detectadas por parte de los operadores del almacén que reportan por vía telefónica al servicio de mantenimiento la incidencia
  El técnico apunta la incidencia a mano y dependiendo de la urgencia de la misma deja sus tareas o la archiva en una cola para ser atendida más adelante; Esto ocasiona **pérdidas de información** por las escasas notas tomadas y cuando llega el momento de solucionarla los mecánicos no tienen toda la información necesaria sobre la misma.

Normalmente los técnicos de mantenimiento no suelen estar en la oficina ya que tienen un volumen importante de trabajo y suelen estar ocupados solucionando incidencias o realizando intervenciones ya programadas en el robot. Cuando estos no se encuentran en la oficina, la secretaria toma nota de la incidencia, y cuando algún técnico vuelve a la oficina le pasa la información que anotó. El principal problema es que muchas veces estas notas se pierden ya que si la secretaria se marcha antes de ver a algún técnico no le pasa la información. Por otro lado y debido a los pocos conocimientos técnicos de esta persona esta **información no es precisa**.

- Otro de los graves problemas es que no hay ningún **control del Stock** de los materiales disponibles en el almacén y esto provoca que algunas veces tenga que estar parada alguna máquina ya que no se posee la pieza necesaria para solventarlo.

- No se tiene ningún **control de la solución** de las incidencias ni del tiempo que se tarda en solucionar las mismas.

- Al haber dos turnos de operadores y mecánicos, estos se transmiten la información de la situación de las distintas averías de viva voz, lo que provoca confusiones y una gran **pérdida de información.**

Todo este sistema rudimentario y deficiente provoca a nuestro cliente un cuello de botella, a la hora de entregar los pedidos, ya que con las breves notas tomadas a mano se producen las siguientes incidencias:

- Pérdida de información
- No se tiene ningún registro incidental
- No se tiene control del tiempo desempeñado por el técnico para solucionar la incidencia
- Pérdida del control de los materiales necesarios
- Desconocimiento del coste de la reparación
- Ausencia de un correcto sistema de priorización de incidencias que provoca averías que son bloqueantes para la salida del producto y que causa una gran demora para solucionarlas, ya que el mecánico es quién decide la prioridad de una avería ante otra.

El principal afectado de todo esto es la propia empresa ya que provoca demoras en la entrega de los pedidos a sus clientes.

Todo ello hace que exista la necesidad de la creación de una aplicación que realice el control incidental, donde se facilite el control de los stocks, costes de las resoluciones de las incidencias y a su vez mejore la comunicación entre los departamentos de finanzas, mantenimiento y logística.

2.3. Funciones

- Área de identificación
- Diferenciación de los diferentes perfiles de usuarios que utilizarán la aplicación.
    1. Súper Usuario
    2. Director Financiero
    3. Director logístico
    4. Mecánico Segundo Nivel
    5. Mecánico Primer Nivel
    6. Operador de Almacén
- Área de creación de incidencias
- Área de edición de incidencias
- Área de consulta de incidencias
- Área de las estadísticas de las incidencias.
- Gestión de materiales.
- Gestión de facturación.
- Gestión de los diferentes perfiles de usuarios.

2.4. Requerimientos específicos

La aplicación tendrá 6 perfiles de usuarios diferentes, Súper Usuario, Director financiero, operador de almacén, mecánico primer nivel, mecánico segundo nivel, y director logístico.

Cantidad de gestores de la aplicación:

1 Súper Usuario
1 Administrador
1 Jefe logística
N perfiles cada uno de los cuales identificará que privilegios tiene en la aplicación cada usuario dependiendo al grupo al que pertenecen.

Los usuarios los dará de alta el Súper Usuario o el Director financiero de la aplicación. El número de perfiles puede crecer o disminuir dependiendo de las necesidades del momento que viva la empresa.
Desde la parte de la gestión de perfiles se permite crear o eliminar los mismos indicando los privilegios que tendrán dentro de la aplicación.

En un primer momento, al inicio de la aplicación, solo existirá el Super Usuario. Desde este perfil se creará el resto de perfiles de que dispondrá la aplicación

- **Súper Usuario:**

  Este es el perfil inicial de la aplicación. Se inserta directamente en la base de datos cuando se crean y posee privilegios máximos sobre todos los módulos de la aplicación.
  Este perfil es el del programador y se utiliza para la instalación de la aplicación en el servidor, y para la detección de posibles errores en la misma así como para hacer el testeo.

  El Super Usuario puede crear incidencias, editar incidencias, consultar incidencias, consultar estadísticas, introducción de materiales, crear facturas, consultar facturas, crear perfiles, consultar perfiles, editar perfiles, eliminar perfiles, crear usuarios, editar usuarios, eliminar usuarios y gestión de los datos de su propia cuenta.

- **Director Financiero:**

  Este perfil es el del Administrador de la aplicación;
  Es el encargado de la gestión de los usuarios, sus perfiles, introducción de los materiales, generación de las facturas y además es a quién le puede interesar más las estadísticas para sacar conclusiones sobre el funcionamiento del sistema.

  Puede crear incidencias, editar incidencias, consultar incidencias, consultar estadísticas, introducción de materiales, crear facturas, consultar facturas, crear perfiles, consultar los perfiles, editar perfiles, eliminar perfiles, crear usuarios, editar usuarios, eliminar usuarios y gestionar los datos de su propia cuenta.

- **Director Logístico:**

  Este perfil vendría ser el perfil del jefe de los operadores y de los mecánicos, su principal función es controlar la buena evolución de las incidencias y supervisar el trabajo de los mecánicos y de los operadores. Dentro de la aplicación es el encargado de controlar las incidencias y las actuaciones que se realizarán. Se encargará de poner las incidencias en estado cerrado una vez haya comprobado que el mecánico ha solucionado correctamente la incidencia.

  Puede crear incidencias, editar incidencias, consultar incidencias, consultar estadísticas, introducción de materiales, crear perfiles, gestión de los datos de su propia cuenta.

- **Mecánico Segundo Nivel:**

  Este perfil correspondería a los mecánicos oficiales de primera, quienes se encargan de solucionar las incidencias correspondientes al robot, también se encargan de los trabajos más complicados que no sean capaces de solucionar los mecánicos de primer nivel. Podrá solucionar las incidencias indicando que materiales ha utilizado.

  Puede crear incidencias, editar incidencias, consultar incidencias, consultar estadísticas y gestionar los datos de su propia cuenta.

- **Mecánico Primer Nivel:**

  Este perfil correspondería a los mecánicos, oficiales de segunda y los aprendices, quienes se encargan de solucionar las incidencias correspondientes al almacén, toros, puentes grúa, etc... Si estos no pueden solucionar alguna incidencia, la escalan a los mecánicos de primer nivel.

  Pueden crear incidencias, editar incidencias, consultar incidencias, consultar estadísticas y gestionar los datos de su propia cuenta.

- **Operador de Almacén:**

  Este perfil correspondería a los usuarios de la maquinara del almacén quienes se encargan de llevar a cabo todos los pedidos, son quienes interactúan directamente con las máquinas y son los que detectan las incidencias en el robot o en los diferentes elementos del almacén.

  Pueden crear incidencias, editar incidencias, consultar incidencias, consultar estadísticas y gestionar los datos de su propia cuenta.

3. Diseños

En este apartado vamos a describir los diferentes diseños que se han creado para poder realizar este proyecto; Se pueden ver dos partes muy diferenciadas la primera corresponde a los diagramas de flujo, donde se detalla la funcionalidad de los diferentes módulos, y la segunda es el diseño de la base de datos.

3.1. Diagramas de flujo

Tal como hemos explicado, se va a detallar en este aparatado la funcionalidad de la aplicación:

3.1.1. Identificación usuarios

En este apartado lo que se realiza, es la identificación de los usuarios en la aplicación; una vez informado el nombre de usuario y la contraseña, se cargan en la aplicación todos los módulos que corresponden al usuario identificado correspondientes a su perfil. En caso contrario se obtiene un error indicando que el usuario no es válido.

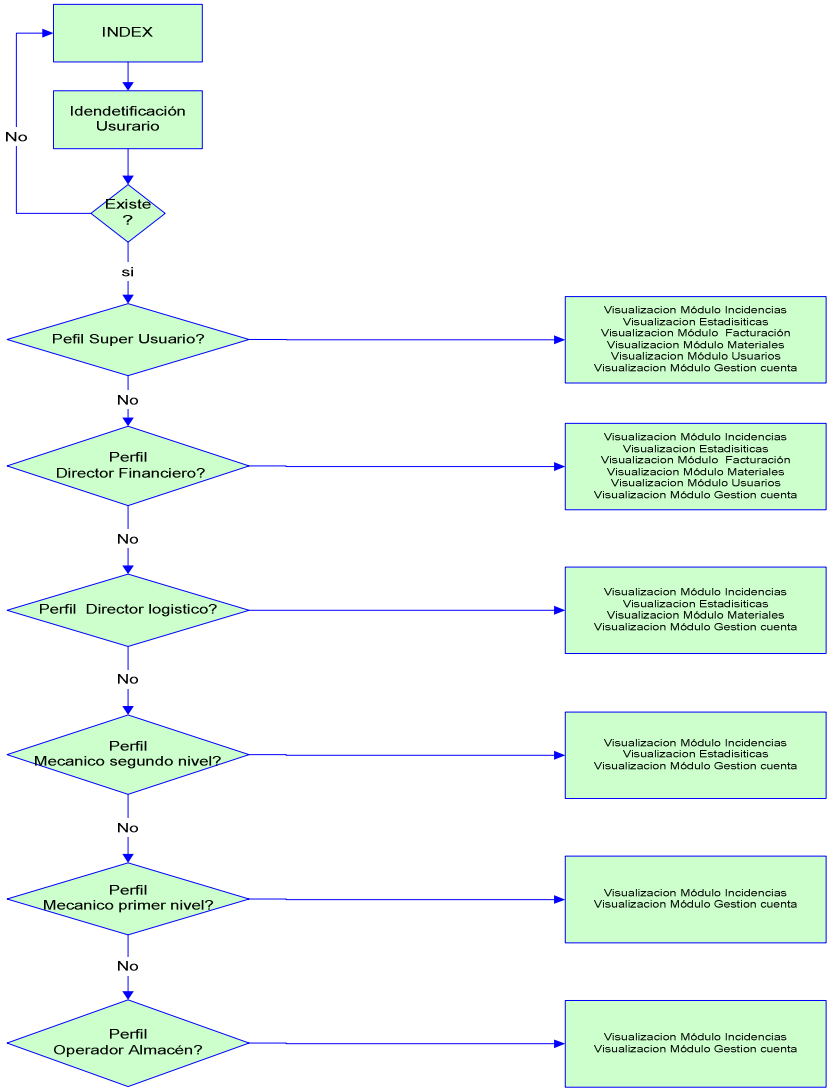

Figura 4 – Identificación Usuarios

3.1.2. Módulo Incidencias

Este módulo nos permite realizar las diferentes operaciones con las incidencias crear, modificar y consultar.

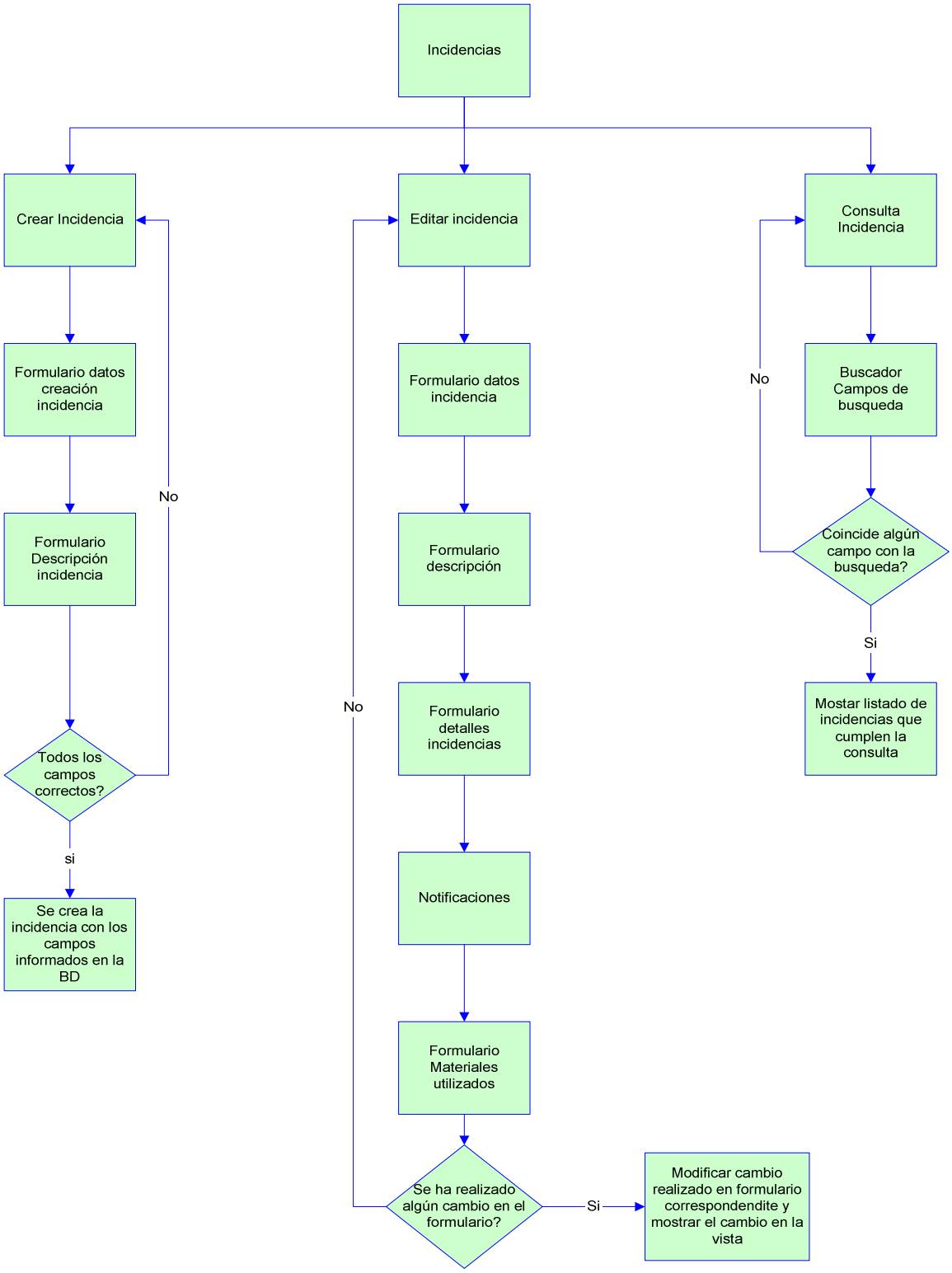

Figura 5 – Módulo de incidencias

3.1.3. Módulo Estadísticas

Este módulo nos da toda la información sobre los datos introducidos en el sistema a nivel incidental. Nos permite saber cuál es la situación y extraer información sobre el funcionamiento del sistema.

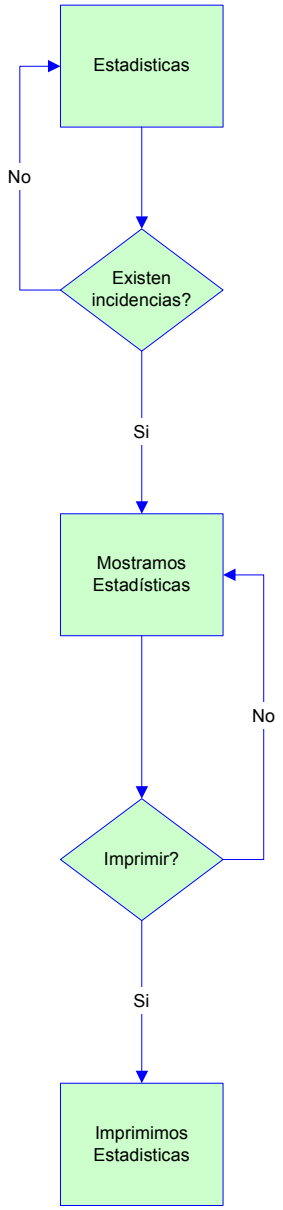

Figura 6– Módulo estadísticas

3.1.4. Módulo Materiales

Este módulo está diseñado para gestionar los materiales disponibles en el almacén, para solucionar las incidencias y poder tener un control del Stock de estos.

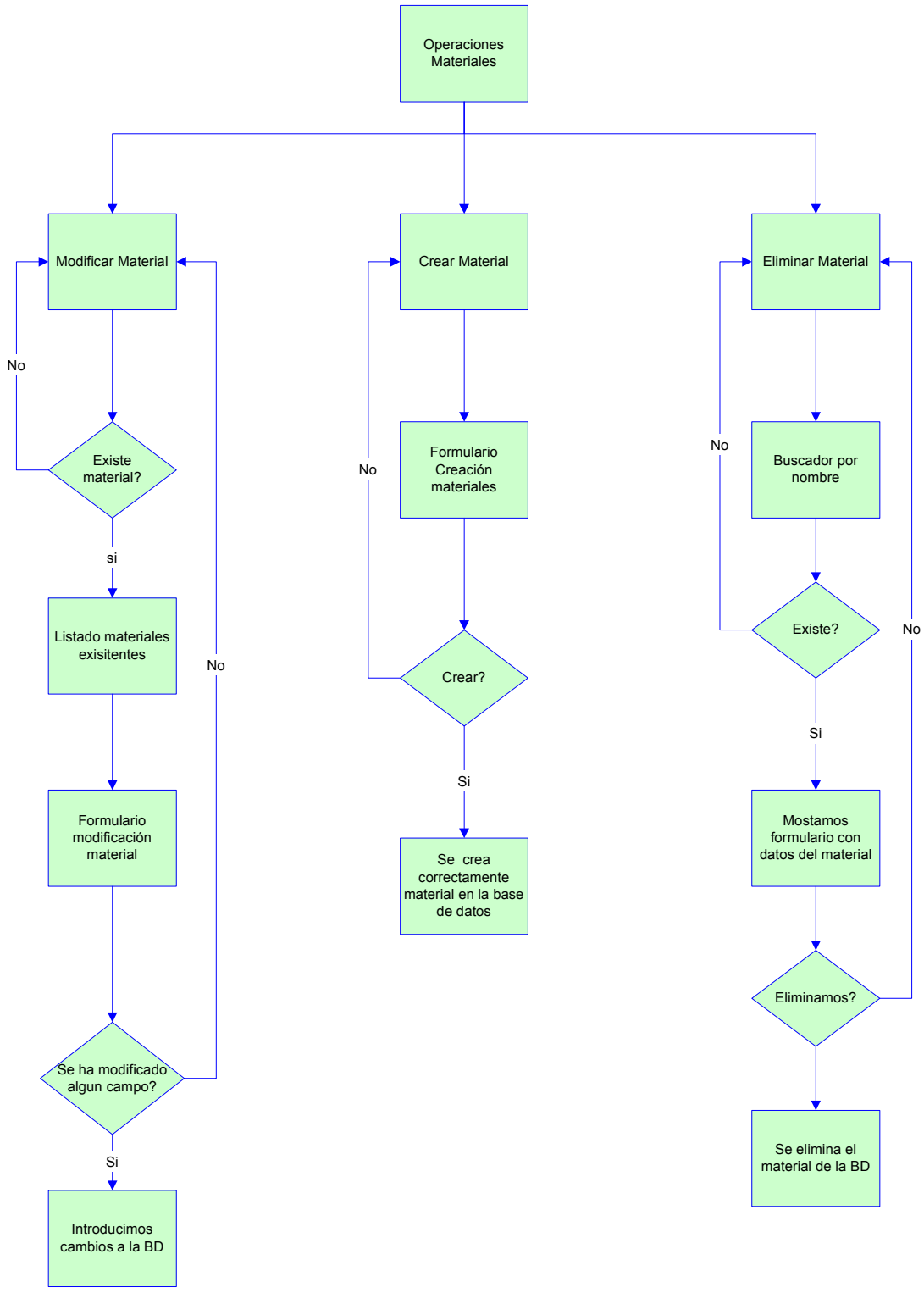

Figura 7 – Módulo Materiales

3.1.5. Módulo usuarios

Este módulo está diseñado para la gestión de los usuarios, tanto para la creación de las nuevas cuentas dentro de la aplicación, como para la gestión de los diferentes perfiles que pueden tener estos usuarios.

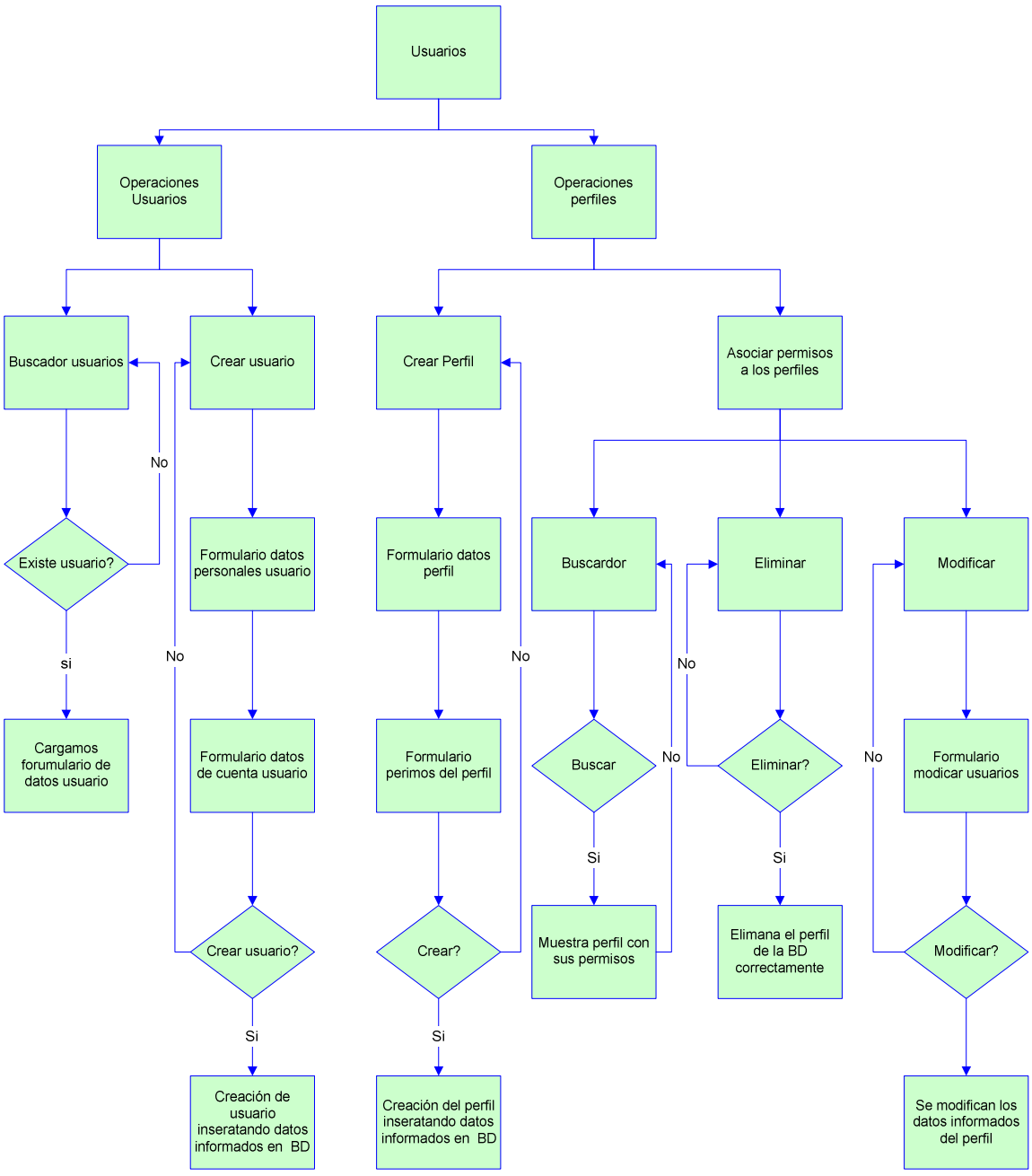

Figura 8 – Módulo Usuarios

3.1.6. Facturación

Este módulo está pensado para la gestión de los costes de cualquier incidencia que se ha solucionado, consulta de las facturas así como la edición de las mismas.

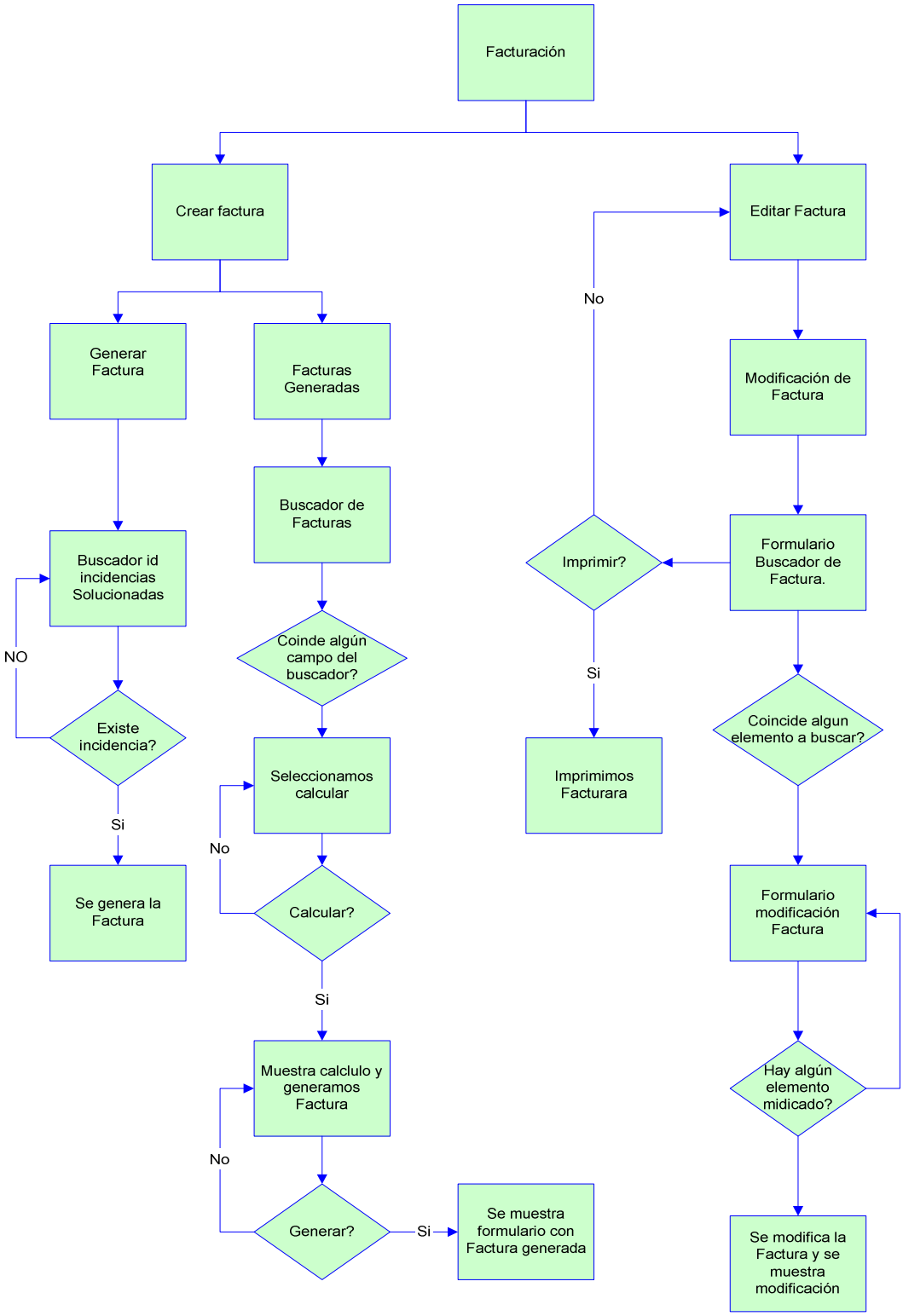

Figura 9 – Módulo Facturación

3.1.7. Gestión Cuenta

Este módulo está diseñado para la gestión propia de los usuarios, tanto de sus datos personales, como de los datos de su cuenta, tales como contraseñas, nombres de usuario, etc...

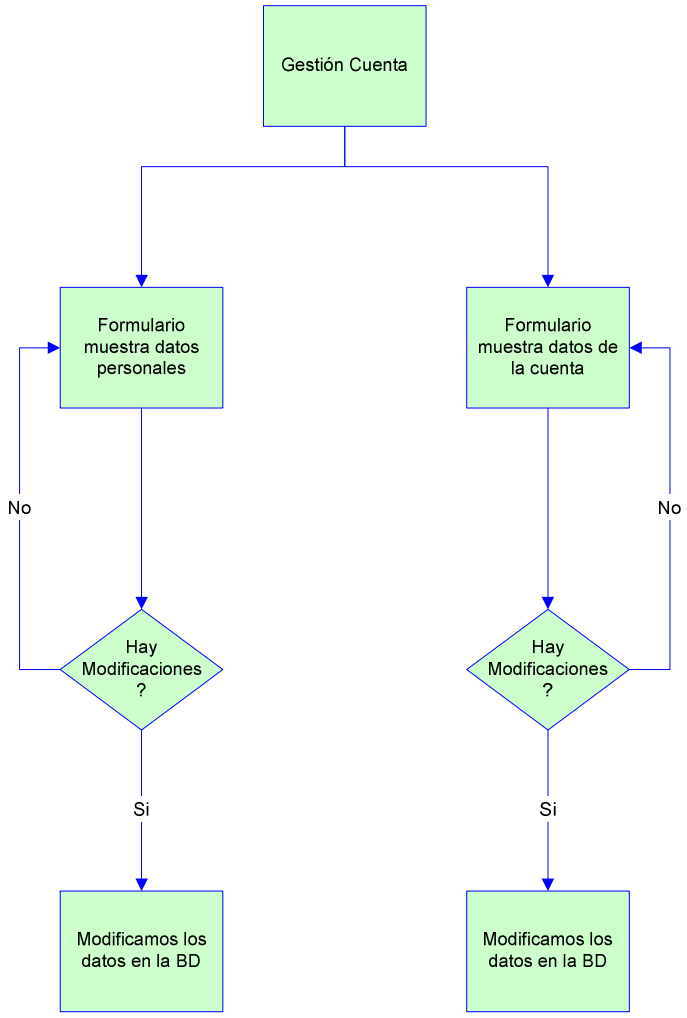

Figura 10 – Módulo Gestión Cuentas

3.1.8. Ciclo de vida de una incidencia

En este flujo se describe el ciclo de vida por el cual pasa una incidencia, desde el momento en que ha sido abierta, pasando por todos los estados hasta que ésta es facturada. Como se puede ver en el flujo, de "estado solucionada" puede pasar a "estado asignada"; Esto sucedería en el caso de que el Director de mantenimiento decidiese que no se ha solucionado correctamente. También indicar que el "estado asignada" puede volver sobre sí mismo, ya que la incidencia se puede ir asignado de un usuario a otro sin necesidad de que cambie su estado.

- **Abierta:** Estado en el que se encuentran las incidencias que se han abierto antes de asignarlas a algún mecánico

- **Asignada:** Estado en el que se encuentra una incidencia que pertenece a un usuario que se espera que realice alguna actuación sobre la misma

- **Solucionada**: Estado en el que se encuentra una incidencia cuando el mecánico ha realizado la intervención y ha asignado los materiales que ha necesitado para solucionar dicha incidencia.

- **Cerrada:** Estado en el que se encuentra la incidencia cuando el Director logístico ha supervisado que la incidencia ha sido solucionada correctamente.

- **Facturada:** Estado en el cual se encuentra una incidencia una vez que el Director financiero ha realizado el cálculo de la factura.

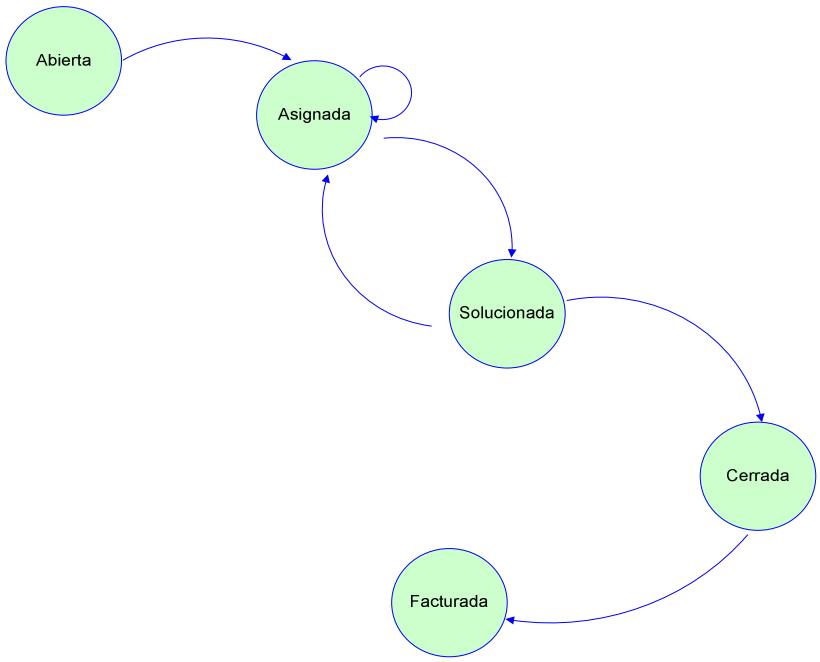

Figura 11– Ciclo de vida de una incidencia

3.2. Diseño de la base de datos.

El diseño de la base de datos es una de las partes más importantes del proyecto ya que de él depende el buen funcionamiento de la aplicación, es por ello de vital importancia un buen diseño ya que nuestra aplicación realiza un gran número de consultas y de inserciones de datos que alimentan a los diferentes formularios que contiene Aplicación.

Todo lo que se registre en la aplicación debe quedar perfectamente reflejado: incidencias, modificaciones de las mismas, materiales, facturas, etc...

Como podremos observar en el diagrama de la base de datos, tenemos dos tablas que identificaremos como las tablas principales del sistema y que son la tabla de incidencias y la tabla de personas. El resto de tablas interactúan con estas dos para poder ofrecer toda la funcionalidad que necesita cumplir el proyecto.

También observaremos a continuación las relaciones entre todas las tablas que conforman el entramado de la base de datos. La tabla incidencia se encuentra relacionada con doce tablas de la base de datos, esto es así ya que incidencias se relacionan con tablas como: Categoría-Incidencia, Tipo-Incidencia, Estado-Incidencia, Prioridad, etc. El Objetivo es conseguir separar los diferentes elementos que se han de ir actualizando en la incidencia durante su ciclo de vida., facilitando las actualizaciones y haciendo más rápida la aplicación

Seguidamente se mostrarán los diferentes diseños que realizamos hasta conseguir nuestra base de datos:

3.2.1. Diseño conceptual

Este diseño es el primero que debemos hacer para identificar las entidades, los atributos y las relaciones existentes para poder llegar a realizar nuestro diagrama entidad-relación.

Las entidades que nos encontramos son:

- Incidencias: Aquí se registraran todas las incidencias que existan indicando las características de ellas. Esta Entidad tiene una relación de especialización con las tablas Reproductividad, Severidad, Prioridad, Estado Incidencia, Categoría Incidencia y Tipo Incidencia.
- Personas: En esta entidad se registran todos los datos referentes a los usuarios que pueden acceder al sistema.
- Perfil: Es utilizada para guardar cada uno de los perfiles que pueden existir en la aplicación.
- Permiso: Entidad con la que registramos los permisos que tiene asociado un perfil.
- Material: Entidad donde se registran los materiales y las cantidades disponibles, realizando la función de almacén virtual.

- Factura: Entidad que contiene los datos de la factura que posee una incidencia; También posee una relación de especialización con las entidades Estado-factura y Detalle-Factura.

Las entidades débiles que nos encontramos son:

- Material Incidencia: Este relaciona Material con Incidencia para indicar que materiales se han utilizado para solucionarlo.
- Crear: Es utilizada para la relación con las incidencias entre Incidencias y Factura para poder facturar una incidencia ya solucionada según los materiales utilizados.
- Comentario Incidencia: Vincula incidencias con personas y guarda un registro del comentario que se realice sobre una incidencia
- Notificación incidencia: Vincula personas con incidencias, registrando las notificaciones que se han realizado a una persona sobre una incidencia.
- Seguimiento incidencia: Vincula persona con incidencias, registrando las actualizaciones que se realizan en la incidencia
- Asignada por: Vincula una Incidencia a una persona

Los atributos de cada una de las entidades son:

Material:

- Id Material: Identificador del material
- Nombre: Nombre del material
- Existencias: Cantidad de material que se tiene en Stock
- Precio: Coste del material
- Descripción: Descripción del material

Perfil:

- Id Perfil: Identificador del perfil
- Nombre Perfil: Nombre del perfil
- Activado: Bloqueo de un usuario en la aplicación
- Fecha Creación: Fecha en que se creó el perfil

Permiso:

- Id Permiso Identificador del permiso
- Nombre Permiso: Nombre del Permiso
- Descripción Permiso: Descripción del permiso
- Fecha Creación: Fecha en que se creó

Categoría Incidencia:

- Id Categoría Incidencia: Identificador de la categoría
- Descripción : Tipos de categorías del sistema

Estado Incidencia:

- Id Estado Incidencia: Identificar de los estados
- Descripción: Tipos de Estados que tiene la incidencia

Tipo Incidencia:

- Id Tipo Incidencia: Identificador del tipo de incidencia
- Descripción: Tipos de incidencia

Prioridad:

- Id Prioridad: Identificador de prioridad
- Descripción: Tipos de prioridades

Severidad:

- Id Severidad: Identificador de severidad
- Descripción: Tipos severidad que existen

Reproductividad:

- Id Reproductividad: Identificador de la reproductividad
- Descripción: Tipos de reproductividad que tenemos

Seguimiento:

- Id Seguimiento: Identificador del seguimiento
- Descripción: Descripción del seguimiento

Estados Factura:

- Id Estado Factura: Identificador de la factura
- Descripción: Tipos de estados que posee una factura

Personas:

- Id Personas: Identificador de persona
- Tipo Identificación: Tipo de documento de identificación
- Identificación: número del documento identificación
- Nombre: Nombre de la persona
- Apellidos: Apellidos de la persona
- Sexo: Tipo de sexo de la persona

- Fecha Nacimiento: Día de nacimiento de la persona
- Correo Electrónico: Correo electrónico
- Teléfono: Teléfono fijo
- Teléfono Móvil: Móvil de la persona
- Dirección: domicilio de la persona
- Población: Población del domicilio
- Código Postal: Código postal del domicilio
- Cuenta Bancaria: Cuenta Bancaria de la persona
- Código Login: Código de usuario
- Contraseña: Pasword para accede al sistema
- Bloqueado: Bloqueo de un usuario para que no accede al sistema
- Fecha Alta: fecha introducción de la persona al sistema
- Código Persona: Código empleado en la empresa

Incidencia:

- Id Incidencia: Identificador incidencia
- Código Incidencia: Código interno de la incidencia
- Id Asignado Por: Identificación de persona
- Id Creador: Identificador de la persona que crea una incidencia
- Id Asignado A: Identificador persona para saber quién asignó
- Id Categoría: Identificador categoría
- Id Tipo Incidencia: Identificador Tipo de incidencia
- Id Estado: Identificador estado de la incidencia
- Prioridad: Identificador de la prioridad de la incidencia
- Severidad: Identificadora de la severidad de la incidencia
- Reproductividad: Identificador de la reproductividad
- Descripción: Descripción incidencia
- Fecha Creación: Fecha creación incidencia
- Fecha Modificación: Fecha en la que se realizan modificaciones

Facturas:

- Id Factura: Identificador de la factura
- Id Incidencia: Identificador incidencia
- Importe Total: Coste de la incidencia
- Fecha Creación: Fecha en que se creó la factura
- Id Estado Factura: Identificador estado factura

Comentario Incidencia:

- Id Comentario Incidencia: Identificador comentario incidencia
- Id Incidencia: Identificador de la incidencia
- Id Persona: Identificador persona
- Fecha Comentario: Fecha se realizó el comentario
- Comentario: Comentario que se añade a la incidencia

Material Incidencia:

- Id Material Incidencia: Identificador material incidencia
- Id Material: Identificado Material
- Id Incidencia: Identificador Incidencia
- Cantidad: Cantidad

Notificación Incidencia:

- Id Notificación Incidencia: Identificador notificación
- Id Incidencia: Identificador incidencia
- Id Personas: Identificador personas
- Enviar Correo: Enviar correo
- Correo Electrónico: correo electrónico a enviar

Perfil Permisos:

- Id Perfil Permisos: Identificador perfil permisos
- Id Perfil: Identificador perfil
- Id Permiso: Identificador permiso

Persona Perfil:

- Id persona perfil: Identificador persona perfil
- Id Persona: Identificador de la persona
- Id Perfil: Identificador del perfil

Detalle Factura:

- Id Detalle: Identificador detalle
- Id Factura: Identificador Factura
- Total: Identificador total
- Descripción: Comentario sobre la factura

Seguimiento Incidencia:

- Id Seguimiento Incidencia: Identificador seguimiento Incidencia
- Id Seguimiento: Identificador seguimiento
- Id Persona: Identificador persona
- Id Incidencia Identificador Incidencia
- Fecha Seguimiento: Fecha Seguimiento
- Seguimiento: Descripción del seguimiento

Asignada Por:

- Id Persona: Identificador persona
- Id Incidencia Identificador Incidencia

Abre Incidencia:

- Id Persona: Identificador persona
- Id Incidencia Identificador Incidencia

Abre Incidencia:

- Id Persona: Identificador persona
- Id Incidencia Identificador Incidencia

Posee:

- Id Factura: Identificador Factura
- Id Incidencia Identificador Incidencia

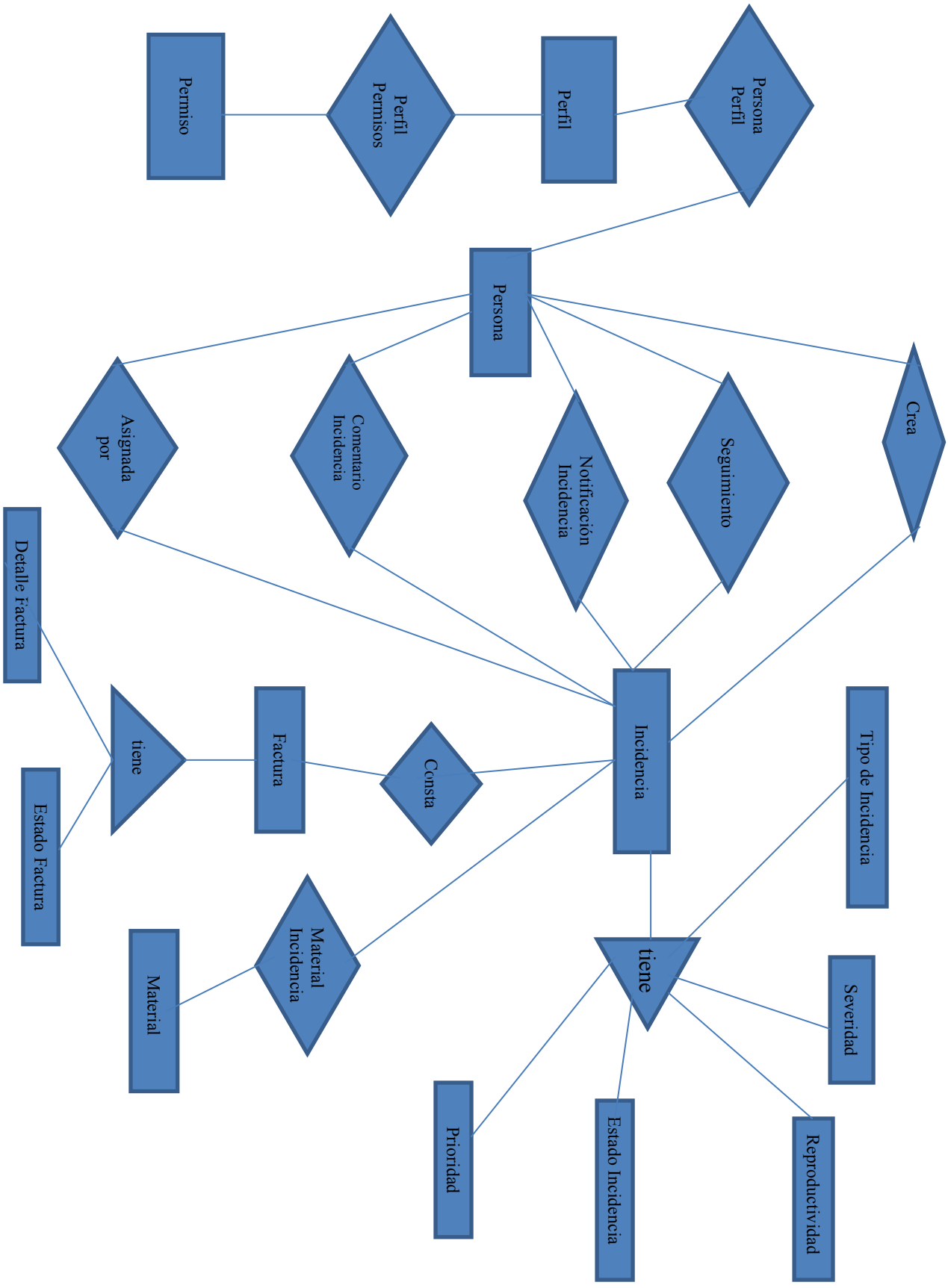

Figura 12 – Diagrama E-R

3.2.2. Diseño Físico

Una vez definido el diseño lógico, implementamos el diseño físico, tal y como se encuentra en nuestro sistema implementado. Como podemos ver, algunas tablas intermedias han sido simplificadas con el fin de optimizar el sistema. También se muestra en la figura como queda la relación de las mismas.

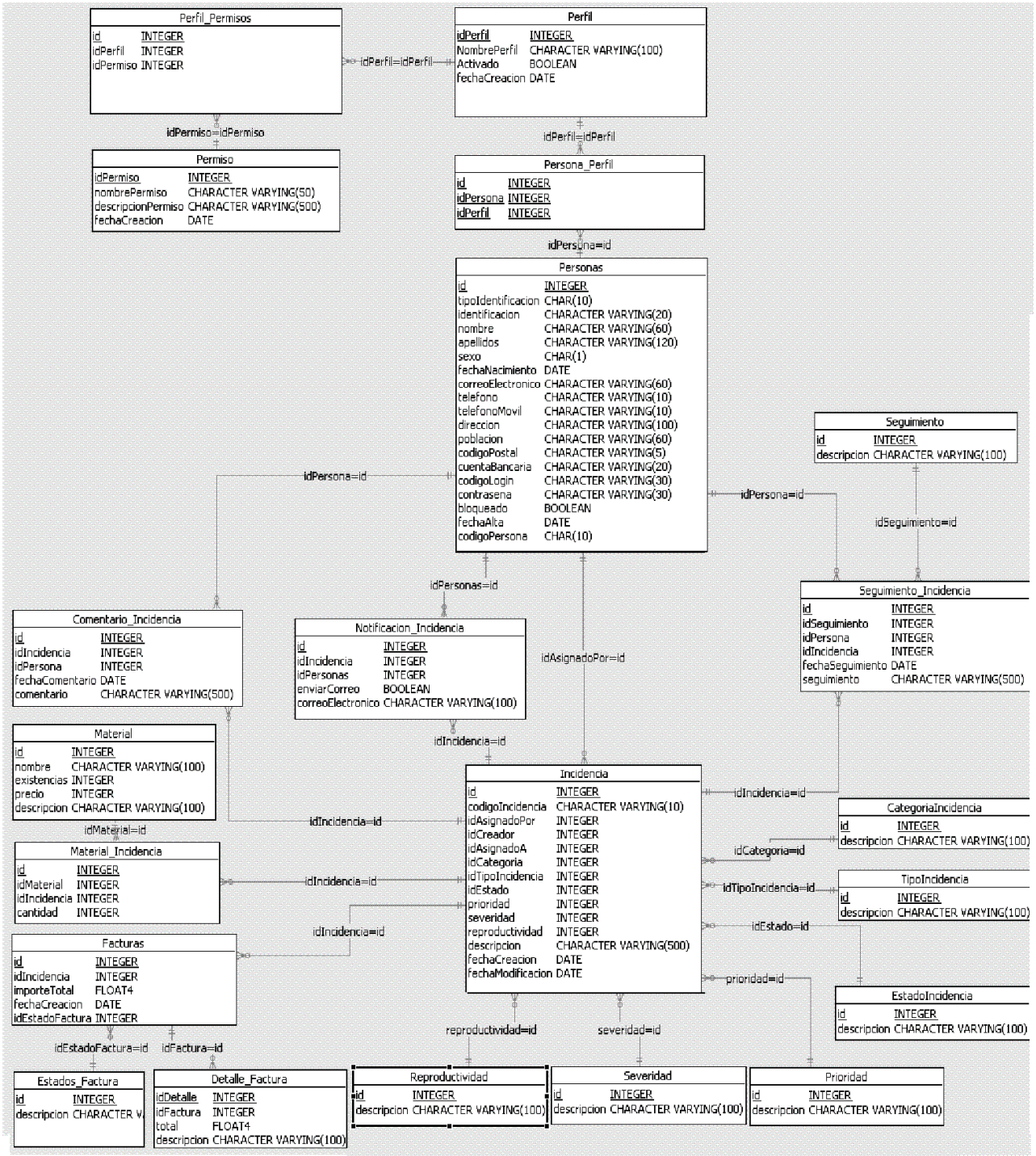

Figura 13 - Diagrama Físico

4. Tecnologías utilizadas

En este apartado se van a describir las tecnologías utilizadas para el desarrollo de este proyecto y la razón por la que se escogieron dichas tecnologías. Me gustaría destacar, que entraré en un detalle más profundo, para explicar el funcionamiento del marco de trabajo Modelo Vista Controlador (MVC) que he utilizado. Creo que es algo a destacar en el proyecto, por ser una tecnología innovadora.

El hecho de utilizar esta aplicación con las últimas tecnologías nos permite cumplir uno de los objetivos marcados para este proyecto, que era realizar una aplicación modular que pueda adaptarse y vaya creciendo según varíen las necesidades del cliente.

Antes de entrar en detalle para explicar cada uno de los bloques anteriores, me gustaría comenzar explicando que partimos del uso del Modelo Vista Controlador (MCV) como modelo a seguir para la creación de nuestra aplicación distribuida.

Definición de MCV: Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML. El control es el código que provee de datos dinámicos a la página, y el modelo contiene clases representativas de la aplicación.

• **El Modelo:** es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos.

• **La Vista:** presenta el modelo en un formato adecuado para interactuar, usualmente un elemento del interfaz de usuario.

• **El Controlador:** responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente como puede ser un Sistema de Gestión de Base de datos para almacenar los datos. MVC no menciona específicamente esta capa de acceso a datos.

4.1. Arquitectura Distribuida

Seguidamente pasamos a estudiar la arquitectura distribuida que hemos seleccionado:

4.1.1. Java 2 Enterprise Edition

J2EE: Es una plataforma que fue creada por SUN en el año 1997. Muchos expertos opinan sobre ella ya que es quizás, una de las mejores cuando se desea buscar un tipo de arquitectura basada en Software libre. Las ventajas que nos ofrece dicha plataforma son:

1. Es un sistema multiplataforma, al estar basado en lenguaje Java nos permite tener la libertad de poder ejecutarlo en cualquier sistema operativo donde sea compatible la ejecución de una máquina virtual Java.

2. Está respaldada y controlada por un organismo, formado por unas 500 empresas donde encontramos las más importantes del mundo de las tecnologías, como SUN, IBM, HP, Oracle , etc., garantizándose de esta manera la evolución de la misma.

3. A los clientes se les ofrece competitividad, hay un amplio abanico de empresas que sirven soluciones de este tipo y esto lógicamente garantiza una buena competitividad a nivel de precios y servicios. A su vez esta plataforma ofrece una garantía de madurez y robustez por el tiempo que lleva en el sector tecnológico.

4. Quizás una de las más importantes, es la posibilidad que nos ofrece de disponer de varias soluciones para cada una de las partes de nuestra Arquitectura.

4.1.2. El modelo de desarrollo de J2EE

Esta plataforma basa su modelo de programación orientándolo a la creación de aplicaciones basadas en N-capas. Suelen poseer cinco capas para realizar esta actividad, aunque en ocasiones, alguna más, como será nuestro caso, dependiendo del marco de trabajo MCV que estemos utilizando.

• **Capa de cliente**: Representa el interfaz de usuario que maneja el cliente.

• **Capa de presentación**: Representa el conjunto de componentes que generan la información que se representará en el interfaz de usuario y del cliente. Típicamente se creará a través de componentes basados en Servlets y JSP.

• **Capa de lógica de negocio**: Contiene nuestros componentes de negocio reutilizables. Normalmente se forma a partir de componentes EJB.

• **Capa de integración**: Aquí se encuentran componentes que nos permiten hacer más transparente el acceso a la capa de sistemas de información. Por ejemplo este es el lugar idóneo para implementar una lógica de objetos de acceso a datos, DAO (Data Access Objects).

• **Capa de sistemas de información**: Esta capa engloba a nuestros sistemas de información: bases de datos relacionales, bases de datos orientadas a objetos.

La principal ventaja que nos proporciona este tipo de modelo, al estar separado por capas, es que nos facilita la realización de modificaciones sobre ellas, ya que existe poco acoplamiento. Esto a su vez nos facilita la extensibilidad y el mantenimiento de la aplicación, cosa que nos ayuda a cumplir los objetivos marcados en nuestro proyecto.

Por último podemos resaltar los componentes Enterprise Java Beans, que facilitan y hacen totalmente transparente al programador la persistencia, seguridad, gestión de las transacciones, etc...

Resumiendo las ventajas que nos ofrece J2EE y las que llevó a mi empresa a la decisión de la utilización de dicha tecnología, pasamos a enumerarlas a continuación:

- Menor coste de producción.
- Menor coste de desarrollo.
- Alta implantación en el mercado.
- Poco coste de aprendizaje
- Actualizable.
- Estándar de código abierto.

4.2. Apache ANT

Apache Ant es una herramienta utilizada en programación para la automatización de tareas mecánicas y repetitivas. Normalmente durante la fase de compilación y construcción (build).

Podemos decir que es similar al make pero escrito en Java y pensado para usarlo en Java. La diferencia entre el Ant y el Make es que Ant utiliza un fichero XML para describir el proceso de construcción (build) y sus dependencias, mientras que Make tiene su propio formato Makefile. Por defecto el fichero XML se llama buid.xml.

Ant es un proyecto de código abierto de Apache Software Foundation. El principal objetivo que se marcaron fue evitar los problemas de portabilidad que generaba el Make. Ant resuelve el problema del Make realizando una gran cantidad de funcionalidades por él mismo

4.3. MCV (Modelo Vista Controlador)

Una vez elegido el servidor J2EE, hemos de buscar un marco de trabajo que nos permita de forma sencilla crear una aplicación que sea rápida y robusta.

La decisión final por parte de la empresa fue la utilización del marco de trabajo JSF (Java Server Faces), es por ello que se brindó una formación para poder desarrollar el aplicativo en este marco de trabajo.

Tal como indiqué anteriormente creo necesario explicar más detenidamente en qué consiste este marco de trabajo ya que es muy interesante.

4.3.1. ¿Qué es Java Server Faces?

JSF es un marco de trabajo de interfaces de usuario para crear aplicaciones java J2EE basadas en el patrón de diseño MVC donde las vistas conocen la acción que se va invocar en su petición. JSF tiene como características principales:

- Una API y una implementación de referencia para representar componentes de interfaz de usuario (UI-*User Interface*) y manejar su estado.
- Manejar eventos, validar en el lado del servidor y convertir datos
- Definir la navegación entre páginas
- Soportar internacionalización y accesibilidad, y proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

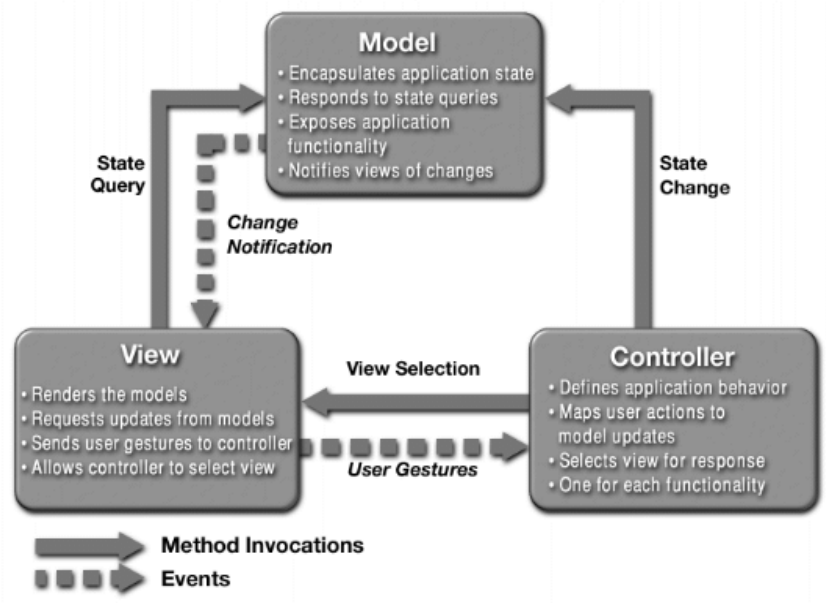

Figura 14 – Patrón MCV

Este modelo de programación bien definido y la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado servidor.

Con un mínimo esfuerzo es posible:

- Conectar eventos generados en el cliente a código de la aplicación en el lado servidor.
- Mapear componentes UI a una página de datos en el lado servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.

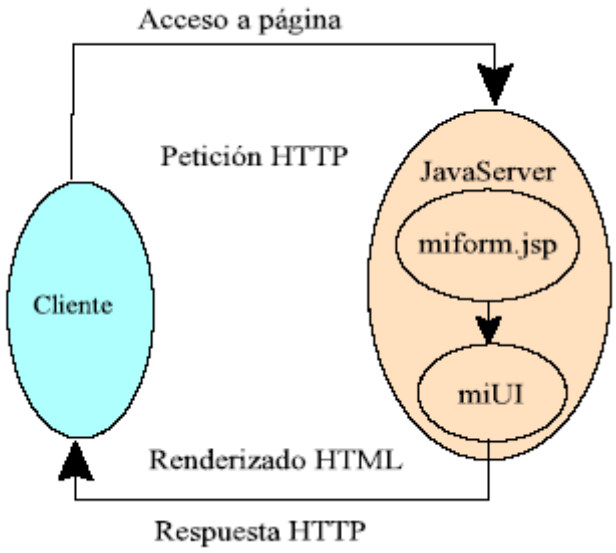

Figura 15 – Módulo Flujo JSF

4.3.2. ¿Cómo funciona JSF?

En la actualidad las webs están creadas como un grupo de pantallas con las que los usuarios interactúan según la navegabilidad que ofrece la aplicación. Estas páginas están formadas por imágenes, botones, tablas, etc.. con las que interactúa el usuario. Todo esto se encuentra agrupado en formularios HTML, utilizados para enviar la información introducida por el usuario al servidor.

Básicamente lo que hace JSF es, asociar a estas pantallas las clases de java, que serán quiénes se encarguen de recoger toda la información que ha introducido el usuario. La ventaja que nos ofrece JSF es que simplifica el envío de estos datos, haciendo que el proceso sea más sencillo ofreciendo soluciones automáticas para facilitarnos el trabajo tales como:

- Las vistas al usuario en cajas de texto y tablas.
- Recopilación de los datos informados en los formularios.
- Realizando validaciones y conversiones de los datos introducidos por el usuario.
- Rellenando de campos automáticos en función de lo que informe el usuario.
- Facilita el control de los eventos que provienen de los periféricos tales como el ratón teclado.

4.3.3. Elementos principales que constituyen una aplicación JSF

- Páginas JSP que contienen los elementos de la aplicación , generarán las vistas
- Beans java que se conectan con los formularios JSF
- Clases java para la lógica de negocio y utilidades.
- Ficheros de configuración, componentes a medida y otros elementos del el marco de trabajo
- Elementos como recursos estáticos, javascript y otros elementos que conforman una web.

4.3.4. Los backbeans

Una parte que no podemos dejar de comentar, para entender cómo funciona la tecnología JSF son los bakbeans, tal como su nombre hace intuir, son las clases java que se encuentran por debajo de los formularios. Estas clases java se asocian a los formularios JSF y sus referencias son indicadas en los ficheros de configuración de los JSF situados en managed Beans. El controlador de JSF es el responsable de la gestión de las clases, controlando la construcción y destrucción de éstas automáticamente, según las necesidades del momento.

4.3.5. Estructura de las páginas

La estructura más básica que podemos encontrar para una página JSF está formada por una página JSP que posee un formulario HTML y un backbean.

Cuando el controlador recibe una llamada del tipo *.jsf en el servidor de aplicaciones sucede lo siguiente:

1. El usuario realiza una navegación y solicita la petición de una dirección url *jsf, y el controlador JSF se activa.

2. El controlador comprueba si es el primer acceso que se hace a esta página. Si es el primer acceso construye en memoria la presentación de la página y sus elementos de control. En ese momento el controlador sabe como pintar la página HTML.

3. Una vez pintada la página se asocian las clases a los formularios, el controlador es capaz de identificar que clase pertenece al formulario, por medio de las peticiones de request y la sesión. Las clases inexistentes son creadas por los constructores, definidos en managed beans.

4. Por último solo queda dar los valores a los elementos JSF que posee la página.

Como resultado de todos los pasos anteriores el servidor nos sirve una página creada desde su página JSP con las etiquetas que han sido asociadas desde la clase java

Las etapas del procesamiento de una petición de una página JSF son:

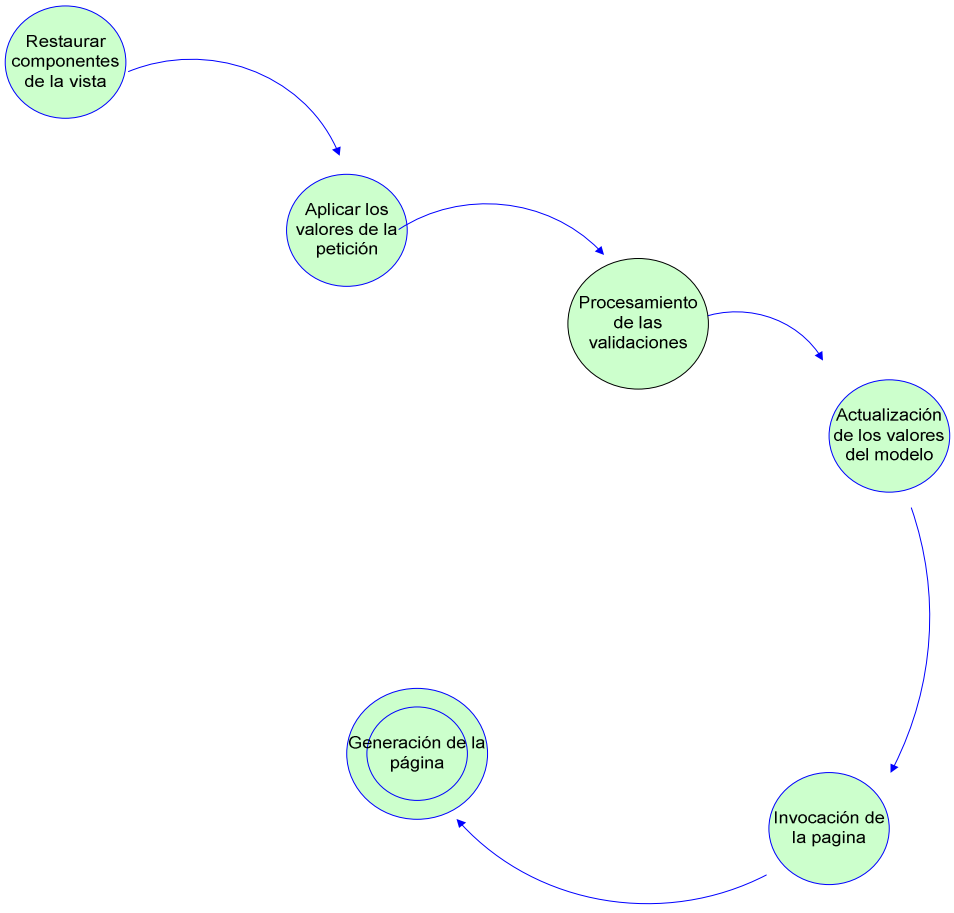

Figura 16– Etapas procesamiento JSF

Resumiendo las ventajas que nos ofrece JSF y las que llevó a mi empresa a la decisión de la utilización de dicha tecnología son:

- La facilidad y rapidez que nos ofrece JSF para el desarrollo de aplicaciones creadas en java, y en la creación de vistas sencillas salvo que introduzcamos mucha maquetización.
- Las vistas JSP parecidas al HTML, son muy fáciles de utilizar
- JSF está integrado dentro del JSP , recogiendo y generando los elementos de la página
- JSF permite introducir javascript mejorando el rendimiento
- JSF es extensible y modificable
- Es fácil de mantener

4.3.6. Richfaces

Para concluir y una vez entendido el funcionamiento de JSF, queda destacar que a nuestra aplicación se le ha intentado dar un estilo de pagina dinámica, dándole un aspecto más amigable y con más agilidad, sin que sea necesario que la aplicación deba refrescarse continuamente y permita al usuario trabajar con más comodidad; es por ello que finalmente se decidió introducir Ajax en las páginas.JSP

Actualmente existe un marco de trabajo Richfaces que casi sin tener que retocar nuestras páginas JSP y solo añadiendo unas etiquetas nos permite introducir Ajax en nuestra aplicación JSF, y que fue una de las razones que nos llevó a introducir este tipo de tecnología para mejorar el rendimiento de la página.

Antes de continuar explicando cómo funciona Richfaces, voy a definir el concepto de Ajax aunque no se comporta de igual manera utilizando los Richfaces.

AJAX (Asynchronous Javascript and XML) traducido como indica el título, no es más que una forma de programar aplicaciones interactivas para web. Esta evolución de DHTML se la ha denominado Web 2.0. Para ello utiliza XHTML y CSS para formatear la información, para interactuar y visualizar dinámicamente la información, se apoya en XML, XSTL para manipular la información mostrada, el objeto XMLHttpRequest (no estándar) y Javascript para actualizar los datos sin necesidad de refrescar la página, y para manipular todas esas tecnologías.

Una vez defino el concepto de Ajax vamos a explicar que nos ofrece Richfaces. Éste marco de trabajo se puede solapar con cualquiera de las librerías myfaces necesarias para la implementación de los JSF.

Richfaces es un marco de trabajo opensource que se integra en la arquitectura de JSF dándoles una tecnología Ajax sin la necesidad de utilizar Javascript, por ello anteriormente expliqué que se comportaba de diferente modo con Richfaces.

Con la introducción de Ajax en JSF se mejora en gran manera los beneficios que nos aportaba el ciclo de vida de JSF, modificándolo en el momento que se define una invocación a una petición de Ajax en nuestra página JSP.
Posteriormente con la respuesta obtenida, se sincroniza con el ciclo de vida de JSF una vez el servidor ha relocalizado el cambio de los datos, y de esta manera optimizaremos la aplicación.

Facilita la escritura de los componentes propios ya que posee un conjunto de componentes de habilitación del Ajax que extiende el soporte de la página. Al estar diseñados para ser compatibles con otras librerías, nos brinda más opciones de desarrollo y muy fáciles de utilizar.

Por último destacar que posee un generador de código para las plantillas JSP, y que proporciona un avanzado soporte a la gestión de diferentes recursos: imágenes, código JavaScript y hojas de estilo CSS.

RichFaces permite definir (por medio de etiquetas de JSF) diferentes partes de una página JSF que se desea actualizar con una solicitud Ajax, proporcionando así varias opciones para enviar peticiones Ajax al servidor.

El esquema de como se solapa con el ciclo de vida de JSF es el siguiente:

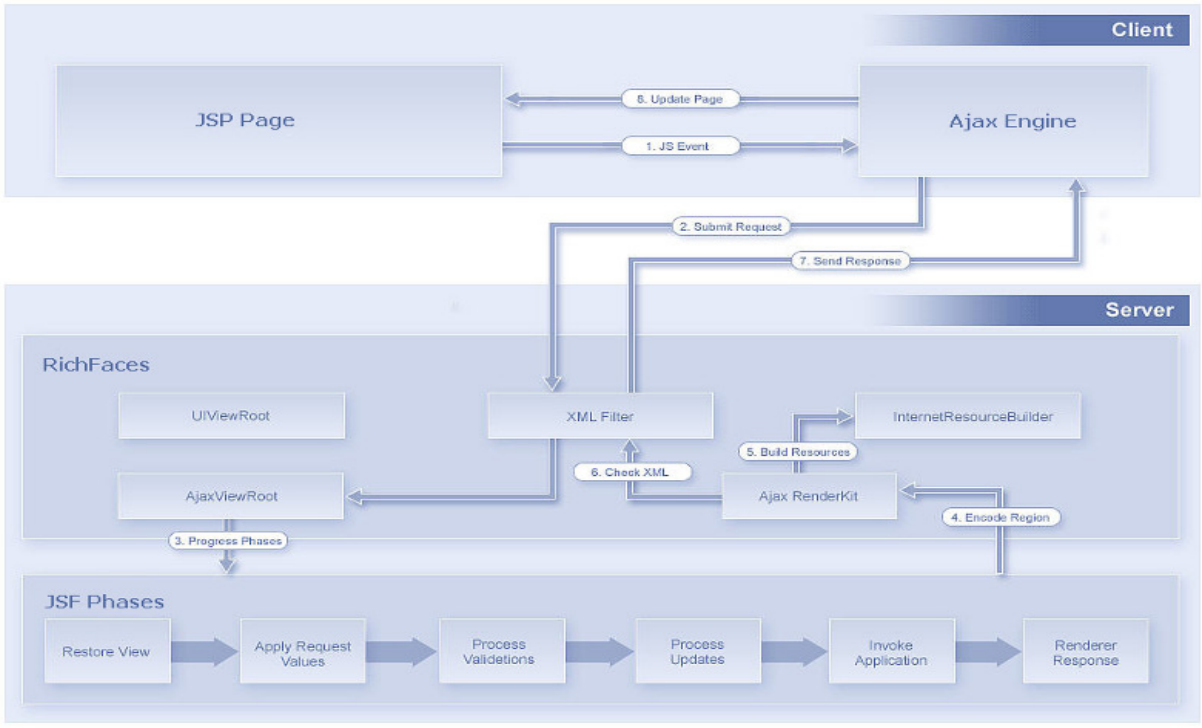

Figura 17– Etapas procesamiento Richfaces

Figura 18 –Componentes Richfaces

4.4. HTML

HTML (Acronimo de Hyper Text Markuo Language) es un lenguaje de marcado que deriva del SGML diseñado para estructurar textos y relacionarlos en forma de hipertexto. Gracias a Internet y a los navegadores web, se ha convertido en una de los formatos más populares que existen en la construcción de los documentos.

4.5. XML

XML, del inglés Markup Language es un metaleguaje extensible de etiquetas de desarrollo por el World Wide Web (W3C). Es una simplificación/adaptación del SGML, permite definir una marca de lenguajes específicos. En fin podemos asegurar que en si no es un leguaje sino una forma de definir los lenguajes. Según las necesidades, como por ejemplo XHTML, SVG, MathML.

A parte de su utilización en Internet XML se utiliza como un Standard que permite la portabilidad entre varias plataformas, también destacar que es una tecnología muy sencilla que se sirve de otras con el fin de que se complementen y esto le proporcina un sinfín de posibilidades. Actualmente está cogiendo mucha importancia por la fiabilidad y facilidad que ofrece al compartir información.

4.6. Eclipse

Eclipse es una plataforma de desarrollo de código libre (OpenSource) basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los ususarios de la aplicación. Es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de los componentes conectados.

Eclipse tiene plugins que permite el desarrollo de Java (JDT Java Development Tools) así como para desarrollar en C/C++,COBOL, etc.

La decisión que me llevó a escoger este entorno de trabajo fue el conocimiento que ya tenía sobre esta herramienta.

4.7. BASE DATOS.

4.7.1. ¿Qué es PostGreSQL?

Es un sistema de base de datos creado en la universidad de Berkley dirigido por el profesor Michael Stonebraker, este sistema ofrece gestión de datos ojeto-relacional.

PostGreSQL es una derivación libre de este proyecto inicial el lenguaje utilizado es el SQL92/SQL99. Pionero en su sector en los inicios principalmente en sistema objeto-relacional, incluyendo características de la orientación a objetos como:

Herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional

En la actualidad ofrece muchas garantías ya que le respaldan 10 años de madurez y podemos asegurar que es una tecnología que ha alcanzado una solida madurez. Haciendo que sea el sistema libre más avanzado con mucha diferencia, atualmente soporta la gran mayoría de las transacciones SQL. Ofrece la posibilidad de control concurrente desde varios legujes como C, C++, Java, Python, PHP y muchos más.

Las características que llevaron a la decisión de utilizar este sistema son:

1. Implementación del estándar SQL92/SQL99.
2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, etc...
3. Permite la creación de tipos propios
4. Permite trabajar con estructuras tipo Array
5. Incorpora funciones de diversa índole: manejo de fechas, geométricas,etc…
6. Soporta declaración funciones propias, así como la definición de disparadores.
7. Soporta el uso de índices, reglas y vistas.
8. Incluye herencia entre tablas
9. Permite la gestión usuarios y sus permisos

Concluyendo podemos decir PostGreSQL es un gestor de base de datos muy eficiente que proporciona una alta escalabilidad, haciéndolo idóneo para sitios en los que no se realicen alrededor 500.000 peticiones por día, dato que nunca llegaremos en nuestra aplicación.

5. Implementación

5.1. Páginas de la aplicación

En este aparatado lo que se pretende es explicar cada una de las funciones que se desempeña en cada una de las páginas que forma parte de la aplicación, por ello podréis ver capturas de pantalla que creo que facilitarán la comprensión de sus funcionalidades:

5.1.1. Index.jsp

Ésta es la página de login de nuestra aplicación, que es invocada desde el navegador con index.jsf; En esta página se introduce el usuario y contraseña para que se tenga acceso a la página de inicio Una vez el usuario hace login si todos los datos son correctos, se asocian los elementos a la clase java LoginBackinBean.java

Como el controlador verá que es la primera vez que accedemos a la página cargará todos los permisos del resto de pestañas que tenga nuestro usuario, asociando las clases java, como por ejemplo en el caso del Super Usuario se cargarán todos Backinbeans que se invocarán de la siguiente forma:

En esta página básicamente lo que podremos hacer es, si somos un usuario registrado, entrar en nuestra sesión, para poder utilizar el aplicativo.

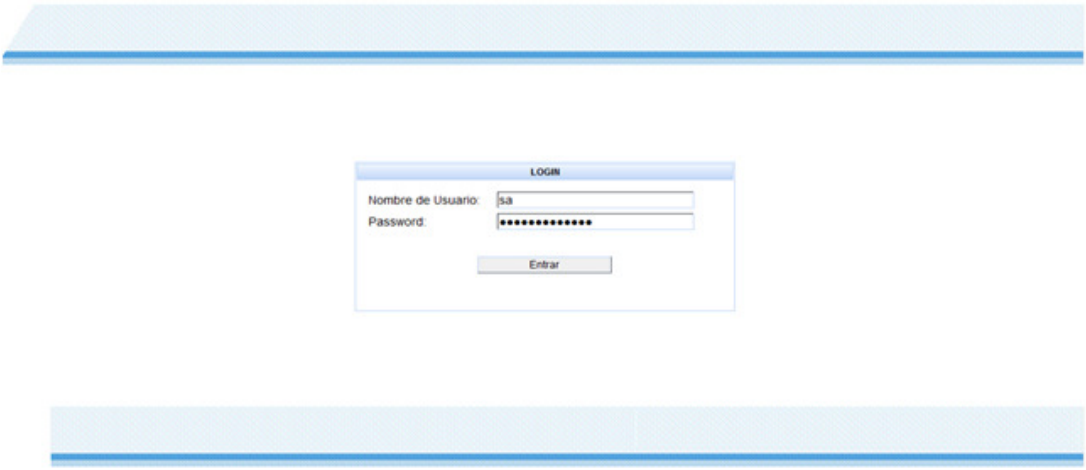

Figura 19 –Index

5.1.2. Inicio.jsp

Es invocado a través de inicio.jsf. Es aquí según los permisos que se han activado tal como se ha indicado anteriormente, donde se cargan los diferentes módulos a los que se tiene acceso y se cargan los BackinBean correspondientes a cada módulo.

En esta página se cargarán las incidencias que posee el usuario en un listado, donde se puede ordenar por cada columna como se desee, si seleccionamos “ver detalle”, se

puede modificar la incidencia accediendo a editar incidencia.jsp invocado por editar incidencia.jsf, más adelante lo explicaremos con más detalle.

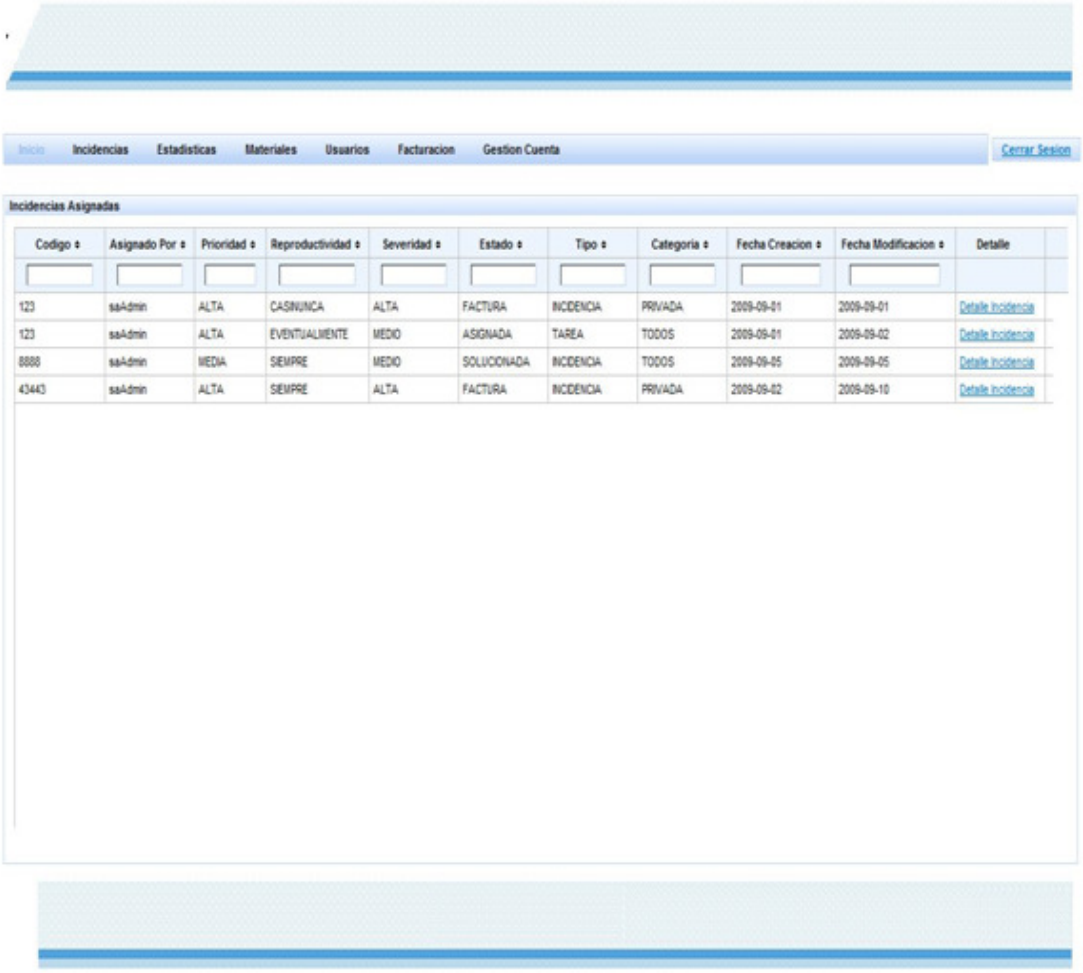

Figura 20- Inicio

5.1.3. moduloCreacionIncidencias.jsp

Es invocado por moduloCreacionIncidencias.jsf. En esta pantalla se nos muestran los campos que deben informarse para poder crear una incidencia y lleva la clase java asociada InsertaIncidenciaBakingBean.java.

Figura 21- Crear Incidencias

5.1.4. moduloConsultaIncidencias.jsp

Invocado a través de moduloconsultaIncidencias.jsf En esta pantalla se nos muestra un buscador y a la derecha aparece un listado con las incidencias que cumplen las

restricciones de búsqueda deseadas. Este tiene asociada la clase java ConsultarIncidenciasBackingBean.java

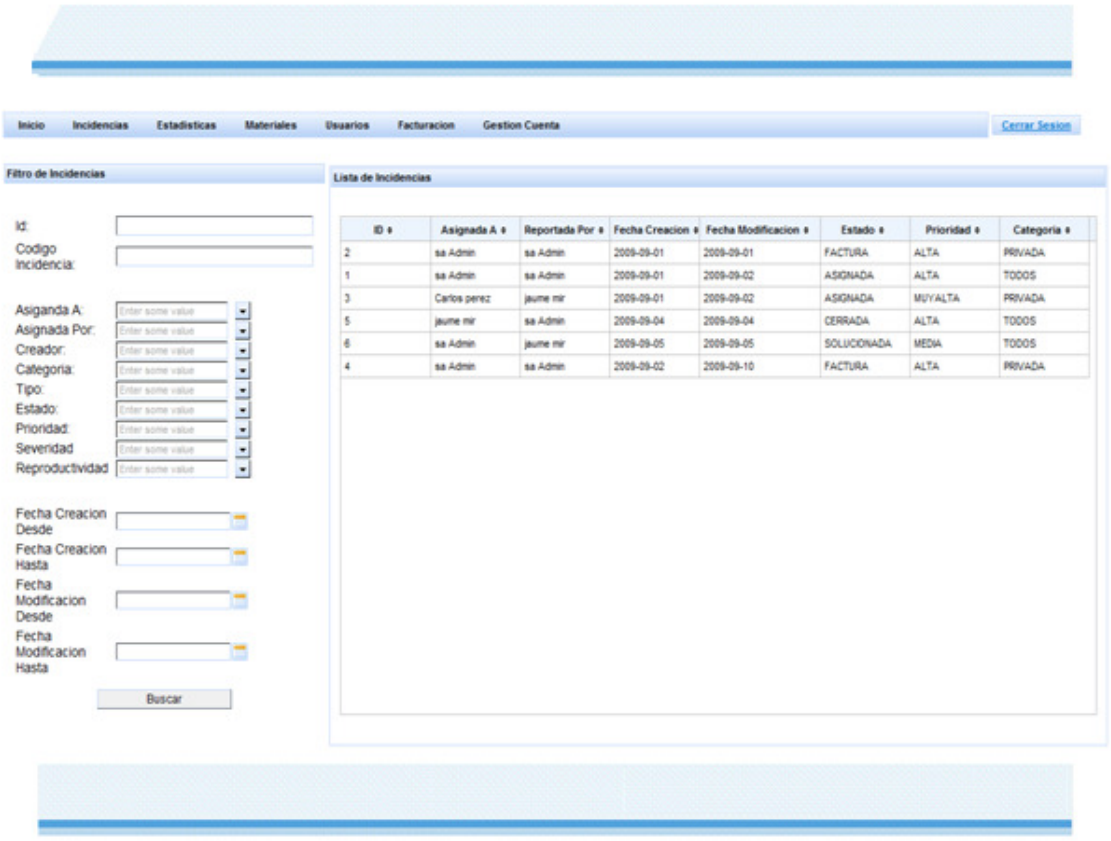

Figura 22- Consulta incidencias

5.1.5. moduloEdicionIncidencias.jsp

Invocado a través de moduloEdiciónIncidencias.jsf esta pantalla nos muestra todos los elementos que forman parte de la incidencia. En este formulario se puede modificar cualquier campo que esté activo, asignar a otros usuarios, indicar comentarios, cambiar estados de las incidencias, realizar las notificaciones a los usuarios por correo, insertar descripciones de cómo se ha solucionado la incidencia y materiales que se han utilizado en la resolución de la incidencia. En la parte baja posee una serie de listados donde se muestran todas las acciones realizadas en la incidencia, como históricos de comentarios, usuarios a los que se ha enviado la incidencia, los materiales que se han utilizado etc.
Cómo dijimos anteriormente en el apartado inicio ver detalle del listado, se accede al mismo módulo. Estos tienen asociado la clase java EditarIncidenciaBackingBean.java.

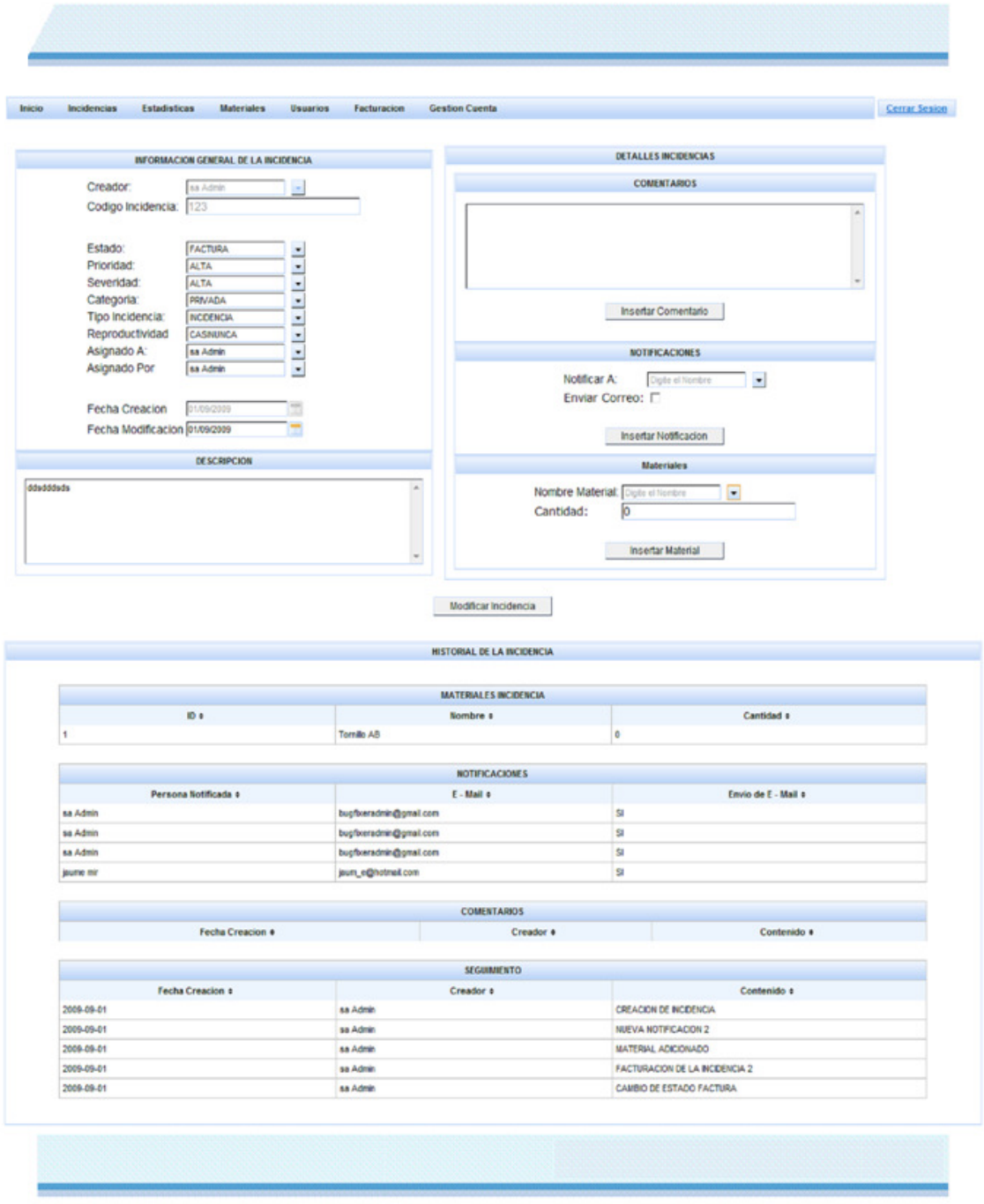

Figura 23- Modificación Incidencias

5.1.6. moduloEstadisticas.jsp

Invocado a través de modulosestadisticass.jsf esta pantalla nos muestran las estadísticas del sistema donde vemos el número de incidencias que tenemos en el sistema y el estado en el que se encuentran. Estos resultados se muestran con una gráfica para que el análisis del sistema sea más fácil. La clase java asociada a este módulo es EstadisticasBackingBean.java

Figura 24- Estadísticas

5.1.7. moduloMateriales.jsp

Invocado a través de moduloMateriales.jsf. En este apartado se realizan todas las operaciones de modificación sobre materiales ya existentes, inserción de nuevos materiales y eliminación de los diferentes materiales. Estos tres formularios se cargan a la vez pero se visualiza uno u otro según la opción selecciona. Los tres están asociados a la clase MaterialesBackingBean.java

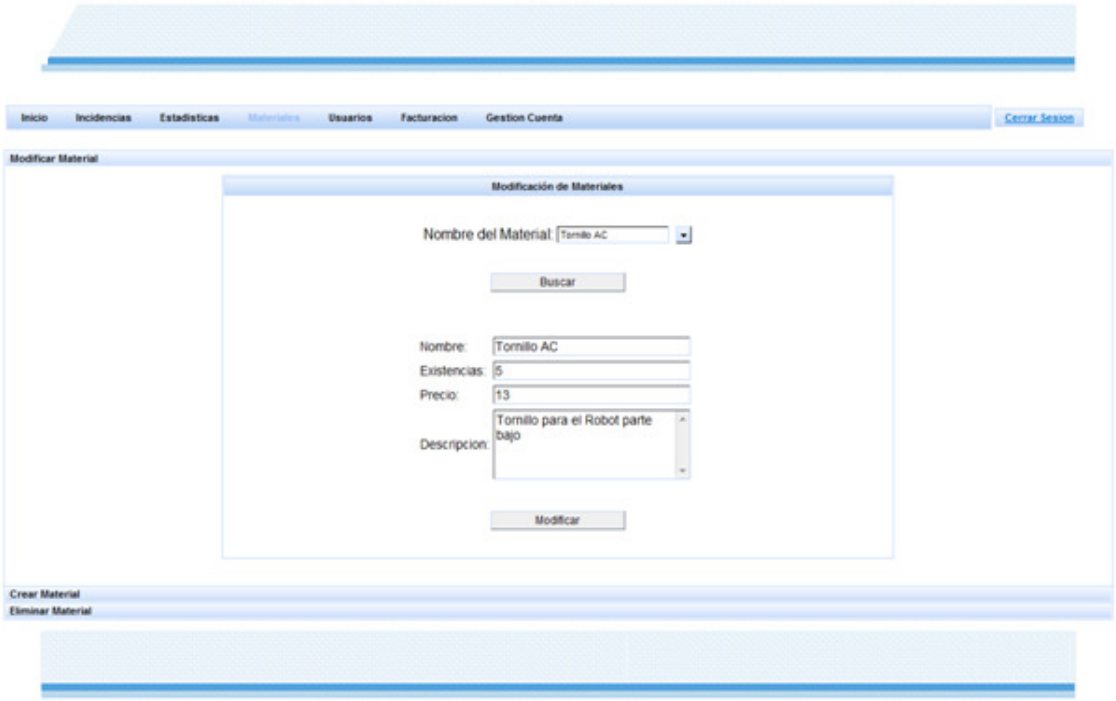

Figura 25- Modificar Material

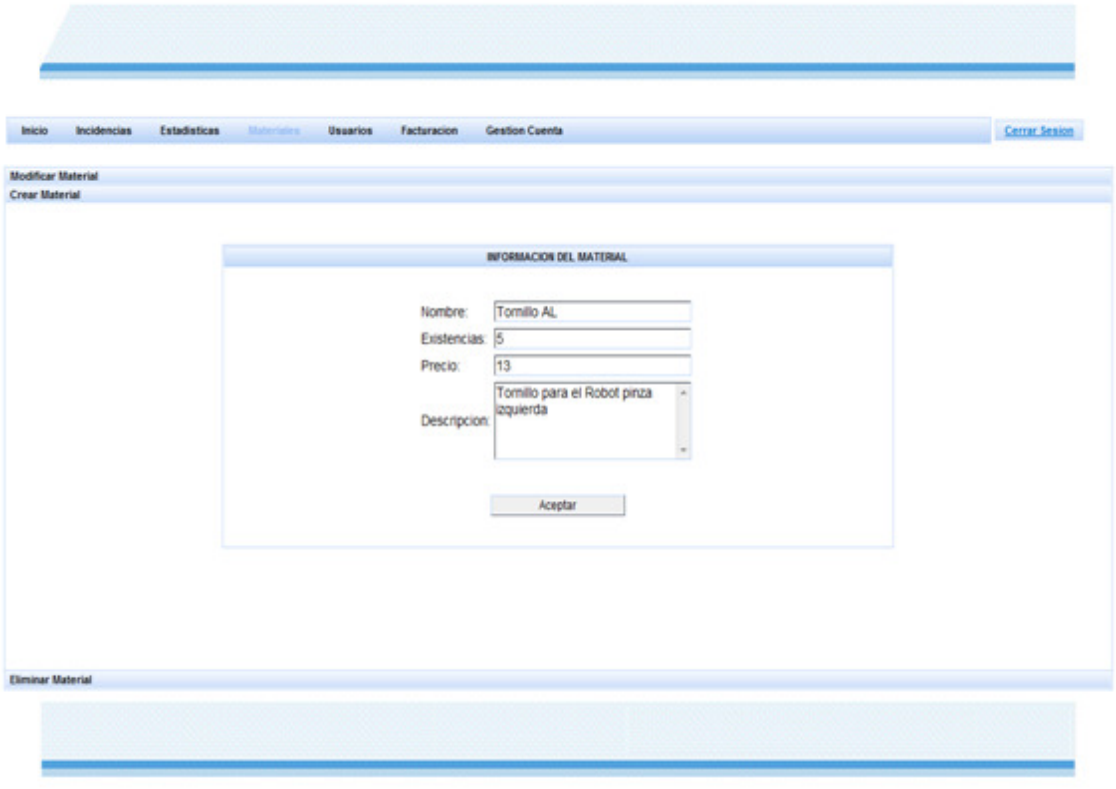

Figura 26- Crear Material

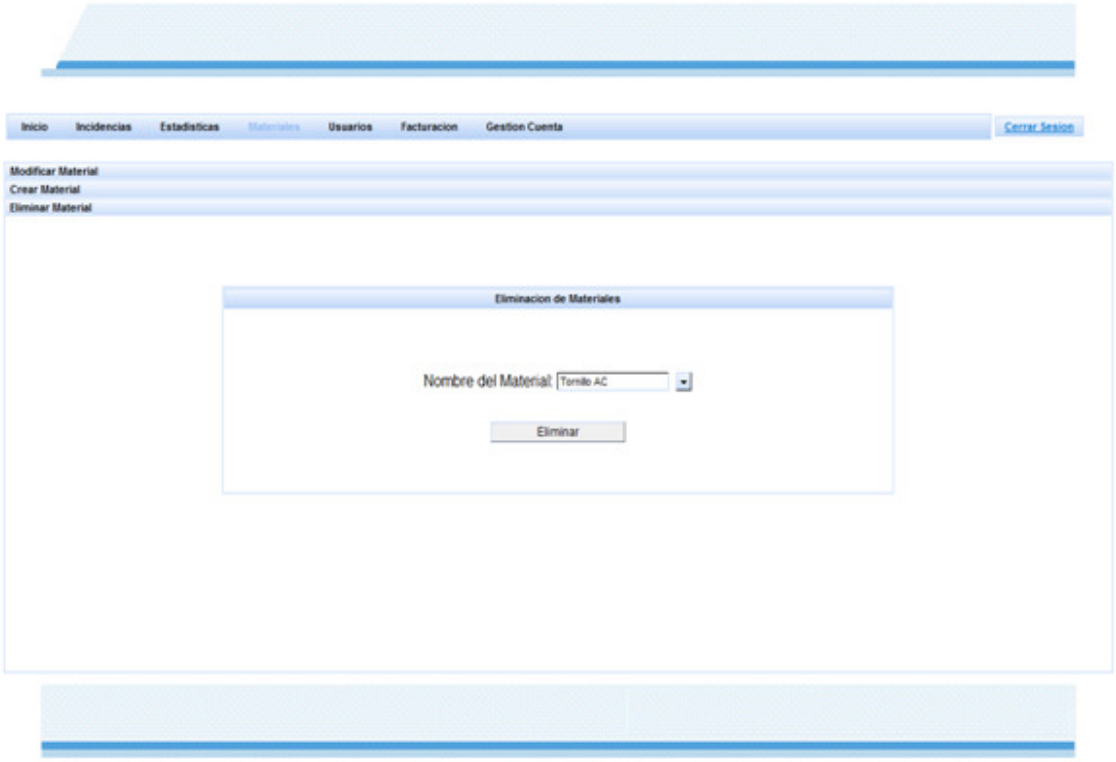

Figura -27 Eliminar Material

5.1.8. admonUsuarios.jsp

Invocado a través de admonUsuarios.jsf. En la parte superior aparece un buscador de usuario por si se desea modificar el mismo. Al buscarlo se cargan los datos en el formulario. De lo contrario si la id es nueva se rellena el formulario, y esto sirve para crear nuevos usuarios guardando sus datos personales, creando su contraseña de conexión e indicando los permisos que tendrá. Está asociado a la clase UsuariosBackingBean.java

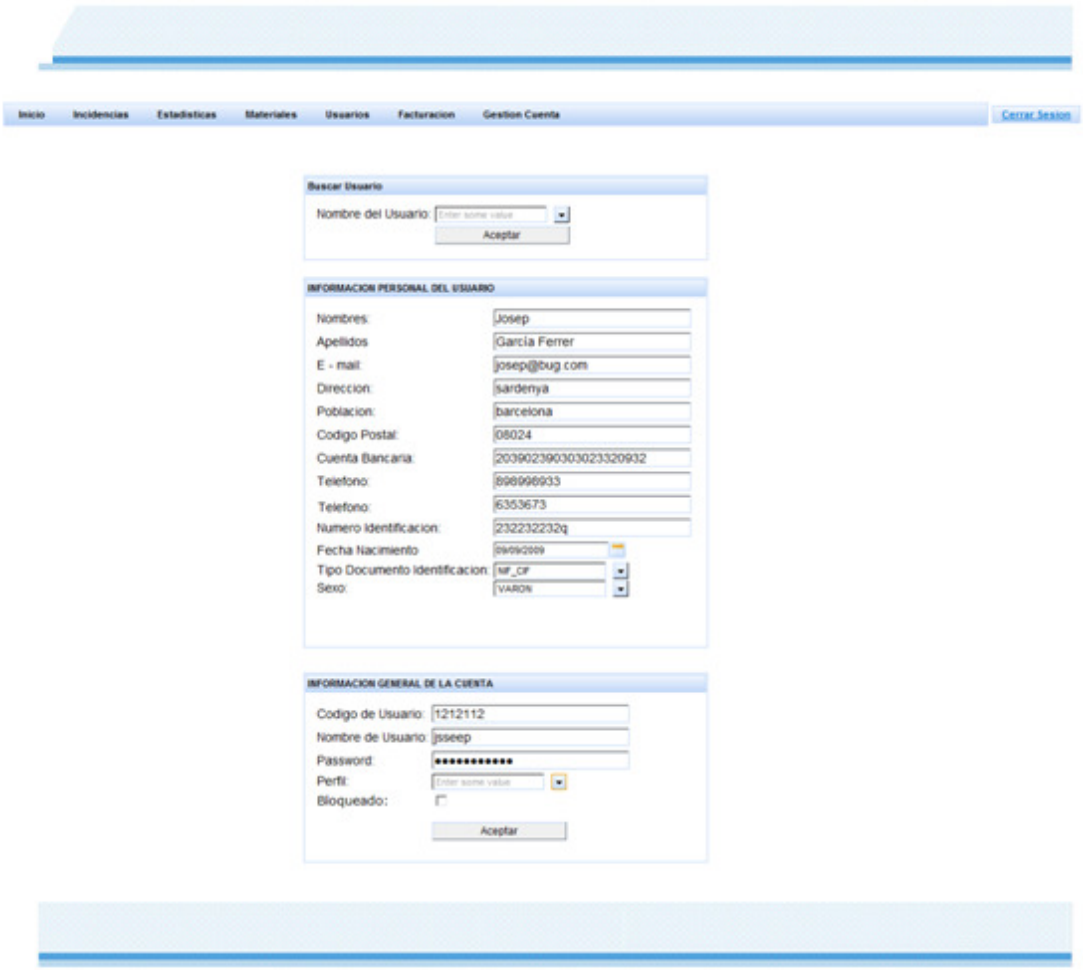

Figura 28- Operaciones usuarios

5.1.9. admonPerfiles.jsp

Invocado a través de admonperfiles.jsf, en este apartado se realizan todas las operaciones con los perfiles de los usuarios del sistema, desde la consulta de estos perfiles donde se muestran los diferentes permisos que tienen a las modificaciones que podemos efectuar sobre ellos, y pasando por la generación de los nuevos perfiles que se deseen crear. Se cargan dos formularios a la vez, que se visualiza uno u otro dependiendo de la selección que realicemos.

Asociar permisos a perfiles donde aparece un buscador que lista los perfiles que existen. Al seleccionar uno de ellos se muestra que permisos tiene y nos da la opción de modificarlos. El otro apartado Operaciones con perfiles nos permite crear los nuevos perfiles asociándoles sus permisos. Ambos apartados están asociados a la clase java PerfilesBackinBean.java

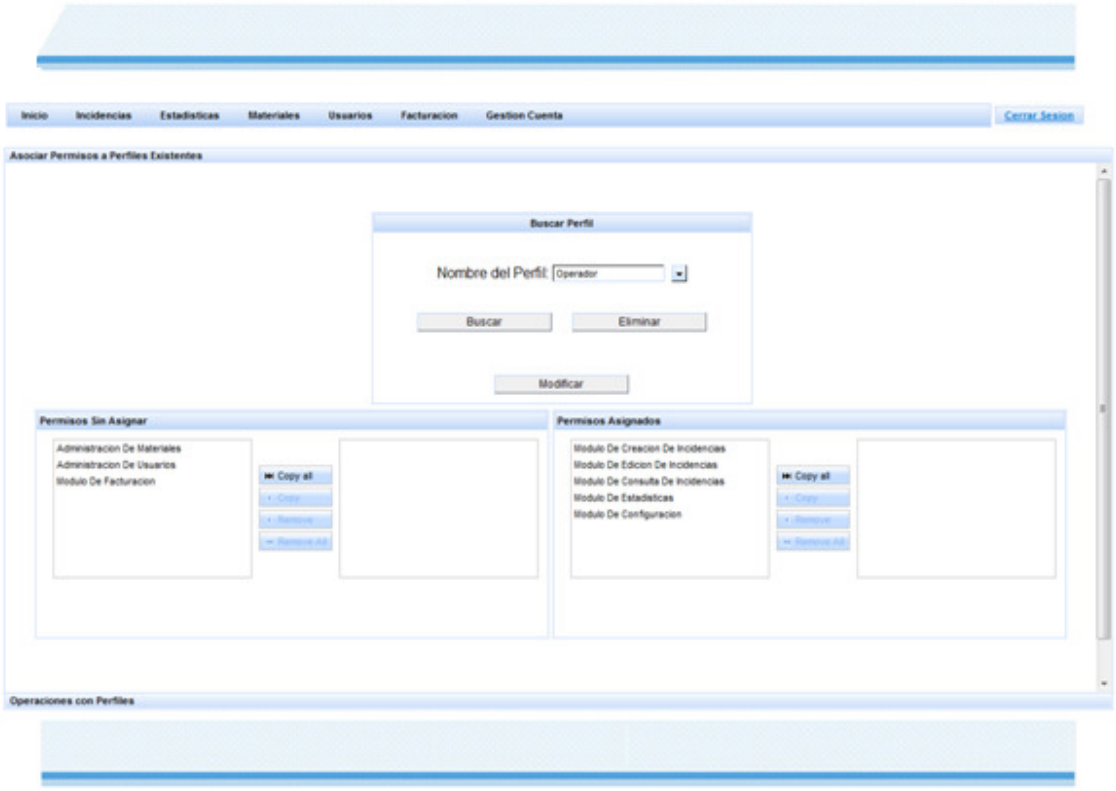

Figura 29- Operaciones perfiles

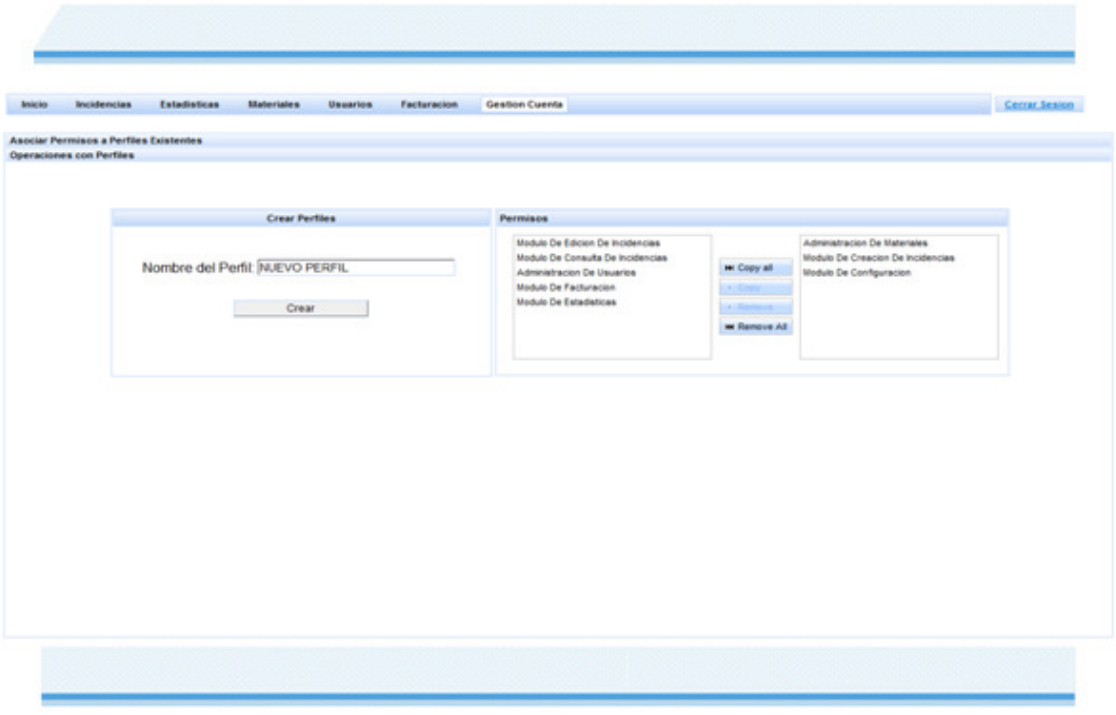

Figura 30- Crear perfiles

5.1.10. moduloFacturacion.jsp

Invocado a través de moduloFacturacion.jsf, En este apartado se realizan todas la operaciones de facturación del sistema desde la consulta de incidencias ya facturadas, hasta la generación de la factura de aquellas incidencias que han sido ya cerradas. Se cargan dos formularios a la vez, que se visualizan uno u otro dependiendo de la selección que realicemos. En "facturas generadas" aparece un buscador donde se listan las facturas que cumplen los criterios de búsqueda en un listado como los anteriormente descritos. Esta opción nos ofrece la posibilidad de ver el detalle de la factura, que invoca a moduloConsultaFacturacion.jsf que explicaremos más adelante.
El otro apartado es "generar Factura", donde una vez escogida la incidencia del listado de las posibles que podemos factura solo aparecerán aquellas que ya estén cerradas, y de esta forma podremos calcular el coste y generar la factura.
Ambos apartados están asociados a la clase java FacturaBackingBean.java

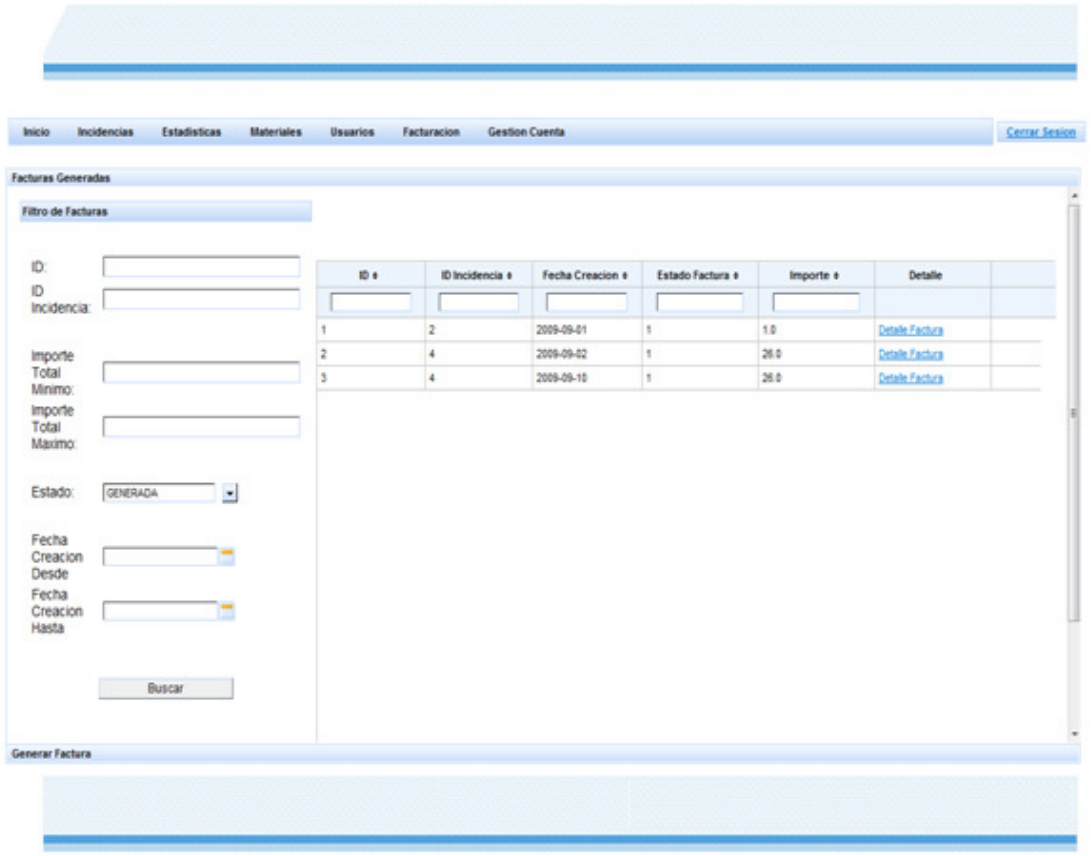

Figura 31- Consulta facturas

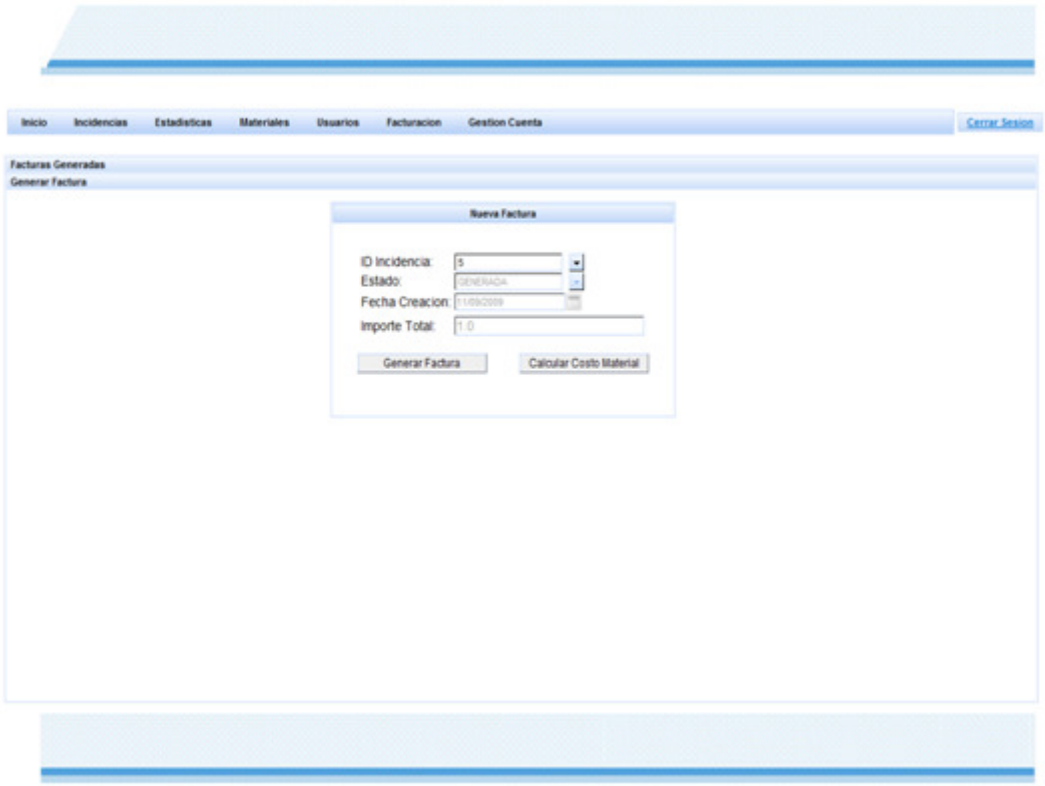

Figura 32- Crear facturas

5.1.11. miCuenta.jsp

Invocado a través de micuenta.jsf. En este apartado los usuarios pueden modificar sus datos personales, su pasword, etc... El formulario está asociado a la clase java MiCuentaBackingBean.java

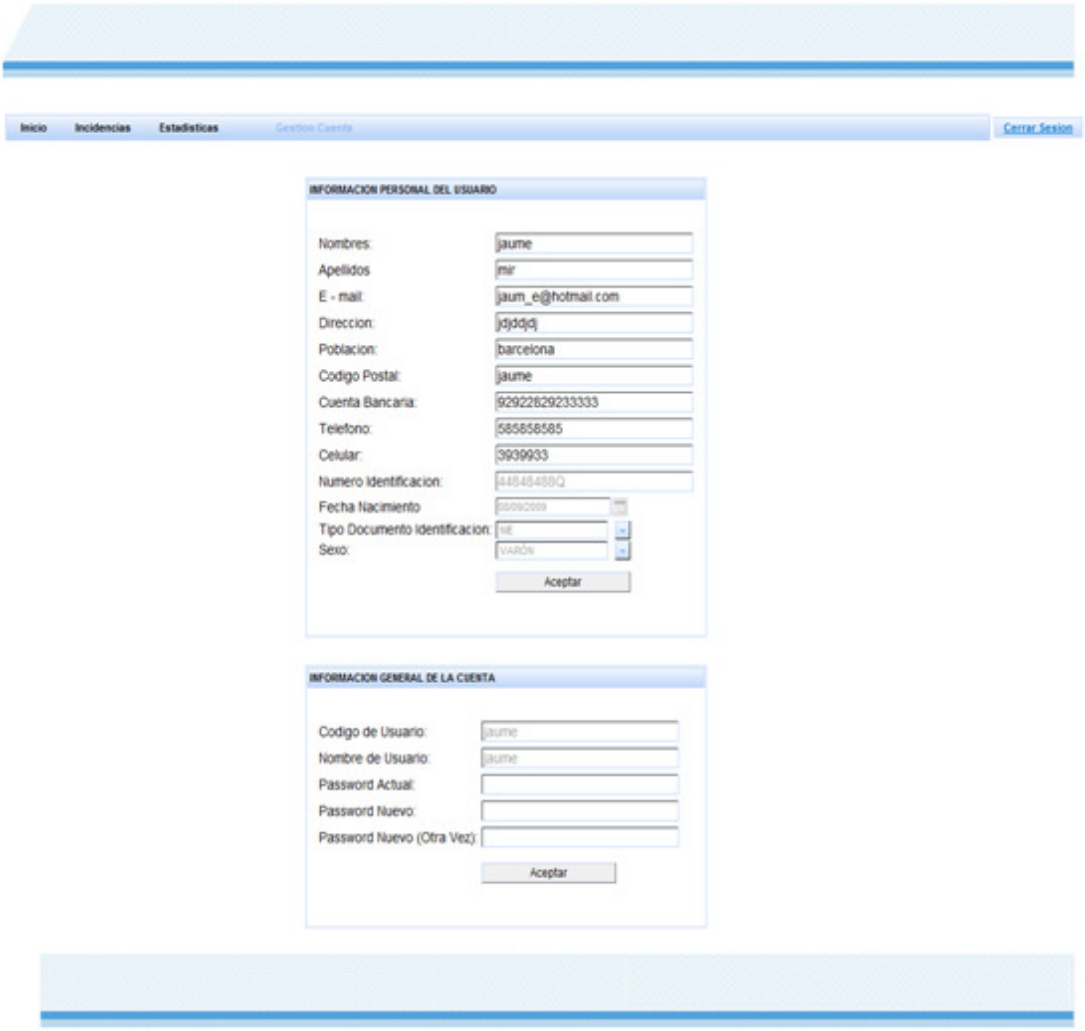

Figura 33- Gestión cuenta

6. Test de la aplicación

Testing es un proceso de planificación, preparación y ejecución que da visibilidad, y asesoramiento de la calidad y los riesgos asociados.

Testing es un proceso cuyo objetivo es encontrar defectos para dar una evaluación de la calidad mostrando las diferencias entre especificaciones y la aplicación desarrollada para dar confianza en el producto y aconsejar sobre la calidad y riesgos y que los responsables puedan tomar decisiones informadas.

Nos ofrece:

- **Aseguramiento de la calidad (QA):** Un patrón planificado y sistemático de todas las acciones necesarias para dar suficiente confianza que el producto cumple sus requerimientos de calidad. (Que no contiene problemas y que es capaz de realizar las tareas para que fue diseñado)

- **Control de la calidad (QC):** La evaluación que el productos (y subproductos) cumplen los requisitos establecidos.

Figura 34- Test

Las fases del testing las podenmos comparar con un Icerberg ya que son más largos los trabajos de preparación del testing, planificación que no lo que realmente parce, ya que el testing solo se ve la ejecución. Por ello es de vital importancia realizar una buena planificación para que se realice un test estructurado y con garantías.

Figura 35 - fases testing

:

En nuestra aplicación hemos realizado dos tipos de testing

- Testing Funcional
- Testing de Carga de la aplicación.

6.1. Test Funcional

Con el test funcional lo que se pretende es demostrar que la aplicación se comporta tal y como se ha indicado en las especificaciones, estos test funcionales los hemos realizado basándonos en la metodología TMAP, que nos indica que pasos se deben seguir para realizar un testing estructurado y con orientación a la calidad.

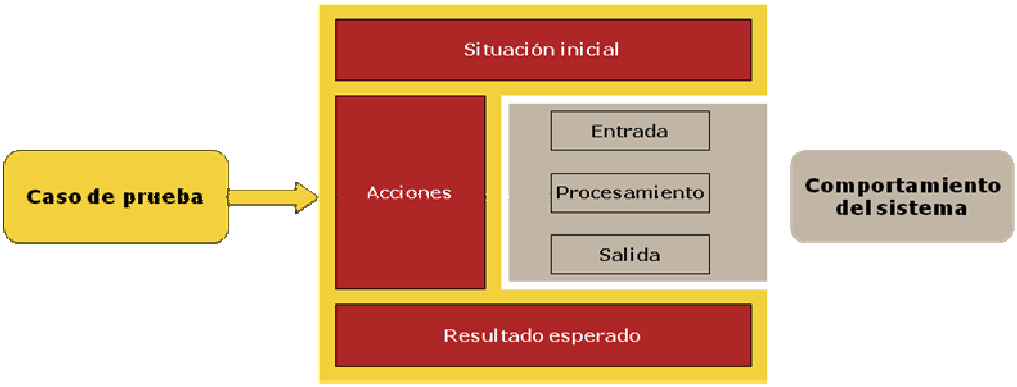

Figura 36- Tmap

A continuación voy a mostrar un ejemplo de cómo se ha realizado el testing de una partes de nuestra aplicación. En concreto el módulo que gestiona los perfiles de los usuarios.

6.1.1. Test funcional perfiles

Caso de Test	Test Case	Resultado Esperado	Resultado
1	**Crear perfil** : -Nombre: Director financiero		

	Seleccionar: Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Administración De Materiales Administración De Usuarios Modulo De Facturación Modulo De Estadísticas Modulo De Configuración	Se crea correctamente el perfil con los permisos indicados	OK
2	**Crear perfil** : -Nombre: Facturador **Seleccionar:** Modulo De Consulta De Incidencias Administración De Materiales Modulo De Facturación Modulo De Estadísticas Modulo De Configuración	Se crea correctamente el perfil con los permisos indicados	OK
3	**Crear perfil** : -Nombre: Director logístico **Seleccionar:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Modulo De Estadísticas Modulo De Configuración	Se crea correctamente el perfil con los permisos indicados	OK
4	**Crear perfil** : -Nombre: Mecánico Segundo Nivel **Seleccionar:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Modulo De Configuración	Se crea correctamente el perfil con los permisos indicados	OK
5	**Crear perfil** : -Nombre: Mecánico Primer Nivel **Seleccionar:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Modulo De Configuración	Se crea correctamente el perfil con los permisos indicados	OK
6	**Crear perfil** : -Nombre: Operador **Seleccionar:** º Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Modulo De Configuración	Se crea correctamente el perfil con los permisos indicados	OK
7	**Asociar Permisos** : **Seleccionar** -Nombre: Mecánico Primer Nivel	**Aparece en permisos Asignados:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Modulo De Configuración **Aparece en no permisos Asignados:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Administración De Usuarios Modulo Estadísticas	OK
8	**Asociar Permisos** : **Seleccionar** -Nombre: Mecánico Primer Nivel **Quitar permiso:** Modulo De Configuración	**Aparece en permisos Asignados:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias **Aparece en no permisos Asignados:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Administración De Usuarios Modulo De Configuración Modulo De Estadísticas	OK
9	**Asociar Permisos** : **Seleccionar** -Nombre: Mecánico Primer Nivel	**Aparece en permisos Asignados:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias	

	Añadir Permiso: Modulo De Estadísticas	Modulo De Consulta De Incidencias Modulo De Configuración Modulo De Estadísticas	OK
10	**Entrar al sistema** : **Login con usuario perfil mecánico:** -Nombre: jaume - Usuario 12345	**Al cargar la aplicación solo se puede tener acceso a las funcionalidades:** Modulo De Creación De Incidencias Modulo De Edición De Incidencias Modulo De Consulta De Incidencias Modulo De Configuración Modulo De Estadísticas	OK

6.2. Test de carga

Para éste apartado nos hemos utilizado una aplicación de carga en concreto Neoload, esta aplicación permite simular la conexión de varios usuarios simultanéate para ver cómo responde aplicación, en un entorno parecido al que se encontrará en la producción.

Otras de la funciones de los test de carga, es buscar cual es el límite de usuarios que permite conectar, las transacciones máximas que permite la base de datos. Todo ello para buscar cual es el umbral máximo de nuestra aplicación, para poder prever posibles problemas en el mercado.

6.2.1. Test de carga 1

Para realizar la Carga se ha hecho un flujo básico completo para 50 usuarios, aumentando de forma incremental cada minuto conectándose 5 usuarios nuevos durante 1 hora

Flujo Realizado:

1. Acceso a Aplicación
2. Login con Súper Usuario
3. Seleccionar Materiales
4. Modificar Material
5. Seleccionar material y buscar

Con este flujo estamos realizando un login, cargando los permisos que pertenecen al Súper Usuario y al realizar la búsqueda estamos cargando la base de datos al consultar una respuesta.

Test 50 usuarios	Resultados
Average pages/s	36.0
Average hits/s	265.0
Total pages	129664
Total hits	954370
Average Request response time	0.033s
Total hit errors	22
Average Page response time	0.175s
Total throughput	15911.0 MB
Average Throughput	35.35 Mb/s
Total users Launched	2569
Total action errors	0

Con los resultados obtenidos no se han detectado ninguna anomalía en todo el proceso y podemos concluir que la prueba ha sido satisfactoria.

6.2.2. Test de carga 2

Para realizar la Carga se ha hecho un flujo básico completo para 80 usuarios, aumentando de forma incremental cada minuto conectándose 5 usuarios nuevos durante 1 hora

Flujo Realizado:

1. Acceso a Aplicación
2. Login con Súper Usuario
3. Seleccionar Materiales
4. Modificar Material
5. Seleccionar material y buscar

Con este flujo estamos realizando un login, cargando los permisos que pertenecen al Súper Usuario y al realizar la búsqueda estamos cargando la base de datos al consultar una respuesta.

Test 50 usuarios	Resultados
Average pages/s	50.8
Average hits/s	283.4
Total pages	183003
Total hits	1020468
Average Request response time	0.063s
Total hit errors	26862
Average Page response time	0.269s
Total throughput	15037.0 MB
Average Throughput	33.41 Mb/s
Total users Launched	3623
Total action errors	0

Con los resultados obtenidos no se han detectado ninguna anomalía en todo el proceso y podemos concluir que la prueba ha sido satisfactoria.

6.2.3. Comparativa entre ambas pruebas

Donde A son pruebas con 50 usuarios y con 80 usuarios.

	A	B	%
Average pages/s	36.0	50.8	+41.1%
Average hits/s	265.0	283.4	+6.9%
Total pages	129664	183003	+41.1%
Total hits	954370	1020468	+6.9%
Average Request response time	0.033s	0.063s	+90.9%
Total hit errors	22	26862	+122,000%
Average Page response time	0.175s	0.269s	+53.7%
Total throughput	15911.0 MB	15037.0 MB	-5.5%
Average Throughput	35.35 Mb/s	33.41 Mb/s	-5.5%
Total users Launched	2569	3623	+41%
Total action errors	0	0	+0%

A: Test con 50 usuarios
B: Test con 80 usuarios

Los resultado obtenidos, son totalmente esperados, vemos que el sistema aguanta perfectamente 80 usuarios conectados a la vez, cosa que nunca pasará. La estimación que se ha realizado es que un pico máximo de uso de la aplicación podrán haber 20 usuarios. Con este estudio se podemos concluir que la aplicación de forma estable y cumple con mucha holgura las expectativas de rendimiento de la misma.

7. Conclusiones

En este proyecto se ha creado un sistema de incidencias que nos permite organizar de forma estructurada un proceso que era caótico y no funcionaba óptimamente. El objetivo principal era crear un entorno que facilitara el día a día de todas aquellas personas que trabajan en el almacén, desde los operadores, quienes reportan las incidencias de los elementos que les afectan en su trabajo, hasta los mecánicos, encargados de solucionar estos problemas.

También sirve como solución para los directivos de la empresa, otorgándoles una herramienta capaz de controlar las incidencias, el tiempo de resolución, el stock de los materiales necesarios para la solución de las incidencias, el coste que conlleva dicha resolución, que defectos pueden existir en la maquinaria, la gestión del trabajo de los usuarios, el control del trabajo realizado y la mejora de los sistemas de trabajo de la empresa.

Podemos decir que se han cumplido todos los objetivos marcados inicialmente tanto por parte de la empresa consultora, como todas las especificaciones que nos ha indicado nuestro cliente para satisfacer sus necesidades.

Exigencias Consultora: La creación de una aplicación que fuera modular, reutilizable y escalable con la intención de poder ser utilizada para futuros clientes, realizando los mínimos cambios posibles. Todos estos objetivos se han cumplido satisfactoriamente.

Exigencias del cliente: Las exigencias del cliente, se han cumplido perfectamente tal como se ha explicado a lo lardo de toda la memoria.

Por último considero muy importante destacar que ha sido para mí una experiencia muy enriquecedora poder realizar este proyecto en una empresa, ya que me ha proporcionado la oportunidad de ver el ciclo de vida de un proyecto desde sus inicios hasta su puesta en producción y trabajar al lado de profesionales con mucha experiencia, factor que facilita en gran medida el trabajo.

Me ha permitido poder desempeñar todas las tareas que conforman un proyecto, como el análisis y la especificación de requisitos, el diseño de una solución concreta delante de una serie de necesidades y requisitos, diseñar una arquitectura que permitiera la transformación de estos requisitos en la forma de la solución. Obteniendo esa experiencia necesaria por la que todo ingeniero tiene que pasar.

Quizás una de las partes que me ha quedado pendiente en el proyecto es la económica,. Desconozco totalmente cuál ha sido el presupuesto de la aplicación, ni de que como se ha diseñado esta oferta para el cliente. Creo que hubiera sido un punto de vista diferente al técnico, que hubiera resultado muy interesante para comprender aún mejor el mundo de los negocios y obtener experiencia en una de los puntos claves de un proyecto: la económica ya que sin esta no se realizarían los proyectos.

7.1. Mejoras Futuras

Las mejoras futuras que se podrían realizar a esta aplicación podrían ser muchas, y fáciles de realizar ya que la misma es totalmente modular y simplemente le deberíamos añadir nuevas herramientas.

Mejoras que a mi entender podrían ser útiles:

El módulo de estadísticas ha sido pensado para que el director financiero saque conclusiones sobre el funcionamiento del sistema y sirva como ayuda ante la toma de decisiones para mejorarlo. Se podrían añadir funcionalidades donde se obtuvieran más datos, por ejemplo realizar gráficas de las facturas para llevar el control económico del gasto mensual de las incidencias resueltas. También una gráfica donde se viera el estado del almacén virtual y de esta forma realizar un control más ágil del Stock de los materiales. Actualmente para ver cantidad de materiales que disponemos debemos mirar material por material.

Otra mejora que se podría realizar, sería situar la aplicación en un dominio con Acceso desde internet, esto sería muy útil ya que los mecánicos poseen un PDA donde reciben las notificaciones de las incidencias por correo electrónico.
Si el sistema fuera accesible desde internet podrían conectarse desde su móvil a Aplicación y leer toda la información de las incidencias, también podrían ver que tareas tienen pendientes sin necesidad de buscar un PC.
Para realizar esto también se debería retocar la aplicación y de esta forma se vería correctamente desde nuestro móvil.
Los pasos que se deberían seguir son: la creación de directorio independiente para la versión móvil de un sitio (WAP), scripts de re-direccionamiento, idea de estructura, esqueleto en hml y css de una posible estructura, cambio de fuentes y plugin.

8. ANEXO 1

En este anexo se van a mostrar algunos ejemplos del código implementado.

8.1. Código Fuente

Los BackingBean que posee nuestra aplicación son:

EditarIncidenciaBackingBean.java
ConsultarIncidenciasBackingBean.java
EstadisticasBackingBean.java
FacturaBackingBean.java
InsertarIncidenciaBackingBean.java
LoginBackinBean.java
MaterialesBackingBean.java
MiCuentaBackingBean.java
PerfilesBackinBean.java
UsuariosBackingBean.java

Los JSP que posee la aplicación son:

admonPerfiles.jsp
admonUsuarios.jsp
index.jsp
inicio.jsp
miCuenta.jsp
moduloConsultaFacturacion.jsp
moduloConsultaIncidencias.jsp
moduloCreacionIncidencias.jsp
moduloEdicionIncidencias.jsp
moduloEstadisticas.jsp
moduloFacturacion.jsp
moduloMateriales.jsp

8.1.1. Un ejemplo de código

A continuación se muestra el código del JSP que muestra la página de las estadísticas y seguidamente se describe la clase EstadisticasBackingBean.java que está asociada a ella

/* Estadisticas JSP*/

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
```

```
<%@ taglib uri="http://sourceforge.net/projects/jsf-comp" prefix="c"
%>
<%@ taglib prefix="f"  uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h"  uri="http://java.sun.com/jsf/html"%>
<%@ taglib prefix="x" uri="http://java.sun.com/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib prefix="rich" uri="http://richfaces.org/rich" %>
<%@ taglib prefix="a4j" uri="http://richfaces.org/a4j" %>

<html>
      <link href="css/panel.css" rel="stylesheet" type="text/css" />
      <link href="css/texto.css" rel="stylesheet" type="text/css" />

      <script type="text/javascript">
            function imprimir(){
                  window.print();
            }
      </script>

      <head>
      <title>Estadisticas</title>
    </head>
      <body>
            <f:view>
                  <rich:panel styleClass="header">
                        <img alt="header" src="images/header.gif"/>
                  </rich:panel>
                  <h:form>
                        <h:panelGrid columns="2">
                              <rich:toolBar width="1100px">
                              <rich:dropDownMenu >
                              <f:facet name="label">
                                    <h:panelGroup>
                                        <h:outputText
value="Inicio"/>
                                    </h:panelGroup>
                                </f:facet>
                              <rich:menuItem submitMode="ajax"
action="#{loginBackinBean.actME}" value="Mis Incidencias"/>
                              </rich:dropDownMenu>
                              <rich:dropDownMenu>
                              <f:facet name="label">
                                    <h:panelGroup>
                                        <h:outputText
value="Incidencias"/>
                                    </h:panelGroup>
                                </f:facet>
                              <rich:menuItem submitMode="ajax"
value="Crear Incidencia"
action="#{insertarIncidenciaBackingBean.actME}"
rendered="#{loginBackinBean.permisoModuloCreacionIncidencias}"/>
                              <rich:menuItem submitMode="ajax"
value="Editar Incidencia"
action="#{editarIncidenciaBackingBean.actME}"
rendered="#{loginBackinBean.permisoModuloEdicionIncidencias}"/>
                              <rich:menuItem submitMode="ajax"
value="Consultar Incidencias"
action="#{consultarIncidenciasBackingBean.actME}"
rendered="#{loginBackinBean.permisoModuloConsultaIncidencias}"/>
```

```
                                </rich:dropDownMenu>
                                <rich:dropDownMenu  disabled="true">
                                <f:facet name="label">
                                        <h:panelGroup>
                                            <h:outputText
value="Estadisticas"/>
                                        </h:panelGroup>
                                    </f:facet>
                                <rich:menuItem submitMode="ajax"
value="Ver Estadisticas"/>
                                </rich:dropDownMenu>
                                <rich:dropDownMenu
rendered="#{loginBackinBean.permisoAdministracionMateriales}">
                                <f:facet name="label">
                                        <h:panelGroup>
                                            <h:outputText
value="Materiales"/>
                                        </h:panelGroup>
                                    </f:facet>
                                <rich:menuItem submitMode="ajax"
value="Operacion Materiales" action="#{materialesBackingBean.actME}"/>
                                </rich:dropDownMenu>
                                <rich:dropDownMenu
rendered="#{loginBackinBean.permisoAdministracionUsuarios}">
                                <f:facet name="label">
                                        <h:panelGroup>
                                            <h:outputText
value="Usuarios"/>
                                        </h:panelGroup>
                                    </f:facet>
                                <rich:menuItem submitMode="ajax"
value="Operaciones Usuarios " action="#{usuariosBackingBean.actME}"/>
                                <rich:menuItem submitMode="ajax"
value="Operaciones Perfiles " action="#{perfilesBackinBean.actME}"/>
                                </rich:dropDownMenu>
                                <rich:dropDownMenu>
                                <f:facet name="label">
                                        <h:panelGroup>
                                            <h:outputText
value="Facturacion"
rendered="#{loginBackinBean.permisoModuloFacturacion}"/>
                                        </h:panelGroup>
                                    </f:facet>
                                <rich:menuItem submitMode="ajax"
value="Crear Factura" action="memf"/>
                                <rich:menuItem submitMode="ajax"
value="Editar Factura" action="memcf"/>
                                </rich:dropDownMenu>
                                <rich:dropDownMenu>
                                <f:facet name="label">
                                        <h:panelGroup>
                                            <h:outputText value="Gestion
Cuenta"/>
                                        </h:panelGroup>
                                    </f:facet>
                                <rich:menuItem submitMode="ajax"
action="memc" value="Mi Cuenta" />
                                </rich:dropDownMenu>
                            </rich:toolBar>
                            <rich:toolBar>
```

```
                              <h:commandLink value="Cerrar Sesion"
action="csme" actionListener="#{loginBackinBean.logOut}"/>
                        </rich:toolBar>
                  </h:panelGrid>
            </h:form>

            <!-- Inicio Logica Pagina -->

            <h:form id="form1" style="text-align: center;">
                  <c:chart id="chart2"
datasource="#{estadisticasBackingBean.categoryDataset}" type="bar"
is3d="true" title="Estado de la Incidencias"/>
                  <tr></tr>
                  <h:commandButton onclick="imprimir();"
styleClass="boton"  value="Imprimir Grafica"/>
            </h:form>

            <!-- Fin Logica Pagina -->

            <rich:panel styleClass="header">
                  <img alt="header" src="images/footer.gif"/>
            </rich:panel>
      </f:view>
   </body>
</html>
```

/* Estadísticas BackingBean*/

```
package business.backingbeans;

import org.jfree.data.category.DefaultCategoryDataset;

import business.ejb.GestorIncidencia;
import business.vo.Estadisticas;

public class EstadisticasBackingBean {

      GestorIncidencia gestorIncidencia = new GestorIncidencia();
      Estadisticas est = new Estadisticas();

      public String actMCI(){
            try {
                  getCategoryDataset();
                  return "mcime";
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return "";
      }

      public String actMEI(){
            try {
                  getCategoryDataset();
                  return "meime";
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return "";
      }

      public String actMBI(){
```

```
		try {
			getCategoryDataset();
			return "mbime";
		} catch (Exception e) {
			e.printStackTrace();
		}
		return "";
	}

	public String actMM(){
		try {
			getCategoryDataset();
			return "mmme";
		} catch (Exception e) {
			e.printStackTrace();
		}
		return "";
	}

	public String actAU(){
		try {
			getCategoryDataset();
			return "aume";
		} catch (Exception e) {
			e.printStackTrace();
		}
		return "";
	}

	public String actAP(){
		try {
			getCategoryDataset();
			return "apme";
		} catch (Exception e) {
			e.printStackTrace();
		}
		return "";
	}

	public String actCF(){
		try {
			getCategoryDataset();
			return "mfme";
		} catch (Exception e) {
			e.printStackTrace();
		}
		return "";
	}

	public String actMCF(){
		try {
			getCategoryDataset();
			return "mcfme";
		} catch (Exception e) {
			e.printStackTrace();
		}
		return "";
	}

	public String actMC(){
		try {
```

```
                getCategoryDataset();
                return "mcme";
            } catch (Exception e) {
                e.printStackTrace();
            }
            return "";
      }

      public String actI(){
            try {
                getCategoryDataset();
                return "ime";
            } catch (Exception e) {
                e.printStackTrace();
            }
            return "";
      }

      public DefaultCategoryDataset getCategoryDataset(){
          try {
                est = gestorIncidencia.getEstadisticas();
            } catch (Exception e) {
                e.printStackTrace();
            }

        String s = "Abiertas";
        String s1 = "Asignadas";
        String s2 = "Cerradas";
        String s3 = "Facturadas";
        String s4 = "Solucionadas";
        String s5 = "Estados";

        DefaultCategoryDataset defaultcategorydataset = new
DefaultCategoryDataset();
        defaultcategorydataset.addValue(est.getAbiertas(), s, s5);
        defaultcategorydataset.addValue(est.getAsignadas(), s1, s5);
        defaultcategorydataset.addValue(est.getCerradas(), s2, s5);
        defaultcategorydataset.addValue(est.getFacturadas(), s3, s5);
        defaultcategorydataset.addValue(est.getSolucionadas(), s4,
s5);

          return defaultcategorydataset;
      }
}
```

8.1.2. Envio de correo para notificar incidencias

```
package business.correo;

public class HelperCorreos {

      private static String FROM = "aplicación@gmail.com";

      protected static void enviarCorreo(Correo correo) {
            try {
                // Propiedades de la conexión
                Properties props = new Properties();

                props.setProperty("mail.smtp.host","smtp.gmail.com");
```

```
                  props.setProperty("mail.smtp.starttls.enable",
"true");
                  props.setProperty("mail.smtp.port", "587");

      props.setProperty("mail.smtp.user","correo@gmail.com");
      props.setProperty("mail.smtp.auth", "true");

                  // Preparamos la sesion
                  Session session = Session.getDefaultInstance(props);

                  // Construimos el mensaje
                  MimeMessage message = getMimeMessage(session,
correo);
                  // Lo enviamos.
                  Transport t = session.getTransport("smtp");
                  t.connect("xxxxx@gmail.com", "pwd");
                  t.sendMessage(message, message.getAllRecipients());

                  // Cierre.
                  t.close();
            } catch (Exception e) {
                  SCILogger.addError("Error enviando correo
electronico", e);
            }
      }

      protected static InternetAddress[] convertToAddress(String[]
tos)
                  throws AddressException {
            ArrayList<InternetAddress> address = new
ArrayList<InternetAddress>();
            for (String to : tos) {
                  address.add(new InternetAddress(to));
            }
            return (InternetAddress[]) address.toArray();
      }

      protected static MimeMessage getMimeMessage(Session session,
Correo correo)
                  throws MessagingException {
            MimeMessage message = new MimeMessage(session);
            message.setFrom(correo.getFrom());
            message.addRecipients(Message.RecipientType.TO,
correo.getTo());
            message.setSubject(correo.getSubject());
            message.setText(correo.getBody());
            return message;
      }

      protected static Correo getCorreo(Seguimiento seguimiento,
                  InternetAddress[] tos) throws AddressException {
            Correo correo = new Correo();
            correo.setBody(seguimiento.getSeguimiento());
            correo.setFrom(new InternetAddress(FROM));
            correo.setSubject("Notificacion Incidencia "
                        + seguimiento.getIdIncidencia());
            correo.setTo(tos);
            return correo;

      }
```

9. Anexo 2

En este anexo se van a mostar las operaciones que se realizan con la base de datos.

9.1. Ejemplos de Querys utilizadas

9.1.1. Query para Insertar datos de la persona nueva en tabla personas

```
SQL_INSERT_PERSONA = "INSERT INTO personas("+ " id,
tipoidentificacion, identificacion, nombre, apellidos,"+ " sexo,
fechanacimiento, correoelectronico, telefono, telefonomovil,
direccion,"+ " poblacion, codigopostal, cuentabancaria, codigologin,
contrasena,"+ " bloqueado, fechaalta, codigopersona) "+ " VALUES
(nextval('seq_personas'), ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?)";
```

9.1.2. Query para devuelve los datos de la persona que coincida con el id correspondiente

```
SQL_GET_PERSONA_IDENTIFICACION = "SELECT id AS TOId,
tipoidentificacion AS TipoIdentificacion, identificacion AS
Identificacion, nombre AS Nombre, apellidos AS Apellidos, sexo AS
Sexo,"+ " fechanacimiento AS FechaNacimiento, correoelectronico AS
CorreoElectronico, telefono AS Telefono, telefonomovil AS
TelefonoMovil,"+ " direccion AS Direccion, poblacion AS Poblacion,
codigopostal AS CodigoPostal, cuentabancaria AS CuentaBancaria,
codigologin AS CodigoLogin,"+ " contrasena AS Contrasena, bloqueado AS
Bloqueado, fechaalta AS FechaAlta, codigopersona AS CodigoPersona "+ "
FROM personas WHERE identificacion=?";
```

9.1.3. Query para modificar un material ya existente en nuestra base de datos

```
SQL_UPDATE_MATERIAL = "UPDATE material" + " SET nombre=?,
existencias=?, precio=?, descripcion=? " " WHERE id=?";
```

9.1.4. Query para el cálculo del numero de incidencias para mostrar las estadísticas

```
SQL_GET_ESTADISTICAS_ABIERTA = "SELECT COUNT(1) AS TO_ABIERTA FROM
incidencia WHERE idestado=1";

SQL_GET_ESTADISTICAS_ASIGNADA = "SELECT COUNT(1) AS TO_ASIGNADA FROM
```

```
incidencia WHERE idestado=2";

SQL_GET_ESTADISTICAS_SOLUCIONADA    = "SELECT COUNT(1) AS
TO_SOLUCIONADA FROM incidencia WHERE idestado=3";

SQL_GET_ESTADISTICAS_CERRADA = "SELECT COUNT(1) AS TO_CERRADA FROM
incidencia WHERE idestado=4";

SQL_GET_ESTADISTICAS_FACTURADA = "SELECT COUNT(1) AS TO_FACTURADA FROM
incidencia WHERE idestado=5";
```

9.1.5. Query para eliminar un material de la base de datos

```
SQL_DELETE_MATERIAL = "DELETE FROM material WHERE id=?";
```

9.1.6. Query para eliminar un material de la base de datos

```
SQL_INSERT_NEW_MATERIAL_INCIDENCIA = "INSERT INTO
material_incidencia(id, idmaterial, idincidencia, cantidad)"+" VALUES
(nextval('seq_materiales_incidencias'), ?, ?, ?)";
```

9.2. Conexión con la BBDD

La conexión de la base se realiza detro de la DataAcces.java y los datos del Datadriver, data basePass, data base user los coge del archivo DBConfig.properties.

```
// cerrar la conexion
      private void closeConnection() {
            try {
                  if (conn != null)
                        conn.close();
                  conn = null;
            } catch (Exception e) {
                  SCILogger.addError("Error cerrando la conexion con la
base de datos ", e);
            }
      }

// Open Connection
      private void openConnection() {
            try {
                  if (conn == null) {
                        Class.forName(dataDriver);
                        String password = databasePass;
                        String hostName =
InetAddress.getLocalHost().getHostName();
                        if("ibcn-pc75".equals(hostName)){
                              password = databasePass2;
                        }
```

```
                        conn =
DriverManager.getConnection(dataSourceName, databaseUser, password);

                    }
                    if (conn != null) {
                        SCILogger.addLog("Conexion con la base de datos
creada con exito.");
                    }
            } catch (Exception e) {
                    SCILogger.addError("Error abriendo la conexion con la
base de datos ", e);
            }

      }
}
```

9.2.1. DBConfig.properties

Archivo de propiedades que establece la conexión con nuestra base de datos PostgreSQL.

```
# ------------------------------------------------------------------------
---------
# DATABASE CONFIGURATION
# ------------------------------------------------------------------------
---------
DATASOURCE=jdbc:postgresql://localhost/aplicación
DATABASEUSER=postgres
DATABASEPASS=sa
DRIVER=org.postgresql.Driver
```

10. BIBLIOGRAFIA

Neil Matthew ab Richard Stons: *Beginig Databases with PostgreSQL: Fom novice to experts*. Apress 2005

John Ferguson Smart JSF *Jumpstart: Getting up to speed with Java Server Faces*. WAKALEO CONSULTING Ltd (2007)

Tim Koomen, Leo van der Aalst, : *TMAPNEXT: for result-driven testing* UTN Publisher's (2006)

11. LINKS EXTERNOS

- Java Platform, Enterprise Edition (Java EE).
  - http://java.sun.com/javaee/javaserverfaces/
- JavaServer Faces Technology 1.2 Maintenance Release Documentation
  - http://java.sun.com/javaee/javaserverfaces/reference/api/
- Tutorial JSF
  - http://www.coreservlets.com/JSF-Tutorial/jsf1/index.html
- Richfaces developer guide
  - http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html_single/
- PorstgreSQL – Todo sobre PotsegreSQL
  - http://www.postgresql.org/
- Apache Tomcat 6.0
  - http://tomcat.apache.org/tomcat-6.0-doc/index.html
- Metodología testing
  - http:/.tmap.net/
- Diseño de la base de datos
  - http://dccs1.dcc.uab.es/teach/a20350/teoria/tema3.pdf
- Ciclo de vida de un proyecto
  - http://www.monografias.com/trabajos4/cicdevida/cicdevida.shtml

Trabajo realizado por: **Jaume Mir Armada**

Bellaterra, 12 de Septiembre de 2009

RESUM

Aquest projecte té com objectiu fer un sistema d'incidències pel control d'un magatzem robotitzat i del cost que comporta la resolució d'aquestes. D'aquesta manera es pretén canviar un mètode de treball rudimentari, passant a un altre totalment estructurat que serveixi per millorar el actual. Per això s' ha creat una aplicació basada en la tecnologia JSF, què permet aconseguir els nostres objectius. L'aplicació inclou una base de dades feta en PostGres SQL on es guardarà tota la informació que s'utilitza en el nostre aplicació. En aquesta memòria s'ha volgut explicar quins han estat els diferents passos que es van donar per poder assolir tots els nostres objectius. En la primera part realitzem una introducció a la situació del problema que es vol solucionar i descriurem els objectius que en hem marcat. Seguidament s'analitzen els requeriments i es detalla com s'implementen les diferents parts de l'aplicació. Per acabar parlem sobre el procés de testing realitzat que ens assegura el correcte funcionament del programa. En conclusió. Resumint el objectiu principal d'aquest projecte consisteix en crear una eina que a la vegada que facilita la feina dels treballadors del magatzem, serveixi per al control de tot el sistema de treball als directius de l'empresa.

RESUMEN

Este proyecto tiene como objetivo crear un sistema de incidencias para el control de un almacén robotizado y del coste que conlleva la resolución de estas. De esta forma se pretende cambiar un método de trabajo rudimentario, pasando a otro totalmente estructurado que sirva para mejorar el actual. Por ello se ha creado una aplicación basada en la tecnología JSF, que permite fácilmente lograr los objetivos marcados. El aplicativo incluye una base de datos en PostGresSQL donde se guardará toda la información que se utiliza en el sistema. En esta memoria se ha pretendido explicar cuáles han sido los diferentes pasos que para lograr todos nuestros objetivos, explicando cada uno de los procesos de análisis y desarrollo para lograrlos. En la primera parte realizamos una introducción a la situación del problema a solventar y describimos los objetivos que nos hemos marcado. Seguidamente se analizan los requerimientos y se detalla cómo se implementan las diferentes partes que tiene nuestra aplicación. Para concluir hablamos del proceso de testing realizado, que nos asegura el correcto comportamiento del programa. Resumiendo el objetivo principal de este proyecto es crear una herramienta que a la vez que facilita el trabajo de los empleados de un almacén, sirva para el control del sistema de trabajo a los directivos de la empresa.

ABSTRACT

This project aims to create a bug system for control an automatic warehouse and the cost involved in the resolution of this bugs. We want change a rudimentary, working method, growing up to another fully structured to improve the current. Therefore we created an application based on JSF, technology that lets us easily achieve our objectives. The application includes a database in PostGreSQL where the information used in the system will be saved. This memory wants to explain what steps we pass to achieve our goals, explaining each analysis and development processes to achieve them. In the first part we make an introduction to the situation of the problem to solve and we describe the objectives we have set ourselves. Then discusses the requirements and detailed how implement the parts of our application. In the conclusion we talk about the testing process, for be sure the program works correctly. For Abstract the main objective of this project is to create a tool that facility the work of warehouse employees, serving to the managers of the company for control the process working.