Gestión de Incidencias

Estudiante
Alfredo Ruiz Egea

Consultor
Antoni Oller Arcas

Fecha: 09/01/2006

Memoria del Trabajo de Fin de Carrera – J2EE
Enginyeria Tècnica en Informàtica de Gestió
Universitat Oberta de Catalunya

Resumen

La presente memoria del TFC “Gestión de Incidencias” muestra el proceso de desarrollo de una aplicación bajo el paradigma de J2EE.

Por un lado describe los pasos realizados en materia de especificación, análisis y diseño utilizando UML como herramienta fundamental de modelado, por otro pretende mostrar que la utilización de frameworks y la aplicación de patrones de diseño facilita sustancialmente el proceso de desarrollo.

La base del desarrollo es la implementación del patrón MVC (Modelo-Vista-Controlador) a partir del Framework Struts, para la vista y el controlador, y la implementación de un EJB (Enterprise Java Bean), para el modelo.

Para la gestión de la persistencia, el uso de un ORM (Hibernate en este caso), permite tener una solución elegante y totalmente desacoplada del gestor de base de datos utilizado.

Palabras clave: Gestión Incidencias, J2EE, MVC, Patrones de diseño, Struts, ORM, Hibernate, EJB.

Area TFC: J2EE

Índice de Contenido

Índice de figuras

1.- Introducción

1.1.- Justificación del TFC y contexto

La selección del área sobre la que realizar el TFC la basé en dos aspectos que consideré importantes. Por un lado, el área debería conjugar la mayor cantidad de conocimientos adquiridos durante estos últimos años de estudio, y por otro, que me permitiera continuar aprendiendo cosas nuevas.

El resultado de la selección fue el área de J2EE que me permitía introducirme en el desarrollo de componentes y aplicaciones distribuidas, y donde además convergen la mayoría de conceptos tecnológicos punteros en la actualidad.

He de reconocer que partía con conocimientos mínimos de J2EE, de las tecnologías que lo envuelven, y de las herramientas de desarrollo que se utilizan habituálmente; pero justamente su descubrimiento ha sido uno de los aspectos más gratos de este proyecto.

El producto final, la aplicación Gestión de Incidencias, pretende crear un sistema que aporte facilidad y agilidad en la introducción de incidencias, comodidad en su seguimiento y sencillez en el registro de intervenciones. Todo ello realizado desde un navegador web (thin client) y por tanto con unos requisitos de hardware y software mínimos para el usuario.

1.2.- Objetivos del TFC

Los Trabajos de Fin de Carrera, tienen dos tipos de objetivos diferenciados: unos derivados de las funcionalidades del producto a desarrollar y otros académicos, como síntesis de la carrera.

Si bien las funcionalidades del producto final no pueden ser, inicialmente, demasiado ambiciosas (al fin y al cabo el tiempo es limitado), sí lo son los objetivos académicos que se desprenden de la realización de este TFC.

Estos objetivos podrían resumirse en conseguir capacitación en el desarrollo de aplicaciones bajo el paradigma de J2EE y para ello será necesario completar los siguientes puntos:

- Seguimiento de la metodología de Gestión de Proyectos para el desarrollo de productos de software orientados a objeto estudiada en la carrera para el desarrollo de un producto real, y de esta forma comprobar su eficacia y asimilar sus procedimientos.

- Adquisición y asimilación de la tecnología J2EE poniéndola en marcha en un producto real.

- Uso de patrones de diseño así como últimas tendencias en el desarrollo de software que pudieran ser interesantes.

El hecho de que el presente proyecto se enmarque dentro de la realización de un TFC y de una temática concreta (J2EE) nos impone unas restriciones muy claras:

- **Restricciones tecnológicas**: La temática del TFC nos impone que la realización del proyecto se enmarque dentro del paradigma de J2EE y por tanto en el momento del diseño y la implementación habrá que tenerlo en cuenta.

- **Restricciones temporales**: Las entregas de los subroductos del TFC así como la entrega final tienen unas fechas concretas a las que habrá que adaptar la planificación del proyecto.

- **Restricciones documentales**: La documentación a entregar está definida en el plan de la asignatura de TFC y por tanto habrá que tenerlo en cuenta a la hora de planificar el proyecto. Además de la documentación habitual en los desarrollos de software, el plan de la asignatura nos pide realizar una memoria y una presentación que explique el proceso de desarrollo.

- **Restricciones organizativas**: Todo el trabajo de planificación, análisis, diseño, implementación y documentación lo realizará el alumno bajo la supervisión del consultor de la asignatura.

1.3.- Enfoque y método seguido.

Para el proceso de desarrollo, he seguido el **ciclo de vida iterativo e incremental**, guiado por los casos de uso, que tan buena prensa tiene actualmente entre la comunidad de desarrolladores.

Este ciclo de vida, pretende evitar la “paralisis by analysis” que puede surgir cuando se pretende definir un sistema de una forma definitiva, al intentando analizar todas las variantes posibles.

He puesto énfasis en seguir todos los pasos “academicos”, utilizando UML como herramienta de modelado, guiado por los casos de usos y viendo que la información que se añade al proceso de desarrollo se desprende del análisis de información de procesos anteriores.

La realización de una rigurosa lista de tareas y la definición de la información a generar en cada paso es fundamental para el cumplimiento de los objetivos.

1.4.- Planificación del proyecto.

1.4.1.- Introducción

Como Proyecto de Fin de Carrera, dentro del area de J2EE, se pretende desarrollar una aplicación de **Gestión de Incidencias** que funcione basada en esta arquitectura.

La líneas generales de la aplicación son las siguientes:

- El usuario final o el personal de soporte puede crear y consultar las incidencias mediante un font-end web.
- Personal de soporte asigna las incidencias a técnicos concretos para su resolución.
- Los técnicos reportan las acciones realizadas y el estado de la incidencia hasta su resolución.

También se pretende que el desarrollo sea fácilmente modificable y extensible para poder añadirle nuevas funcionalidades de forma fácil y poco costosa. En este punto nos ayudará la arquitectura J2EE.

1.4.2.- Organización del Equipo

Debido a la última restricción organizativa del punto 1.2 no tenemos margen para elegir una organización de equipo diferente de la aquí expuesta. Como mucho podríamos decir que en caso de que el trabajo hubiera sido en grupo y debido al carácter académico del proyecto, podríamos haber escogido una estructura democrática con o sin coordinador rotativo.

Por tanto, todo el volumen de trabajo recaerá en el alumno bajo la supervisión del consultor. Se mantendrá una comunicación fluida con el consultor de forma que esté informado de la evolución del proyecto y se le enviará periódicamente los subproductos que se vayan generando para su supervisión con la idea de que pueda orientar a tiempo al alumno en la realización del proyecto.

1.4.3.- Recursos de Software y Hardware

Los recursos de hardware para el desarrollo del proyecto están limitados a los recursos informáticos que posea el alumno, siendo los mínimos requeridos los que la UOC recomienda para la realización de la carrera.

Si bien el proyecto a realizar pretende implementar una arquitectura distribuida (J2EE), eso no impide que su desarrollo y pruebas se puedan realizar en un único ordenador que funcionará de forma simultánea como Front End, Servidor de Aplicaciones y Servidor de Bases de Datos. La propia arquitectura J2EE permitirá desplegar posteriormente el producto en un entorno distribuido real en caso de que fuera necesario.

Será suficiente en cuanto **hardware**, pues, un único ordenador para el desarrollo del proyecto que disponga de conexión a internet para las comunicaciones con el campus y el consultor .

En cuanto a requisitos de **software** utilizaremos los siguiente productos:

- Campus de la UOC para la comunicación con el aula y el consultor.
- Microsoft Word para la elaboración de la documentación textual del proyecto y la memoria.
- Microsoft Project para la planificación del proyecto.
- Microsoft PowerPoint para la elaboración de la presentación del proyecto.
- La plataforma J2EE para las librerías de desarrollo.
- JBoss como servidor de aplicaciones.
- MySQL como servidor bases de datos.
- IExplorer como Front-End de usuario.
- Eclipse como entorno de desarrollo.
- Plug-in de Eclipse UML Omondo para la elaboración de los diagramas UML.
- Internet para la busqueda de información y recursos.

1.4.4.- Descomposición del Proyecto en Tareas

Las tareas a realizar se enmarcan en tres grandes áreas:

- **Tareas del proceso de desarrollo de software:**
  - Planificación
    1. Revisión asignaturas EPI, TDP y GOI
    2. Definición tareas a realizar
    3. Elaboración Plan de Projecto
  - Especificación de requisitos
    - Documentación de Requisitos
      1. Descripción del sistema y contexto
      2. Identificación de actores y casos de uso
      3. Modelo de Dominio
      4. Documentación de los casos de uso
    - Análisis de Requisitos
      1. Identificación de las Clases de Entidades
      2. Diagrama Estático de las Clases Entidad
      3. Diagrama de Interacción de los casos de uso
      4. Diagrama Estático de Análisis
      5. Diagramas de estados y transiciones

6. Diagramas de Actividad

- Diseño
    1. Adaptación de la herencia al lenguaje
    2. Diseño de la persistencia
    3. Modelo Estático de Diseño
    4. Diseño de los subsistemas
    5. Diseño de la interficie de usuario
- Implementación
    1. Evaluación de la corrección de los modelos
    2. Creación de la base de datos y tablas.
    3. Implementación de las clases y pruebas unitarias
    4. Diagrama de componentes y de despliegue
- Pruebas de Integración
- Documentación
    1. Revisión de la documentación de análisis y diseño
    2. Manual del Usuario

- **Tareas de formación e instalación de la infraestructura J2EE.**
    - Estudio de Patrones de Diseño
    - Evaluación y pruebas de MySQL
    - Evaluación y Pruebas de Tomcat y Jboss
    - Evaluacion y Pruebas del frameworks J2EE Struts
    - Maqueta del framework Struts
    - Instalación de la instraestructura seleccionada

- **Tareas de documentación del proceso de desarrollo del proyecto.**
    - Realización de la Memoria
    - Presentación PowerPoint

1.4.5.- Schedule del Proyecto

Podemos ver la planificación en el tiempo de la anterior lista de tareas en el siguiente diagrama de Gantt, donde se han aplicado las precedencias oportunas:

Id	Nombre de tarea	Duración
1	**Planificación**	**6 días**
2	Revisión EPI-TDP-GOI	3 días
3	Definición tareas a realizar	1 día
4	Elaboración Plan de Projecto	2 días
5	***PAC1 - Entrega Plan de Trabajo***	1 día
6	**Especificación de requisitos**	**23 días**
7	**Documentación de los Requisitos**	**10 días**
8	Descripción del sistema y contexto	2 días
9	Identificación de actores y casos de uso	2 días
10	Modelo de Dominio	1 día
11	Documentación de los casos de uso	5 días
12	**Analisis de Requisitos**	**13 días**
13	Identificación de las Clases de Entidades	1 día
14	Diagrama Estático de las Clases Entidad	1 día
15	Diagrama de Interacción de los casos de uso	7 días
16	Diagrama Estático de Análisis	2 días
17	Diagramas de estados y transiciones	1 día
18	Diagramas de Actividad	1 día
19	**Diseño**	**20 días**
20	Adaptación de la herencia al lenguaje	2 días
21	Diseño de la persistencia	2 días
22	Modelo Estatico de Diseño	10 días
23	Diseño de los subsistemas	1 día
24	Diseño de la interficie de usuario	5 días
25	***PAC2 - Entrega del Analisis y Diseño***	1 día
26	**Implementación**	**26 días**
27	Evaluación de la corrección de los modelos	2 días
28	Creación de la Base de Datos y Tablas	1 día
29	Implementación de las clases y pruebas unitarias	21 días
30	Diagrama de componentes y de despliegue	2 días
31	**Pruebas de Integración**	5 días
32	***PAC3 - Entrega de la Implementación***	1 día
33	**Documentación del Proyecto**	**30 días**
34	Revisión de la documentación de análisis y diseño	2 días
35	Manual del Usuario	8 días
36	Memoria	12 días
37	Presentación PowerPoint	8 días
38	***Entrega Final del Proyecto***	1 día
39	**Formación en J2EE e Instalación Infraestructura**	**42 días**
40	Estudio de Patrones de Diseño	5 días
41	Evaluación y pruebas con MySQL o Postgres	5 días
42	Evaluación y Pruebas con Tomcat o Jboss	10 días
43	Evaluacion y Pruebas de frameworks J2EE (String, Strut	15 días
44	Maqueta del framework seleccionado	5 días
45	Instalación de la instraestructura seleccionada	2 días

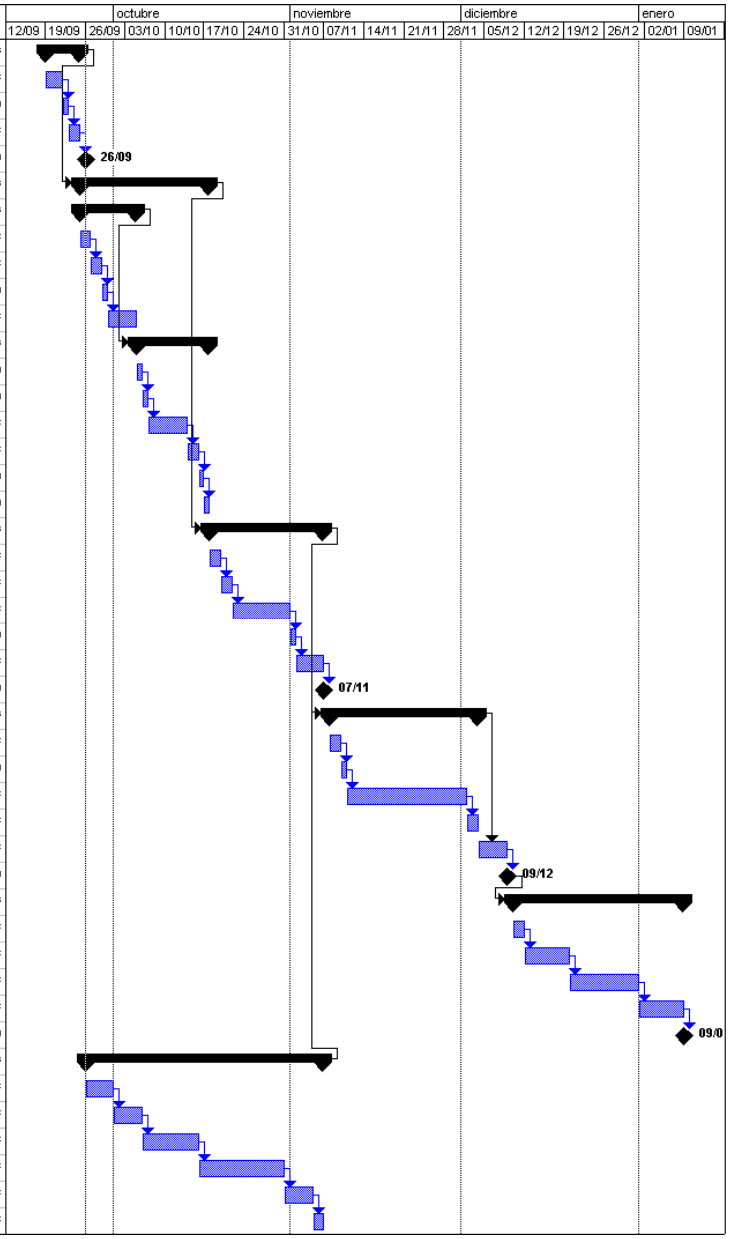


Fig. 1: Diagrama de Gantt

1.4.6.- Temporización

La temporización, las entregas parciales del proyecto y la entrega final vienen determinados por el Plan de la Asignatura TFC-J2EE. Está información ha sido utilizada para establecer los hitos en el schedule del punto anterior, y constan como restricciones temporales del proyecto.

Dichas fechas y entregas son las siguientes:

Fecha	Acontecimiento
26/09/2005	PAC1 – Plan de Trabajo
17/10/2005	Finalización del análisis
07/11/2005	PAC2 – Análisis y Diseño
09/12/2005	PAC3 – Implementación
09/01/2006	Memoria y Presentación

Los contenidos de las entregas parciales son los siguientes:

PAC1 - Plan de Trabajo: Debe contener básicamente los objetivos del proyecto así como la planificación temporal de las tareas a realizar para conseguirlos.

PAC2 – Especificación, Análisis y Diseño: Diagramas y Modelos técnicos que muestren como será la solución del problema planteado.

PAC3 - Implementación: El código desarrollado, los diagramas de componentes y de despliegue, así como los correspondientes paquetes de instalación.

1.5.- Productos obtenidos.

Los productos finales obtenidos son los siguientes:

Memoria: Síntesis del trabajo realizado en el TFC, mostrando claramente que se han conseguido los objetivos propuestos. Desde un punto de vista formal la memoria ha de contener aquella información relevante que permita comprender el problema planteado en el TFC, la metodología que se ha empleado para su resolución y muestre la solución del problema planteado.

Presentación : Síntesis clara y concisa del trabajo realizado a lo largo del semestre y de los resultados obtenidos, ofreciendo una perpectiva general del TFC.

Aplicación Gestión Incidencias, compuesto por:

- Proyecto Eclipse con el código desarrollado y la estrutura de desarrollo.
- GestionIncidencias.war: Componente a desplegar en el contenedor web (vista y controlador).

- GestionIncidencias.jar: Componente a desplegar en el contenedor de EJB (modelo y persistencia).
- El manual de instalación (anexo en la memoria).
- El manual del usuario (anexo en la memoria).

1.6.- Descripción del resto de capítulos de la memoria.

Una vez definido el **Plan de Proyecto** ya tenemos todas las tareas y pasos a seguir para cumplimentar todo el proceso de desarrollo.

El capítulo 2 contiene la **Documentación de Requisitos** y el capítulo 3 contiene el **Modelo de Análisis** generado a partir de esa documentación.

El siguiente paso en el proceso de desarrollo de software es el **Diseño** de aplicación que queda documentado en el capítulo 4. Para poder realizar un diseño directamente implementable es necesario tener muy clara la **arquitectura del sistema** sobre la que se va que se va a desarrollar. Es por ello que iniciamos este capítulo describiendo esta arquitectura y que nos ayudará a tomar las decisiones de diseño que vamos a aplicar
.

En el capítulo 5 mostramos el **Modelo de Implementación** y se explican algunos puntos interesantes sobre el producto desarrollado. En este punto se finaliza el proceso de desarrollo. Es momento, pues, de sacar **conclusiones** y es en el capítulo 6 donde expongo sus resultados.

Para finalizar esta memoria incluimos el **glosario** en el capítulo 7, la **bibliografía** en el capítulo 8 y un apartado de anexos en el capítulo 9. Hay que decir que estos anexos son material que forma parte del producto software obtenido ya que son el **Manual de Instalación** y el **Manual del Usuario**.

2.- Documentación de Requisitos.

2.1.- Descripción del sistema

La aplicación Gestión de Incidencias nace con la idea de ayudar a un departamento de soporte interno en la gestión de peticiones de soporte que sus clientes (los usuarios del sistema) le remiten ante los problemas informáticos con los que se encuentran en su sistema.

El sistema debe facilitar el registro de dichas incidencias, la asignación de los técnicos que deben resolverlas, registrar las acciones realizadas y proveer al cliente de un sistema de información que le muestre el estado de sus peticiones.

2.2.- Descripción de Requisitos

Requisitos funcionales

Los clientes, tras su identificación en el sistema, deben poder registrar las incidencias informáticas que se les produzcan, consultar de las tienen en curso y poder acceder a su histórico de incidencias.

La información que deberá aportar el cliente es la siguiente: Código de máquina donde se requiere la intervención, teléfono de contacto y descripción de la incidencia.
Las incidencias de nueva creación quedará en estado “pendiente” hasta el momento en que el personal del CAU, la catalogue (asignación de subsistema y tipo de incidencia) y le asignen un técnico de soporte para su resolución; en ese momento pasarán a estado de “abierta”. La catalogación de la incidencia pretende agilizar posteriores búsquedas que permitan al Cau y a los técnicos utilizar la base de datos para ver que acciones se realizaron sobre problemas parecidos.

Los técnicos de Soporte deberán poder registrar las intervenciones que vayan realizando en la resolución de cualquier incidencia abierta y no solo las en las que tienen asignadas. Para cada intervención quedará registrada la fecha y la hora y el tiempo invertido. Cuando la incidencia quede solucionada el técnico de soporte deberá registrarlo, quedando la incidencia como “cerrada”. La reapertura de una incidencia solo podrá ser realizada por personal del CAU.

Para los casos en que el cliente no pueda utilizar este servicio para registrar sus incidencias, el propio personal del CAU registrará la incidencia en su nombre. También podrá registrar intervenciones y cerrar incidencias, ya que puede haber casos en que la resolución sea realizada por ellos mismos por vía telefónica o deban cerrar incidencias que se abrieron por error o porque estaban provocadas por efectos colaterales de otros sistemas, que al ser solucionados ya no requieren de la intervención de ningún técnico.

Requisitos no funcionales

La arquitectura del sistema se requiere que sea bajo el paradigma de J2EE con utilización de Entreprise Java Beans. Esto implica que la lógica del proceso deberá ejecutarse en un servidor de aplicaciones que proporcione un contenedor web y un contenedor de EJB's.

La interficie con el usuario será mediante paginas web generadas en el servidor y por tanto solo se requerirá un explorador web para ejecutar la capa de presentación de la aplicación.

La validación de usuarios en producción se realizará contra un servidor LDAP, si bien en la etapa de desarrollo se realizará contra una tabla de la base de datos. El mantenimiento de tablas auxiliares que aparezcan (técnicos, personal CAU, subsistemas, tipos de incidencia...) se realizará directamente desde el gestor de base de datos. Si el mantenimiento fuera importante debería desarrollarse un módulo de aplicación para mantener dichas tablas.

2.3.- Identificación de los actores y casos de uso

De los requisitos funcionales expuestos anteriormente vemos que tenemos tres tipos de actor claramente diferenciados: Cliente, personal del CAU (Cau) y técnico de soporte (técnico).

Identificamos los siguiente casos de uso por actor:

- Cliente: Identificación en el sistema, Registro de incidencias, Consulta de incidencias.
- Cau: Identificación en el sistema, Registro de Incidencias, Consulta de Incidencias, Asignación de Incidencias, Registro de Intervenciones.
- Técnico: Identificación en el sistema, Consulta de Incidencias, Registro de intervenciones.

Vemos que necesitaremos además un caso de uso Relación de Incidencias para ayudar al usuario a buscar cualquier incidencia antes de actuar sobre ella.

Como podemos observar el actor Cau ejecuta tanto los casos de uso de Cliente como los de Técnico por lo que vemos que tiene relación de herencia con estos.

2.4.- Modelo de Dominio

El modelo de dominio recoge los tipos de objeto más importantes. De los requisitos funcionales detectamos los siguientes: Cliente, Cau, Técnico, Incidencia e Intervención. Estos tipos de objetos se relacionan de la siguiente manera:

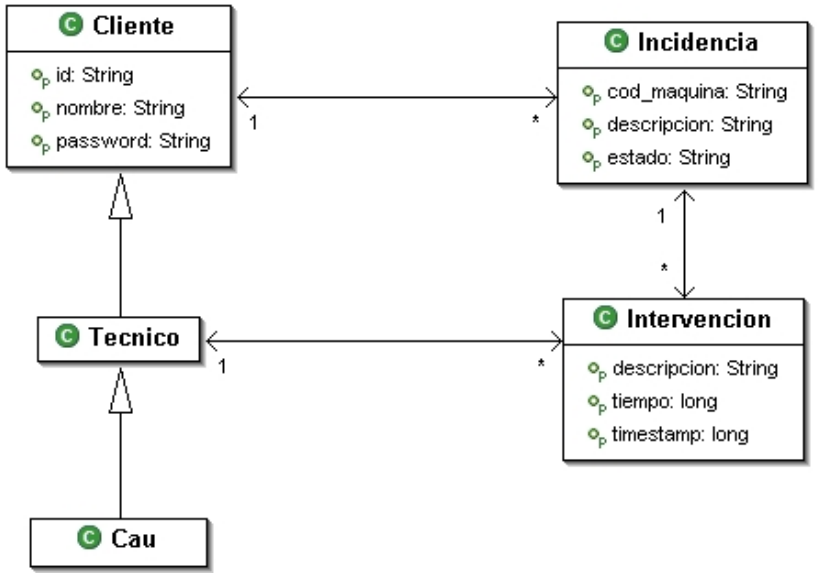


Fig. 2: Modelo de Dominio

2.5.- Diagrama de Casos de Uso

Veamos ahora como se relacionan los actores y los casos de uso que hemos detectado:

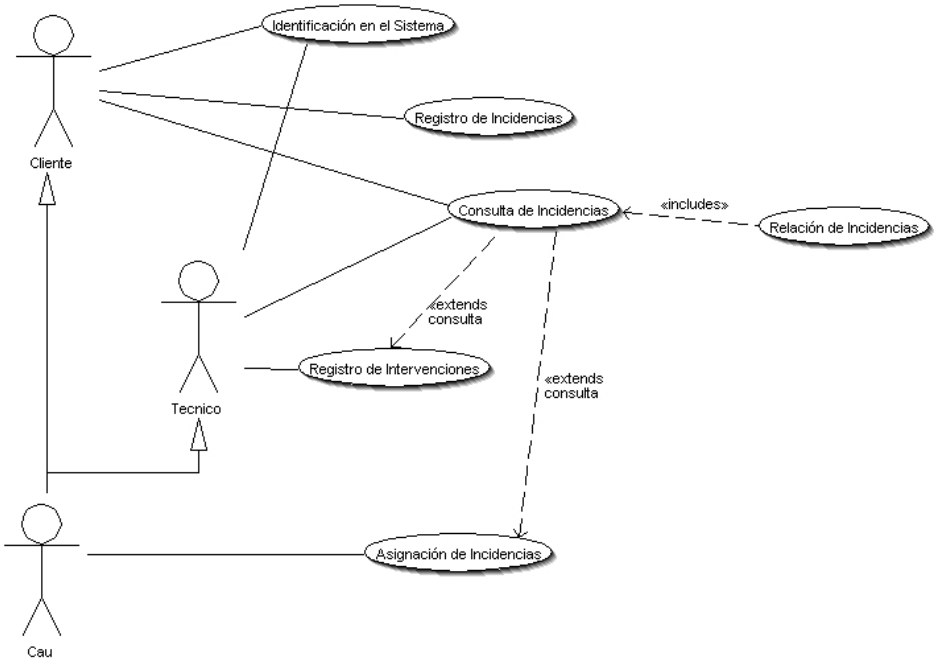


Fig. 3: Diagrama de Casos de Uso

2.6.- Descripción de los casos de uso

1.- Identificación en el Sistema

Resumen de la funcionalidad	Permite al usuario identificarse en el sistema
Papel dentro del trabajo del actor	Función principal
Actores	Cliente, Cau, Tecnico
Casos de uso relacionados	
Precondición	El usuario no está identificado en el sistema.
Poscondición	El usuario ha sido identificado en el sistema y se han cargado sus privilegios.
Proceso normal principal	1.- El usuario introduce su identificador de usuario y su password. 2.- El sistema valida sus datos y permite la entrada en el sistema
Alternativas de proceso y excepciones	2.- El usuario no existe o el password es incorrecto: el sistema le notifica el error y lo devuelve a pantalla de identificación.

2.- Registro de Incidencias

Resumen de la funcionalidad	Permite al usuario registrar una incidencia en el sistema
Papel dentro del trabajo del actor	Función principal
Actores	Cliente, Cau
Casos de uso relacionados	Consulta de incidencias
Precondición	El usuario está identificado en la aplicación.
Poscondición	La incidencia queda registrada en estado de “pendiente”
Proceso normal principal	1.- El usuario introduce su teléfono de contacto, código de máquina y la descripción del problema. 2.- El sistema la almacena en la base de datos, con estado “pendiente”, además guarda la fecha y hora en que se ha registrado la incidencia. 3.- Se le notifica al usuario que la incidencia ha sido registrada.
Alternativas de proceso y excepciones	1.- Si el usuario es Cau debe introducir además el identificador del Cliente al que se refiere la incidencia.

3.- Consulta de Incidencias

Resumen de la funcionalidad	Permite consultar una incidencia mostrando al usuario los datos relacionados con ella.
Papel dentro del trabajo del actor	Función principal
Actores	Cliente, Cau, Técnico
Casos de uso relacionados	Relación de incidencias
Precondición	El usuario está identificado en la aplicación.
Poscondición	
Proceso normal Principal	1.- El usuario se le muestra una Relación de Incidencias y el usuario selecciona la que desea visualizar. Si es tipo Cliente solo puede ver sus incidencias. 2.- El sistema comprueba la existencia de la incidencia y si el usuario tiene permiso para consultarla. 3.- El sistema muestra la información de la incidencia.
Alternativas de proceso y excepciones	3.- La incidencia no existe o el usuario no tiene permiso para consultarla: el sistema le notifica el error.

4.- Asignación de Incidencias

Resumen de la funcionalidad	Permite asignar al técnico encargado de resolver una incidencia.
Papel dentro del trabajo del actor	Función principal
Actores	Cau
Casos de uso relacionados	Consulta de Incidencias
Precondición	El usuario se ha identificado en la aplicación y es Cau.
Poscondición	El técnico, sistema y tipo de incidencia queda registrado en la base de datos y la incidencia queda en estado de “abierta”.
Proceso normal principal	1.- El usuario selecciona la incidencia sobre la que va a asignar un técnico. 2.- El sistema comprueba la existencia de la incidencia y muestra información que la identifica. 3.- El usuario asigna a uno de los Técnicos registrados en la aplicación. Además debe clasificar la incidencia asignándole sistema y tipo. 4.- El sistema actualiza la información.
Alternativas de proceso y excepciones	2a.- La incidencia no existe: el sistema le notifica el error. 2b. La incidencia está cerrada: El sistema permite reabrirla y continua con el punto 3, pudiendo modificar la información previamente introducida.

5.- Registro de Intervenciones

Resumen de la funcionalidad	Permite registrar las intervenciones realizadas para conseguir solucionar una incidencia.
Papel dentro del trabajo del actor	Función principal
Actores	Cau, Técnico
Casos de uso relacionados	Consulta de Incidencias
Precondición	El usuario está identificado en la aplicación y es Técnico o Cau.
Poscondición	La Intervención queda registrada en la base de datos.
Proceso normal principal	1.- El usuario selecciona la incidencia sobre la que va a registrar la Intervención. 2.- El sistema comprueba la existencia de la incidencia y muestra su información. 3.- El usuario introduce la descripción de la intervención y el tiempo dedicado. Además puede cambiar el estado de la Incidencia a “cerrado”. 4.- El sistema registra la Intervención, añadiendo el técnico que la ha registrado, la fecha y la hora. Actualizará el estado de la incidencia en caso necesario.
Alternativas de proceso y excepciones	2.- La incidencia no existe o ya está cerrada: el sistema le notifica el error y lo devuelve a pantalla de petición de código.

6.- Relación de Incidencias

Resumen de la funcionalidad	Permite mostrar una lista de incidencias para su posterior selección.
Papel dentro del trabajo del actor	Función secundaria
Actores	Usuario, Cau, Técnico
Casos de uso relacionados	Consulta de Incidencias
Precondición	El usuario está identificado en la aplicación.
Poscondición	
Proceso normal principal	1.- El sistema recibe como parámetro el filtro que debe aplicar 2.- El sistema muestra la relación de incidencias que cumplan con dicho filtro.
Alternativas de proceso y excepciones	

3.- Análisis de Requisitos.

3.1.- Revisión de los Casos de Uso

Antes de iniciar el análisis hay que detenerse a revisar los casos de uso de los requisitos presentados al cliente, para ver si encontramos algún fallo, indeterminación o contradicción que deberíamos solucionar antes de proseguir.

En nuestro caso, los casos de uso ya fueron revisados antes de cerrar la Documentación de Requisitos , siendo está la base para nuestro análisis.

3.2.- Identificación de las Clases de Entidad

A partir de los Casos de Uso y del Modelo de Dominio vemos que clases entidad candidatas tenemos así como sus atributos:

- Modelo de Dominio: Cliente, Cau, Técnico, Incidencia e Intervención.
- Caso de Uso Identificación en el sistema: Usuario (idenficador, password)
- Caso de Uso Registro de Incidencias: Incidencia (telefono del usuario, código de maquina, descripción, estado, fecha y hora), Cau, Cliente(identificador).
- Caso de Uso Consulta de Incidencias: Incidencia (código).
- Caso de Uso Registro de Intervenciones: Intervención (descripción, tiempo dedicado, técnico, fecha y hora), Incidencia(estado).
- Caso de Uso Asignación Incidencia: Incidencia(codigo, técnico, sistema, tipo)
- Caso de Uso Relación de Incidencias: Incidencia.

Podemos ver que los tres tipos de actores (Cliente, Cau, Técnico) son los diferentes usuarios del sistema que requerirán un identificador y un password para acceder al sistema. Será conveniente, por tanto, definir una superclase Usuario.

El Cau, en ocasiones actúa como técnico y por tanto podemos asignarle una relación de herencia.

El teléfono del usuario se pide para cada incidencia; si bien podríamos pensar que podría ser conveniente que fuera un atributo del usuario, hay que decir que un usuario puede tener incidencias con diferentes máquinas y por tanto en diversas localizaciones. El teléfono de la incidencia es un teléfono de contacto para que los técnicos puedan hablar con el usuario en el lugar donde tiene el problema y poder recabar mayor información.

El **Diagrama Estático de Entidades** será el siguiente:

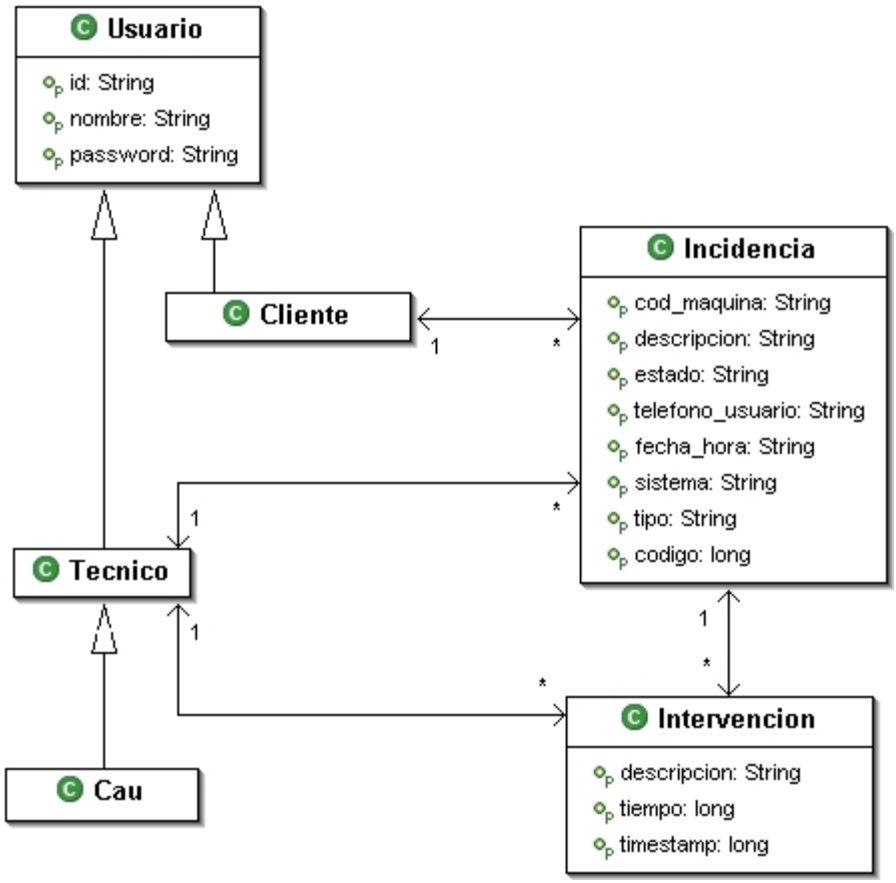


Fig. 4: Diagrama Estático de Entidades

3.3.- Diagramas de Interacción

Para ver la dinámica de la aplicación vamos a utilizar diagramas de colaboración simplificados para cada Caso de Uso.

Caso de Uso 1: Identificación en el Sistema

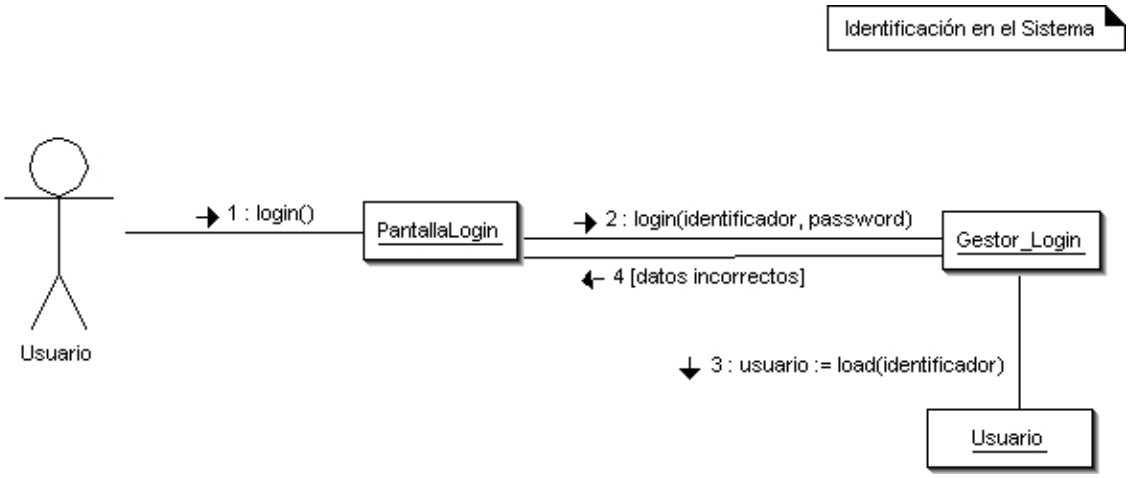


Fig. 5: Diagrama de colaboración Identificación en el Sistema

Caso de Uso 2: Registro de Incidencias

Fig. 6: Diagrama de colaboración Registro de Incidencias

Caso de Uso 3: Consulta de Incidencias

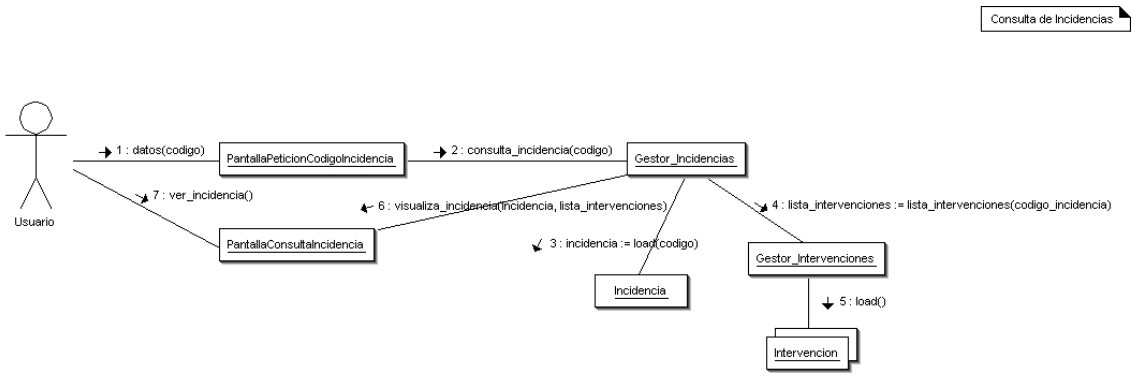


Fig. 7: Diagrama de colaboración Consulta de Incidencias

Caso de Uso 4: Asignación de Incidencias

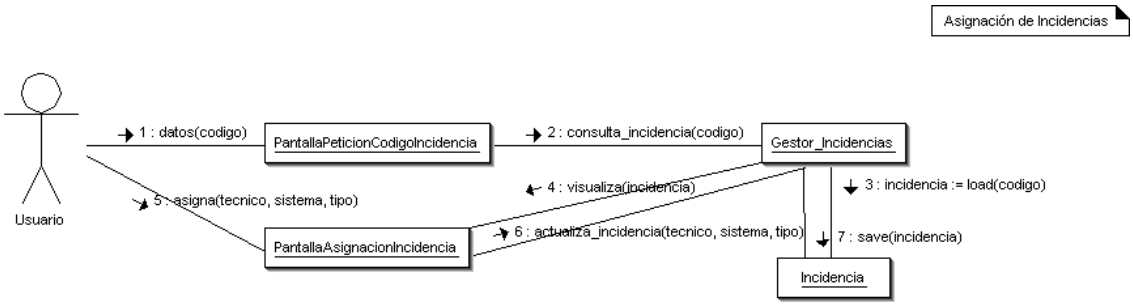


Fig. 8: Diagrama de colaboración Asignación de Incidencias

Caso de Uso 5: Registro de Intervenciones

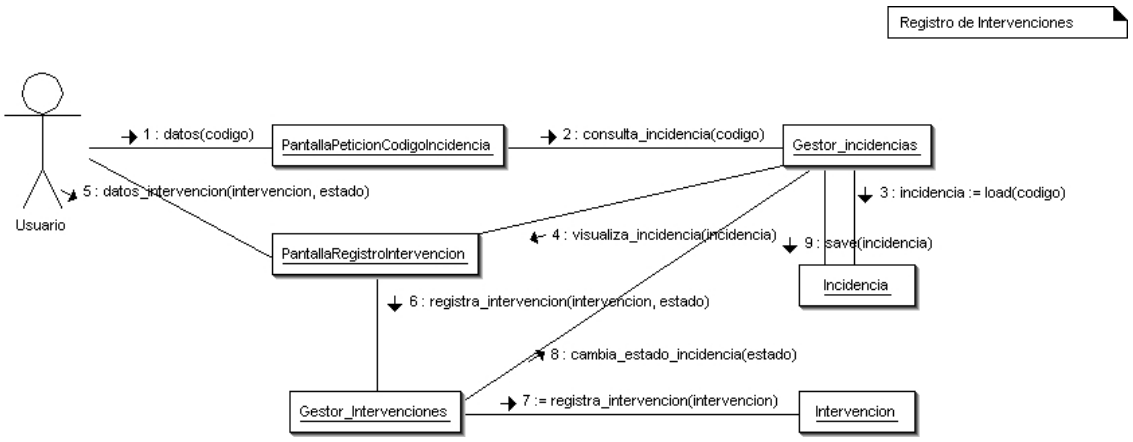


Fig. 9: Diagrama de colaboración Registro de Intervenciones

Caso de Uso 6: Relación de Incidencias

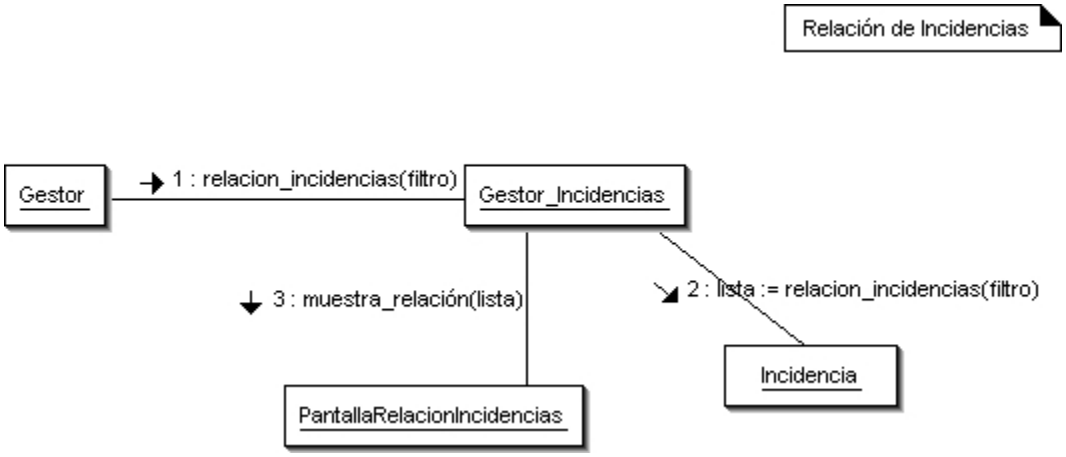


Fig. 10: Diagrama de colaboración Relación de Incidencias

3.4 .-Identificación de las Clases de Análisis

A partir de los diagramas de colaboración descritos en el apartado anterior, aparecen las clases de análisis detectadas hasta el momento. La clases entidad ya las habíamos detectado anteriormente, pero en este momento nos aparecen las clases Control y Frontera. Además aparecen los métodos que se precisan para la interacción entre las clases. Organizando esta información en un diagrama de clases obtenemos el **Diagrama Estático de Análisis**

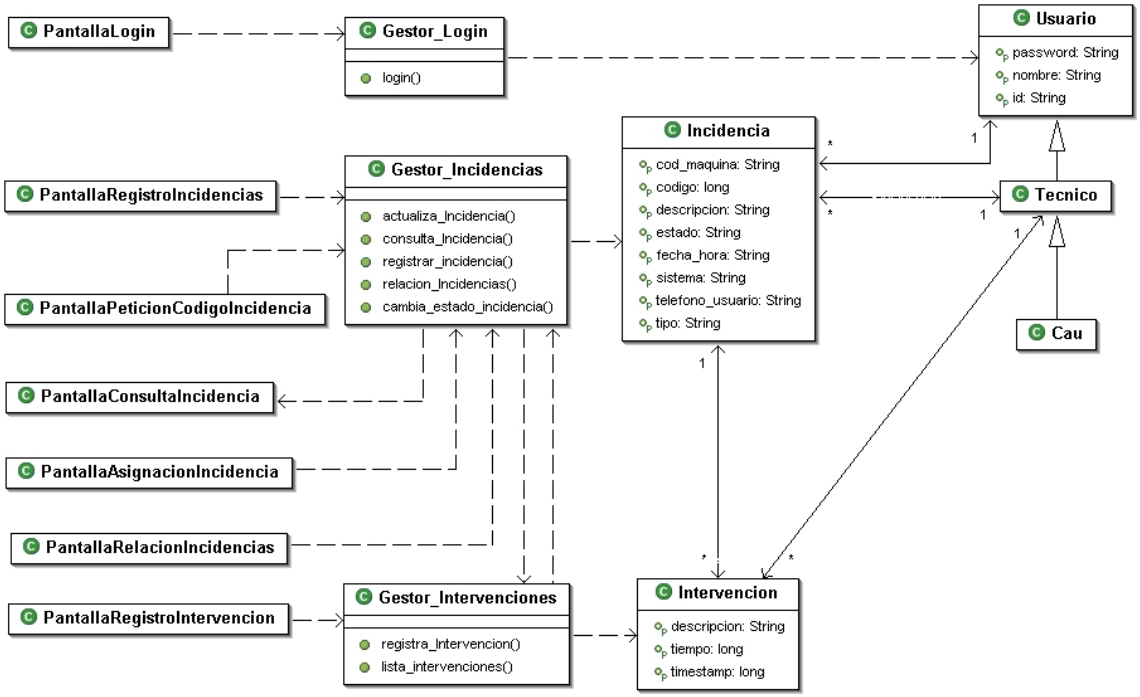


Fig. 11: Diagrama Estático de Análisis

3.5.- Diagramas de Estado

Como podemos comprobar solo hemos hablado de estados al referirnos a las incidencias. Parece ser que es la única clase que tiene estado. Vemos cual es su diagrama:

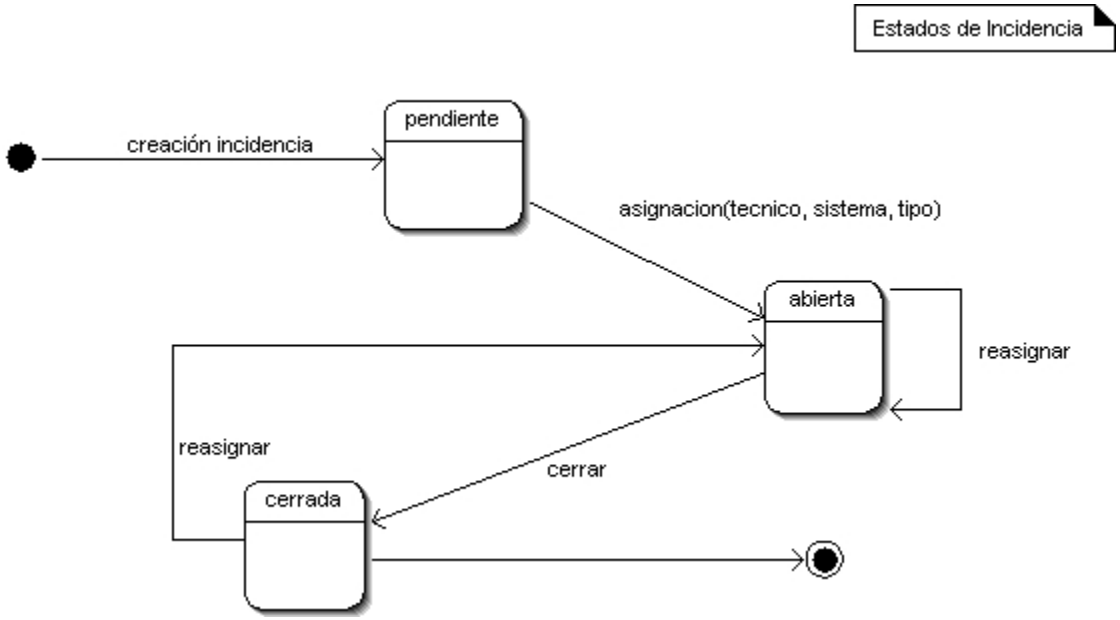


Fig. 12: Diagrama de Estado de Incidencia

4.- Diseño.

4.1.- Diseño Arquitectónico

La especificación de requisitos, en el apartado de Requisitos no funcionales, ya nos dice que se precisa que la arquitectura sea bajo el paradigma de J2EE, con la implementación de Entreprise Java Beans para la lógica de negocio, y cliente web para la interficie de usuario. Veamos de que estamos hablando y que implicaciones tiene en nuestro proyecto.

Arquitectura J2EE

J2EE es un grupo de especificaciones diseñadas por Sun que permiten la creación de aplicaciones empresariales. Esto es: acceso a base de datos (JDBC), utilización de directorios distribuidos (JNDI), acceso a métodos remotos (RMI/CORBA), funciones de correo electrónico (JavaMail), aplicaciones Web (JSP y Servlets), etc. Aquí es importante notar que J2EE es solo una especificación, esto permite que diversos productos sean diseñados alrededor de estas especificaciones; la especificación más reciente de Sun es J2EE 1.4 , la cual esta conformada por: JSP 2.0 ,Servlet 2.4, EJB 2.1 y Connector 1.5 entre otros API's, los detalles se encuentran en http://java.sun.com/j2ee

Las aplicaciones J2EE están formadas por un conjunto de componentes que se ensamblan y se despliegan en contenedores J2EE. Un componente J2EE es una unidad autocontenida que necesita del contenedor para funcionar. Tenemos dos tipos de contenedores:

- Contenedores Web: que dan un entorno de ejecución a los Servlet y JSP's
- Contenedores EJB: dan un entorno de ejecución a los Enterprise Java Beans

El esquema de la arquitectura J2EE es el siguiente:

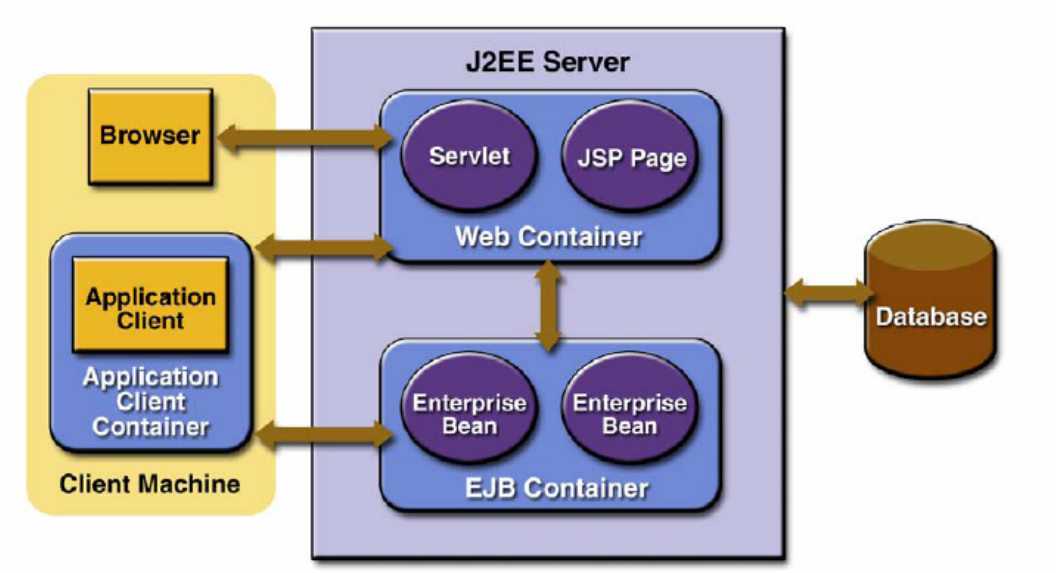


Fig. 13: Esquema arquitectura J2EE

Los Servlets son clases escritas en Java que especializadas en procesar comandos dinámicamente y construir respuestas.

Los JSP son documentos de texto que se ejecutan como servlets. Nos permiten escribir codigo servlet directamente en un documento de hipertexto.

Un EJB es una clase escrita en Java, con sus atributos y métodos, que se utiliza para implementar las reglas de negocio. El contenedor hará que sea accesible de forma remota. Existen tres tipos de EJB: de sesión, de entidad y de mensaje.

Diseño con patrones

El objetivo es disponer de una colección de soluciones de diseño que son válidas en distintos contextos y que han sido aplicadas con éxito en otras ocasiones.

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Los patrones son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además facilitan la comunicación entre diseñadores, pues establecen un marco de referencia (terminología, justificación).

Uno de los patrones de diseño fundamental en la arquetictura de aplicaciones distribuidas es el Modelo Vista Controlador.

Modelo Vista Controlador (MVC) es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos.

En términos generales, construir una aplicación usando una arquitectura MVC implica definir tres clases de módulos.

- **Modelo**: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. Contiene el núcleo de la funcionalidad de la aplicación. Encapsula el estado de la aplicación y no sabe nada del Controlador y la Vista.

- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario. Es la presentación del Modelo.

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

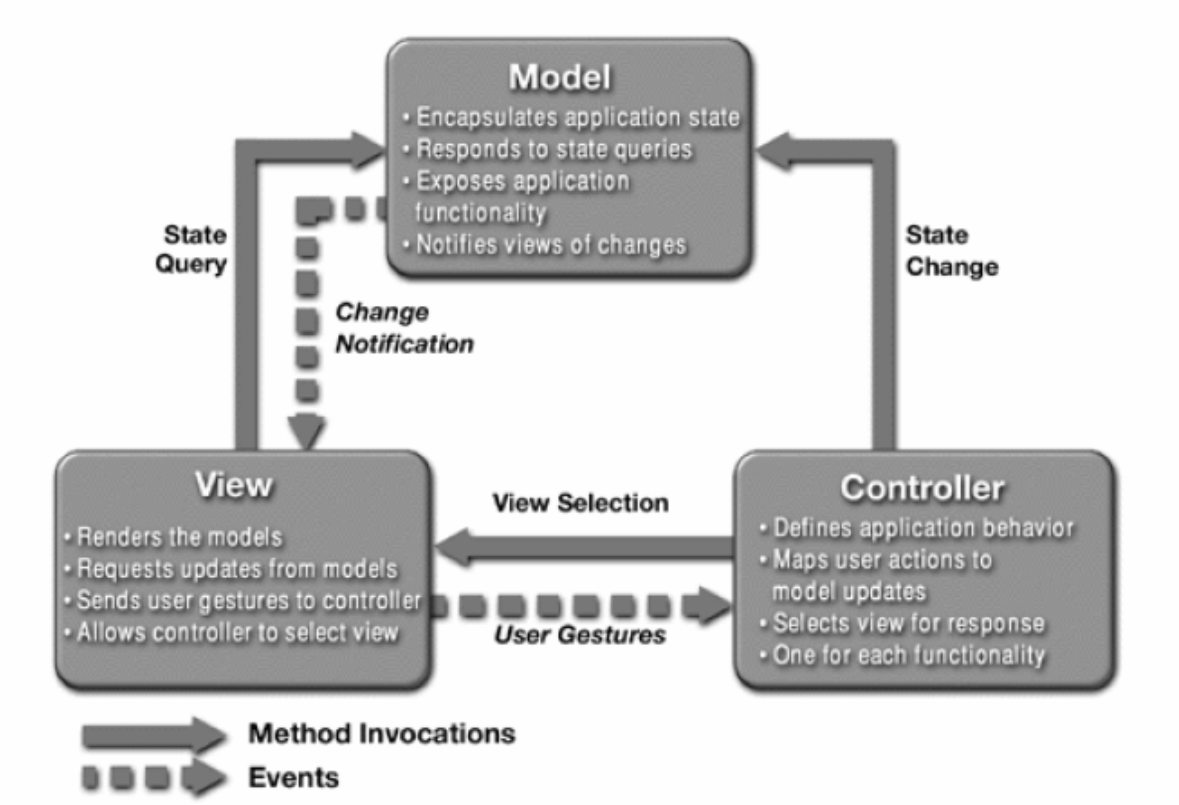


Fig. 14: Esquema MVC

Por otro lado Sun, en sus BluePrints, ha catalogado una serie de patrones muy útiles en la implementación de aplicaciones J2EE que se relacionan de la siguiente forma:

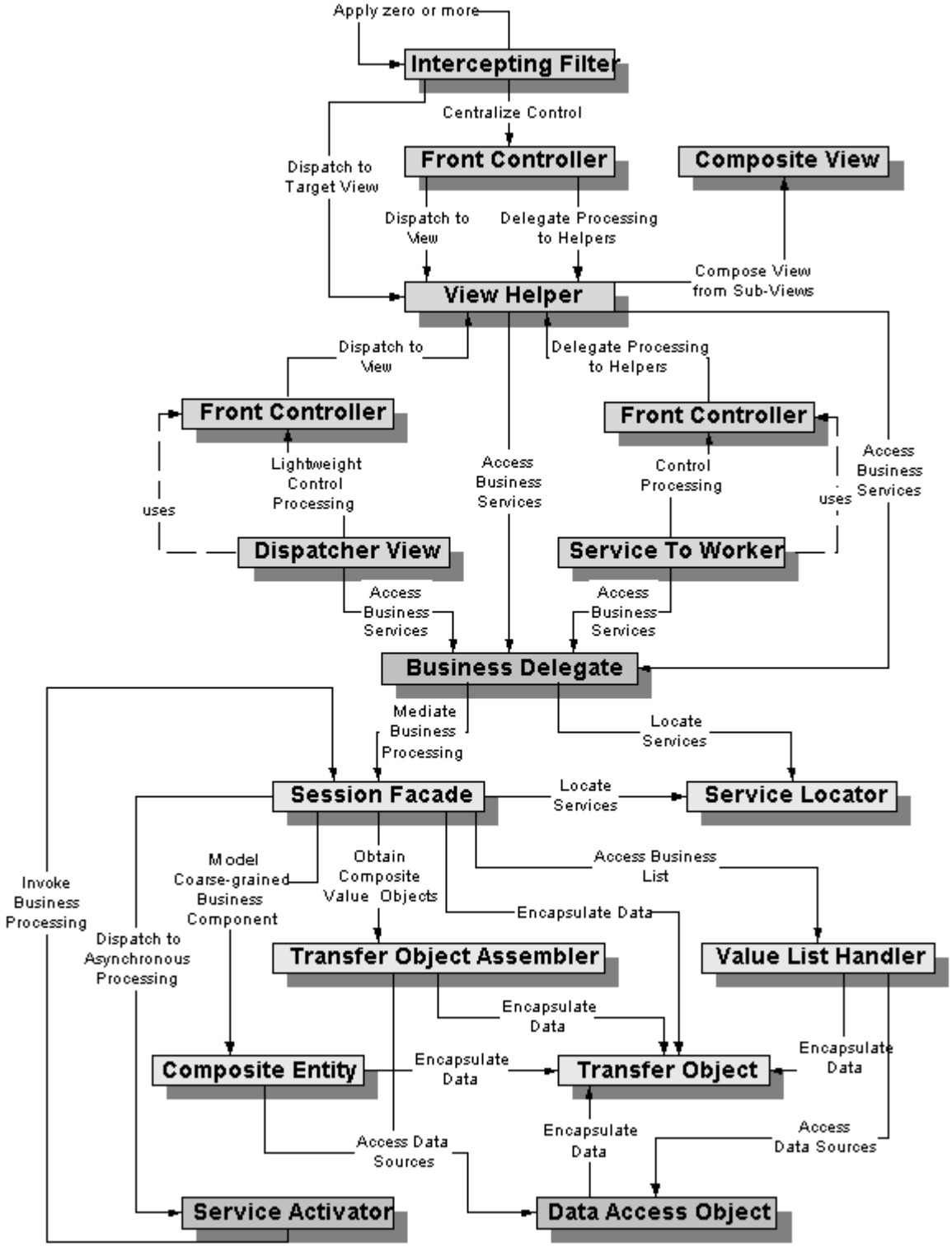


Fig. 15: Patrones para J2EE

Struts

Struts es un framework para aplicaciones web java que implementa el patrón MVC. Realmente lo que provee es un conjunto de clases y TAG-LIBS que conforman el Controlador y facilitan la construcción de Vistas. Naturalmente, el Modelo o lógica de negocio es la parte que nos corresponde desarrollar completamente.

Dentro de la arquietectura J2EE Struts se situa dentro del Contenedor Web:

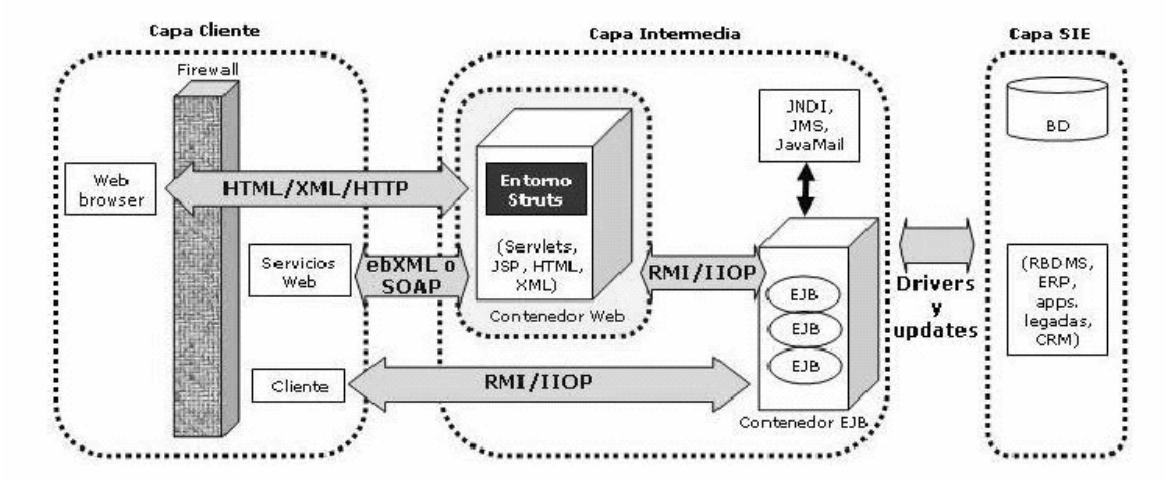


Fig. 16: Struts en J2EE

Struts implementa el patrón MVC siguiendo este esquema:

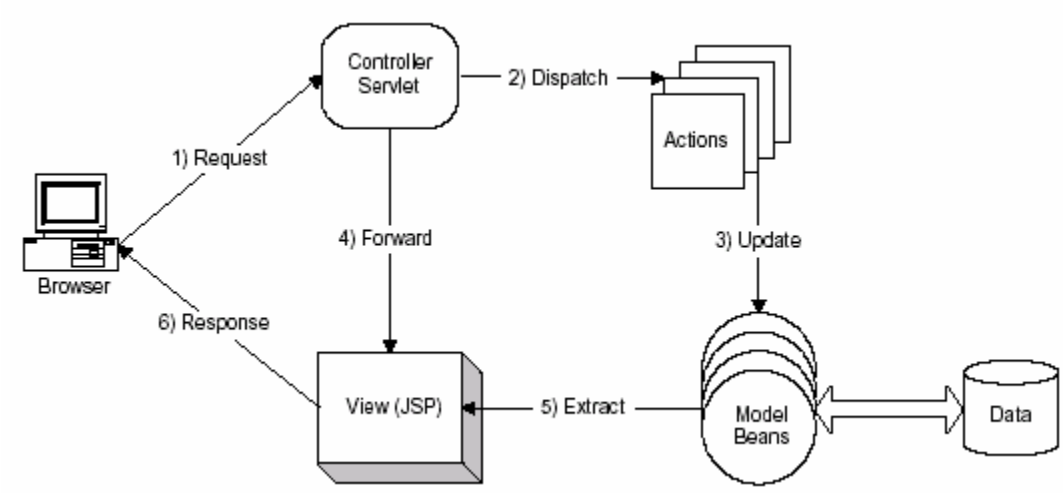


Fig. 17: Esquema MVC Struts

Para ello Struts define completamente la estructura de clases y artefactos a utilizar para las capas de Vista y Controlador. Este es el diagrama de clases:

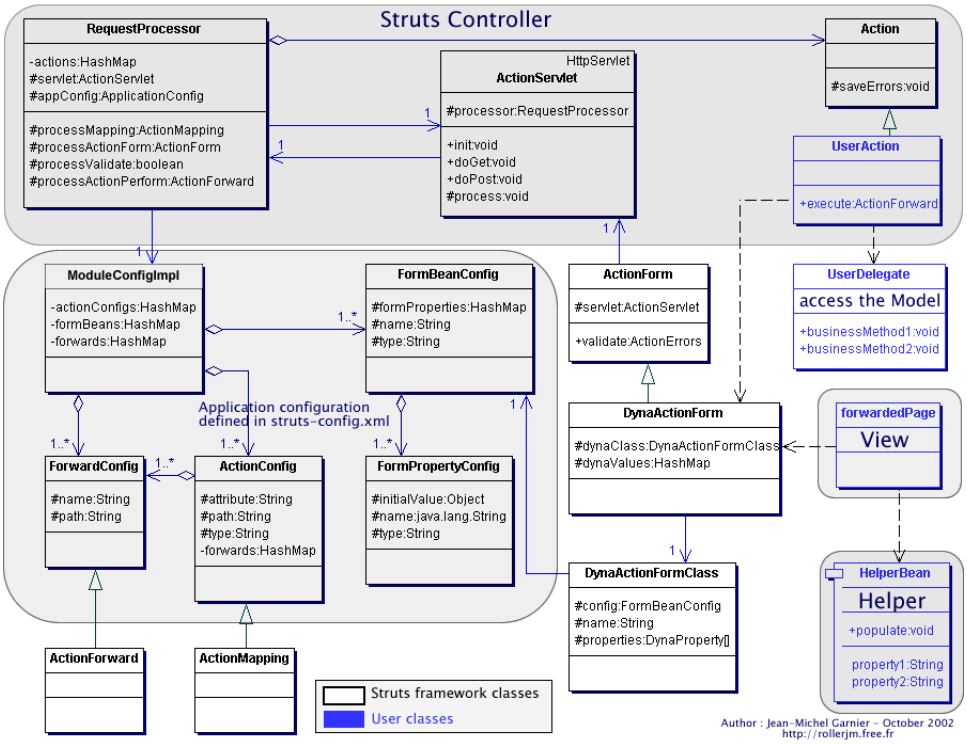


Fig. 18: Diagrama de Clases Struts

Como vemos Struts nos proporciona la parte genérica del controlador y las herramientas necesarias para montar el flujo deseado en la aplicación. El diagrama genérico de secuencia correspondiente es el siguiente:

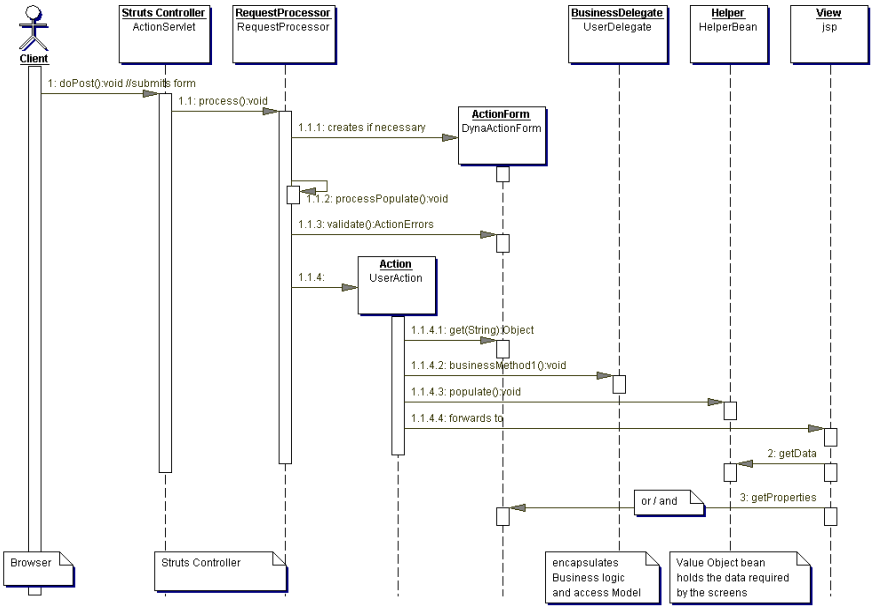


Fig. 19: Diagrama de Secuencia Struts

Struts funciona de la siguiente manera:

1. Un usuario hace una petición.
2. Cada petición está asociada a una acción que ejecutará un regla de negocio en particular.
3. La acción suele necesitar un conjunto de datos que se encapsulan en un objeto llamado Form, que puede ser una clase física o un elemento general (dynaform) cuyo comportamiento se define en el fichero de configuración.
4. Los datos se pueden validar y si falla la validación se puede volver a mostrar el formulario de entrada.
5. Una vez ejecutada la acción, puede el resultado sea favorable o desfavorable por lo que se delega sobre distintos elementos de presentación en cada caso.
6. Todo mensaje mostrado puede sacarse a un fichero de recursos para favorecer su mantenimiento y la internacionalización de la aplicación.
7. Cada elemento se construye por separado y se relacionan en el fichero struts-config.xml

Como era de esperar, Strust implementa varios de los patrones recomendados por Sun para el desarrollo J2EE. Podemos destacar:

Pattern(s)	Struts component(s)
Service to Worker	ActionServlet, Action
Command [Go4], Command and Controller, Front Controller, Singleton, Service Locator	ActionServlet, Action
Dispatcher, Navigator	ActionMapping, ActionServlet, Action, ActionForward
View Helper, Session Facade, Singleton	Action
Transfer Objects (fka Value Objects), Value Object Assembler	ActionForm, ActionErrors, ActionMessages
View Helper	ActionForm, ContextHelper, tag extensions
Composite View, Value Object Assembler	Template taglib, Tiles taglib

Hibernate

Trabajar con software orientado a objetos y bases de datos relacionales puede hacernos invertir mucho tiempo en los entornos actuales. Hibernate es una herramienta que realiza el *mapping* entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. El término utilizado es **ORM** (object/relational mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

Hibernate no solo realiza esta transformación sino que nos proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reducen el tiempo de desarrollo.

Hibernate funciona asociando a cada tabla de la base de datos un Plain Old Java Object (POJO, a veces llamado Plain Ordinary Java Object). Un POJO es similar a un Java Bean, con propiedades accesibles mediante métodos setter y getter. Para poder asociar el POJO a su tabla correspondiente en la base de datos, Hibernate usa los ficheros hbm.xml.

Los parámetros de conexión con la Base de Datos se configuran en el fichero hibernate.cfg.xml. Esto tiene la ventaja de hacer totalmente transparente la base de datos a la que se accede, pudiendo cambiar de base de datos sin necesidad de cambiar una línea de código de nuestra aplicación, simplemente cambiando los ficheros de configuración de Hibernate.

La arquitectura que proporciona hibernate es la siguiente:

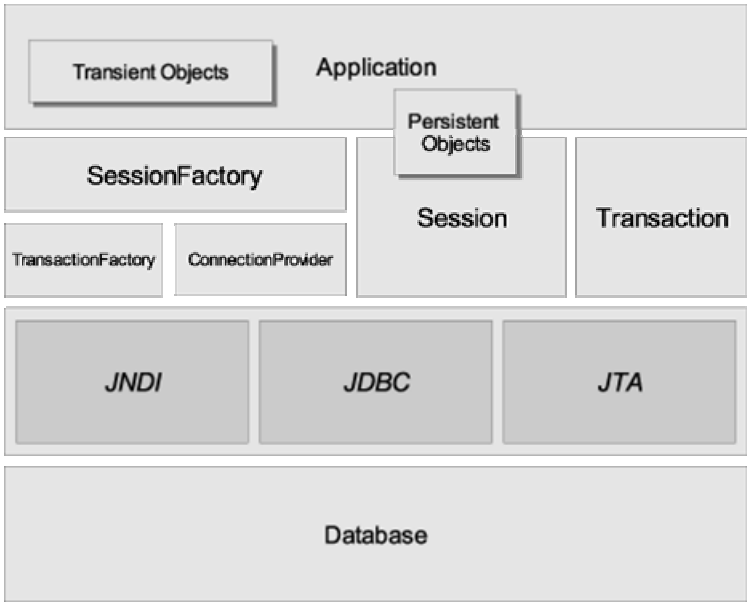


Fig. 20: Arquitectura de Hibernate

Decisiones de diseño

Visto lo anterior, ya podemos definir las líneas generales de diseño que vamos a seguir:

1. Los casos de uso serán desglosados en acciones que se implementarán mediante la clase **Action** correspondiente. Se creará una superclase BaseAction que implemente servicios comunes y centralice la gestión de excepciones. Cada acción puede tener asociado un **ActionForm** (físico o dinámico) para el traspaso de datos entre la Vista (JSP) y el Controlador (Action).

2. La Vista se implementará mediante **JSP** con las librerías de tags standard, las que proporciona struts y tiles.

3. El Modelo se implementará en un **EJB** de Sesión que actuará de **Facade.** Éste accederá a los datos mediante **DAO**'s que ocultarán la implementación del acceso a la base de datos.

4. El Controlador accederá al Modelo mediante **Bussines Delegate** para ocultar todo el tratamiento de llamadas remotas al EJB de sesión, desacoplándolo del controlador. Se utilizarán **Value Objects** (también llamados DTO**)** para la transferencia de información, y se implementará un **ServiceLocator** para centralizar la obtención de referencias de cualquier interficie Home y hacer de caché de las referencias obtenidas (implementará el patrón de diseño **Singleton**).

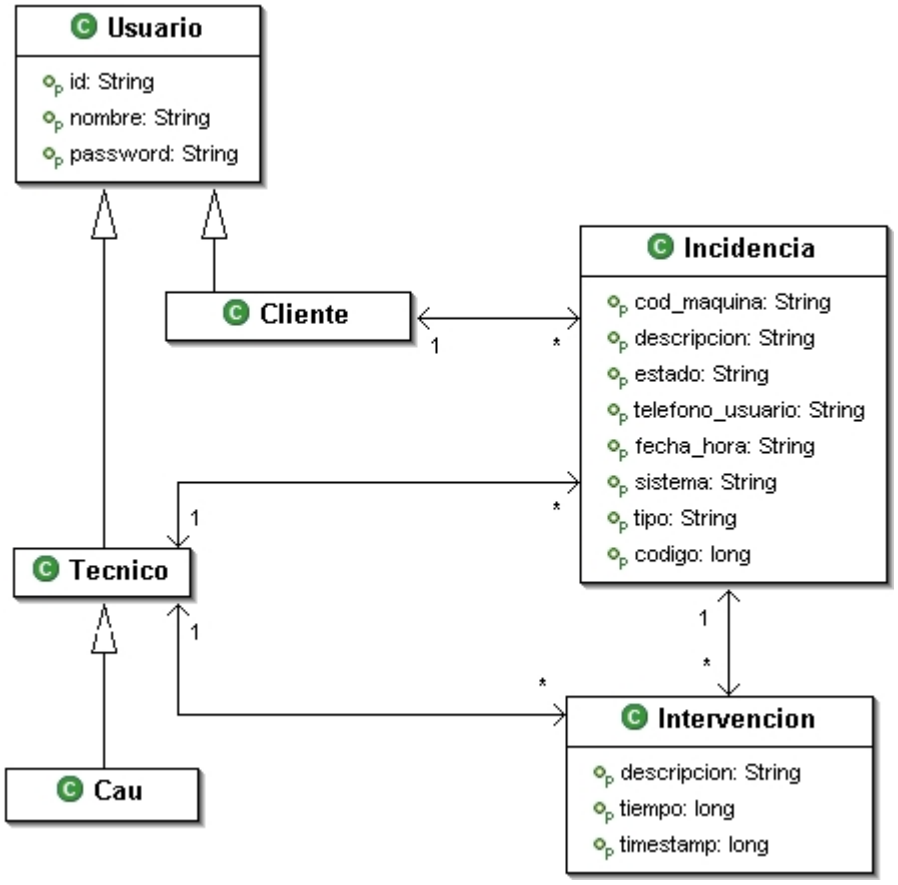
5. Los DAO's implementarán la persistencia en la base de datos (MySQL) mediante **Hibernate**. No se utilizarán Entity Beans para la implementación de la persistencia.

Con todo esto, tendremos la lógica de la Vista y el Controlador en el Contenedor Web y la lógica de Negocio en el Contenedor de EJB. La lógica de negocio podría ser accedida directamente por otros tipos de cliente, si fuera necesario.

Por otro lado, si en un futuro se quisiera traspasar la lógica de negocio al Contenedor Web (y eliminar de esta manera el Contenedor EJB), solo habría que traspasar la lógica del EJB de Facade al Bussines Delegate, reutilizando todo el acceso a los DAO's e Hibernate.

4.2.- Diseño de la Persistencia

Partimos del Modelo Estático de Entidades:



Supresión de la Herencia

Si revisamos el Modelo Estático de Entidades vemos que tenemos relación de herencia entre Usuario y Cliente; entre Usuario y Técnico; y entre Técnico y Cau.

Para suprimir la herencia tenemos tres métodos:

1. Definir una clase para subclase que contendrá tanto los atributos propios como los heredados.
2. Crear una clase para cada subclase que tenga los atributos propios más los que identifican la superclase.
3. Crear una única clase para toda la jerarquía con todos los atributos existentes más otro que indicara el tipo de clase.

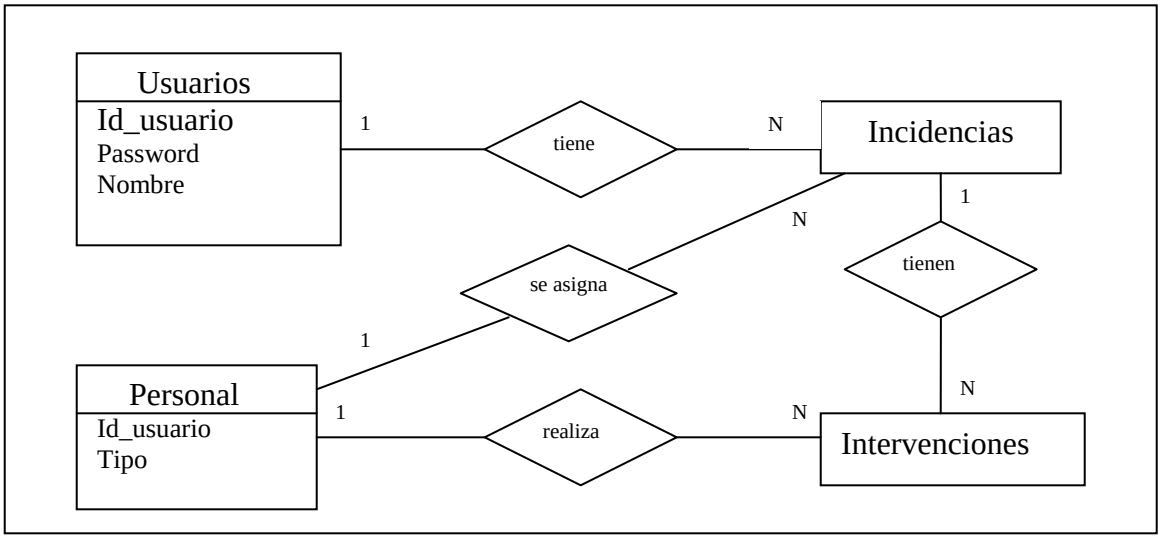
Si bien el método 3 parece que sería el idóneo en nuestro caso (subclases sin atributos) no podemos utilizarlo ya que la clase Usuario debemos considerarla como externa (LDAP) en el sistema productivo y no podemos tocarla.

Sin embargo, si podemos aplicarlo entre Técnico y Cau y por tanto los sustuimos por una nueva clase Personal que tendrá un atributo "tipo" que nos dirá si es Cau o técnico.

Ahora aplicaremos el método 2 y le añadiremos el atributo id_usuario a Personal y como vemos la Clase Cliente aporta información redundante (de usuario) podemos eliminarla, ya que todo usuario es cliente del sistema.

Transformación del Modelo Estático a Entidad-Relación

Tras la supresión de la herencia, para pasar al modelo E-R hay que crear una entidad por cada clase y añadir nuevas entidades para los atributos múltiples (no tenemos en nuestro caso). El diagrama queda como sigue:



Transformación de E-R a Relacional

Vemos que todas las interrelaciones son 1-N y sin atributos. En este tipo de interrelaciones, para pasar al modelo relacional hay que incluir en la tabla del lado N la clave primaria del lado 1.

Se cree conveniente que los atributos sistema y tipoincidencia de la entidad Incidencia deban existir en tablas de validación, de forma que los atributos contengan una referencia a la tabla, que será la que contenga la descripción correspondiente.

Las tablas quedarán de la siguiente forma:

Tabla USUARIO:
- id_usuario: String, (Clave Primaria)
- nombre: String
- password: String.

Tabla PERSONAL:
- id _ usuario: String (Clave Primaria)
- tipo: char.

Tabla INCIDENCIA:
- Código: long (Clave Primaria)
- id_usuario: String, (Clave foránea de USUARIO)
- id_personal: String, (Clave foránea de PERSONAL)
- cod_maquina: String
- descripción: String
- telefono_usuario: String
- fecha_hora: Date
- sistema: String (Clave foránea de SISTEMA)
- tipo: String (Clave Foranea de TIPOINCIDENCIA)
- estado: integer,

Tabla INTERVENCION:
- codigo: long (Clave Primaria)
- cod_incidencia: long (Clave foránea de INCIDENCIA)
- descripción: String,
- fecha_hora: Date
- tiempo: integer.

Tabla SISTEMA:
- codigo: String (Clave Primaria)
- descripción: String;

Tabla TIPOINCIDENCIA:
- codigo: String (Clave Primaria)
- descripción: String;

Definición de los gestores de persistencia

Para acceder a los datos aplicaremos el patrón de diseño DAO. Para ello definiremos los correspondientes DAO para trabajar con el modelo. La persistencia se implementará mediante hibernate, por lo que deberán configurarse los archivos correspondientes y crear los pojos necesarios.

UsuarioDAO, accederá a las tablas USUARIO y PERSONAL para gestionar la persistencia, utilizando un objeto UsuarioVO.

IncidenciaDAO accederá a la tabla INCIDENCIA y a IntervencionDAO para gestionar la persistencia y utilizará el objeto IncidenciaVO para la transferencia de la información.

IntervencionDAO accederá a la tabla INTERVENCION para gestionar la persistencia y utilizará el objeto IntervencionVO para la transferencia de la información

.

Para acceder a las tablas de validación SISTEMA y TIPOINCIDENCIA se crearán sus correspondientes DAO, SistemaDAO y TipoIncidenciaDAO, y los correspondientes VO.

4.3.- Diseño de los Casos de Uso

Teniendo clara la arquitectura es necesario descubrir que clases y artefactos necesitaremos en nuestra aplicación, para ello utilizaremos los diagramas de secuencia, que darán una visión de la dinamica de la aplicación, qué artefactos concretos necesitaremos y qué metodos habrá que implementar en cada clase.

No incluiremos en el diagrama las clases que proporciona Strust. Sabemos que el Requests Procesor llamará a nuestras Actions y cargará nuestras JSP cuando se necesario y estás invocarán Actions según las peticiones de los usuarios.

Caso de Uso 1: Identificación en el Sistema

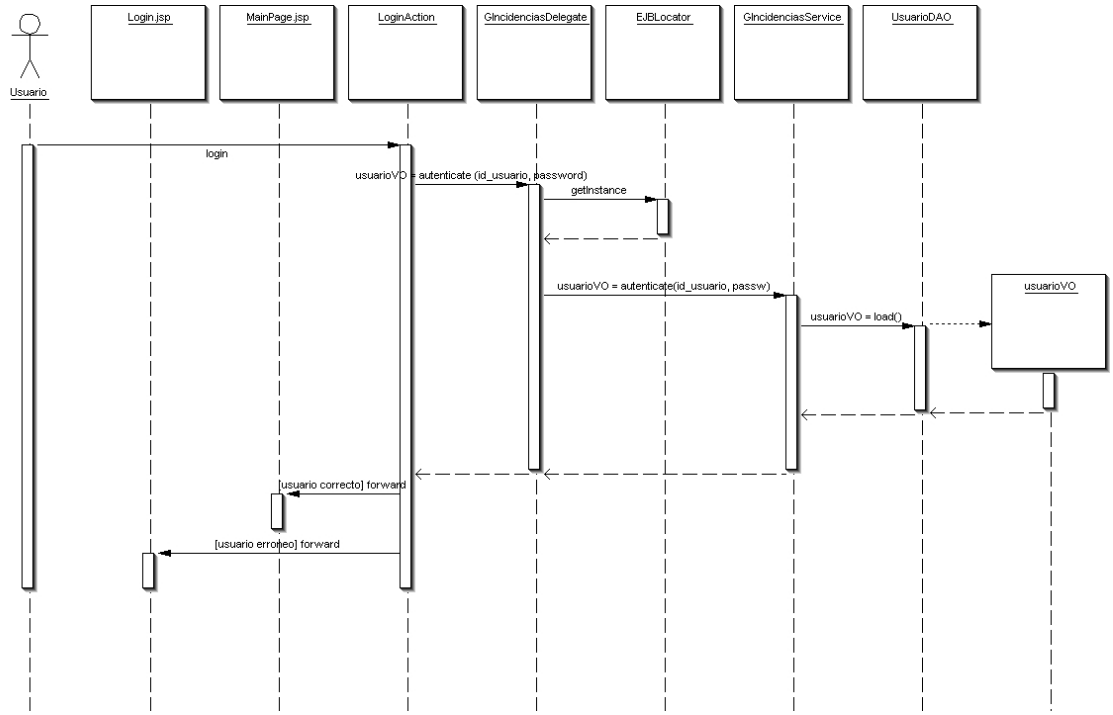


Fig. 21:Diagrama de secuencia Identificación en el Sistema

Caso de Uso 2: Registro de Incidencias

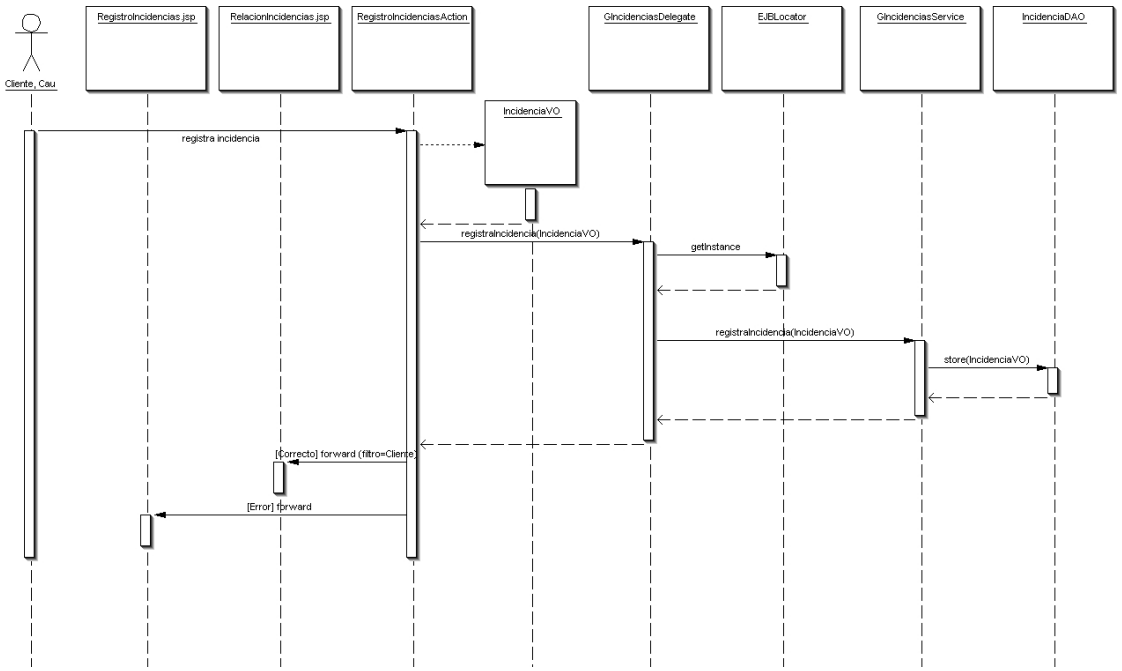


Fig. 22:Diagrama de secuencia Registro de Incidencias

Caso de Uso 3: Consulta de Incidencias

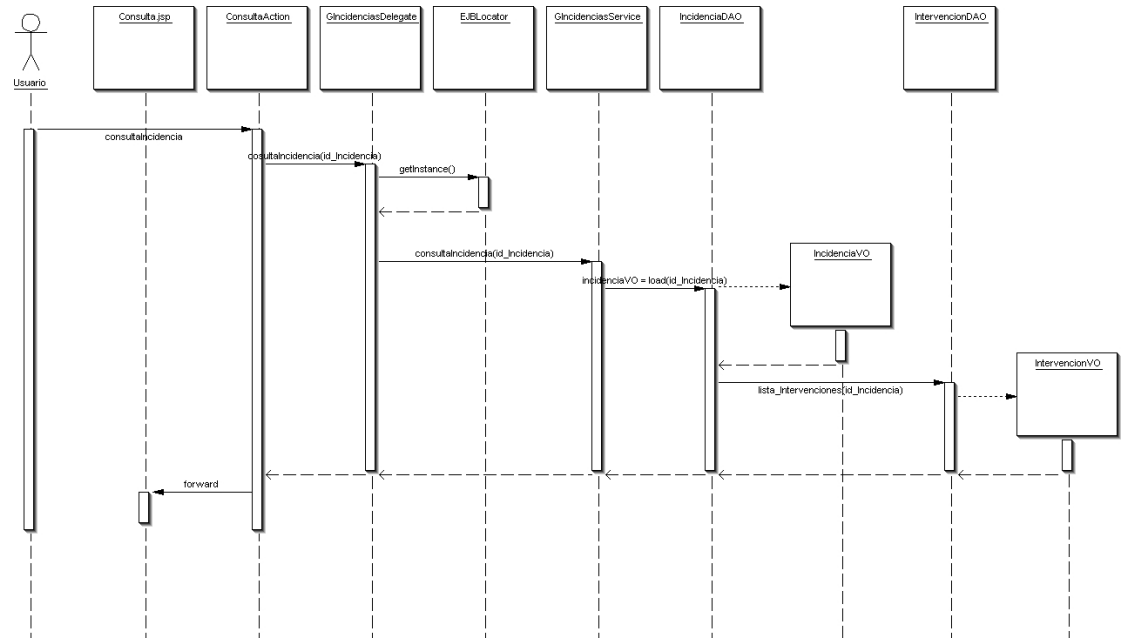


Fig. 23:Diagrama de secuencia Consulta de Incidencias

Caso de Uso 4: Asignación de Incidencias

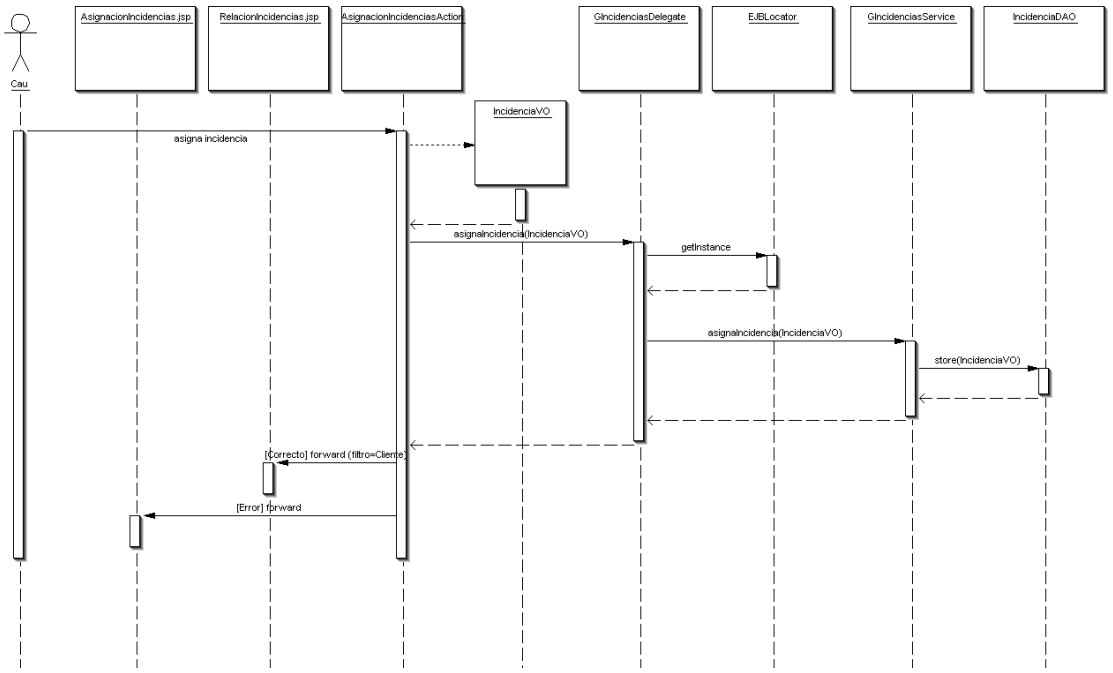


Fig. 24:Diagrama de secuencia Asignación de Incidencias

Caso de Uso 5: Registro de Intervenciones

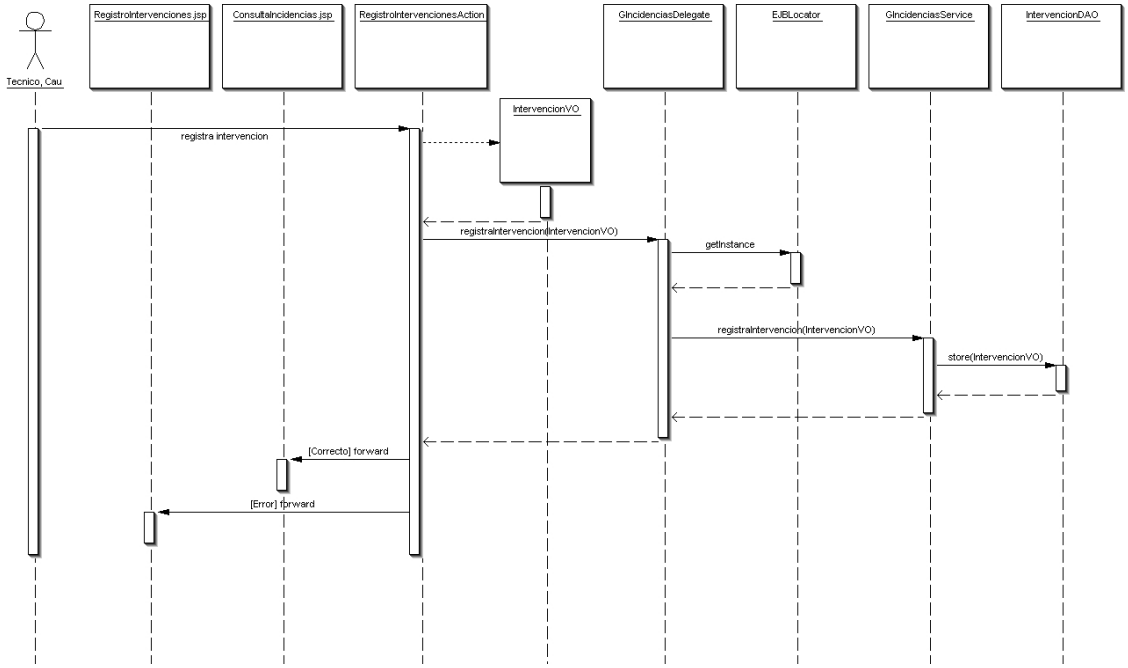


Fig. 25:Diagrama de secuencia Registro de Intervenciones

Caso de Uso 6: Relación de Incidencias

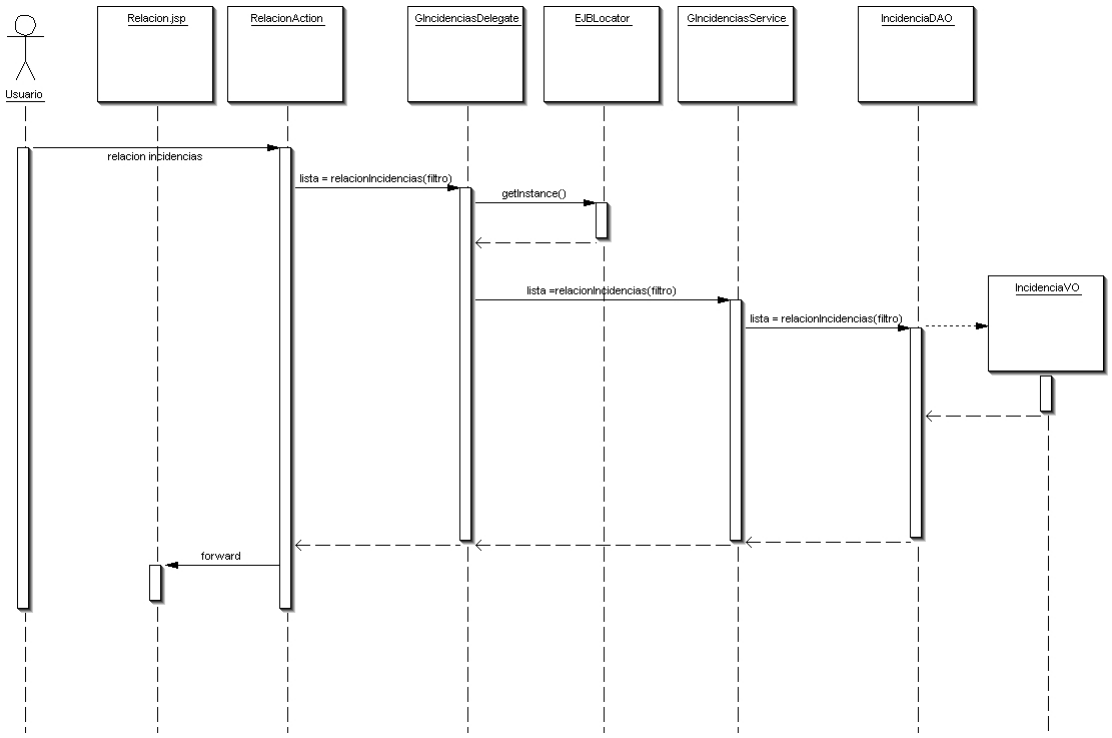


Fig. 26: Diagrama de secuencia Relación de Incidencias

4.4.- Diagrama Estático de Diseño

A partir de los diagramas de secuencia anteriores y una vez definido el acceso al modelo, podemos realizar el diagrama de clases de diseño.

Podemos identificar varios paquetes de diseño:

- JSPs: Paquete de paginas JSP que forman parte de la Vista de la aplicación. Se ejecutarán en el Contendor Web.

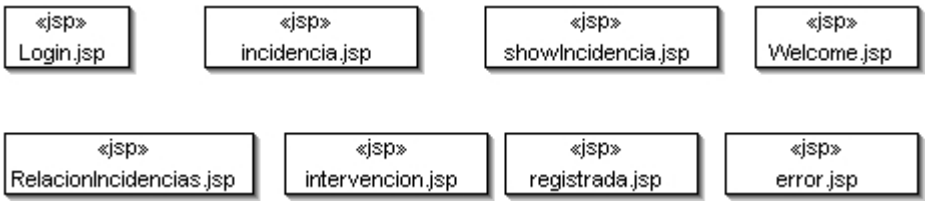


Fig. 27: Paquete JSPs

- Control: Paquete compuesto por las Actions, GISLocator y GIncidenciasDelegate que configuran el Controlador.

Fig. 28: Paquete Control

- Modelo: Paquete compuesto por el EJB de Sesión GIncidenciasService y los DAO para la gestión de la persistencia, la Factoria de Sesiones Hibernate y los pojos necesarios, que configuran el Modelo de la aplicación y que se ejecutan en el contenedor EJB.

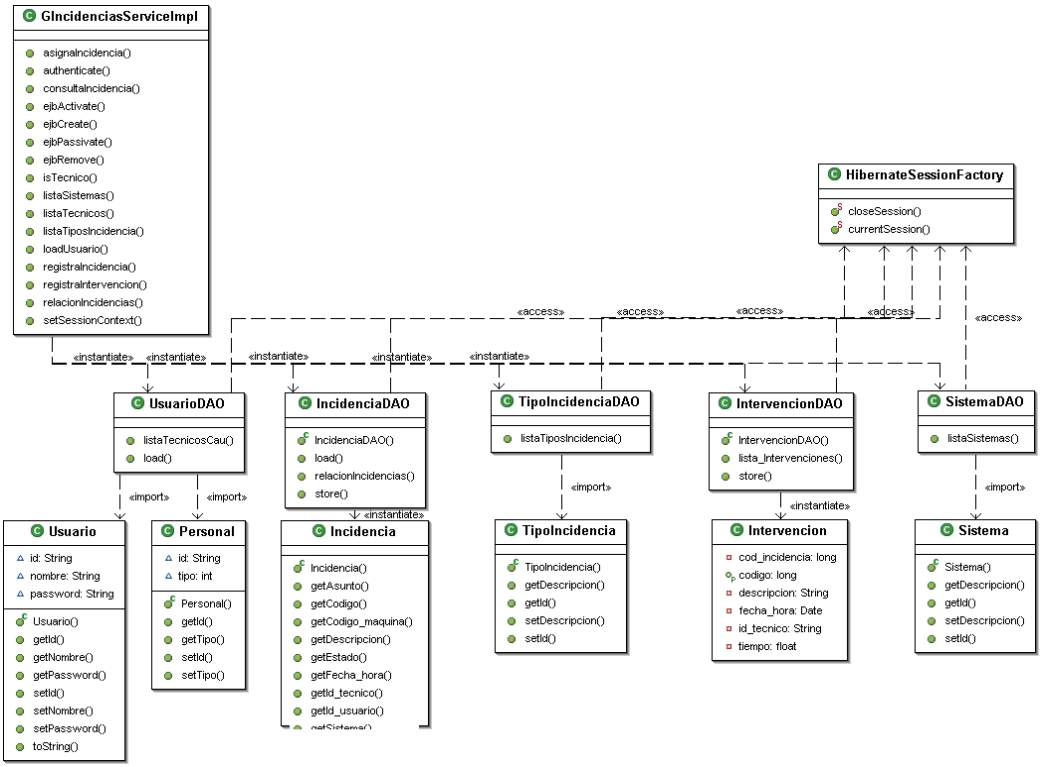


Fig. 29: Paquete Modelo

- VO: Paquete compuesto por los Value Objects (DTO) que al usarse para la transferencia de los datos se utilizan en ambos contenenedores..

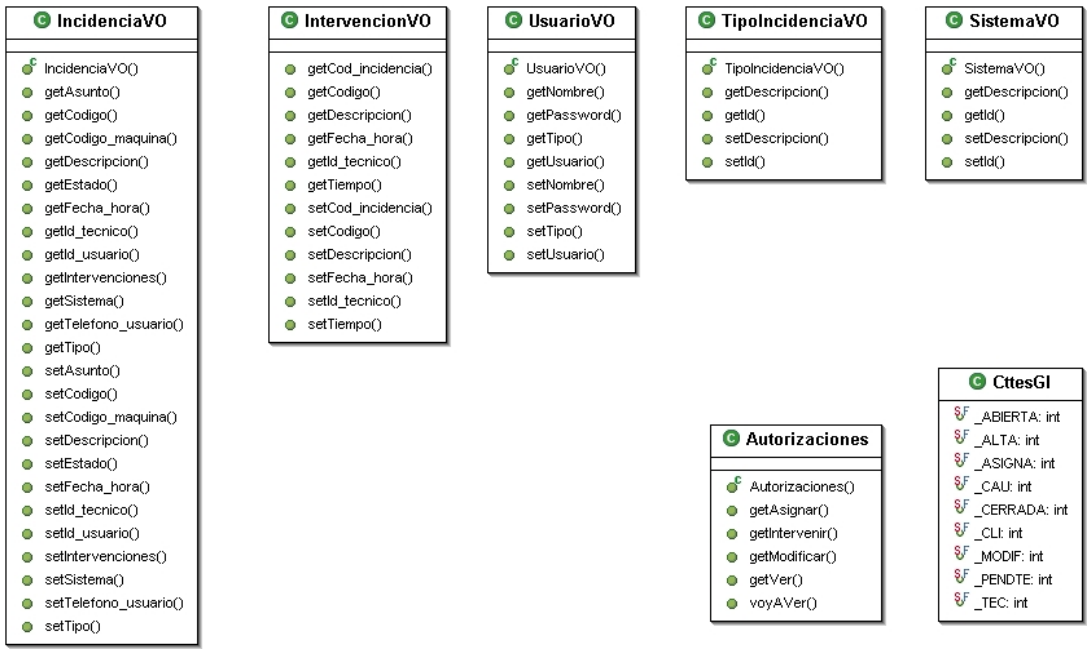


Fig. 30: Paquete VO

- Interfaces: Paquete compuesto por los Interfaces del GIncidenciasService, necesarios tanto en el contenedor web para localizar y usar el servicio remoto, como en el contenedor EJB para informar a éste de los servicios disponibles.

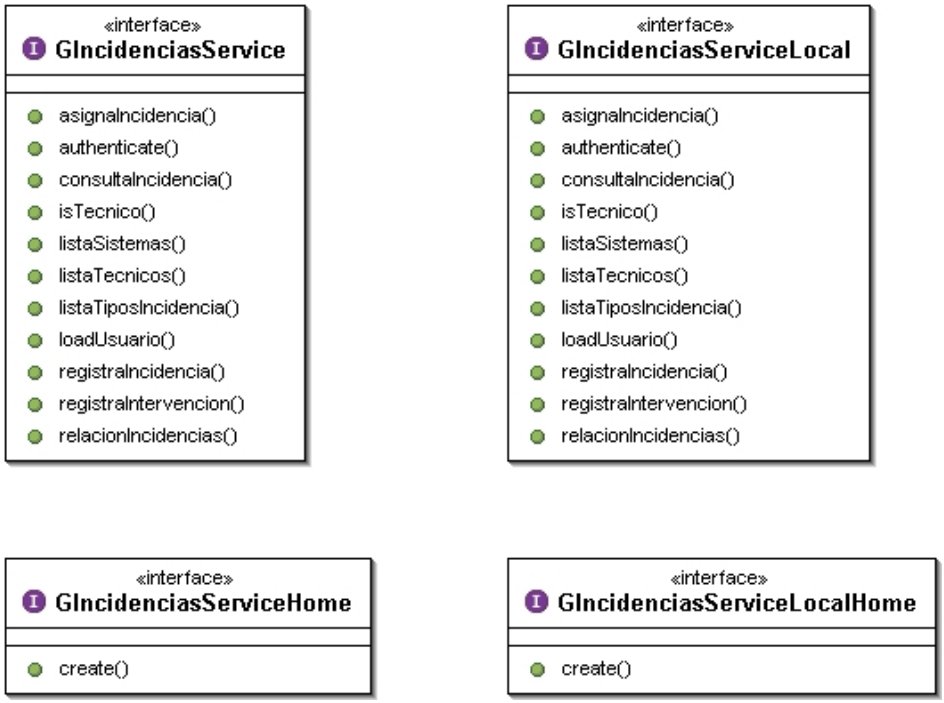


Fig. 31: Paquete Interfaces

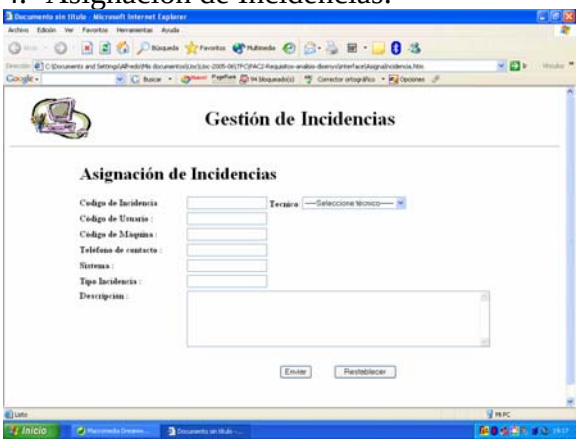
4.5.- Diseño de la Interficie de Usuario

1.- Identificación en el sistema

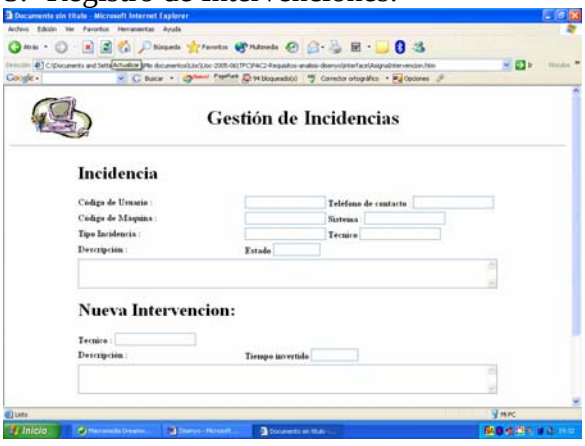
2.- Registro de incidencias:

3.- Consulta de Incidencias:

4.- Asignación de Incidencias:



5.- Registro de Intervenciones:



6.- Relación de Incidencias:

Fig. 32: Interficie de Usuario

5.- Implementación.

5.1.- Estructura de Desarrollo

Para la construcción de la aplicación Gestión de Incidencias se ha utilizado la plataforma de desarrollo Eclipse en su versión 3.0.2

La estructura de directorios utilizada para almacenar los artefactos dentro del proyecto es la siguiente:

Nombre	**Módulo**	Descripción	**Path**
JSRC	Desarrollo Java	**Fuentes Java**	/src/
JBIN	Desarrollo Java	Binarios Java	/bin/
STRUTS-LIB	Framework Struts	Librerías	/WEB-INF/lib/
STRUTS-CFG	Framework Struts	**Configuración**	/WEB-INF/
STRUTS-REC	Framework Struts	**Archivo de recursos**	/WEB-INF/classes/
STRUTS-TAG	Framework Struts	Librerías de tags	/WEB-INF/
HIB-LIB	Hibernate	Librerías	/lib-hibernate/
HIB-CFG	Hibernate	**Configuración**	/cfg-hibernate/
JSP-IDX	Desarrollo JSP	**Index.jsp**	/
JSP-PAG	Desarrollo JSP	**Páginas jsp**	/pages/
JSP-IMG	Desarrollo JSP	**Imágenes para jsp**	/img/

En negrita constan los artefactos que han tenido que ser desarrollados o modificados para la implementación de la aplicación.

Para la compilación de la aplicación es necesario configurar el proyecto eclipse de la siguiente forma:

Source path: path de JSRC
Output path: path de JBIN
Librerias JRE: path de JRE System Library
Librerias de J2EE: path de J2EE 1.4 Library
Librerias Struts: path de STRUTS-LIB
Librerias Hibernate: path de HIB-LIB

El desarrollo de las clases java se realiza bajo el paquete gincidencias que contiene 4 subpaquetes cuyas dependencias son las siguientes:

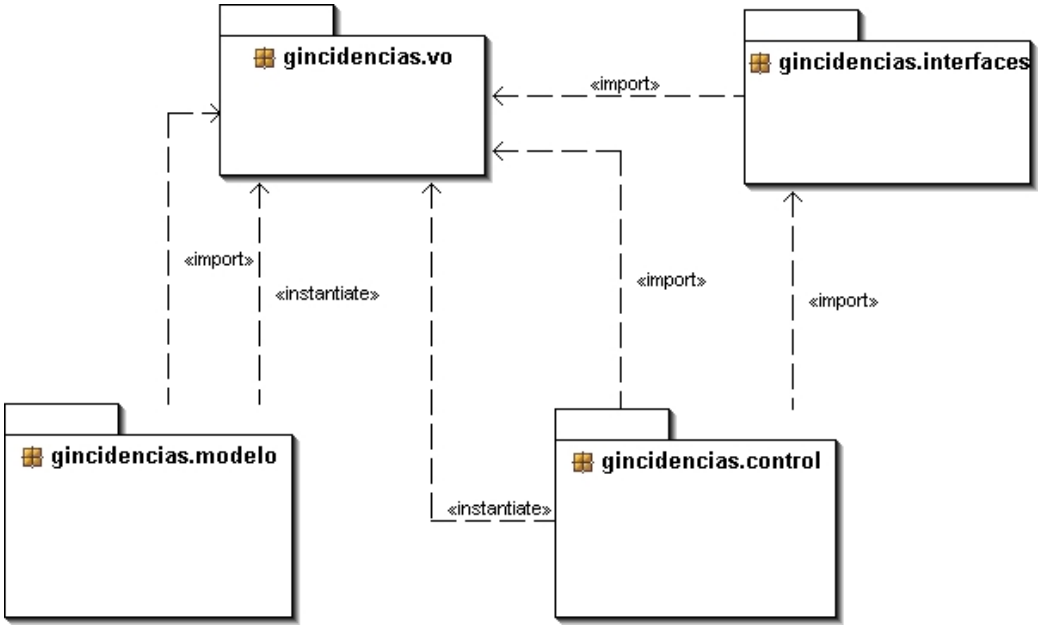


Fig. 33: Distribución de paquetes

El paquete **control** contiene las Actions de la aplicación así como los helpers beans necesarios para las paginas JSP, y el bussines delegate para acceder al modelo.

El paquete **modelo** contiene el java bean de sesión que implementa la lógica de negocio, junto con los DAOs y Pojos necesarios para la implementación de la persistencia.

El paquete **interfaces** contiene las interficies remotas, local y home del java bean de sesión, así que aunque en el gráfico no aparezcan relacionados, hay que tener en cuenta que su relación es muy estrecha ya que el java bean de sesión del modelo implementa dichos interfaces.

El paquete **vo** contiene los Value Objects que se utilizan para el traspaso de información entre las diferentes clases de toda la aplicación. Es por esto por lo que está referenciado por todos los demas.

Debido al uso común del paquete **vo,** se ha incluido en él la clase CttesGI, que contiene las constantes utilizadas en la aplicación y la clase Autorizaciones que centraliza la lógica de autorización de la aplicación.

5.2.- Diagrama de Componentes

En el siguiente gráfico vemos la distribución de los componentes de la aplicación:

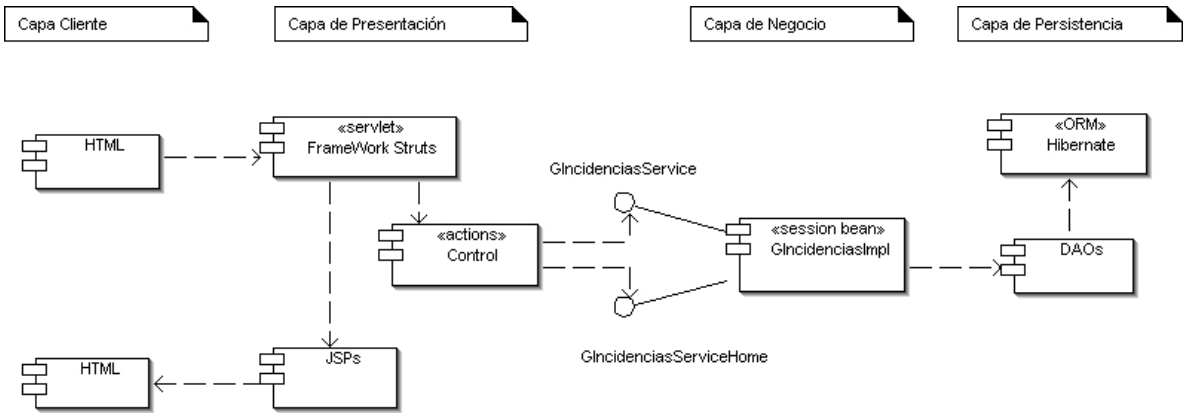


Fig. 34: Diagrama de Componentes

Podemos observar que realmente hemos implementado el Patrón MVC de Nivel 2 donde tenemos separada la Vista, el Controlador y el Modelo.

La **Vista** esta formada por el componente JSP, que utilizará los helpers beans que le proporciona el controlador.

El **Controlador** esta formado por el componente FrameWork Struts y nuestro componente control (compuesto por los paquetes *control*, *vo* e *interfaces*).

El **Modelo** esta formado por los componentes GincidenciaServiceImpl, el componente DAOs (implementados en el paquete *modelo*, *vo* y directamente relacionado con *interfaces*) y el ORM Hibernate.

5.3.- Diagrama de Despliegue

La aplicación está diseñada para que se ejecute en un servidor que posea un contenedor web y un contenedor de EJB.

La capa de Presentación (Vista y Controlador) se ejecutarán en el Contenedor Web y las Capas de Negocio y Persistencia (Modelo) se ejecutarán en el Contendor EJB. Para poder desplegar la aplicación es necesario generar los componentes de despligue GestionIncidencias.war para el contendor web y GestionIncidencias.jas para el contendor de EJB.

Además se debe configurar los descriptores de despligue correspondientes: web.xml (ubicado en /WEB-INF/) para el war y ejb.xml (ubicado en /META-INF/) para el jar.

La creación de dichos componentes de despliegue se ha realizado utilizando las herramientas de packaging que proporciona Eclipse.

Para ello hemos configurado la opción de Packaging Configuration de la siguiente forma:

Path Proyecto	**GestionIncidencias.war**
/bin/*	/WEB-INF/classes
/WEB-INF/*	/WEB-INF
/index.jsp	/
/pages/*	/pages
/img/*	/img

Path Proyecto	**GestionIncidencias.jar**
/bin/*	/
/lib-hibernate/*	/lib
/cfg-hibernate/*	/
/META-INF/*	/META-INF

El diagrama de despligue es el siguiente:

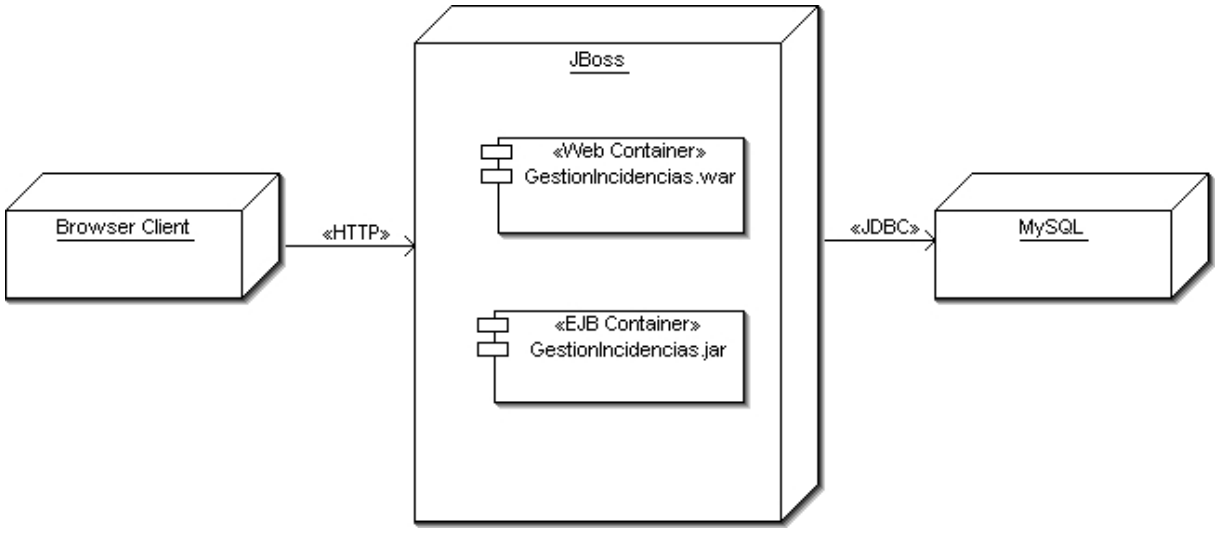


Fig. 35: Diagrama de Despliegue

5.4.- Apuntes sobre la Implementación

Patrones implementados

En el desarrollo de las diferentes clases desarrolladas se han implementado diversos patrones, siguiendo las recomendaciones de Sun para el desarrollo de aplicaciones J2EE.

Bussines Delegate: es implementado por la clase GIncidendiasDelegate. Encapsula la localización de servicios y los independiza de los errores de ejecución remota.

Facade: es implementado por GIncidenciasService, el EJB de sesión, que proporciona la interfaz de servicios del modelo.

Service Locator: implementado por GISLocator. Singleton que se guarda la referencia a los sevicios remotos.

DAO: Es implementado por las clases DAO del modelo y permite ocultar los detalles de la implementación del acceso a datos.

Value Objects: implementado por las clases vo; toda la transferencia de información se realiza mediante objetos de transferencia, optimizando las comunicaciones y minimizando las llamadas de acceso a datos.

Helper View: Ayudante de vista. A parte de los ActionForm que incorpora Struts se han añadido dos clases en el paquete control (cfgShowIncidenciaForm y cfgIncidenciaForm) que son creadas e inicializadas en las Action y utilizadas como configuración para las JSP correspondientes.

Implementación de la seguridad

La seguridad de acceso a la aplicación se realiza mediante la validación del usuario y password correspondiente. La acción ActionLogin inicializa objetos a nivel de sesión que posteriormente son comprobados y utilizadas por BaseAction, clase de la que heredan el resto de actions.

Para evitar el acceso a opciones no permitidas se ha implementado la clase Autorizaciones que centraliza la lógica de acceso en función del usuario que accede (fundamentalmente su tipo), la incidencia (su estado y usuario) y el tratamiento que se requiere.

Todos los Action crean un objeto de autorización que les dice si el usuario puede o no puede realizar la acción correspondiente. La superclase dispone de un metodo de utilidad al que llamar si el acceso es no permitido.

Si bien la aplicación muestra al usuario la opciones permitidas, con estas verificaciones se pretende evitar violaciones de acceso mediante mediante la modificación directa de parámetros en la url.

Control de Excepciones

Los métodos de la clases de la aplicación capturan sólo las Excepciones que pueden tratar y relanzan o dejar fluir el resto. Estas excepciones se capturan en la superclase BaseAction, que compone la traza del error, lo recoge en el log y hace un forward a la pantalla error.

De esta forma conseguimos que la aplicación no se rompa por un error, recoja la información suficiente para reportarlo y permita continuar con la aplicación si es posible (si el tipo de error lo permite).

6.- Conclusiones

Una vez completado el proceso de desarrollo es necesario analizar el recorrido realizado y sacar las conclusiones pertinentes:

Sobre el Plan de Proyecto

El Plan de Proyecto y su temporización se han seguido tal y como se definieron inicialmente. Ha habido los lógicos desplazamientos en las fechas de inicio y fin para algunas tareas que, por el desconocimiento inicial de la tecnología a aplicar, eran difíciles de evaluar. De todas formas, su impacto en el proyecto no ha sido relevante.

La carga de las tareas de formación ha resultado mayor de lo esperado, por lo que se han tenido que dedicar más horas de las previstas al proyecto, sobre todo en la fase inicial, para no dilatar la fecha de finalización de las mismas, ya que tenían que estar listas antes de comenzar la implementación.

Todo ello no ha impedido que se hayan cumplido todos los hitos programados en la planificación del proyecto.

Sobre la metodología utilizada

El seguimiento del ciclo de vida iterativo e incremental para el desarrollo del software, con la utilización de UML como herramienta de modelado del sistema y la aplicación de los pasos en especificación, análisis y diseño, ha sido fundamental para la gestión y el control de proyecto.

En todo momento se sabía en que punto del proyecto se estaba trabajando, cual era su cometido y que resultado debía ofrecer a la tarea que le seguía.

La sensación de control del proceso de desarrollo ha sido total y para ello ha sido fundamental tener claro el desglose de las tareas a realizar, las precedencias correspondientes y la documentación a generar en cada momento.

Sobre el alcance de los objetivos

Tal y como ya expusimos en los Objetivos del TFC, punto 1.2, tenemos dos tipos de objetivos diferenciados: los relacionados con la funcionalidad del software y los académicos.

Sobre los objetivos relacionados con la **funcionalidad** del software podemos decir que se han desarrollado todas las especificaciones y casos de uso relacionados en la Documentación de Requisitos. Fue acertado incluir solo los casos de uso de índole operativa en esta iteración del proceso de construcción, dejando para un futuro ampliar las funcionalidades.

Con respecto a los objetivos **académicos,** las espectativas se han visto desbordadas. Esperaba adquirir y prácticar una serie de conocimientos concretos: poner en práctica la metodología de gestión de proyectos en un desarrollo real, aprender qué es J2EE y de qué se compone, diseñar usando patrones de diseño, poner en marcha un ORM… y todo ello dentro de un framework concreto. Esto se ha conseguido ampliamente.

Estoy fráncamente satisfecho con el trabajo realizado, con la tecnología aprendida y aplicada y con la base conceptual que se desprende del paradigma J2EE. He visto como las ideas y los conceptos, integran y diseñan sistemas. He visto como herramientas, librerías y frameworks colaboran en la creación de aplicaciones. He visto profesionalidad e ingenio. He "mamado" Ingeniería..., que al fin y al cabo, es de lo que se trata.

Personalmente he notado como a medida que me introducía en esta tecnología volvía a oir esa especie de música que envuelve a los desarrolladores cuando andan por una senda acertada. Al principio eran intrumentos sueltos… el "bajo" va dando algunos toques (Hibernate y su sonido ORM), las flautas empiezan a marcar un ritmo de frescura y orden (patrón, patrón), juntándose enseguida la percusión (Struts) y la cuerda como hilo integrador (Jboss) . Mas tarde se unió la "prima dona" con sus fantásticas arias (EJB), para dar color a la orquesta. Sin darte cuenta te encuentras al frente del teclado, tecleando y tecleando las notas de una melodía pensada para "orquesta y piano"... y te das cuenta de que eres el "teclista de una banda de software".

Líneas futuras a seguir

Para poner en **producción** la aplicación Gestión de Incidencias, todavía hace falta dar algunos pasos más:

1.- Tal y como se comenta en los requisitos del proyecto hay que sustituir la tabla usuarios por el acceso a un LDAP empresarial, antes de pasarlo a producción. El patrón DAO implementado en la clase UsuarioDAO va a ser fundamental para que esta modificación minimice el impacto en el proyecto.

2.- Ampliar el control de errores aplicando algunos de los principios más importantes para el trabajo con excepciones en una aplicación empresarial:

- Desarrollar un árbol de clases Exception
- Independizar las excepciones de usuario de las del sistema.
- Darle a las excepciones un contexto, un tipo y un nivel de seguridad.
- Independizar el manejo de excepciones y el logging a la aplicación.
- Crear una facilidad para el encadenado de excepciones.
- Externalizar los strings de las excepciones para su internacionalización.

3.- Incorporar logging a la aplicación, para poder activarlo en caso de problemas cuando pase a producción.

4.- Realizar pruebas de carga y tunning de la aplicación.

Por otro lado hay que ir pensado en añadir funcionalidades que faciliten la configuración, la busqueda de información y la obtención de resultados de gestión:

a) Mantenimiento de Cau, Técnicos, Sistemas y Tipos de Incidencia
b) Opcion de busqueda de incidencias según diversos criterios
c) Elaboración de resultados sobre incidencias según diversas agrupaciones (temporales, por tipos, técnicos, sistemas, etc), que permitan ver el nivel de calidad de servicio.

Tecnológicamente hay que continuar trabajando, queda mucho por hacer ya que en el fondo no hemos hecho más que empezar, con el objetivo de completar la definición de nuestro entorno de desarrollo:

1. Incorporación de Ant para la creación y despliegue del proyecto, de forma que se independice de un entorno gráfico como es eclipse.
2. Estudio e incorporación de herramientas de loggin (log4j).
3. Estudio e incorporación de heramientas de prueba y stress (JUnit, Cactus).
4. Ampliación y consolidación de los conceptos de ORM e Hibernate.
5. Ampliación y consolidación del conocimiento de Struts

No hay que perder de vista que es fundamental estar al tanto de nuevas especificaciones, frameworks y apis, hacerles un seguimiento e incorporar a nuestro entorno aquello que creamos imprescindible.

Con este proyecto se cierra un ciclo formativo formal, que se inició hace años con ilusión y esfuerzo. Ahora se abre otro sin fecha de finalización y que consiste en continuar aprendiendo, creciendo e integrando los elementos necesarios para la generación de aplicaciones empresariales de calidad.

7.- Glosario.

Incidencia	Problema informático que tiene un usuario con el sistema que precisa de la ayuda o intervención de un especialista para su resolución.
Cliente	Persona que utiliza el sistema informático y que puede tener incidencias en su uso.
CAU	Centro de Atención al Usuario. Departamento que se encarga de gestionar la incidencias de los usuarios, asignándoles técnicos de soporte y clasificándolas
Técnico de Soporte	Especialista informático encargado de solucionar los problemas que los usuarios tienen con el sistema.
Intervención	Acción realizada para solucionar parte o toda la problemática de una incidencia de un usuario.
J2EE	Java 2 Enterprise Edition – Especificaciones para el desarrollo de aplicaciones empresariales
ORM	Object-Relational Mapping – Puente entre el mundo de los objetos y las bases de datos relacionales
MVC	Modelo-Vista-Controlador – Patrón de diseño que consiste en desacoplar los datos, la logica de la aplicación y el interface del usuario.
Hibernate	ORM open source de reconocido prestigio en la comunidad de desarrolladores
Struts	Framework que implementa el patrón MVC para J2EE

8.- Bibliografía y enlaces.

J2EE

The J2EE™ 1.4 Tutorial; Sun Microsystems

http://java.sun.com/j2ee/index.jsp
http://java.sun.com/products/jsp/jstl/
http://www.programacion.com/java/tutorial/javabeans/2/

Patrones

Thinking in Patterns; Bruce Eckel
Core J2EE™ Patterns: Best Practices and Design Strategies, Second Edition; By Deepak Alur, John Crupi, Dan Malks, Publisher: Prentice Hall PTR

http://java.sun.com/blueprints/patterns/index.html
http://java.sun.com/blueprints/corej2eepatterns/index.html
http://www.programacion.com/java/tutorial/patrones

Struts

Struts in Action; Ted Husted, Cedric Dumoulin, George Franciscus, David Winterfeldt; Manning Publ.Co.
Programming Jakarta Struts; By Chuck Cavaness; O'Reilly & Associates, Inc.
Mastering Jakarta Struts; James Goodwill; Wiley Publishing, Inc.

http://struts.apache.org/
http://jakarta.apache.org/taglibs/index.html
http://www.programacion.net/java/tutorial/struts/
http://www.programacion.com/java/articulo/tips_struts/

Hibernate

http://hibernate.org/
http://www.hibernate.org/hib_docs/reference/en/pdf/hibernate_reference.pdf
http://www.programacion.com/java/tutorial/hibernate/
http://www.programacion.com/java/articulo/jap_persis_hib/#3_ficchero

UML

Aprendiendo UML en 24 horas; Joseph Schmuller; Prentice Hall

http://www.clikear.com/manuales/uml/index.asp

JBoss

http://www.jboss.org/

9.- Anexos

9.1.- Manual de Instalación

Para la instalación la aplicación es necesario lo siguiente:

1. Disponer de un servidor Jboss instalado. La versión 4.0.2 funciona perfectamente.
2. Disponer de un servidor MySql instalado. La versión 4.1 funciona perfectamente.
3. Ejecutar el script de creación de la base de datos de Gincidencias en el servidor MySql. Este script hace lo siguiente:
   - Crea la Base de Datos y las tablas.
   - Inserta registros en las tablas de validación Sistemas y TipoIncidencia, y crea los usuarios de prueba (usuario1, usuario2, tecnico1, tecnico2, cau1 y cau2 todos con el mismo password que el usuario).
   - Crea el usuario que desde Hibernate debe acceder a la base de datos (usuario: gincid; password: gi)
4. Desplegar los componentes war y jar en el servidor Jboss.
5. Ejecutar en el cliente la url: http://localhost:8080/GestionIncidencias donde localhost deberá sustituirse por el servidor que tenga el Jboss.

El script de creacion de la base de datos se denomina GIcreaDB.sql y está ubicado en el directorio raiz del Proyecto Eclipse y éste es su contenido:

```
--
-- Crea el Schema GIncidencias
--

CREATE DATABASE gincidencias;
USE gincidencias;

--
-- Table structure for table `gincidencias`.`personal`
--

DROP TABLE IF EXISTS `personal`;
CREATE TABLE `personal` (
  `id_personal` varchar(20) NOT NULL default '',
  `tipo` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`id_personal`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `gincidencias`.`personal`
--

INSERT INTO `personal` (`id_personal`,`tipo`) VALUES
 ('Cau1',2),
 ('Cau2',2),
 ('Tecnico1',1),
 ('Tecnico2',1);
```

```
--
-- Table structure for table `gincidencias`.`sistemas`
--

DROP TABLE IF EXISTS `sistemas`;
CREATE TABLE `sistemas` (
  `id` varchar(5) NOT NULL default '',
  `descripcion` varchar(45) NOT NULL default '',
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `gincidencias`.`sistemas`
--

INSERT INTO `sistemas` (`id`,`descripcion`) VALUES
 ('EMAIL','Correo Electronico'),
 ('FW','Firewall'),
 ('INET','Red Internet'),
 ('NET','Red Corporativa'),
 ('OFI','Ofimática'),
 ('PROD','Produccion'),
 ('SAP','Administracion - SAP'),
 ('WS','Estacion de Trabajo');

--
-- Table structure for table `gincidencias`.`tiposincidencia`
--

DROP TABLE IF EXISTS `tiposincidencia`;
CREATE TABLE `tiposincidencia` (
  `id` varchar(5) NOT NULL default '',
  `descripcion` varchar(45) NOT NULL default '',
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `gincidencias`.`tiposincidencia`
--

INSERT INTO `tiposincidencia` (`id`,`descripcion`) VALUES
 ('CFG','Configuracion'),
 ('EHARD','Error Hardware'),
 ('ESOFT','Error Software'),
 ('IHARD','Instalacion Hardware'),
 ('ISOFT','Instalacion Software'),
 ('QRY','Consulta');

--
-- Table structure for table `gincidencias`.`usuarios`
--

DROP TABLE IF EXISTS `usuarios`;
CREATE TABLE `usuarios` (
  `id_usuario` varchar(20) NOT NULL default '',
```

```
  `nombre` varchar(45) NOT NULL default '',
  `password` varchar(45) NOT NULL default '',
  PRIMARY KEY  (`id_usuario`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `gincidencias`.`usuarios`
--

INSERT INTO `usuarios` (`id_usuario`,`nombre`,`password`) VALUES
 ('Cau1','Nombre del Cau Uno','cau1'),
 ('Cau2','Nombre del Cau Dos','cau2'),
 ('Tecnico1','Nombre del Tecnico Uno','tecnico1'),
 ('Tecnico2','Nombre del Tecnico Dos','tecnico2'),
 ('Usuario1','Nombre del Usuario Uno','usuario1'),
 ('Usuario2','Nombre del Usuario Dos','usuario2');

--
-- Table structure for table `gincidencias`.`incidencias`
--

DROP TABLE IF EXISTS `incidencias`;
CREATE TABLE `incidencias` (
  `codigo` bigint(20) unsigned NOT NULL auto_increment,
  `id_usuario` varchar(20) NOT NULL default '',
  `telefono_usuario` varchar(20) default NULL,
  `codigo_maquina` varchar(20) default NULL,
  `tipo` varchar(20) default NULL,
  `sistema` varchar(20) default NULL,
  `asunto` varchar(80) NOT NULL default '',
  `descripcion` text NOT NULL,
  `id_tecnico` varchar(20) default NULL,
  `estado` int(10) unsigned NOT NULL default '0',
  `fecha_hora` datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (`codigo`),
  KEY `FK_incidencias_Tipo` (`tipo`),
  KEY `FK_incidencias_Sistema` (`sistema`),
  KEY `FK_incidencias_Tecnico` (`id_tecnico`),
  CONSTRAINT `FK_incidencias_Sistema` FOREIGN KEY (`sistema`) REFERENCES
`sistemas` (`id`),
  CONSTRAINT   `FK_incidencias_Tecnico`   FOREIGN   KEY   (`id_tecnico`)
REFERENCES `personal` (`id_personal`),
  CONSTRAINT  `FK_incidencias_Tipo`  FOREIGN  KEY  (`tipo`)  REFERENCES
`tiposincidencia` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='InnoDB free: 4096 kB';


--
-- Table structure for table `gincidencias`.`intervenciones`
--

DROP TABLE IF EXISTS `intervenciones`;
CREATE TABLE `intervenciones` (
  `codigo` bigint(20) unsigned NOT NULL auto_increment,
  `descripcion` text NOT NULL,
  `id_tecnico` varchar(20) NOT NULL default '',
```

```
  `tiempo` float NOT NULL default '0',
  `fecha_hora` datetime NOT NULL default '0000-00-00 00:00:00',
  `cod_incidencia` bigint(20) unsigned NOT NULL default '0',
  PRIMARY KEY  (`codigo`),
  KEY `FK_intervenciones_Incidencia` (`cod_incidencia`),
  KEY `FK_intervenciones_Tecnico` (`id_tecnico`),
  CONSTRAINT       `FK_intervenciones_Incidencia`       FOREIGN       KEY
(`cod_incidencia`) REFERENCES `incidencias` (`codigo`),
  CONSTRAINT   `FK_intervenciones_Tecnico`   FOREIGN   KEY   (`id_tecnico`)
REFERENCES `personal` (`id_personal`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Crea el usuario gincid para acceder a la base de datos
--

GRANT all ON GIncidencias.* TO gincid IDENTIFIED BY "gi";
```

9.2.- Manual del Usuario

El objeto de la aplicación Gestión de Incidencias es permitir el registro de incidencias por parte de los usuario a los que se les presentan, la asignación de las mismas a los técnicos encargados de solventarlas y el registro de las acciones que se han realizado para su resolución.

Tenemos tres tipos de usuario en esta aplicación:

- Usuario Cliente: Es el usuario al que se le presentan las incidencias y por tanto primer proveedor de información de esta aplicación.

- Usuario Cau: Es el usuario que verifica la corrección de los datos introducidos y responsable de la correcta escalación del problema a los técnicos más indicados.

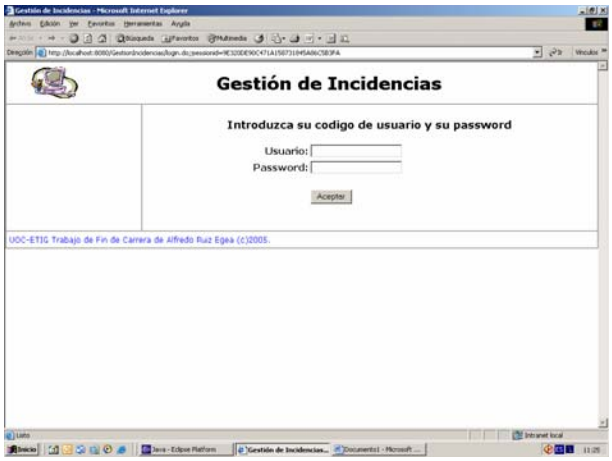
- Usuario Técnico: Son los encargados de reolver las incidencias y de registrar en el sistema la descripción de sus intervenciones y el tiempo que han empleado para ello.

Para cada tipo de usuario la aplicación provee de las opciones necesarias para la realización de sus acciones.

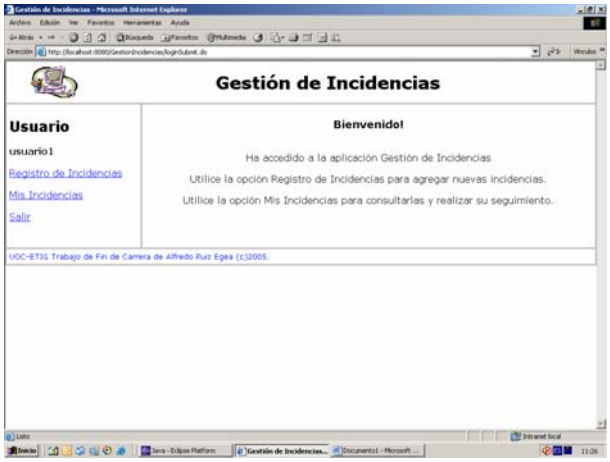
Hay que tener en cuenta que los técnicos pueden actuar como clientes del sistema (puede tener incidencias que resolverán ellos mismos u otros técnicos), y los Cau pueden ser clientes (también pueden tener incidencias propias) y técnicos (pueden registrar intervenciones en incidencias que no requieran la intervención de técnicos especialistas, o actuar de técnicos de primer nivel mediante soporte telefónico).

Opciones de Cliente

Identificación en el sistema:

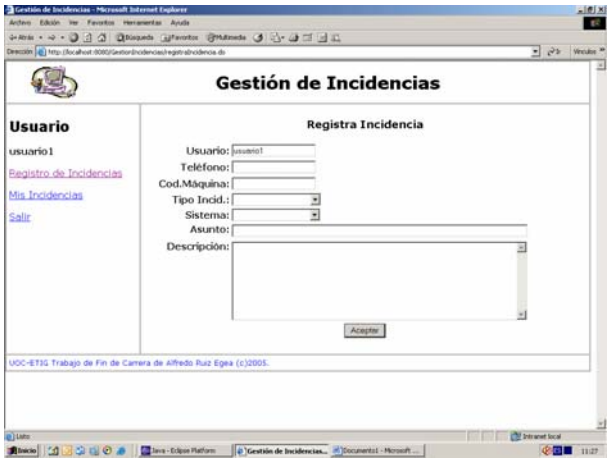


Para poder utilizar la aplicación cualquier usuario debe identificarse en el sistema introduciendo su identificador de usuario y su password.

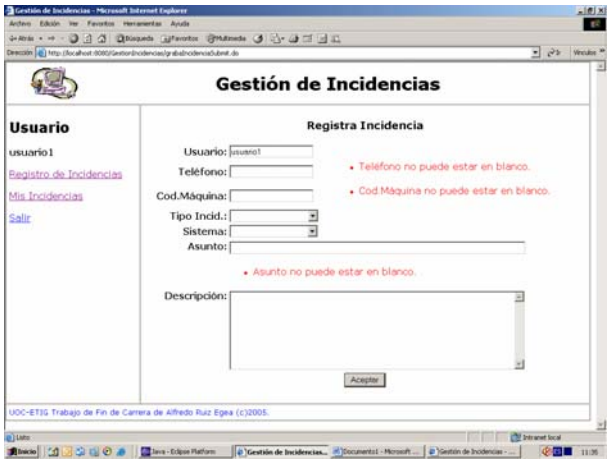


Tras la validación de la identificación el sistema da la bienvenida y muestra un menú personalizado según el tipo de usuario.

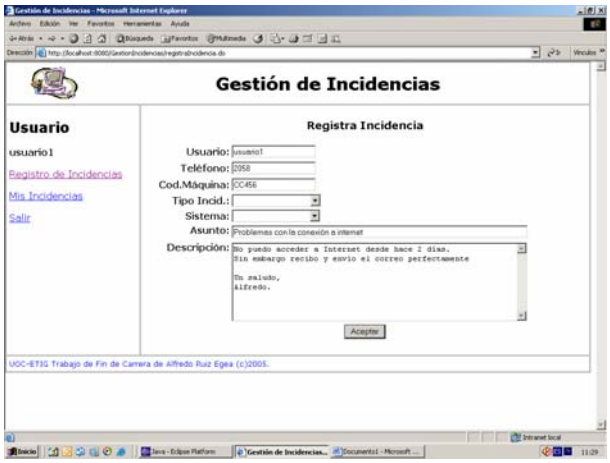
Registro de Incidencias:



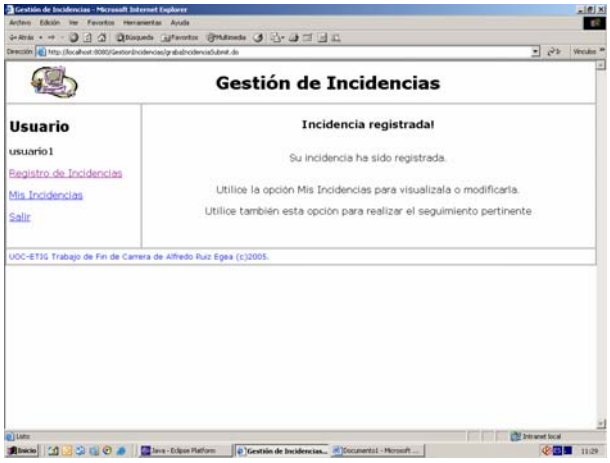
Tras pulsar la opción de “Registro de incidencias” el sistema le muestra la pantalla de entrada de datos que el usuario deberá cumplimentar.



Los campos obligatorios son, a parte del Usuario al que se refiere la incidencia, que cumplimenta automáticamente el sistema, el teléfono de contaco, el código de maquina y el asunto de la incidencia.

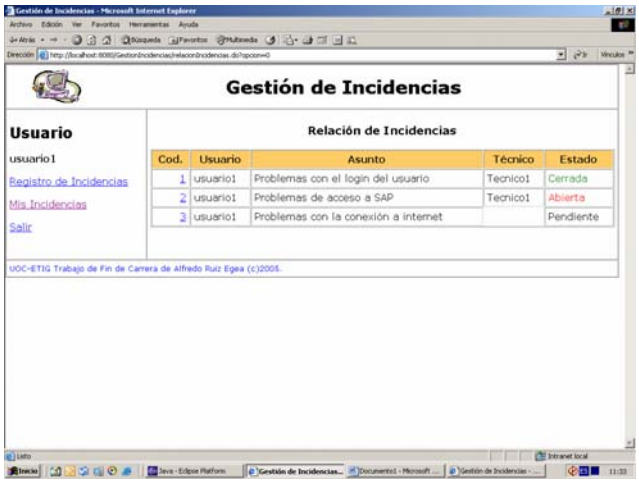


Una vez cumplimentados los datos el usuario debe pulsar el botón “Aceptar” para introducir la incidencia en el sistema.



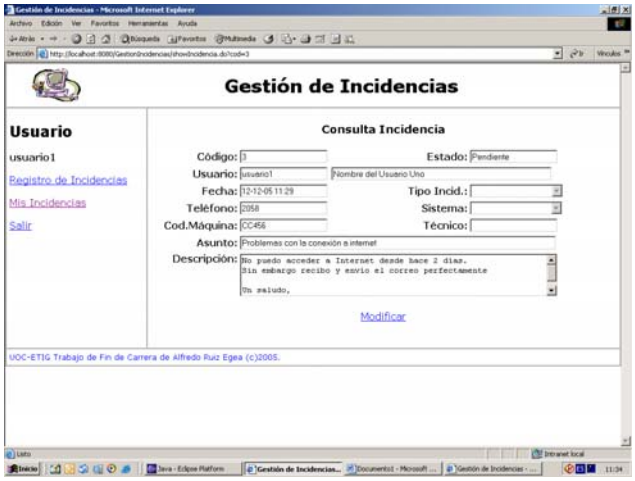
El sistema le notifica que la incidencia ha sido registrada en el sistema y que puede visualizarla mediante la opción “Mis Incidencias”.

Mis Incidencias:



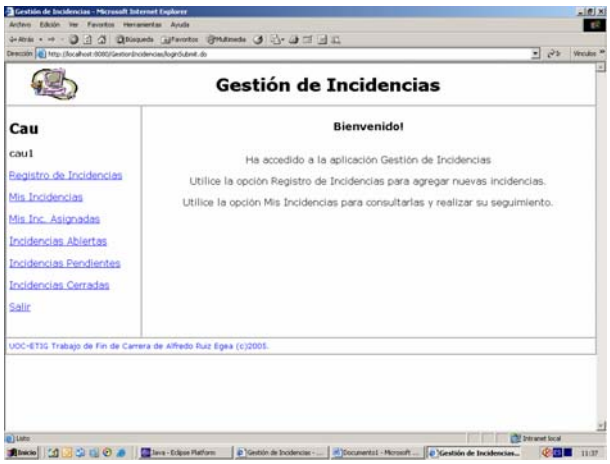
Mediante la opción “Mis Incidencias” obtendrá la relación de incidencias introducidas por él (o en su nombre, por un Cau).

Consulta Incidencia:



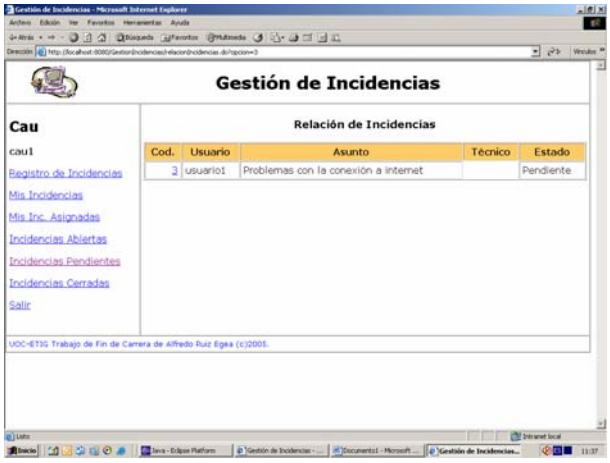
Pulsando sobre el código de incidencia esta se visualiza. Mientras su estado sea “pendiente” podrá modificarla (volverá a la pantalla de introducción de datos).

Opciones de Cau



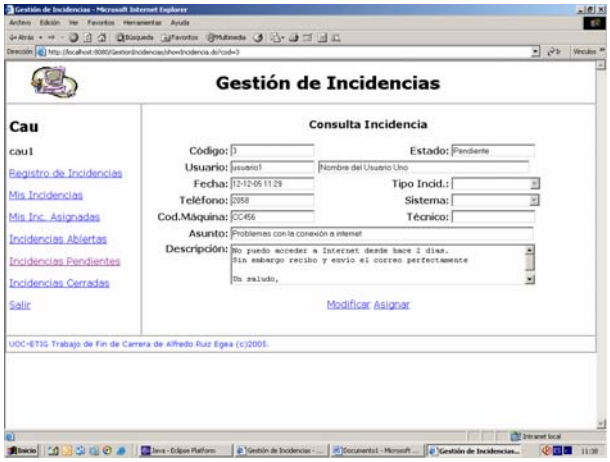
Una vez identificado un Cau le aparece un menú con sus opciones permitidas.
La tarea princpal del usuario Cau es revisar las incidencias pendientes (introducidas por los usuarios clientes) y asignarla técnicos de soporte para su resolución.

Incidencias Pendientes:



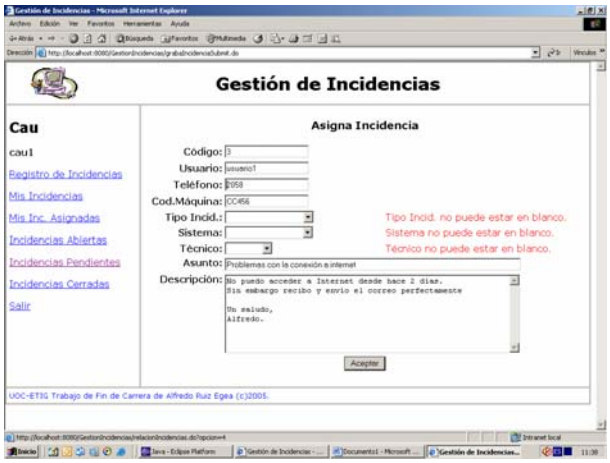
Mediante la opción "Incidencias pendientes" podrá obtener la relación de incidencias en este estado para su gestión.
"Clickando" sobre el código de incidencia podrá visualizarla.

Consulta Incidencia:

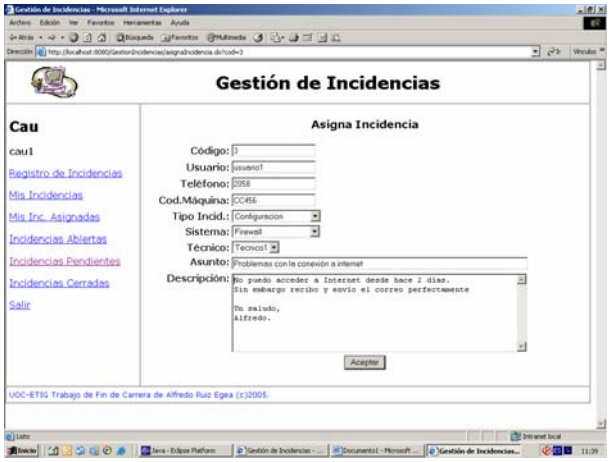


Un vez visualizada, puede modificarla para corregir cualquier error o entrar en la opción de asignación.

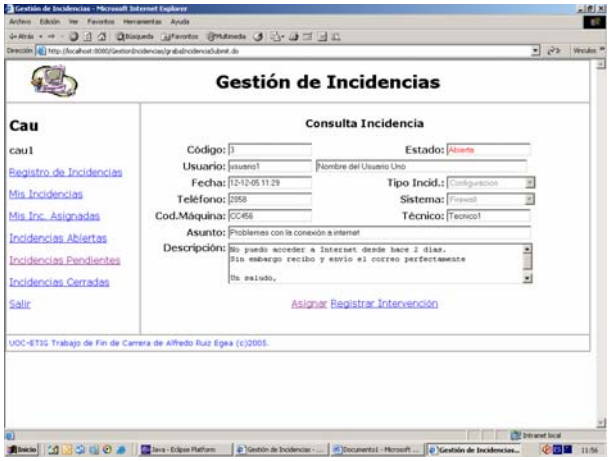
Asigna Incidencia:



La opción de asignación además de permitirle su modificación le obliga a asignar un tipo de incidencia, el sistema involucrado y el técnico principal que deberá resolverla.



Una vez introducidos estos datos deberá pulsar el botón aceptar para que sean recogidos por el sistema.



El sistema actualiza los datos y pasa la incidencia a estado de “Abierta”.
A partir de ahora cualquier técnico o cau pueden realizar intervenciones sobre esta incidencia, si bien el técnico asignado será el principal y al que le aparecerá en su relación de “Incidencias asignadas”

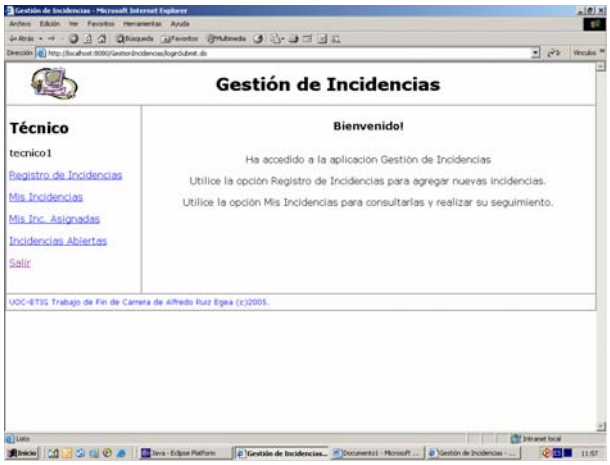
Opciones no habituales:

- Registro de incidencias: A parte de poder registrar incidencias propias como usuario, está opción en los cau permite dar de alta una incidencia en nombre de cualquier usuario dado de alta en el sistema. Esto permite a los cau registrar incidencias en nombre de usuarios que no tengan acceso al sistema y hayan contactado en el servicio de Cau mediante una llamada telefónica.

- Incidencias Abiertas: Permite obtener la relación de todas las incidencias abiertas en ese momento para poder reasignarlas a otro técnico o incluso para registrar intervenciones en caso de haber participado en su resolución (por ejemplo telefónicamente).

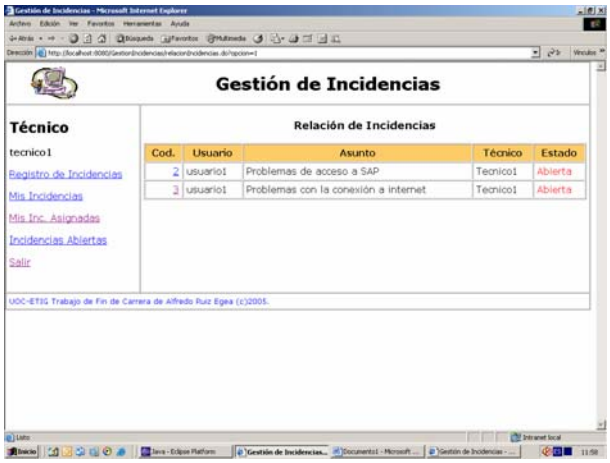
- Incidencias Cerradas: Permite obtener la relación de todas las incidencias cerradas. Esto permitirá a los Cau localizar una incidencias cerrada indebidamente y reabrirla de nuevo asignándola al mismo técnico o a otro que se considere más idoneo. El sistema incorpora una intervención automática en su nombre (y con tiempo dedicado 0) para que conste la reapertura de la incidencia.

Opciones de Técnico

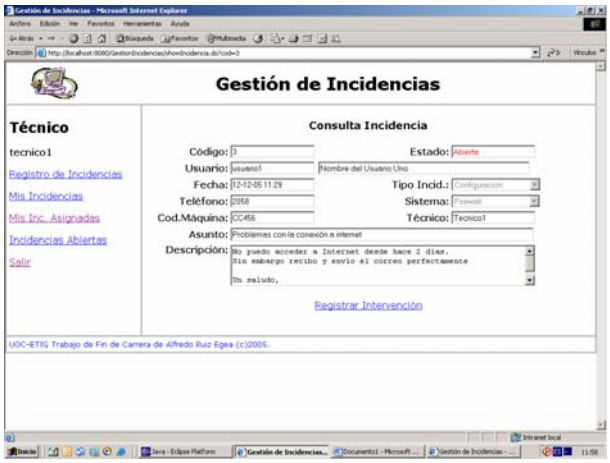


Una vez identificado un técnico en el sistema, le aparecen sus opciones permitidas.
A parte de las opciones de usuario cliente, para registrar sus propias incidencias, le aparecen dos opciones desde donde obtener las relaciones de incidencias a las que realizar intervenciones.

Mis Incidencias Asignadas:

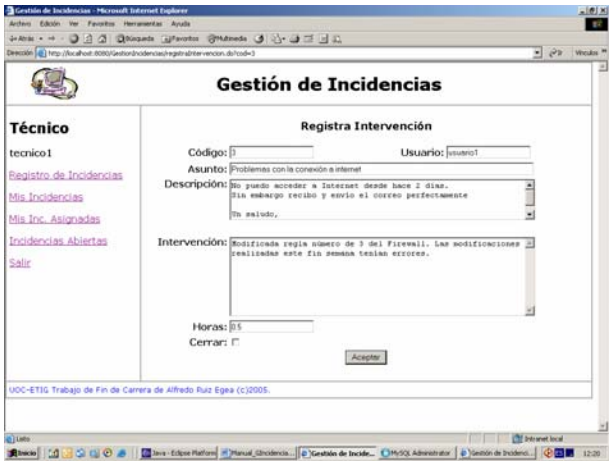


Mediante la opción “Mis Inc. Asignadas” obtiene la relación de incidencias asignadas a él directamente y donde tiene la responsabilidad de su resolución.

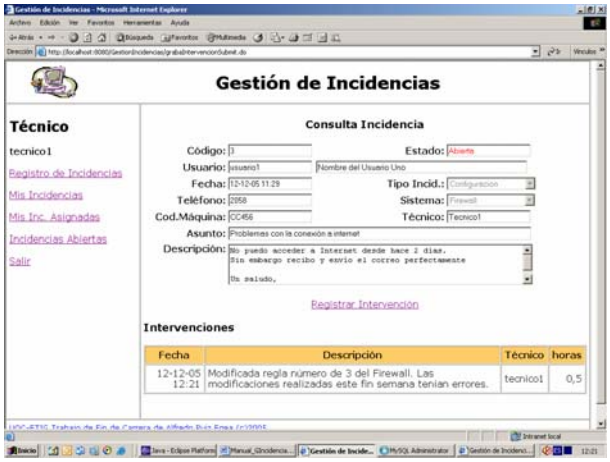


Tras pulsar sobre el código de incidencia, se visualizarán todos sus datos.
Pulsando sobre “Registrar Intervención”, podrá asignarle intervenciones a la incidencia.

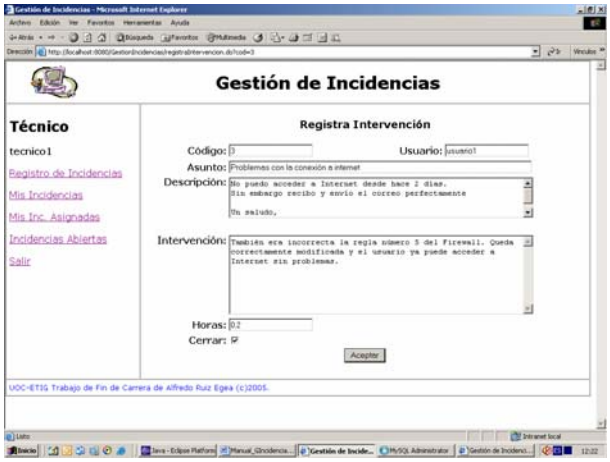
Registrar Intervención:



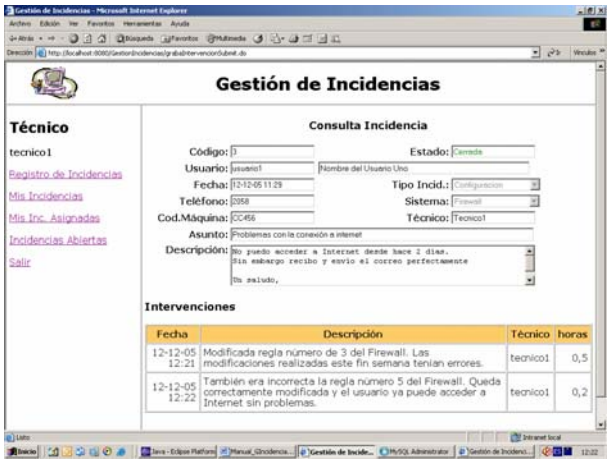
Los datos a cumplimentar para registrar una intervención son el texto de la intervención, el tiempo dedicado (en horas) y si esta intervención resuelve la incidencia y por tanto debe cerrarse.



Tras registrar la intervención se vuelve a la pantalla de consulta, donde veremos también las intervenciones realizadas.



Se pueden realizar tantas intervenciones como se precisen mientras no se marque la incidencia como cerrada.



En caso de marcar la casilla de “Cerrar”, el sistema cambia el estado de la incidencia a “Cerrado” y no permite realizar más intervenciones (a no ser que sea reabierta por un Cau).

Opciones no habituales:

- **Incidencias Abiertas**: Permite obtener la relación de todas las incidencias abiertas en ese momento para poder localizar alguna en concreto para visualizarla o incluso asignarle intervenciones y cerrarlas. Esto permite la colaboración de técnicos en la resolución de problemas y actitudes proactivas sin necesidad de que los cau reasignen incidencias a otros técnicos que las solucionan al encontrarse con ellas o al serles solicitada su colaboración. Las intervenciones quedan asignadas al técnico que la realiza y no al asignado en la incidencia.