Intranet Gestión de Incidencias con J2EE

Francisco Puertas Calvero

ETIG

Consultor: Jordi Ceballos Villach

16 de Junio de 2006

Resumen del Proyecto

El proyecto consiste en el diseño e implementación de una aplicación web utilizando la tecnología *open source* en el desarrollo, J2EE, que cuenta con una base instalada muy amplia y que está soportada por importantes empresas constructoras de software.

La aplicación que se define aquí pretende responder a la demanda de servicio que requieren los empleados de una organización por el uso de maquinaria y programario para la realización de sus tareas en la organización. La aplicación normalizar el procedimiento de solicitudes de asistencia de los usuarios, así como el soporte que el departamento les ofrece para su resolución. La aplicación estará disponible en la intranet de la empresa y se accederá a través de un navegador. La aplicación solicita la identificación del usuario para poder mostrar su menú de trabajo. Cada menú tendrá definidas unas tareas a realizar como: abrir incidencias, mantener la configuración de la aplicación y resolver las incidencias.

El administrador será el encargado de configurar el entorno de trabajo de todo el aplicativo. Este entorno está compuesto por mantenimiento de usuarios, departamentos y oficinas que componen la empresa. Una vez creado este entorno, la aplicación será accesible para los usuarios autorizados.

Cuando un empleado de la empresa tenga que notificar alguna incidencia efectuará el login y le aparecerá un menú principal con las opciones a realizar, en su caso, tendrá disponibles dos, una para introducir las incidencias y otra para consultar las ya realizadas. Para crear una incidencia, el empleado indicará un título y la descripción de la anomalía que tiene, así como otros datos que la identifiquen.

Además de las tareas de configuración del entorno, el administrador es el responsable de asignar las incidencias a los técnicos disponibles.

Los técnicos son los encargados de resolver las incidencias que llegan al departamento de soporte. Al validarse en la aplicación, los técnicos dispondrán de una opción para consultar las incidencias que tienen asignadas para su resolución. Para cada incidencia se pueden ir creando las tareas que se realizan para su resolución y documentación. Cuando la incidencia se finaliza, el aplicativo envía un correo electrónico al empleado para notificación.

El aplicativo genera los informes y estadísticas en formato Excel para que pueda tratarse más adecuadamente con las herramientas ofimáticas habituales.

La aplicación está diseñada en capas, inspirada en la arquitectura Modelo – Vista – Controlador lo que permite su portabilidad y ser multiplataforma. El aprovechamiento del modelo OO, garantiza un mantenimiento y reutilización más óptimos.

Área del TFC: Tecnología J2EE

Palabras claves: Gestión Incidencias; J2EE; Java; aplicación web

1. ÍNDICE

2. Introducción

2.1. Justificación del proyecto: punto de partida y aportación

El proyecto parte de una idea propia a partir de las experiencias vividas en el departamento de soporte a usuarios. Se pretende canalizar las peticiones de soporte de los usuarios a través de un único canal de comunicación, para de esa manera conseguir una gestión eficiente y ordenada del servicio que presta el departamento de soporte dentro de una organización.

El punto importante de esta gestión es registrar todas las incidencias que se realizan, además, que estas sean introducidas por el propio usuario que efectúa la solicitud de asistencia. Los estados y las modificaciones que se van incorporando a la incidencia desde su inicio hasta su finalización nos aportarán una base de datos de conocimiento. Con esta base de datos se podrá valorar el nivel de servicio que se presta, además, a través de los diferentes atributos que contienen las incidencias se podrá realizar una medición para poder crear modelos que nos identifiquen comportamientos o carencias en la organización, tales como: falta de formación en determinadas áreas, anomalías en el programario o maquinaria utilizada, etc.

He utilizado la tecnología J2EE para la construcción de la aplicación porque es la plataforma utilizada en el software libre, disponen de multitud de recursos y además está amparada por importantes compañías del mundo del software que le dan soporte. Otra circunstancia adicional es que desde hace unos años he usado Java como lenguaje de programación en las asignaturas que requerían programación.

La aportación de esta TFC es la posibilidad de normalizar y canalizar el tratamiento que se da a las incidencias en las organizaciones que no disponen de un procedimiento para su gestión. La aplicación tal y como está estructurada puede ir creciendo en su entorno con nuevos usuarios, departamentos, oficinas y tipo de incidencias para ir detallando o agrupando la información según las necesidades de estudio posterior que se desee obtener.

Hay definidos tres tipos de perfil de usuario: el administrador que será el encargado de mantener el entorno de la aplicación, el empleado que creará las incidencias y el técnico que será el encargado de solucionarlas.

2.2. Objetivos

El objetivo del proyecto es conocer la tecnología J2EE, sobre la que no tenía ninguna experiencia, desarrollando y aplicando el modelo de patrón MVC en una aplicación web.

Por otra parte, considero que tal y como he estructurado los servlet que hacen de controlador en la aplicación y el control de las acciones que solicitan las vistas, dan una visión muy clara del modelo de patrón MVC.

Además, se pueden reutilizar las clases siguientes: Email_inc para el envío de correos electrónicos, HojaCal para la generación de informes con formato Excel y ListaTabla para mostrar una lista por pantalla de los elementos de una tabla.

En desarrollo de la vista se ha creado una interfaz que puede servir como plantilla para futuras aplicaciones. En este apartado por falta de conocimiento y por no considerar oportuno gastar recursos en la búsqueda de productos específicos de gestión de menús, se ha optado por una gestión muy sencilla. Se considerará como una mejora para futuras adaptaciones.

2.3. Enfoque y método seguido

El enfoque y el método seguido se basan en el sistema estándar clásico (o en cascada) del ciclo de vida, es decir siguiendo las siguientes fases: estudio de oportunidad, análisis, diseño, implementación y pruebas.

Este proyecto de TFC se diferencia de un proyecto normal, que pudiera solicitarse por cualquier otra persona o empresa, en el hecho de que al estar desarrollado por uno mismo no se produce lo que habitualmente se conoce como las indefiniciones y los cambios de requerimientos una vez finalizado el proyecto.

El modelo mental desarrollado por un técnico de informática permite que un proyecto solicitado por el mismo, en su fase del estudio de oportunidad y de su análisis los requerimientos estén bastante bien definidos, no quedando partes del proyecto indefinidas.

Por otra parte, el enfoque de la aplicación busca obtener una herramienta que permita una interacción intuitiva y autoexplicativa persiguiendo en todo momento la claridad de los objetivos, su visibilidad, su consistencia, su flexibilidad y su eficacia en el uso.

2.4. Planificación del proyecto

La planificación está definida por el calendario de las PAC's y se ha mantenido durante el semestre. Dentro de la temporalización se definen las tareas para la consecución de los hitos.

Entrega del Plan de trabajo (PAC 1): 13 de marzo
En este documento se presenta una descripción general del proyecto y su planificación.

Entrega del Documento de Análisis y primera versión del prototipo: 3 de abril
En este documento se entregan las especificaciones de las funcionalidades, mediante casos de uso junto con un primer prototipo de la interficie de la aplicación.

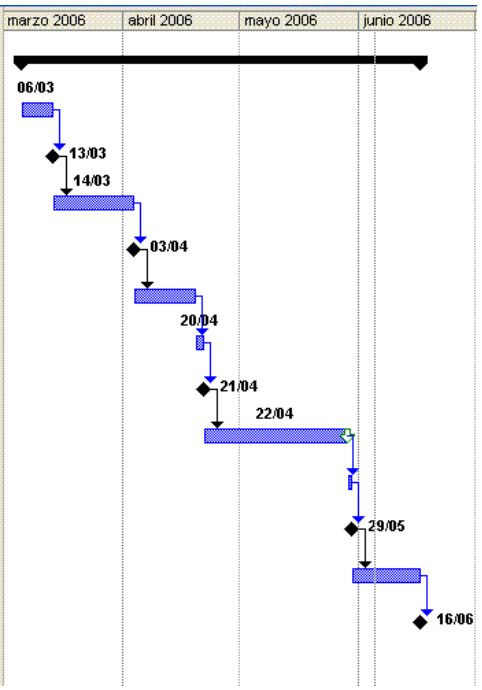
Entrega del Documento de Diseño y prototipo definitivo (PAC 2): 21 de abril
Se especifican cómo se implementará las funcionalidades descritas en la fase anterior. Se adjuntan diagramas de clases y el diseño de la base de datos.

Entrega de la Implementación (PAC 3): 29 de mayo
Se entrega la aplicación implementada, aunque no sea la versión definitiva y pueda someterse a alguna rectificación.

Entrega de la Memoria y la Presentación en Powerpoint y la implementación definitiva: 16 de junio
Se entregan los documentos definitivos del proyecto, así como la ejecución en real de la aplicación.

El diagrama de GANTT correspondiente a las entregas y su planificación es el siguiente:

	Nombre de tarea	Duración	Comienzo	Fin
1	**TFC**	**103 días**	**lun 06/03/06**	**vie 16/06/06**
2	Plan de Trabajo	8 días	lun 06/03/06	lun 13/03/06
3	Entrega Plan de Trabajo	0 días	lun 13/03/06	lun 13/03/06
4	Análisis	21 días	mar 14/03/06	lun 03/04/06
5	Entrega del documento de Análisis	0 días	lun 03/04/06	lun 03/04/06
6	Diseño y prototipo definitivo	16 días	mar 04/04/06	mié 19/04/06
7	Prototipo definitivo	2 días	jue 20/04/06	vie 21/04/06
8	Entrega Diseño y Prototipo definitivo	0 días	vie 21/04/06	vie 21/04/06
9	Implementación	37 días	sáb 22/04/06	dom 28/05/06
10	Manual instalación	1 día	lun 29/05/06	lun 29/05/06
11	Entrega Implementación y Manual Instalación	0 días	lun 29/05/06	lun 29/05/06
12	Memoria y presentación powerpoint	18 días	mar 30/05/06	vie 16/06/06
13	Entrega Memoria y Presentación powerpoint	0 días	vie 16/06/06	vie 16/06/06



2.5. Productos obtenidos

Durante el desarrollo del proyecto se han obtenido los siguientes productos, ya comentados en el apartado de la planificación:

- Plan de Trabajo: breve descripción del proyecto y planificación temporal.
- Análisis: conjunto de los requerimientos y especificaciones de las funcionalidades mediante los casos de uso.
- Prototipo: diseño de la interfaz con ejemplos de las funcionalidades en HTML estático.
- Diseño: diagrama de clases y diagrama de la base de datos.
- Implementación: la aplicación operativa.
- Manual de instalación: guía de la instalación y de los componentes necesarios para instalar.
- Memoria del proyecto: el contenido de este documento.
- Presentación del proyecto: documento que complementa a la memoria destacando los puntos más importantes, diseñado para ser presentado en público.

2.6. Descripción de los capítulos siguientes

En los siguientes capítulos se explicará la arquitectura de la aplicación, con las consideraciones a tener en cuenta en referencia a la arquitectura Modelo – Vista – Controlador, el diseño de la base de datos, el diseño de las clases y la estructura de una página modelo con las características de la interfaz gráfica usada en la aplicación.

A continuación se comentan las características de seguridad que componen el aplicativo relativo a su autentificación y a los tres tipos de perfiles de usuario que soporta.

Cómo arquitectura de la aplicación, también se relacionan todos los componentes de terceros que han sido utilizados o adaptados en la aplicación.

Además, se presenta una descripción de la aplicación con sus funcionalidades para a continuación efectuar una valoración económica junto con las conclusiones del proyecto una vez cubierto todo el ciclo de vida de una aplicación.

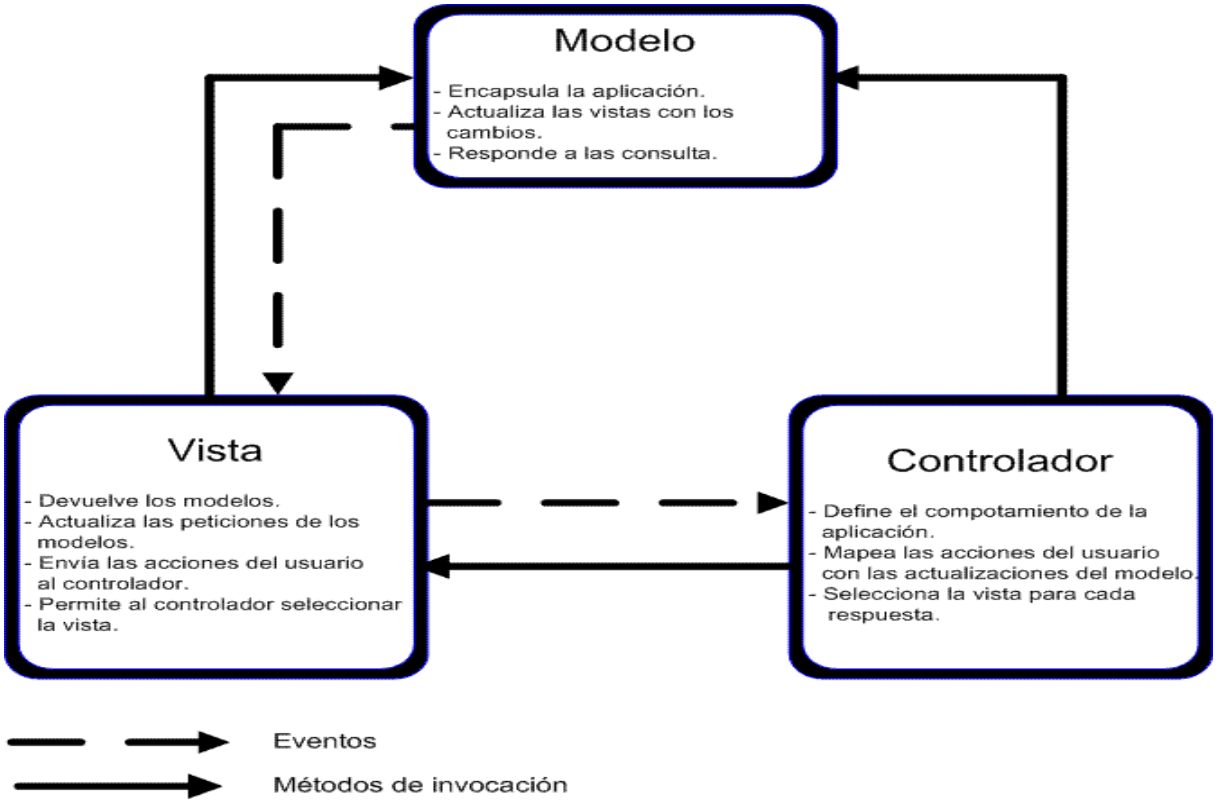
Finalmente, se incluyen unas posibles mejoras para desarrollos futuros, un glosario, la bibliografía y las referencias de Internet utilizadas.

3. Arquitectura de la aplicación

3.1. Aproximación a MVC

La arquitectura usada es el patrón MVC (Modelo – Vista – Controlador) que separa los datos de la aplicación, el interfaz del usuario y la lógica de control de los tres componentes.

El **Modelo** representa el dominio específico sobre el cual funciona la aplicación, el modelo es el que añade significado a los datos.
La **Vista** representa el formato adecuado para que el usuario interaccione con la aplicación.
El **Controlador** es el que responde a los eventos que usualmente realiza el usuario a través de acciones de su interfaz gráfico y que además pueden invocar cambios en el modelo.



La arquitectura resultante de aplicar el patrón MVC es la siguiente:

- **Vista:** paginas jsp donde se encuentra la interfaz del usuario y donde se capturan los datos entrados por el usuario y las acciones realizadas.

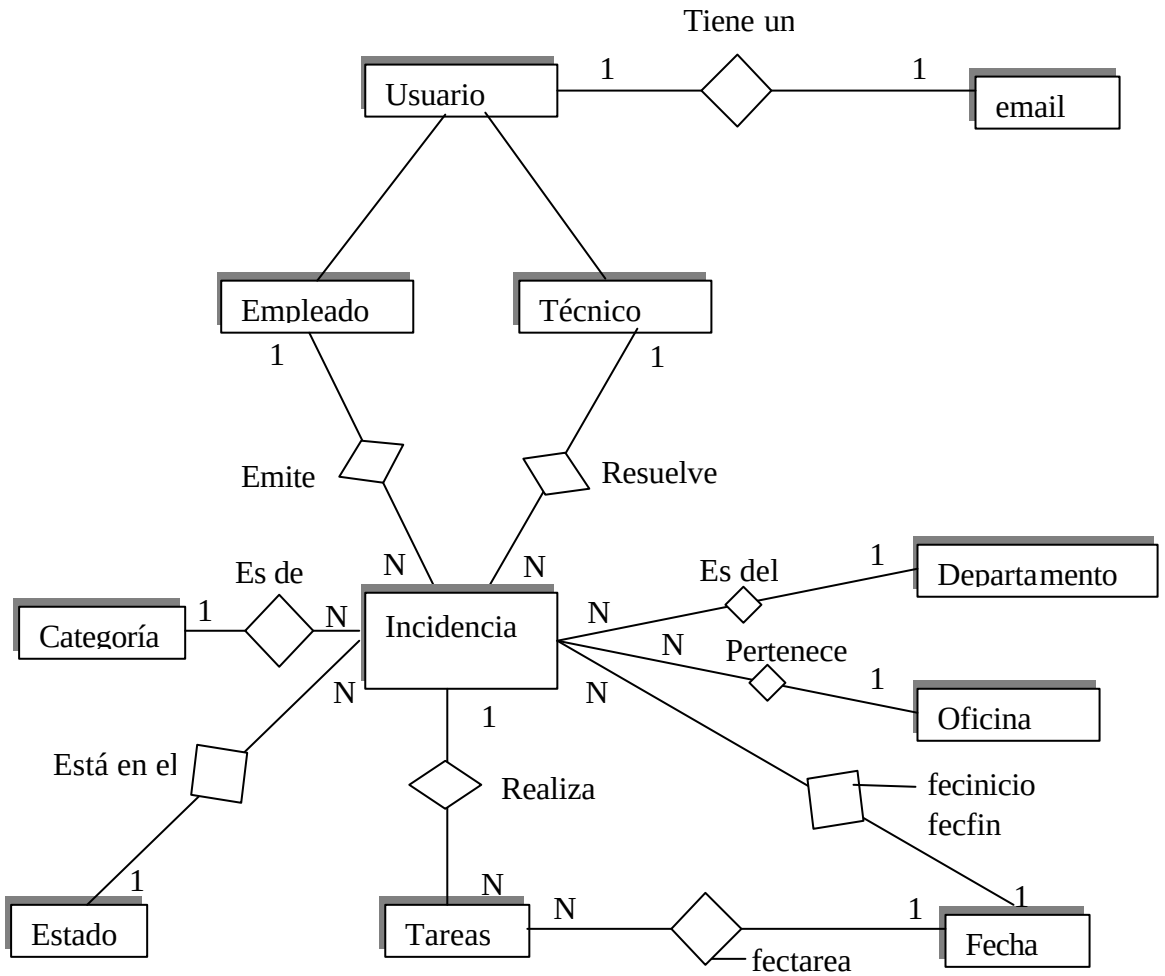
- **Controlador:** clases de tipo servlet que interaccionan con el modelo de datos y la interfaz del usuario, estas clases determinan el flujo de la aplicación.

- **Modelo:** clases que representan el modelo de datos y la lógica del dominio. Se componen de las clases de entidad, las gestoras de las clases de entidad y las que interactúan con el SGBD.

3.2. Diseño de la base de datos

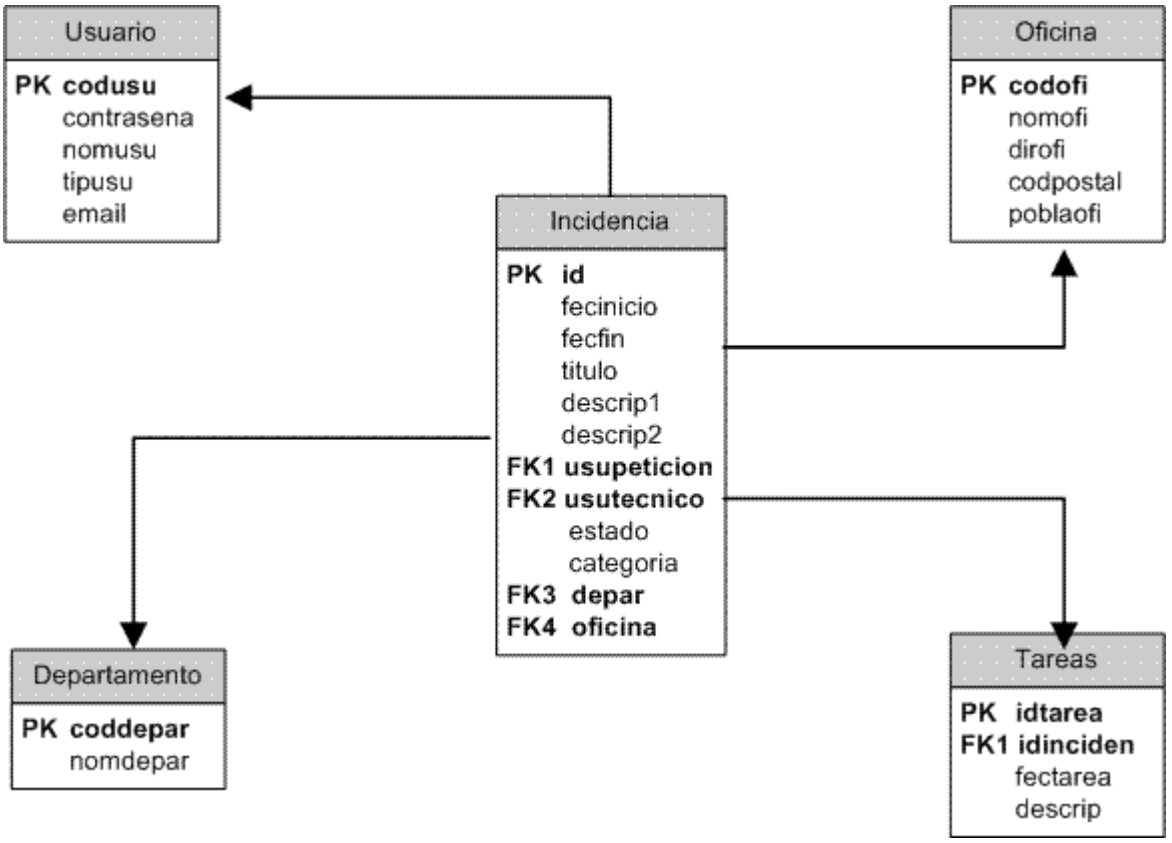
A partir del análisis anterior y lo que hemos ido definiendo podemos especificar el diseño conceptual de la base de datos, conocido como diagrama ER (Entidad – relación).

3.2.1. Diagrama ER



A partir de este diagrama conceptual podemos definir el diseño lógico efectuando las transformaciones a partir de las entidades y las relaciones aquí indicadas.

3.2.2. Diseño lógico de la base de datos

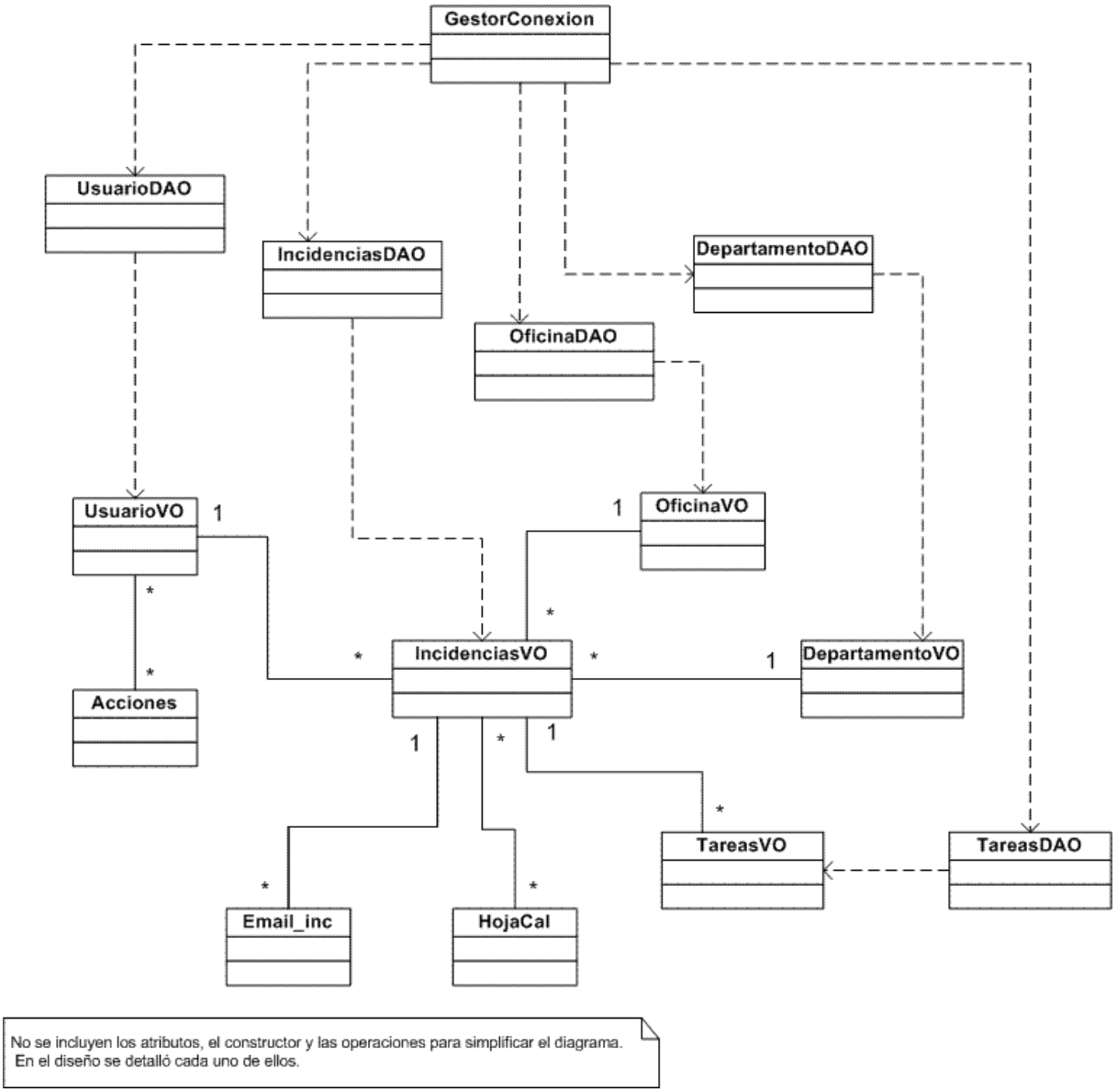


Donde PK son las claves primarias de las tablas y FK las claves foráneas que apuntan a otras tablas.

3.3. Diseño de clases

3.3.1. Clases del modelo

Las clases del Modelo son todas las que intervienen en los datos y en el tipo de problema que soluciona. El diagrama de clases es el siguiente:



Dentro de este diagrama se distinguen dos grupos de clases:

- Las clases que corresponden a entidades del problema.
- Las clases gestoras que corresponden a las entidades que requieren operaciones especiales a la SGBD como: selección, inserción, actualización y borrado. Estas clases se llaman xxxxDAO, siendo xxxx la clase de la entidad a la cual da soporte.

A continuación se explican y detallan las clases que intervienen en el Modelo.

3.3.1.1. GestorConexion

GestorConexión
+getConexion(): Connection

Esta clase se encarga de crear los pools de conexión con el SGBD.

3.3.1.2. UsuarioDAO

UsuarioDAO
-conn :Connection
+findByPrimaryKey(usu :string): usuarioVO +getAll(smt: string): ArrayList +insert(usuario: usuarioVO): void +delete(usuario: usuarioVO): void +update(usuario: usuarioVO): void +getConn(): Connection +setConn(): void +closeConn(): void

Esta clase se encarga de interaccionar con la tabla Usuario del SGBD, por lo tanto, contiene los métodos para realizar las consultas, inserciones, modificaciones y borrado de los datos.

3.3.1.3. DepartamentoDAO

DepartamentoDAO
-conn :Connection
+findByPrimaryKey(depar :string): departamentoVO +getAll(smt: string): ArrayList +insert(depar: departamentoVO): void +delete(depar: departamentoVO): void +update(depar: departamentoVO): void +getConn(): Connection +setConn(): void +closeConn(): void

Esta clase se encarga de interaccionar con la tabla Departamento del SGBD, por lo tanto, contiene los métodos para realizar las consultas, inserciones, modificaciones y borrado de los datos.

3.3.1.4. OficinaDAO

OficinaDAO
-conn :Connection
+findByPrimaryKey(ofi :string): oficinaVO +getAll(smt: string): ArrayList +insert(ofi: oficinaVO): void +delete(ofi: oficinaVO): void +update(ofi: oficinaVO): void +getConn(): Connection +setConn(): void +closeConn(): void

Esta clase se encarga de interaccionar con la tabla Oficina del SGBD, por lo tanto, contiene los métodos para realizar las consultas, inserciones, modificaciones y borrado de los datos.

3.3.1.5. IncidenciaDAO

IncidenciaDAO
-conn :Connection
+findByPrimaryKey(inc :string): incidenciaVO +getAll(smt: string): ArrayList +insert(inc: incidenciaVO): void +delete(inc: incidenciaVO): void +update(inc: incidenciaVO): void +getConn(): Connection +setConn(): void +closeConn(): void

Esta clase se encarga de interaccionar con la tabla Incidencia del SGBD, por lo tanto, contiene los métodos para realizar las consultas, inserciones, modificaciones y borrado de los datos.

3.3.1.6. TareasDAO

TareasDAO
-conn :Connection
+findByPrimaryKey(tarea :string): tareasVO +getAll(smt: string): ArrayList +insert(tarea: tareasVO): void +delete(tarea: tareasVO): void +update(tarea: tareasVO): void +getConn(): Connection +setConn(): void +closeConn(): void

Esta clase se encarga de interaccionar con la tabla Tareas del SGBD, por lo tanto, contiene los métodos para realizar las consultas, inserciones, modificaciones y borrado de los datos.

3.3.1.7. UsuarioVO

UsuarioVO
-codusu :String -contrasena: String -nomusu: String -tipusu: String -emailusu: String
+getCodusu(): string +getContrasena(): string +getNomusu(): string +getTpusu(): string +getEmail(): string +setCodusu(codusu: string): void +setContrasena(contrasena: string): void +setNomusu(nomusu: string): void +setTipusu(tipusu: string): void +setEmail(email: string): void

El UsuarioVO representa a un usuario del sistema con sus datos personales y de autentificación. Según el tipo de usuario se determinará a qué tipo de operaciones puede acceder del aplicativo.

3.3.1.8. DepartamentoVO

DepartamentoVO
-coddepar :String -nomdepar: String
+getCoddepar(): string +getNomdepar(): string +setCoddepar(coddepar: string): void +setNomdepar(nomdepar: string): void

El DepartamentoVO representa a los departamentos que dispone la organización en todas las sedes que dispone.

3.3.1.9. OficinaVO

OficinaVO
-codofi :String -nomofi: String -dirofi: String -codpostal: String -poblaofi: String
+getCodofi(): string +getNomofi(): string +getDirofi(): string +getCodpostal(): string +getDirofi(): string +setCodofi(codofi: string): void +setNomofi(nomofi: string): void +setDirofi(dirofi: string): void +setCodpostal(codpostal: string): void +setPoblaofi(poblaofi: string): void

La OficinaVO representa a las oficinas que dispone la organización.

3.3.1.10. IncidenciaVO

IncidenciasVO
-id :int -fecinicio: date -fecfin: date -titulo: String -descrip1: String -descrip2: String -usupeticion: String -usutecnico: String -estado: String -categoria: String -depar: String -oficina: String
+getId(): int +getFecinicio(): date +getFecfin(): date +getTitulo(): string +getDescrip1(): string +getDescrip2(): string +getUsupeticion(): string +getUsutecnico(): string +getEstado(): string +getCategoria(): string +getDepar(): string +getOficina(): string

+setId(id: int): void +setFecinicio(feciniico: date): void +setFecfin(fecfin: date): void +setTitulo(titulo: string): void +setDescrip1(descrip1: string): void +setDescrip2(descrip2: string): void +setUsupeticion(usupeticion: string): void +setUsutecnico(usutecnico: string): void +setEstado(estado: string): void +setCategoria(categoria: string): void +setDepar(titulo: string): void +setOficina(oficina: string): void

La IncidenciaVO representa a las incidencias realizadas por el empleado y resultas por el departamento de soporte.

3.3.1.11. TareasVO

TareasVO
-idtarea :int -idinciden :int -fectarea :date -descrip: String
+getIdtarea(): int +getIdinciden(): int +getFectarea(): date +getDescrip(): string +setIdtarea(idtarea: int): void +setIdinciden(idinciden: int): void +setFectarea(fectarea: date): void +setDescrip(descrip: string): void

La TareasVO representa a la tarea realizada para solucionar una incidencia creada.

3.3.1.12. Acciones

Acciones
-menu :String -stack: String
+getMenu(): string +getStack(): string +setMenu(menu: string): void +setStackl(stack: string): void

Las Acciones representan el control del flujo de las acciones a realizar por parte de la clase controlador a partir de las acciones solicitadas desde la vista.

3.3.1.13. Email_inc

Email_inc
-destino : string -origen: string -asunto: string -texto: string
+setDestino(destino: string):void +setOrigen(origin: string):void +setAsunto(asunto: string):void +setTexto(texto: string):void +enviarMail():void

La Email_inc representa al email que se va a realizar al empleado para notificarle el cambio de estado de su incidencia.

3.3.1.14. HojaCal

HojaCal
+crearEx(lista: ArrayList): HSSFWorkbook

La HojaCal genera la hoja de cálculo a partir de los criterios indicados en la selección de informes y estadísticas.

3.3.2. Clases Controlador y Vista

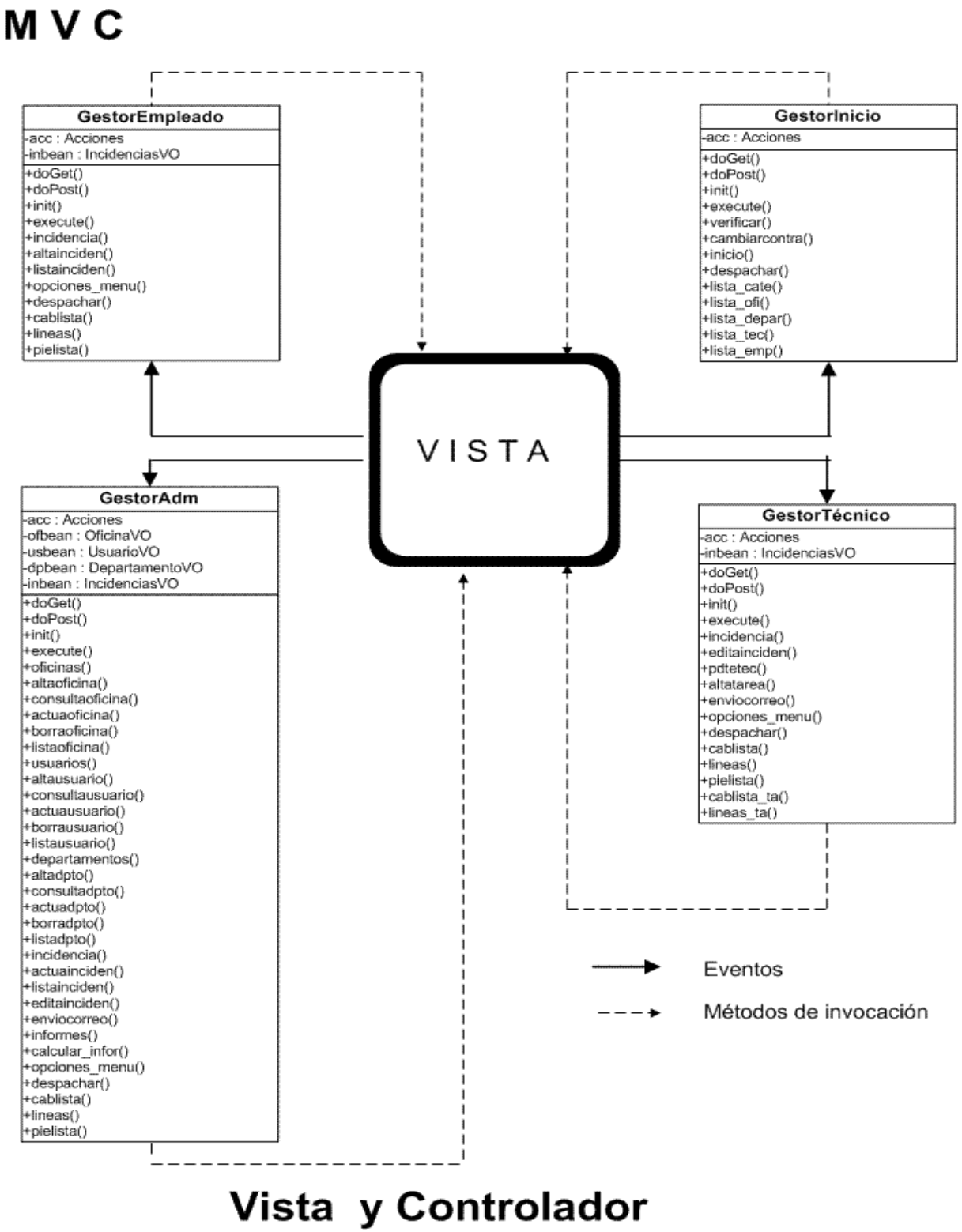
La arquitectura MVC (Modelo – Vista – Controlador) separa la interfaz del usuario (Vista) de la gestión de la lógica del modelo de datos. Para efectuar esta separación es necesario crear los controladores que gestionan la interacción entre la vista y el modelo de datos. A partir del prototipo diseñado se puede dar una relación de las clases que se encargan del control del flujo entre la interfaz del usuario y el modelo de datos, estas clases son:

Clase controlador	Vista (.jsp)	Descripción
GestorEmpleado		
	emp_menu	Pantalla menú para el empleado.
	inc_alta	Pantalla para dar de alta una incidencia.
	inc_selemp	Pantalla para seleccionar una consulta.
	inc_listaemp	Pantalla con la lista de incidencias realizadas.
GestorTecnico		
	tec_menu	Pantalla menú para el técnico.
	inc_pdtetec	Pantalla con la lista de incidencias pendientes

		de un técnico.
	inc_edicion	Pantalla que muestra la incidencia para su edición.
	inc_contarea	Pantalla con la lista de tareas realizadas en una incidencia.
	inc_altatarea	Pantalla para dar de alta una tarea.
GestorAdm		
	adm_menu	Pantalla menú para el administrador.
	usu_alta	Pantalla para dar de alta un usuario.
	usu_consel	Pantalla para seleccionar una consulta de usuarios.
	usu_convis	Pantalla con los datos de una consulta de un usuario.
	usu_edicion	Pantalla para editar un usuario.
	usu_lista	Pantalla con la lista de usuarios.
	dpto_alta	Pantalla para dar de alta un departamento.
	dpto_consel	Pantalla para seleccionar una consulta de departamentos.
	dpto_convis	Pantalla con los datos de la consulta de un departamento.
	dpto_edicion	Pantalla para editar un departamento.
	dpto_lista	Pantalla con la lista de departamentos.
	ofi_alta	Pantalla para dar de alta una oficina.
	ofi_consel	Pantalla para seleccionar una consulta de una oficina.
	ofi_convis	Pantalla con los datos de la consulta de una oficina.
	ofi_edicion	Pantalla para editar una oficina.
	ofi_lista	Pantalla con la lista de oficinas.
	inc_pdte	Pantalla con la lista de incidencias pendientes de asignar a un técnico.
	inc_asig	Pantalla con los datos de la incidencia para efectuar la asignación a un técnico.
	inf_sel	Pantalla para la selección de los informes de la aplicación que nos indicarán los niveles de servicio.
GestorInicio		
	inicio	Pantalla para efectuar el login en la aplicación.
	camb_contra	Pantalla para cambiar la contraseña del usuario.

3.3.2.1. Diagrama MVC

El diagrama del flujo entre las clases controladoras y las vistas de la arquitectura MVC es la siguiente:



El controlador es el encargado de procesar las acciones solicitadas desde la vista. Cada perfil de usuario es gestionado por un controlador que dispensa nuevas vistas a partir de los eventos que procesa o que son solicitados por acciones de otras vistas.

El controlador para determinar los procesos a realizar identifica las siguientes acciones y valores de la vista:

- Identifica el nombre de la vista actual o formulario
- Acciones realizadas en el menú
- Acciones realizadas en los botones del formulario

El controlador a partir de estos datos determina cual es el flujo de proceso a realizar y cual será la vista siguiente a dispensar si fuera necesario.

Desde la vista solamente se realizan las siguientes funciones o controles:

- La validación de los campos obligatorios a través de javascript

- La carga de los elementos que componen los combos de los formularios a través scriptlets

El ciclo de vida del controlador se inicia cuando un formulario es sometido con acciones, a partir de ese instante el controlador las procesa. Se finaliza una vez efectuado ese proceso y dispensa otra nueva vista, si así lo requiere el proceso iniciado.

Los métodos de las clases controladores según el tipo de usuario son los siguientes:

Para la clase controladora que gestiona las acciones del perfil administrador, sus métodos son:

GestorAdm
-acc :Acciones -ofbean: OficinaVO -dpbean: DepartamentoVO -ofbean: OficinaVO -usbean: UsuarioVO -inbean: IncidenciaVO
+doGet(req: HttpServletRequest, resp: HttpServletResponse resp): void +doPost(req: HttpServletRequest, resp: HttpServletResponse resp): void +init(): void +execute(req: HttpServletRequest, resp: HttpServletResponse resp): void +oficinas(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +altaoficina(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +consultaoficina(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void

+actuaoficina(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+borraoficina(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+listaoficina(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+usuarios(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+altausuarios(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+consultausuarios(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+actuausuarios(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+borrausuarios(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+listausuarios(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+departamentos(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+altadpto(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+actuadpto(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+consultadpto(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+borradpto(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+listadpto(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+incidencia(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+actuainciden(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+listainciden(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+editainciden(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+enviocorreo(in: IncidenciaVO): void
+informes(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+calcular_infor(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+opciones_menu(req: HttpServletRequest, resp: HttpServletResponse resp,
session: HttpSession): void
+despachar(req: HttpServletRequest, resp: HttpServletResponse resp, session:
HttpSession): void
+cablista(): string
+lineas(id: integer, fec: string, tit: string, cat: string): string
+pielista(): string

Para la clase controladora que gestiona las acciones del perfil empleado, sus métodos son:

GestorEmpleado
-acc :Acciones -inbean: IncidenciaVO
+doGet(req: HttpServletRequest, resp: HttpServletResponse resp): void +doPost(req: HttpServletRequest, resp: HttpServletResponse resp): void +init(): void +execute(req: HttpServletRequest, resp: HttpServletResponse resp): void +incidencia(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +altainciden(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +listainciden(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +opciones_menu(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +despachar(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +cablista(): string +lineas(id: integer, fec: string, tit: string, cat: string): string +pielista(): string

Para la clase controladora que gestiona las acciones del perfil técnico, sus métodos son:

GestorTecnico
-acc :Acciones -inbean: IncidenciaVO
+doGet(req: HttpServletRequest, resp: HttpServletResponse resp): void +doPost(req: HttpServletRequest, resp: HttpServletResponse resp): void +init(): void +execute(req: HttpServletRequest, resp: HttpServletResponse resp): void +incidencia(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +editainciden(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +pdtetec(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +altatarea(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +consultatarea(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +enviocorreo(in: IncidenciaVO): void +opciones_menu(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void

+despachar(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void
+cablista(): string
+lineas(id: integer, fec: string, tit: string, cat: string): string
+pielista(): string
+cablista_ta(): string
+lineas_ta(fec: string, descr: string): string

La clase controladora que gestiona el login, identifica el tipo de usuario y carga los elementos de las tablas de la base de datos para utilizarlos en los componentes combos de las pantallas es la siguiente:

GestorInicio
-acc :Acciones +doGet(req: HttpServletRequest, resp: HttpServletResponse resp): void +doPost(req: HttpServletRequest, resp: HttpServletResponse resp): void +init(): void +execute(req: HttpServletRequest, resp: HttpServletResponse resp): void +verificar(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +cambiarcontra(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession, usu: UsuarioVO): void +inicio(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void +lista_cate(session: HttpSession): void +lista_ofi(session: HttpSession): void +lista_depar(session: HttpSession): void +lista_tec(session: HttpSession): void +lista_emp(session: HttpSession): void +despachar(req: HttpServletRequest, resp: HttpServletResponse resp, session: HttpSession): void

3.4. Estructura de la interfaz: diseño y CSS

La interfaz queda identificada de la siguiente forma:

- **En la claridad de objetivos**
  Los objetivos quedan definidos en tres bloques: el frontal superior donde aparece el nombre, el bloque vertical izquierdo donde aparecen las opciones de menú a realizar por el usuario y la central donde aparecerán los contenidos que se hayan solicitado desde el menú o bloque izquierdo.

- **En su visibilidad**
  La interfaz se identifica claramente por sus tres bloques indicados anteriormente, bloque de identificación, bloque de menús o acciones y bloque de contenidos.

- **En su adecuación a los usuarios**

Los enlaces son textuales, los colores y tamaños de las letras son uniformes y comunes en todo el conjunto de la interfaz.

- **En la consistencia y estándares**

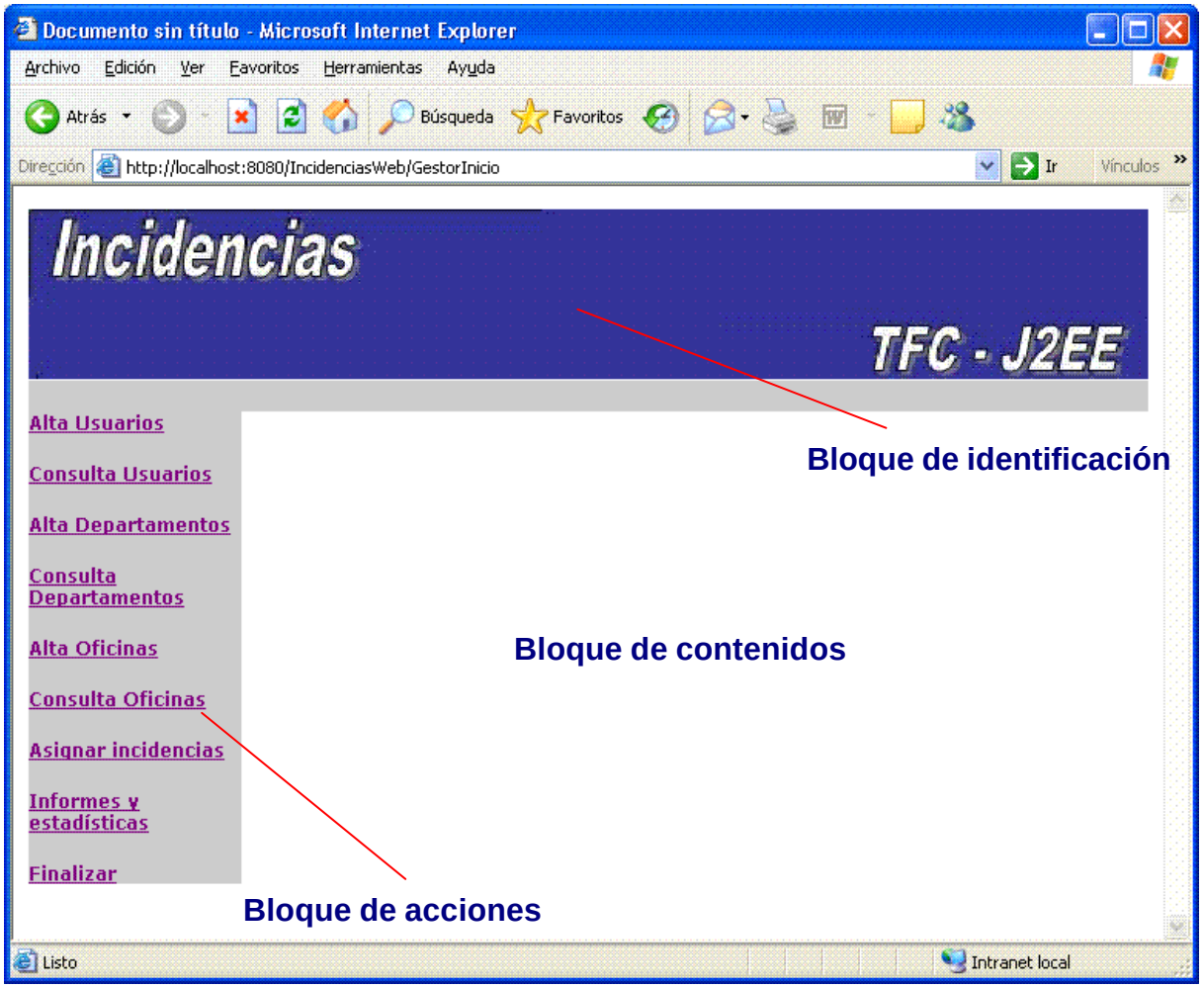
Los vínculos hipertextuales se usan solamente en el bloque vertical izquierdo, se utiliza una única metáfora en el bloque de contenidos identificada por una lupa para efectuar las búsquedas.

- **En su flexibilidad y eficacia de uso**

El usuario puede acceder a todas las acciones y contenidos de manera sencilla y sin repetir pasos.

- **En el diseño minimalista**

No existe ninguna página que cargue imágenes u otros contenidos que dificulten el acceso rápido a la aplicación.



Otra de las características de la interfaz es la facilidad del cambio de formatos y posición de la mayoría de los elementos y componentes de la aplicación, gracias al uso de un fichero de estilos CSS. Un simple cambio en este fichero se hace efectivo en toda la aplicación asegurando de esa manera una aplicación homogénea.

3.5. Seguridad

La aplicación se ha diseñado para ser ejecutada dentro de la intranet de la empresa, por lo tanto, no se han incorporado controles específicos de acceso desde Internet, la seguridad será la propia que incorpore el servidor sobre el que vaya a colocarse la aplicación y que esté diseñado para trabajar en una intranet.

3.5.1. Autentificación

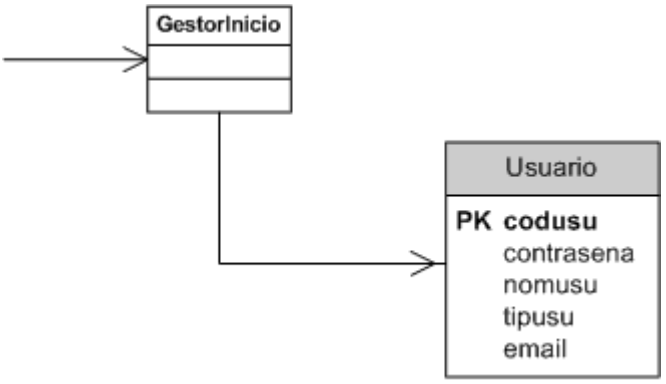
La autentificación de la aplicación se lleva a cabo a través de una página de inicio que pide el usuario y la contraseña. Ambos campos tienen una longitud de 10 dígitos. Para el usuario solamente se permiten letras mayúsculas y números, para la contraseña se permite la combinación de mayúsculas, minúsculas y números.

El usuario y la contraseña son creadas la primera vez por el administrador, pero su modificación la debe realizar el propio usuario a través de una página de inicio específica para esa tarea. La contraseña nunca se muestra por pantalla.

3.5.2. Perfiles de usuario

La aplicación dispone de tres perfiles de usuario: administrador, empleado y técnico.

La arquitectura de los perfiles está basada en la tabla Usuario y el controlador *GestorInicio*.



A través de la página de login, el *GestorInicio* accede a la tabla Usuario para verificar la autenticidad del usuario y de su contraseña, una vez verificada la autenticidad, se comprueba a que tipo de perfil pertenece comprobando el atributo *tipusu*. En función del tipo de atributo se muestra un menú específico para cada uno de los tres perfiles disponibles en la aplicación.

Cada perfil es gestionado por un controlador independiente: *GestorAdm* para el perfil de administrador, *GestorEmpleado* para el perfil de empleado y *GestorTecnico* para el perfil del técnico.

Las acciones de menú de un perfil determinado están accesibles en todo momento desde cualquiera de las páginas de la aplicación.

Las acciones por perfil son:

Clase controlador	Descripción
Administrador	
	Alta de usuarios.
	Consulta y búsqueda de usuarios.
	Edición de usuarios.
	Supresión de usuarios.
	Alta de departamentos.
	Consulta y búsqueda de departamentos.
	Edición de departamentos.
	Supresión de departamentos.
	Alta de oficinas.
	Consulta y búsqueda de oficinas.
	Edición de oficinas.
	Supresión de oficinas.
	Asignación de incidencias a técnicos.
	Selección de informes.
Empleado	
	Creación de incidencias
	Consulta de incidencias creadas
Técnico	
	Edición de incidencias
	Creación de tareas
	Consulta de tareas
	Finalización de tareas

3.6. Requerimientos técnicos

El programario utilizado será todo libre (open source).

- **Lenguaje de desarrollo:** Java y J2EE. La programación bajo J2EE nos permite trabajar en multicapa. Cada una de ellas compuesta por: páginas HTML potenciadas con JSP, servlet y bean que se encargan de la lógica y los cálculos del negocio.

- **Entorno de desarrollo:** Eclipse 3.1.0 y Lomboz 3.1.2

- **Servidor web:** Apache – Tomcat 5.5

- **Sistema Gestor de Base de Datos:** MySQL 5.0

Además, hará falta un servidor de correo SMTP. La aplicación hace servir el propio servidor de la UOC para realizar las pruebas, por lo que el emisor y receptor han de estar en el mismo dominio *uoc.edu*.

En cuanto a la maquinaria, será necesario un servidor para la puesta en marcha del proyecto. En la parte cliente, como requisito mínimo sólo será necesario un navegador actualizado. La aplicación se ha probado con Internet Explorer 6.0 SP2 y Opera 8.51.

3.7. Componentes externos

Con el fin de obtener una mejor productividad, se ha intentado aprovechar los recursos gratuitos disponibles en Internet para la creación de ciertas funciones.

A pesar de eso, algunos componentes no están debidamente documentados y codificados o no terminan de ajustarse a nuestros requerimientos. Esto ha obligado a efectuar modificaciones para adaptarlo a nuestras especificaciones.

A continuación se comentan los dos componentes de terceros utilizados:

3.7.1. Commons email

Proporciona una API para enviar correos electrónicos. Está construido sobre la API principal de Java Mail API de Sun.

Este componente nos permite crear email con el texto básico en el cuerpo del correo, componer uno o varios ficheros adjuntos en el correo, hacer referencia a una URL de ficheros que no se tienen localmente y que son descargados en el momento del envío del mensaje y crear mensajes en formato HTML.

El servidor utilizado para realizar las pruebas ha sido el servidor de correo de la UOC(smtp.uoc.edu) por lo que tanto el emisor como el receptor han de estar en el dominio de la UOC. Se ha utilizado una variable de contexto (web.xml) para indicar este valor, por lo tanto, cuando se realice el despliegue podrá indicarse cualquier otro servidor de correo.

El contenido de la variable es:

```
<context-param>
      <param-name>servidor_smtp</param-name>
      <param-value>smtp.uoc.edu</param-value>
</context-param>
```

3.7.2. POI - HSSF

POI es un proyecto que consiste en un conjunto de APIs para manipular diferentes tipos de formatos de ficheros basados en formatos de Documento Compuesto OLE 2 de Microsoft, utilizando java puro. Dentro de este proyecto se usa HSSF cuando se desea leer o escribir un fichero Excel (xls) utilizando Java.

HSSF provee a la API para leer, modificar y crear hojas de un fichero xls. Además, permite formatear y diseñar las hojas con los objetos *stylesheet* que incorpora la API.

4. Descripción de la aplicación de Incidencias

La aplicación pretende canalizar las peticiones de servicios que realizan los empleados de una organización al departamento encargado de dar soporte. Las peticiones pueden ser sobre sus herramientas de maquinaria como las de programario que tienen disponibles para sus tareas diarias.

Dada la arquitectura de los perfiles ya definidos anteriormente la aplicación no se estructura en módulo sino en funcionalidades propias de cada perfil.

A continuación se verán las funcionalidades principales de la aplicación, identificando el perfil que se dispone para el acceso a esa funcionalidad.

4.1. Funcionalidades

Las funcionalidades, tal y como se ha indicado anteriormente, están determinadas por el perfil de usuario, por lo tanto, éstas son: las del administrador, las del empleado, las del técnico y otra inicial de login que se realiza para entrar en la aplicación.

4.1.1. Login

Se trata de la pantalla de inicio a la aplicación. Todos los usuarios que accedan a la aplicación deben estar registrados. A través del usuario y contraseña se verifica si existe en la base de datos (BD), si es así, se comprueba el tipo de usuario y se le asigna el perfil de trabajo sobre el que va a poder operar.

Si se desea cambiar la contraseña, a través de botón de cambio contraseña se accede a otra pantalla donde debe de indicarse el usuario, la contraseña actual y por dos veces la nueva contraseña que se desea.

La pantalla de inicio está gestionada por el controlador GestorInicio. Este controlador se encarga de las siguientes funciones:

- Validar usuario y contraseña.
- Preparar la carga de los elementos que se visualizarán en los componentes combo de las pantallas, como: tipo usuario, técnico, oficina y departamento.
- Determinar el tipo de usuario para redireccionar a la pantalla de menú adecuada según su perfil de usuario.

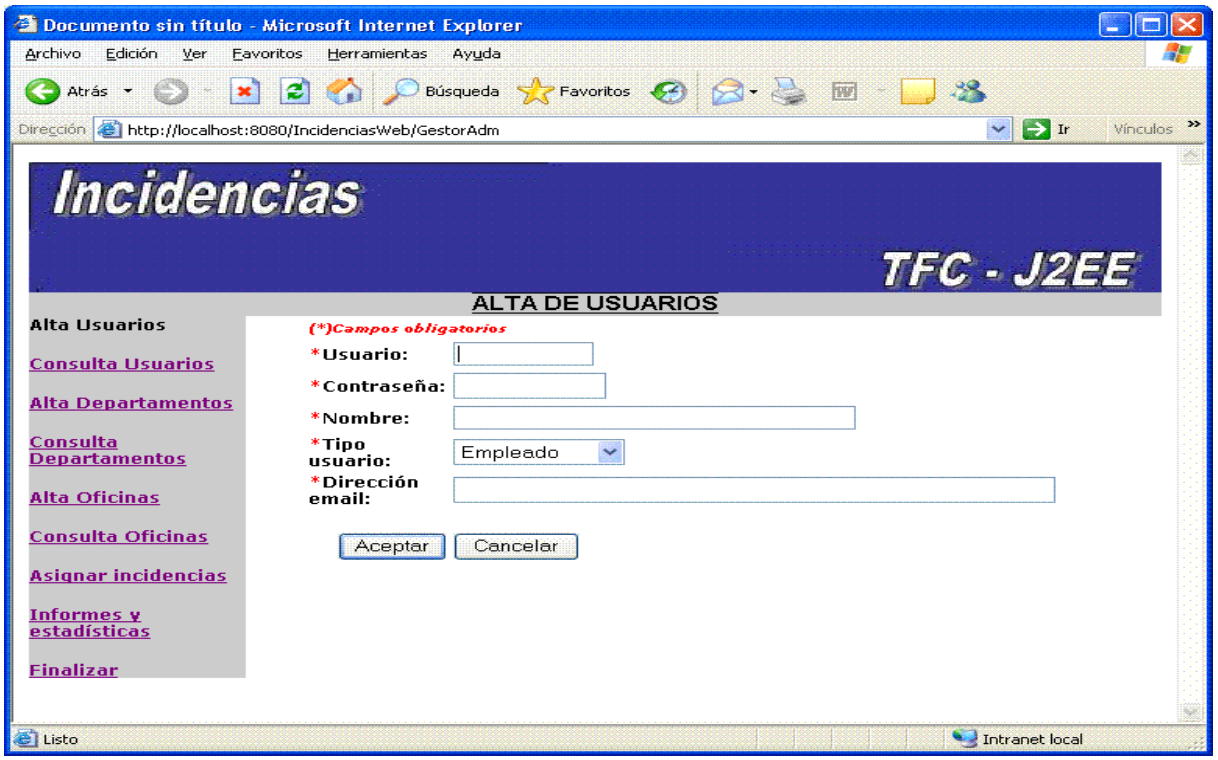
4.1.2. Funcionalidades del perfil Administrador

A través del menú del administrador se accede a las siguientes funcionalidades de que dispone.

Las funciones del administrador están gestionadas por el controlador GestorAdm. Este controlador se encarga del mantenimiento del entorno de la aplicación, como el mantenimiento de usuarios, de oficinas y de departamentos. Además, gestiona la asignación de incidencias que se realiza a los técnicos que estén definidos en el aplicativo. También se encarga de la gestión y confección de los informes.

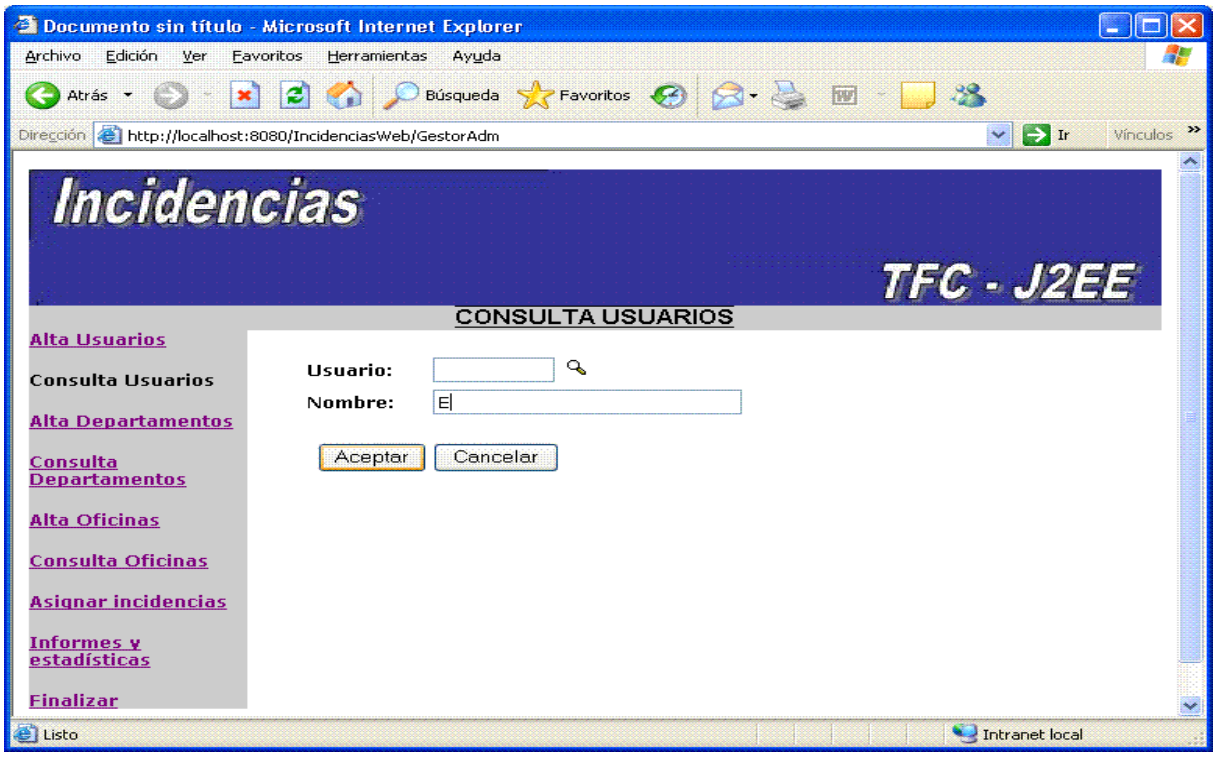
4.1.2.1. Creación de usuarios

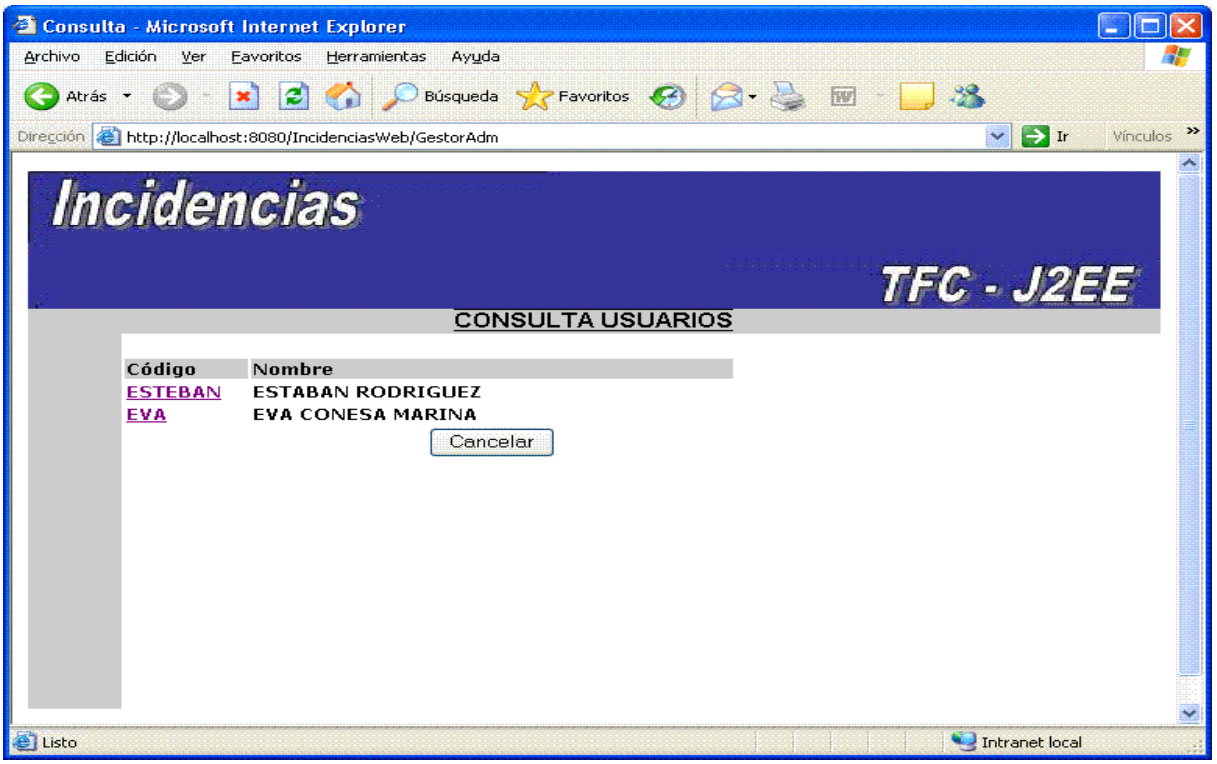
Desde la siguiente pantalla se realizan las altas de usuario que tendrán acceso a la aplicación. La pantalla muestra los campos a introducir, con el botón de aceptar se confirman los datos y se crea el usuario en la BD.



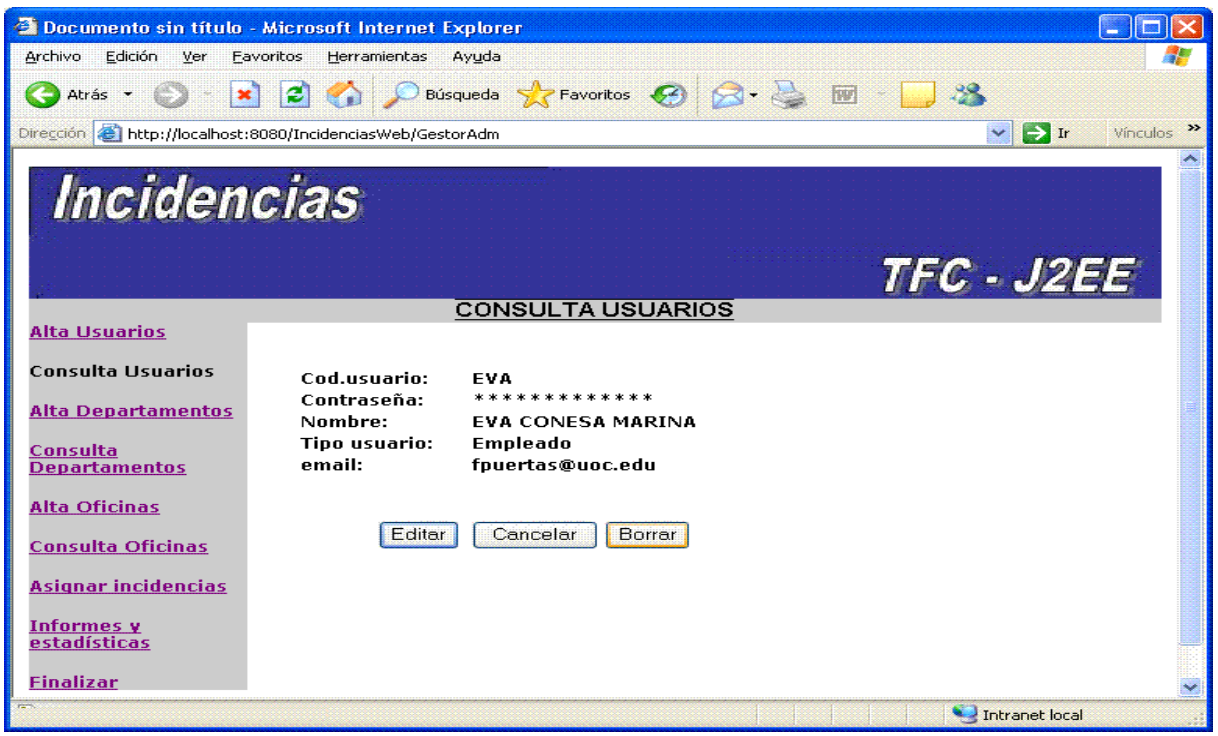
4.1.2.2. Consulta de usuarios

Desde las siguientes pantallas se permite la consulta, modificación y supresión de un usuario. A través del código de usuario o realizando una búsqueda total o parcial por el nombre se obtienen los datos del usuario. Las siguientes pantallas muestran la secuencia.



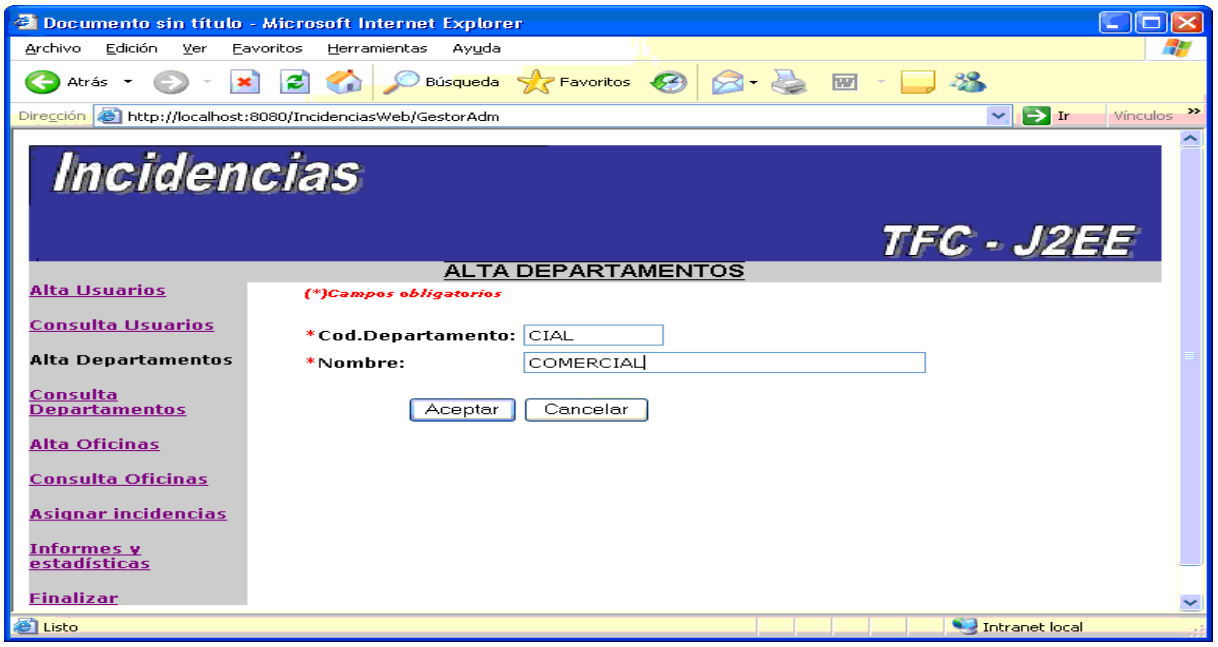


Desde la pantalla de consulta se visualizan los atributos del usuario, desde aquí se permiten realizar varias acciones a través de los botones de control.
El botón de editar presenta una pantalla con los atributos del usuario para su modificación.
El botón de borrar elimina el registro del usuario de la base de datos, esta acción solamente estará permitida si el usuario a suprimir no tiene ninguna incidencia creada.
El botón de cancelar, cancela la acción sin realizar ninguna acción.



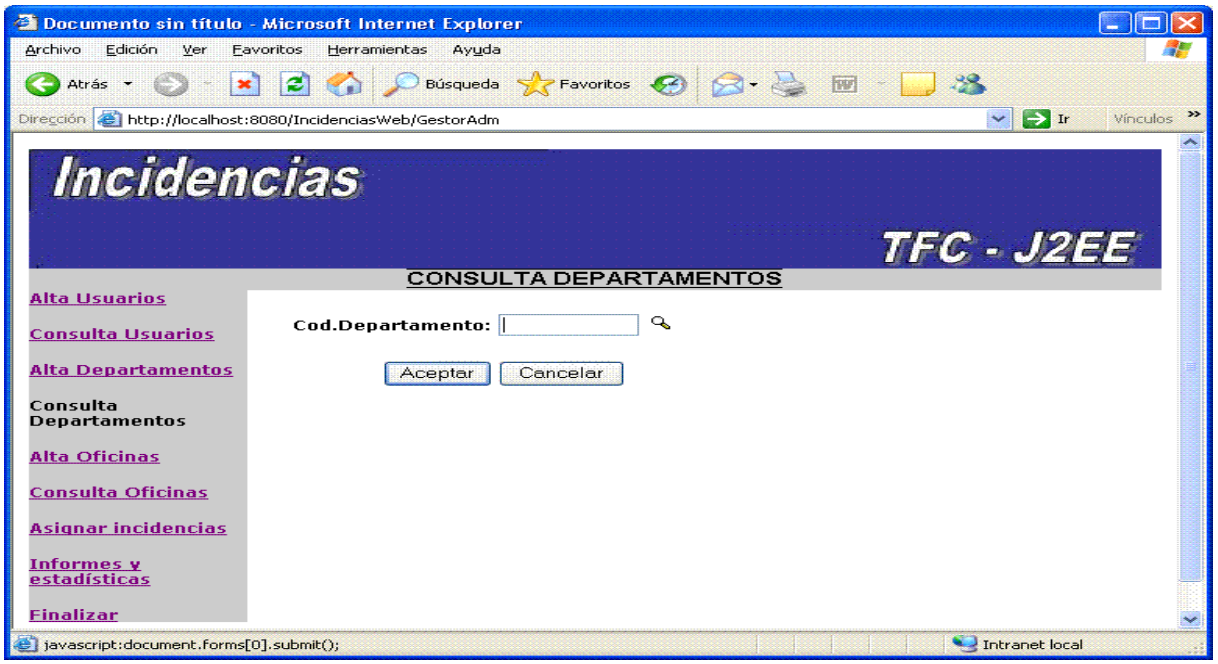
4.1.2.3. Creación de departamentos

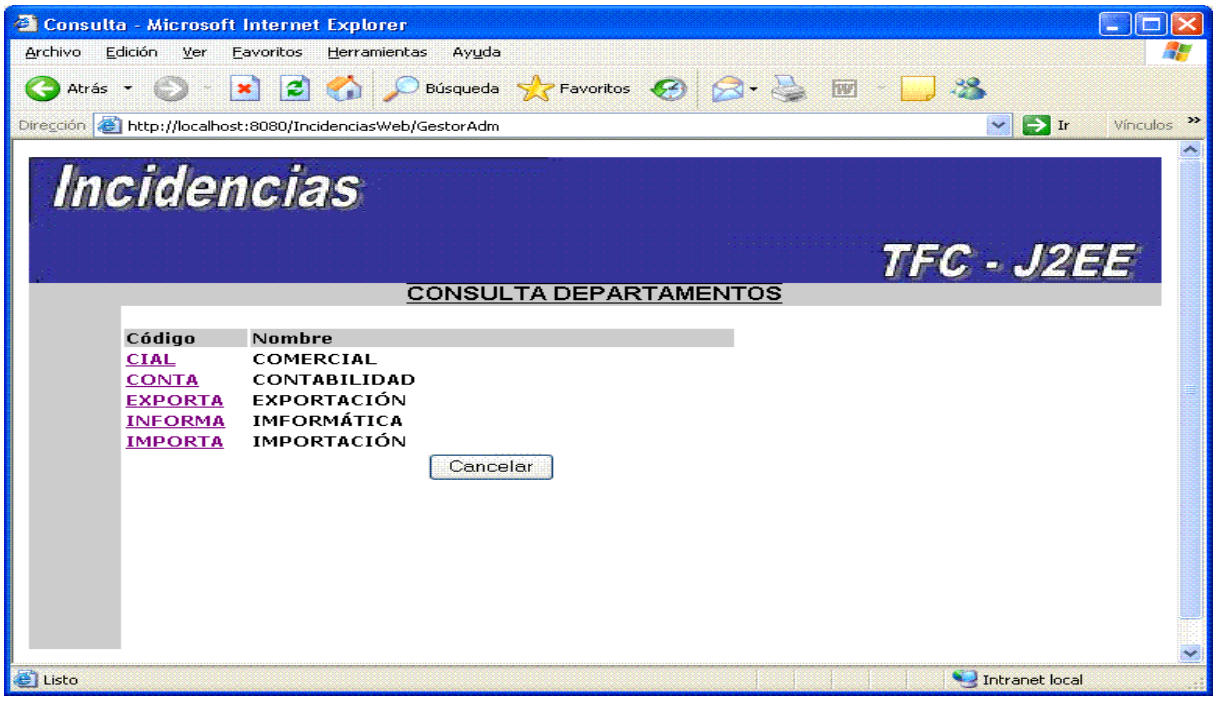
Desde la siguiente pantalla se realizan las altas de departamentos que dispondrá la aplicación. La pantalla muestra los campos a introducir, con el botón de aceptar se confirman los datos y se crea un departamento en la BD.



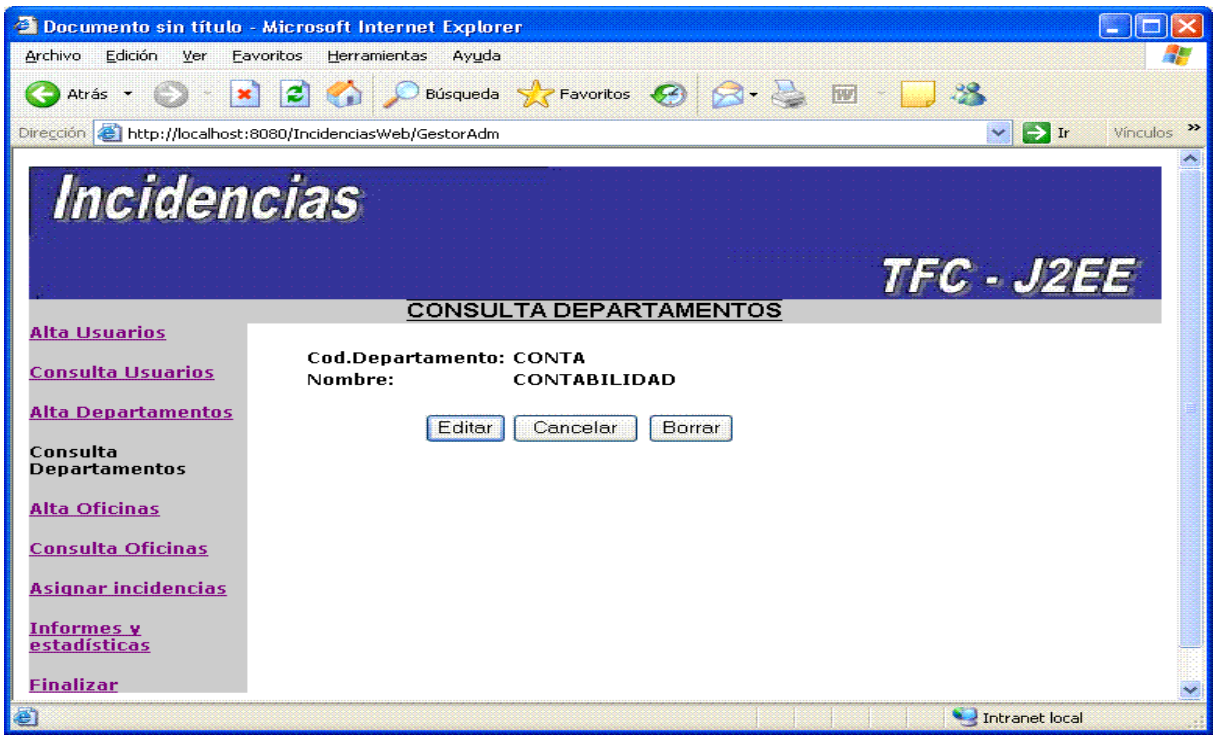
4.1.2.4. Consulta de departamentos

Desde las siguientes pantallas se permite la consulta, modificación y supresión de un departamento. A través del código de departamento o realizando una búsqueda total, se obtiene el código o la lista de departamentos para su selección. Las siguientes pantallas muestran la secuencia.



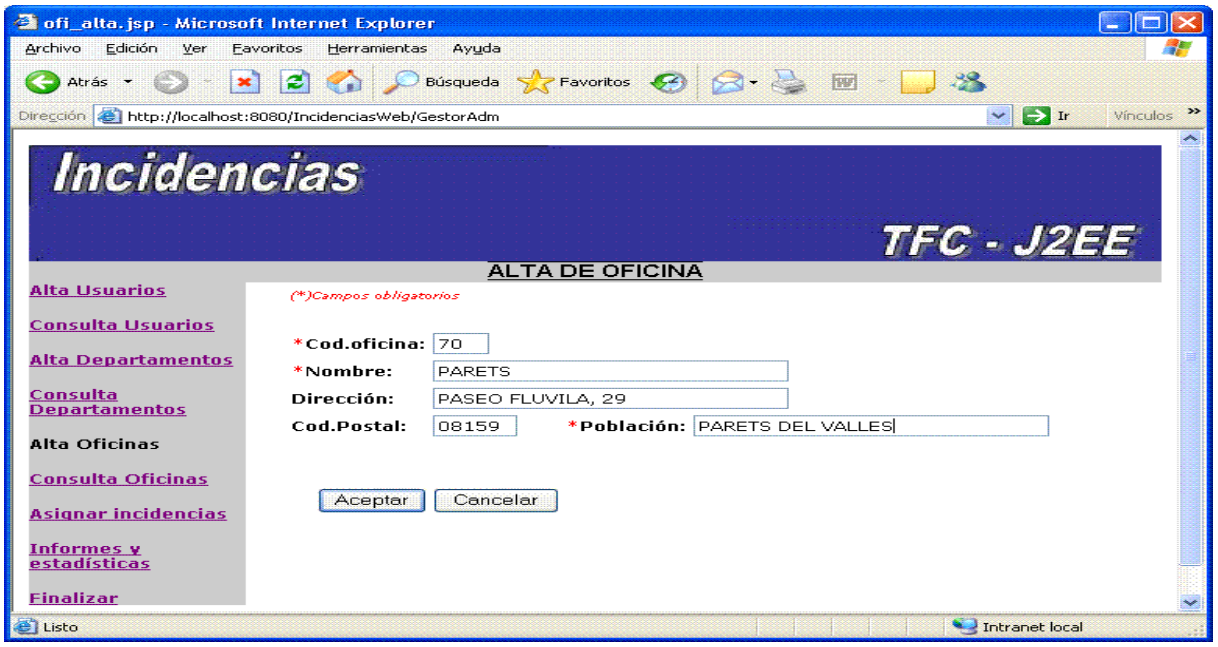


Desde la pantalla de consulta se visualizan los atributos del departamento, se permiten realizar varias acciones a través de los botones de control.
El botón de editar presenta una pantalla con los atributos del departamento para su modificación.
El botón de borrar elimina el registro del departamento de la base de datos, esta acción solamente estará permitida si el departamento a suprimir no tiene ninguna incidencia asignada.
El botón de cancelar, cancela la acción sin realizar ninguna acción.



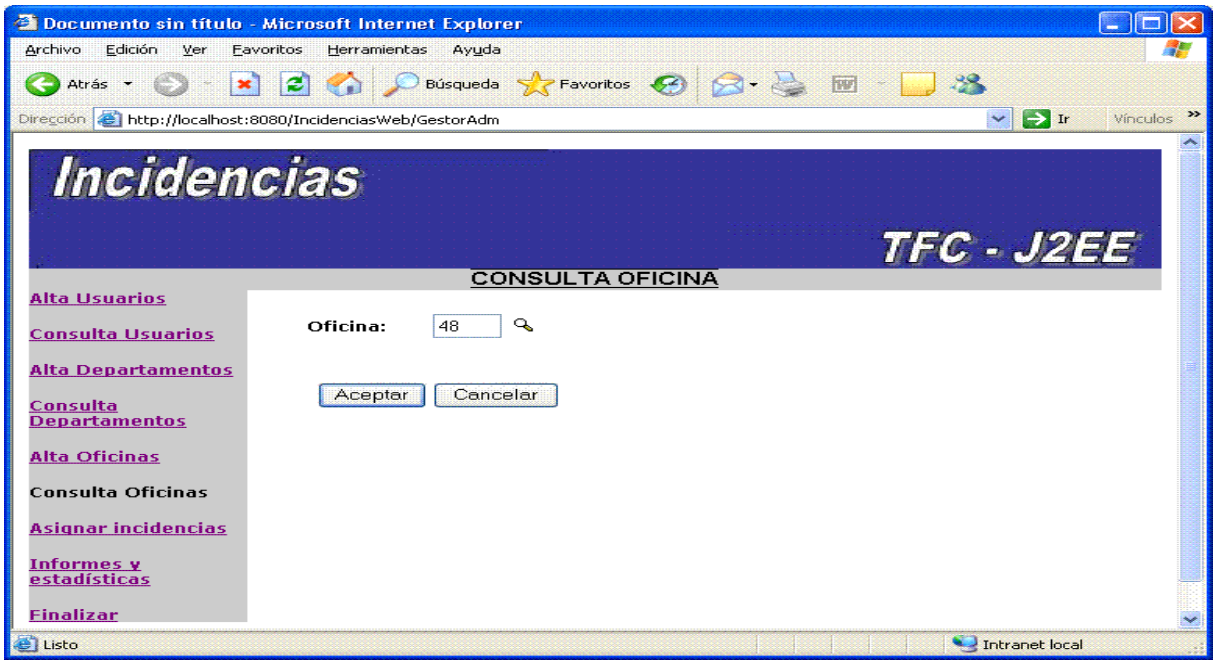
4.1.2.5. Creación de oficinas

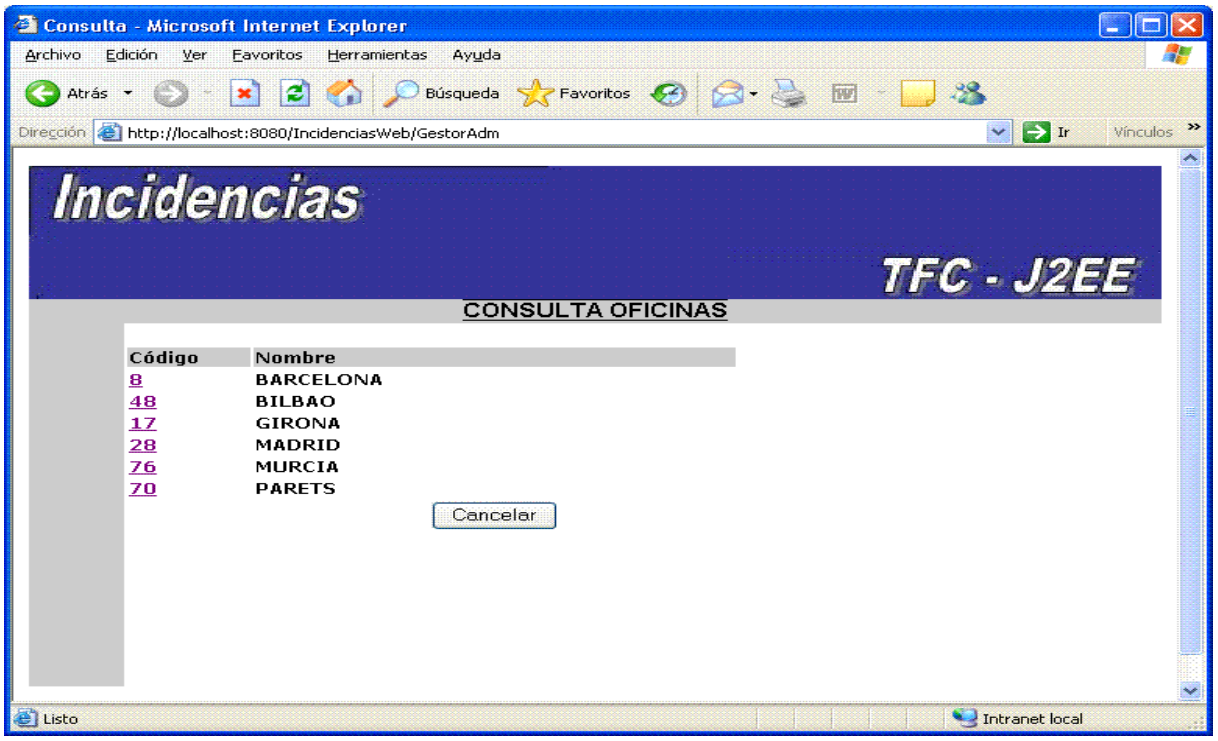
Desde la siguiente pantalla se realizan las altas de las oficinas de la organización que dispondrá la aplicación. La pantalla muestra los campos a introducir, con el botón de aceptar se confirman los datos y se crea una oficina en la BD.



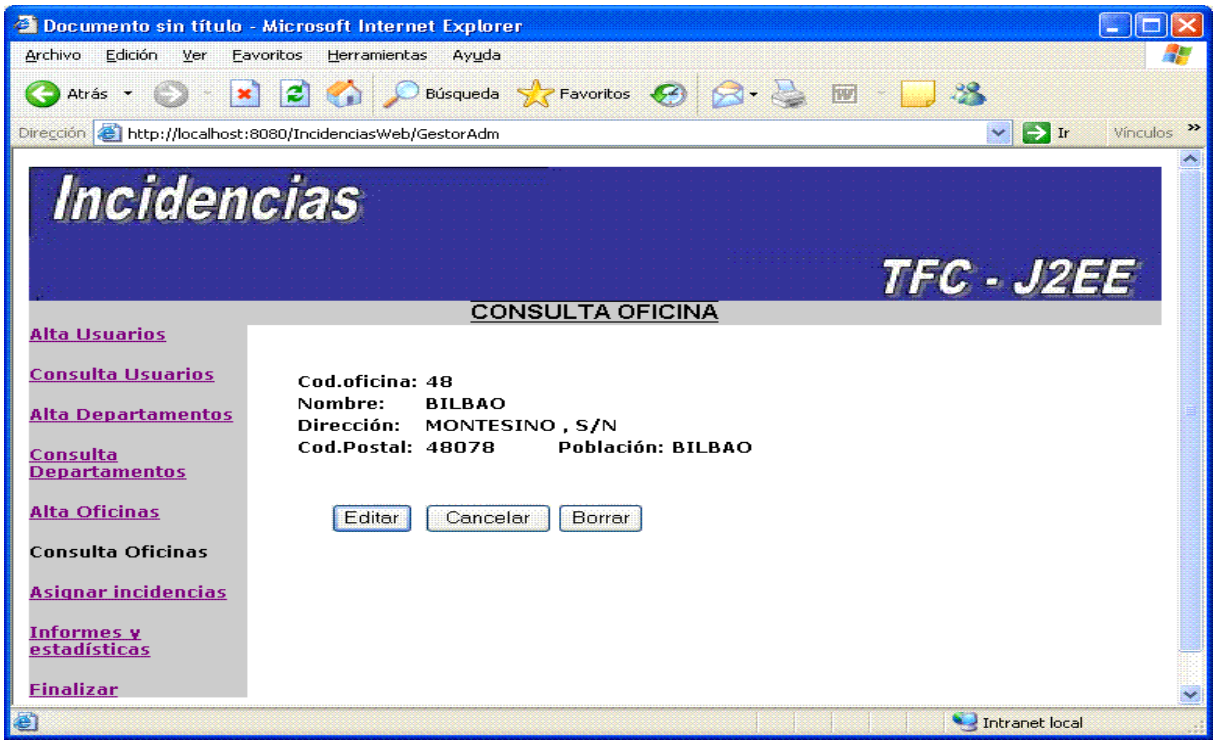
4.1.2.6. Consulta de oficinas

Desde las siguientes pantallas se permite la consulta, modificación y supresión de una oficina. A través del código de oficina o realizando una búsqueda total se obtiene una lista para seleccionar el código que se desee. Las siguientes pantallas muestran la secuencia.



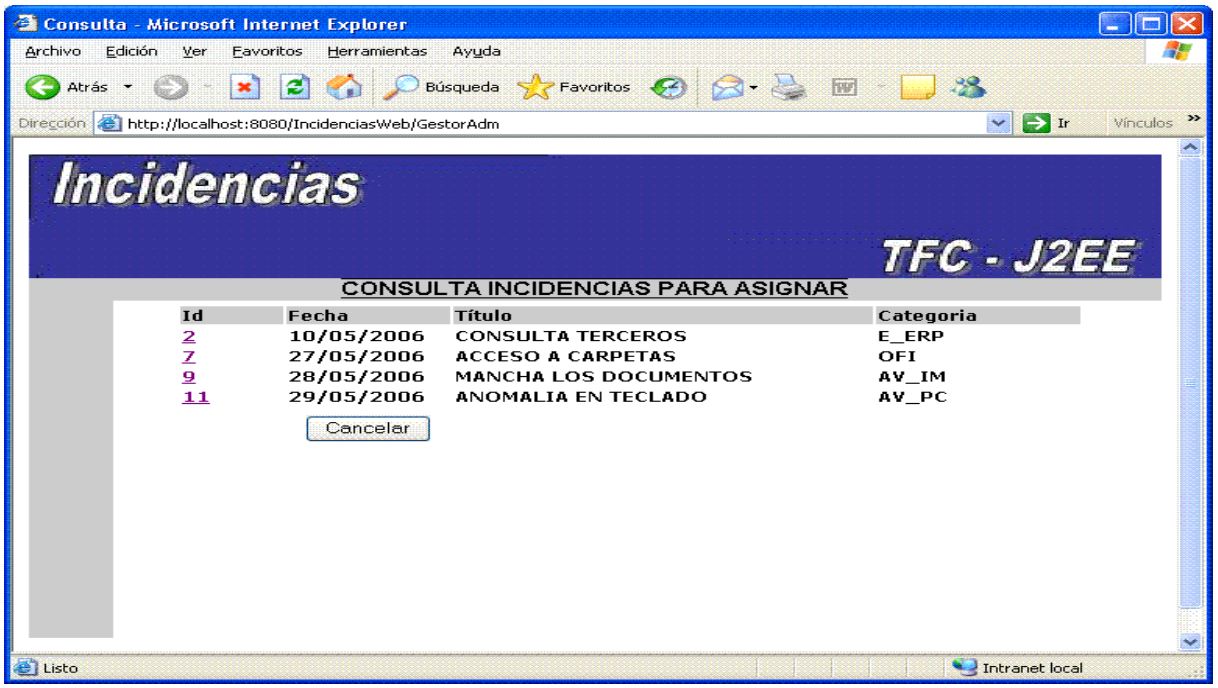


Desde la pantalla de consulta se visualizan los atributos de la oficina, se permiten realizar varias acciones a través de los botones de control.
El botón de editar presenta una pantalla con los atributos de la oficina para su modificación.
El botón de borrar elimina el registro de la oficina de la base de datos, esta acción solamente estará permitida si la oficina a suprimir no tiene ninguna incidencia creada.
El botón de cancelar, cancela la acción sin realizar ninguna acción.

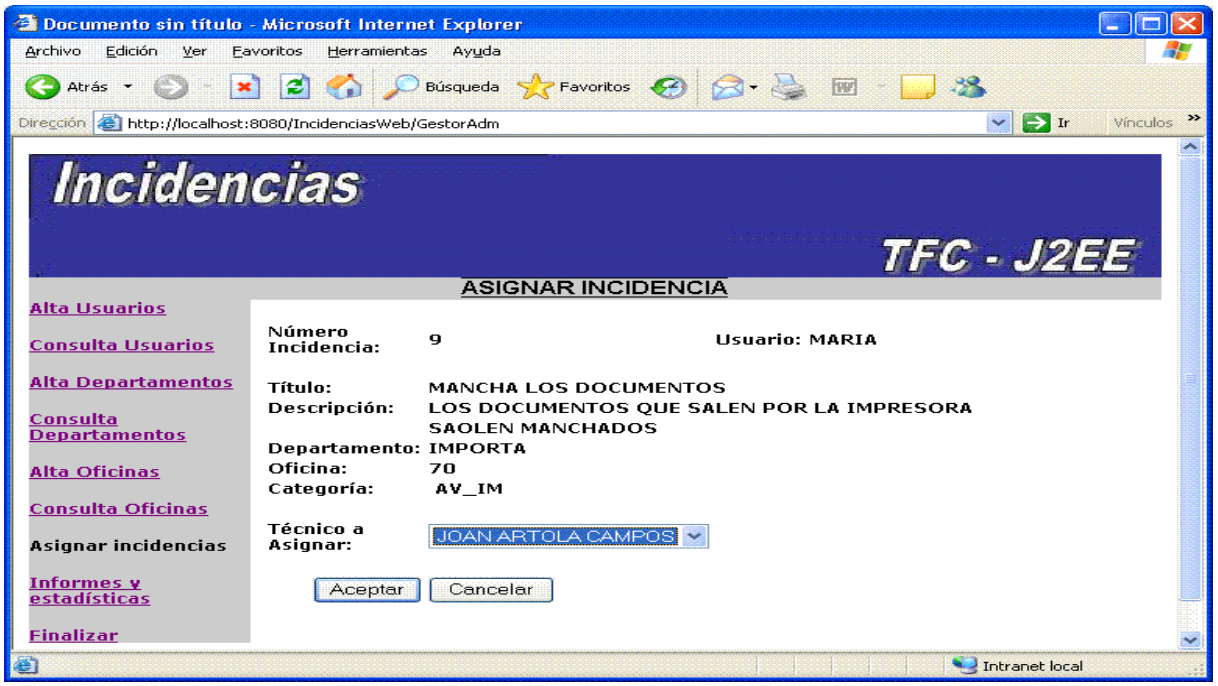


4.1.2.7. Asignar incidencias

Muestra una pantalla con la lista de incidencias generadas por el empleado y pendientes de asignar a un técnico. Al seleccionar una de ellas, muestra otra pantalla con todos los datos de la incidencia creada por el empleado.

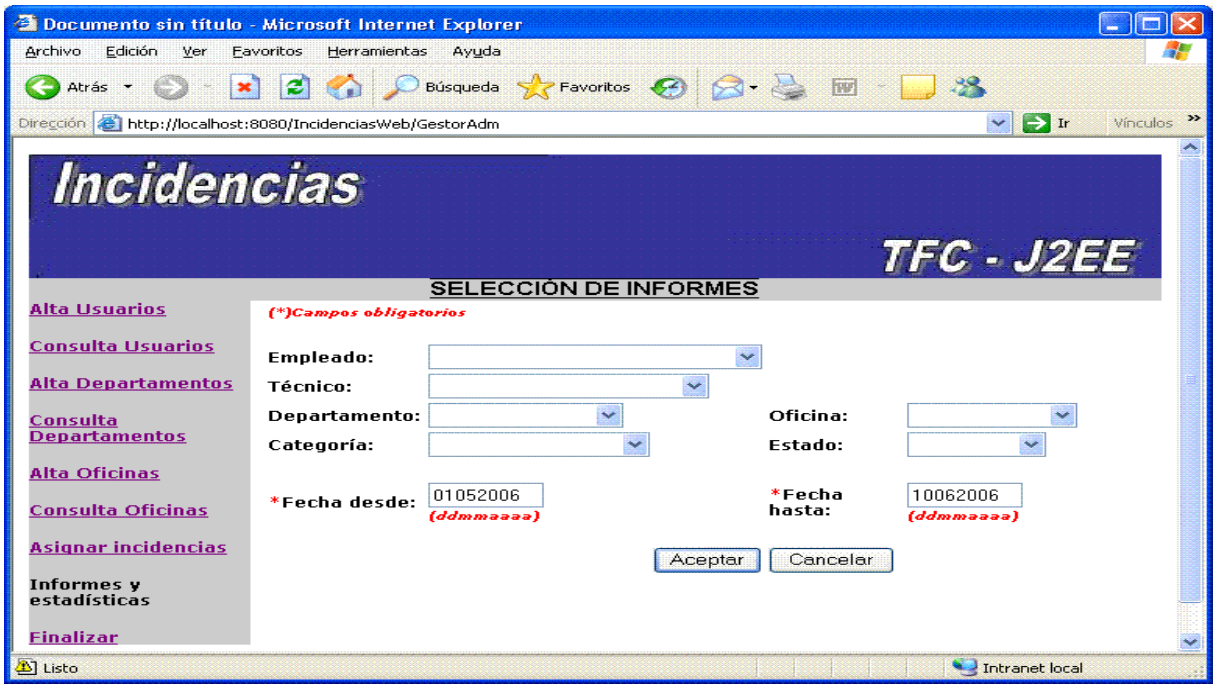


Para asignar la incidencia se seleccionará uno de los técnicos que están disponibles en el componente combo de la pantalla. Con el botón de aceptar la incidencia quedará asignada al técnico con el estado de asignada. En ese instante la aplicación generará un correo electrónico destinado al empleado notificándole esta última acción.



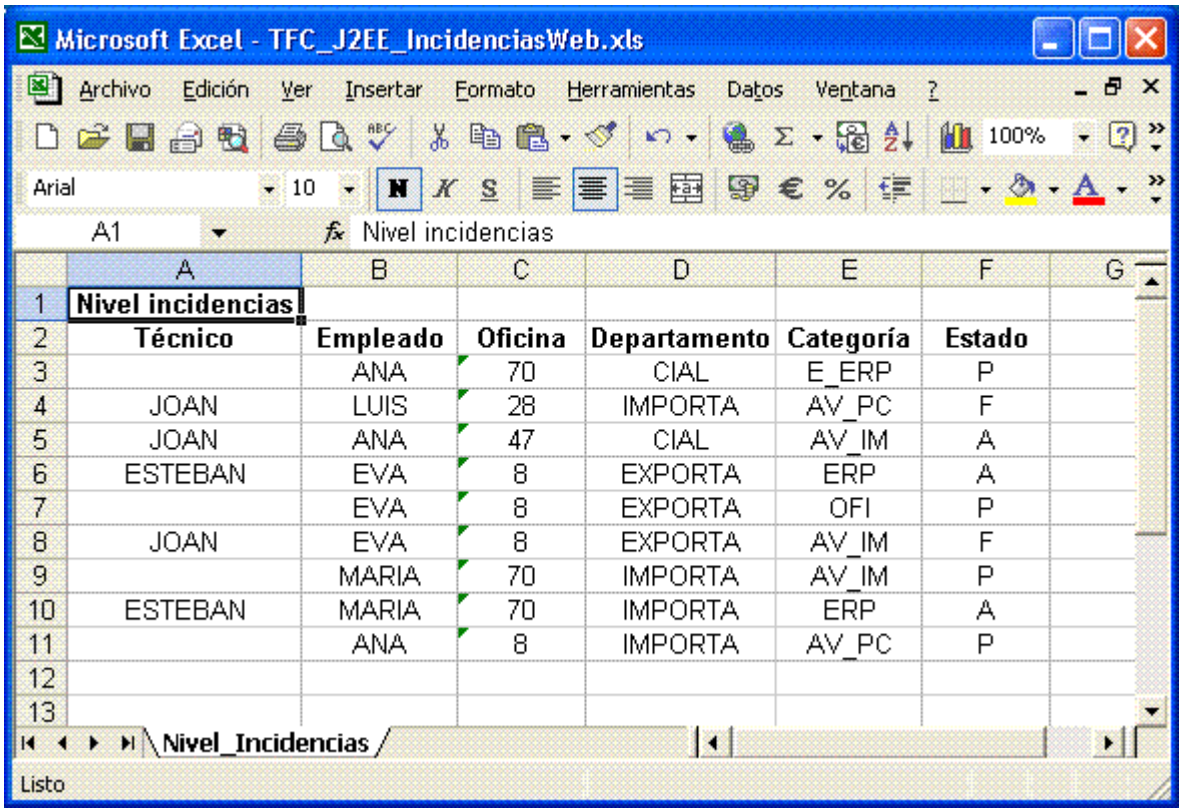
4.1.2.8. Informes y estadísticas

Desde esta función se generan los informes de la aplicación. En la pantalla de solicitud de informes se muestra los criterios de selección por los que se puede obtener la información para su posterior estudio. Los componentes combos de la pantalla son cargados con los elementos de la BD en el proceso de login que se efectúa cuando se entra a la aplicación.



La aplicación genera un documento .xls con todos los registros de la tabla de incidencia que cumplan con los criterios de selección indicados en la pantalla de solicitud.
Los criterios de selección se combinan usando el operador lógico “y”.
El documento puede abrirse o guardarse a partir del cuadro de dialogo que muestra la aplicación en el momento de su creación.

La información generada en todo el proceso de incidencias permitirá el estudio y conocimiento del comportamiento del personal de la organización y de sus herramientas de trabajo.



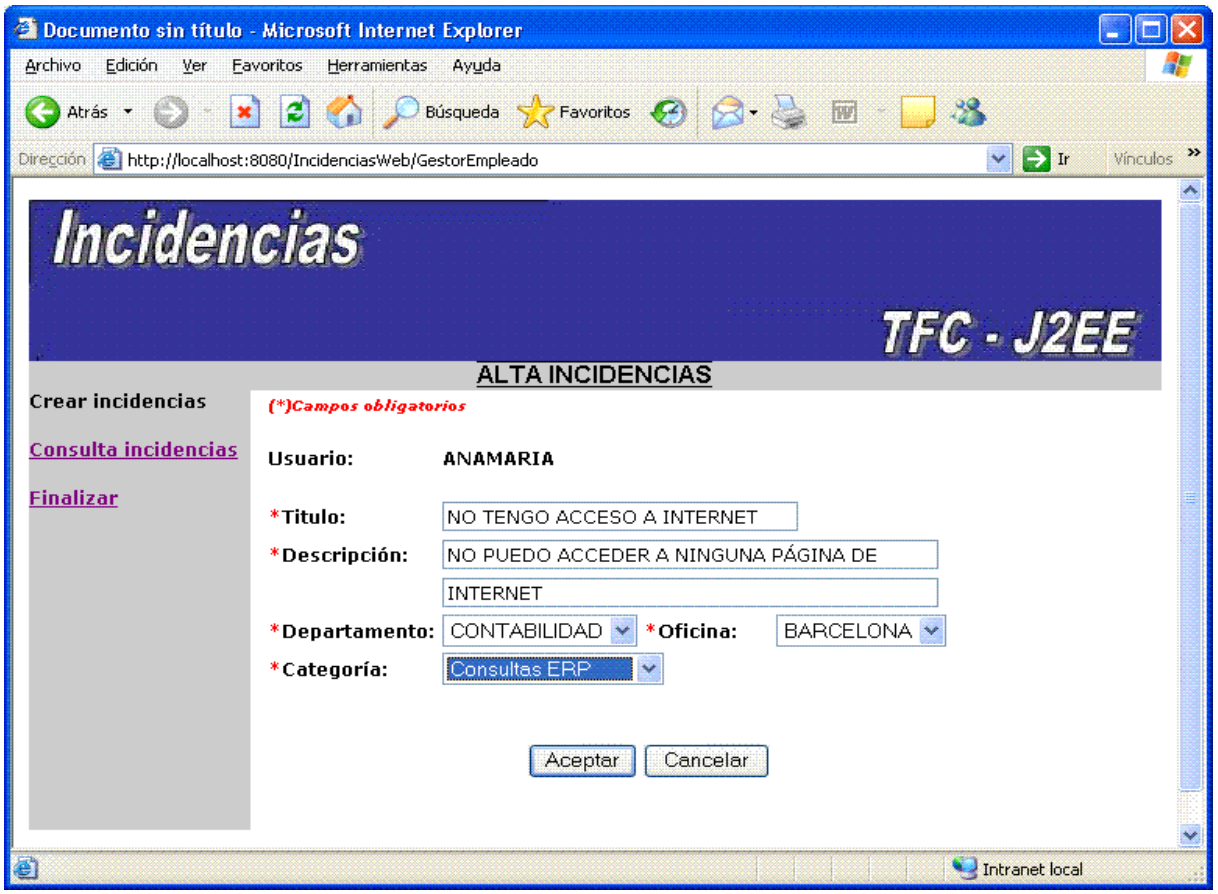
	A	B	C	D	E	F	G
1	**Nivel incidencias**						
2	**Técnico**	**Empleado**	**Oficina**	**Departamento**	**Categoría**	**Estado**	
3		ANA	70	CIAL	E_ERP	P	
4	JOAN	LUIS	28	IMPORTA	AV_PC	F	
5	JOAN	ANA	47	CIAL	AV_IM	A	
6	ESTEBAN	EVA	8	EXPORTA	ERP	A	
7		EVA	8	EXPORTA	OFI	P	
8	JOAN	EVA	8	EXPORTA	AV_IM	F	
9		MARIA	70	IMPORTA	AV_IM	P	
10	ESTEBAN	MARIA	70	IMPORTA	ERP	A	
11		ANA	8	IMPORTA	AV_PC	P	
12							
13							

4.1.3. Funcionalidades del perfil Empleado

A través del menú del empleado se acceden a las siguientes funcionalidades de que dispone.

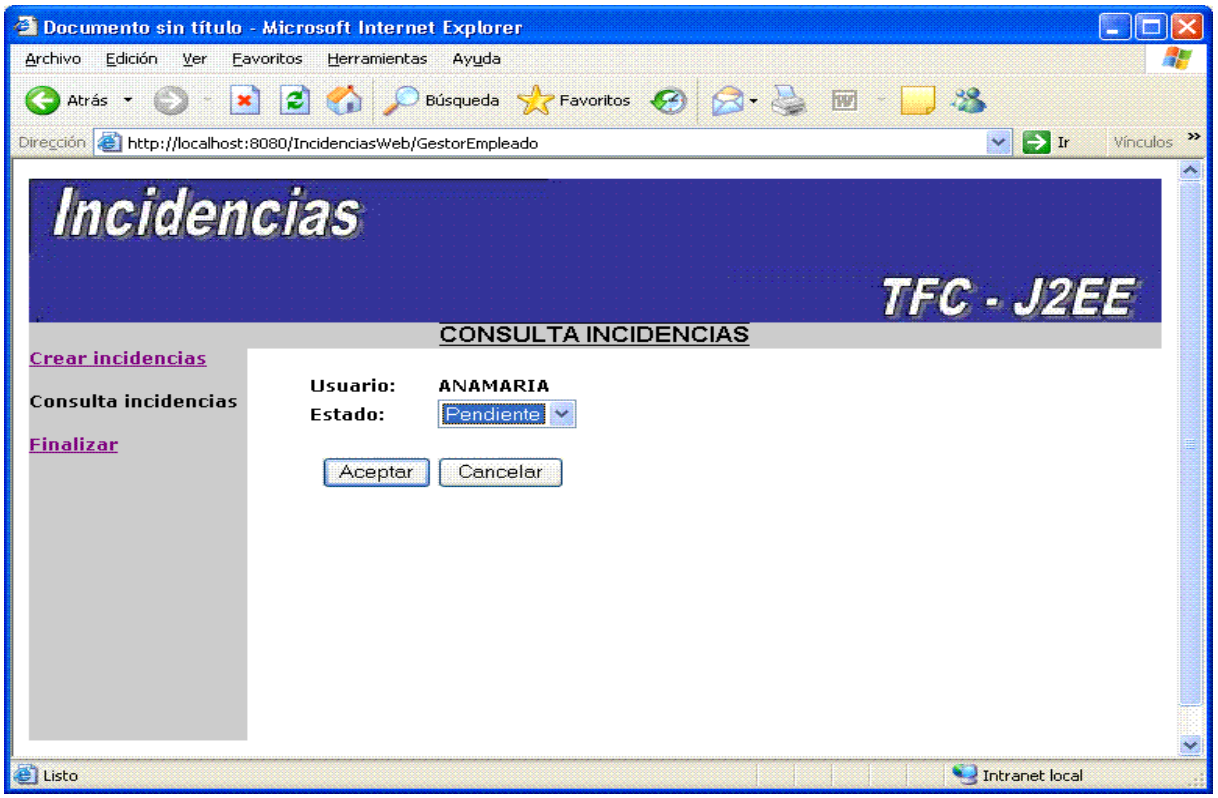
4.1.3.1. Creación de incidencias

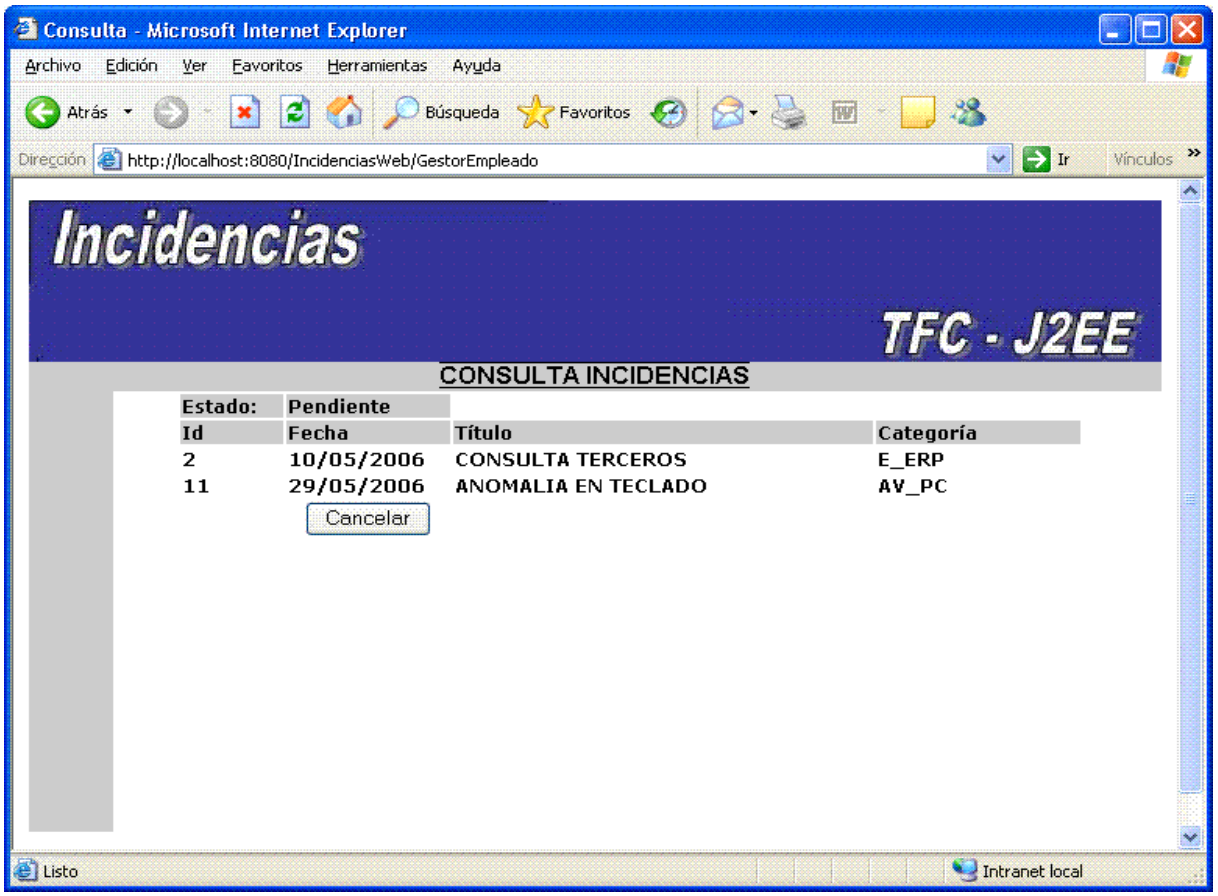
El alta de incidencias se realiza desde esta funcionalidad. La pantalla muestra los campos a introducir para efectuar el registro de petición de asistencia solicitado por el empleado.
Una vez introducidos los datos en el formulario, con el botón de aceptar se genera una incidencia en la BD con el estado de pendiente. A partir de aquí la incidencia queda disponible para que el administrador le asigne un técnico para su resolución.



4.1.3.2. Consulta de incidencias

El empleado puede en todo instante consultar el estado de sus peticiones de asistencia. Desde la siguiente pantalla se muestran los tres tipos de consulta que pueden realizar, siempre a partir de su estado. Seleccionando el estado a consultar desde el componente combo de la pantalla, se muestra una pantalla con la lista de incidencias creadas por el empleado con el estado solicitado.


Documento sin título - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://localhost:8080/IncidenciasWeb/GestorEmpleado
Ir Vínculos
Incidencias
TFC - J2EE
CONSULTA INCIDENCIAS
Crear incidencias
Consulta incidencias
Finalizar
Usuario: ANAMARIA
Estado: Pendiente
Aceptar Cancelar
Listo
Intranet local

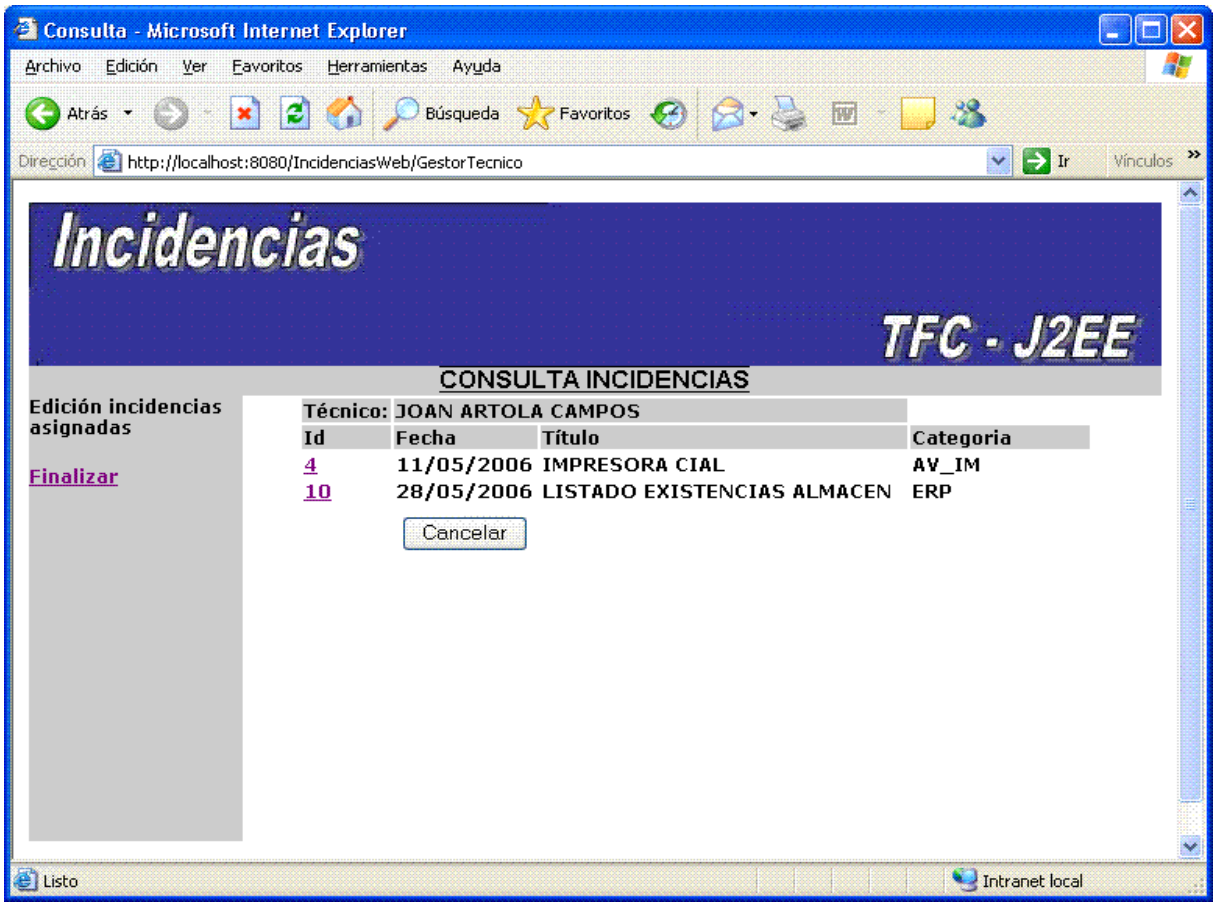

Consulta - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://localhost:8080/IncidenciasWeb/GestorEmpleado
Ir Vínculos
Incidencias
TFC - J2EE
CONSULTA INCIDENCIAS
Estado: Pendiente
Id Fecha Título Categoría
2 10/05/2006 CONSULTA TERCEROS E_ERP
11 29/05/2006 ANOMALIA EN TECLADO AV_PC
Cancelar
Listo
Intranet local

4.1.4. Funcionalidades del perfil Técnico

A través del menú del técnico se acceden a las siguientes funcionalidades de que dispone.

4.1.4.1. Edición incidencias asignadas

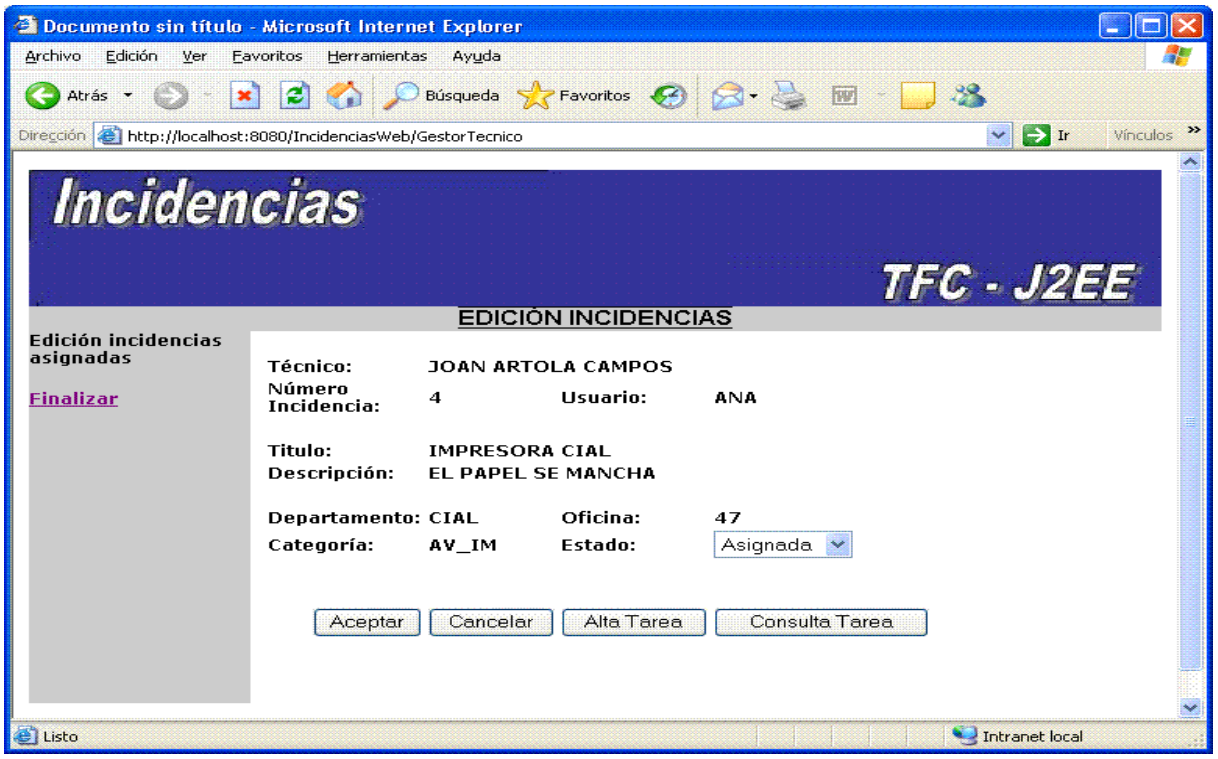
Desde esta opción el técnico conoce las incidencias que tiene asignadas y que están pendientes de resolver. Se muestra una pantalla con la lista de las incidencias que tiene asignadas. Para trabajar sobre una de ellas deberá seleccionarse una de la lista.



A continuación se muestra una pantalla con los datos de la incidencia a resolver y con las acciones que puede realizar.

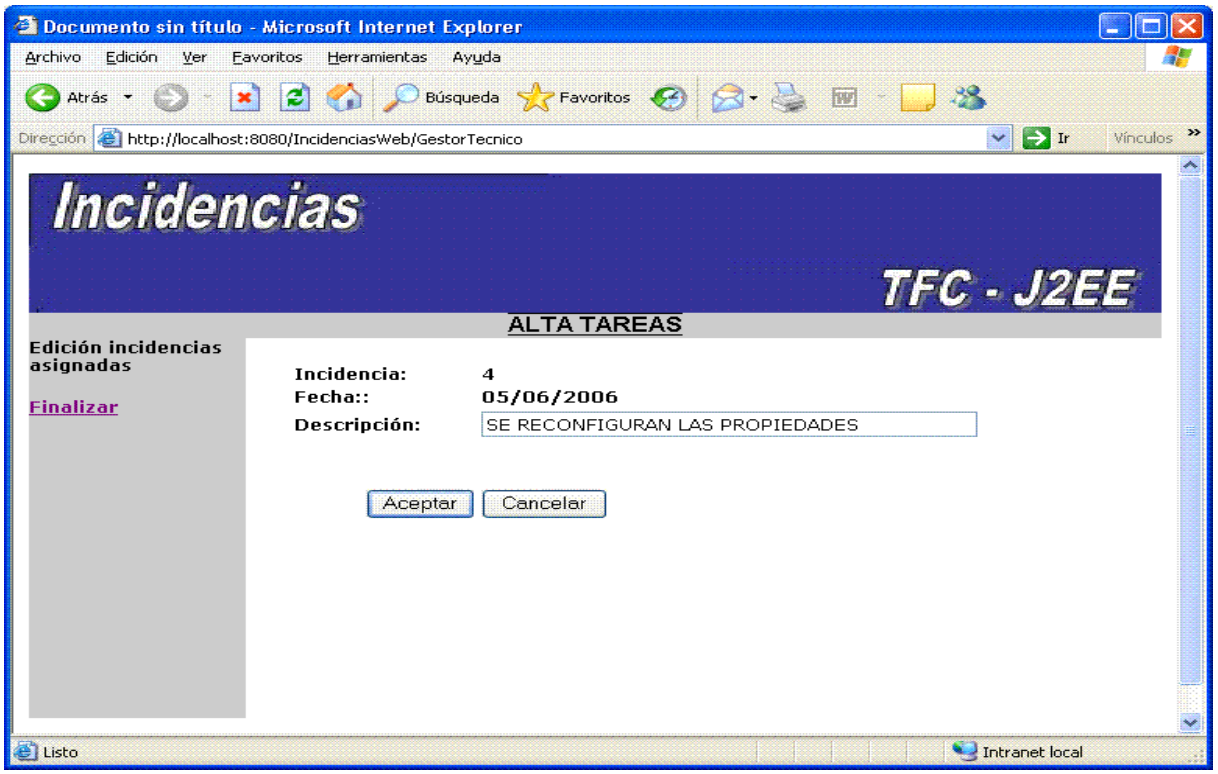
Para la finalización de la incidencia debe seleccionarse el estado Finalizada desde el componente combo de la pantalla.

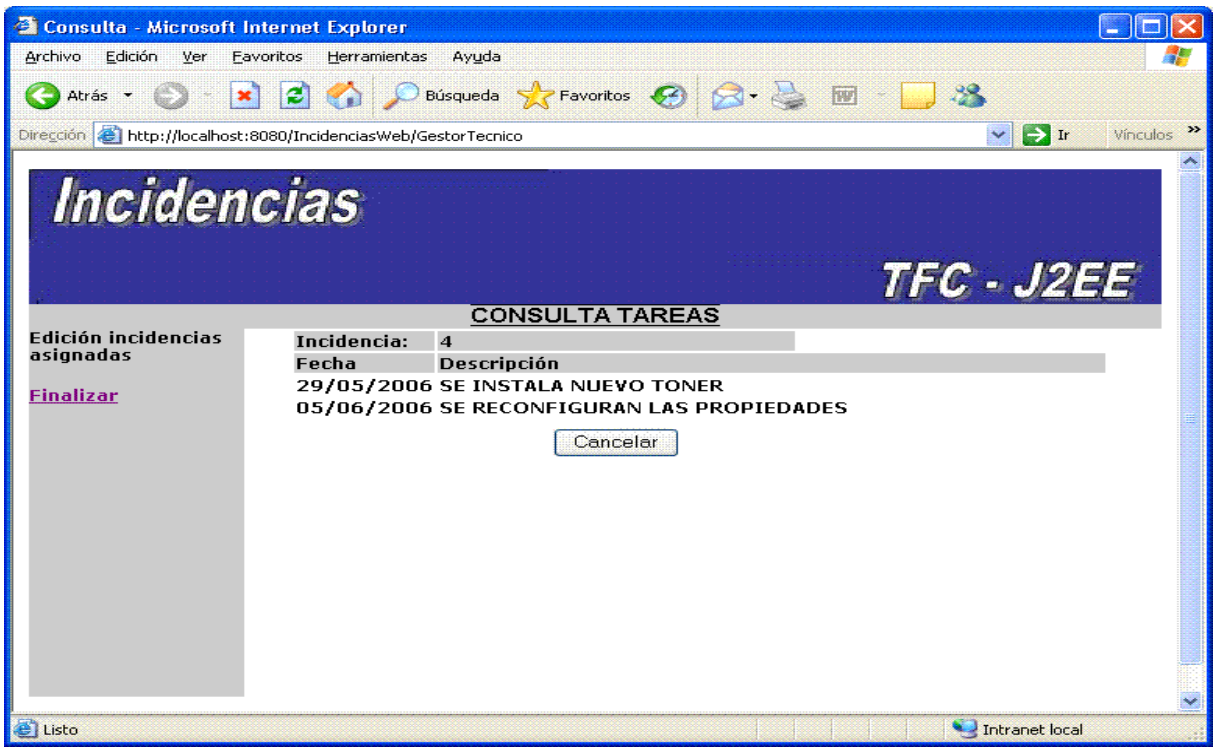
Al finalizar la incidencia, la aplicación genera un correo electrónico destinado al empleado, donde se le informa de la finalización de su incidencia.



Desde la edición de la incidencia, a través del botón Consulta Tarea se puede consultar las tareas ya realizadas sobre esa incidencia. Con el botón Alta Tarea se puede generar una nueva tarea indicando qué acción se va a realizar para su resolución.

A continuación se muestran las pantallas de la gestión de Tareas.




Consulta - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://localhost:8080/IncidenciasWeb/GestorTecnico
Ir Vínculos
Incidencias
TFC - J2EE
CONSULTA TAREAS
Edición incidencias asignadas
Finalizar
Incidencia: 4
Fecha Descripción
29/05/2006 SE INSTALA NUEVO TONER
05/06/2006 SE RECONFIGURAN LAS PROPIEDADES
Cancelar
Listo
Intranet local

5. Valoración económica

Es difícil realizar una valoración económica, pero se podría realizar una estimación a partir de una media diaria o semanal, y de los días que tenemos definidos en el diagrama de GANTT según la planificación inicialmente definida.

El número de días son 103, y en mi caso no hay distinción entre días laborales y festivos. Estimo que la dedicación semanal ha sido de unas 26 horas, si el total de semanas son aproximadamente 15, tenemos un total de 390 horas/hombre en todo el ciclo de vida del proyecto.

Si se estima un coste medio de 40 euros/hora entre los diferentes perfiles que hayan podido intervenir en el proyecto (jefe de proyecto, analista, programador), tenemos que la aplicación podría costar en el mercado del orden de 15.600 euros.

Lógicamente este coste podría ser mucho menor porque han existido muchos factores que han causado un exceso de horas en la finalización del proyecto como: el desconocimiento previo de la tecnología, los componentes que han tenido que incorporarse, el despliegue y la configuración del servidor de aplicaciones.
Por el contrario, no es habitual la realización de un proyecto con el nivel de detalle que se ha aportado en todos los entregables de los hitos de control.

A parte de todo esto, si se pretendiera comercializar la aplicación como un producto estándar, se deberían de tener en cuenta otras consideraciones de marketing como los gastos de promoción y precios de licencia, siempre teniendo en cuenta los precios de mercado de un producto de estas características.

6. Conclusiones

La experiencia del TFC ha sido positiva y ha cubierto mucho más de lo esperado las expectativas que había puesto de antemano en está asignatura.

He adquirido un conocimiento más que suficiente en la arquitectura MVC, que me ha parecido muy interesante y portable a cualquier plataforma. En el apartado de la Vista de la aplicación, no me he encontrado cómodo porque nunca antes había trabajado con javascript, existen multitud de componentes que podían ayudarme pero el tiempo que tenía estimado para la parte de la Vista no me ha permitido trabajar y modelar esos componentes. He optado, por ejemplo, con un menú muy simple y no gastar el tiempo en localizar componentes que le dieran un aspecto más profesional, eso lo dejaré como una mejora a realizar en el aplicativo.

He desarrollado clases que podrán reutilizarse en otras aplicaciones como: Email_inc, HojaCal, ListaTabla. También he utilizado recursos de terceros, buscando por la red el que mejor se adaptaba a las características de la aplicación. He valorado las ventajas de utilizar un lenguaje OO con la facilidad de incorporar componentes y la reutilización de las clases.

Por primera vez he trabajado con un servidor de aplicaciones (Tomcat) que he tenido que configurar y sobre el que he desplegado la aplicación.

Por último, es el primer trabajo en el que realizo completamente el ciclo de vida de un proyecto. Para su realización me han sido de ayuda la experiencia adquirida en las asignaturas de Ingeniería del Programario I y la de Técnicas de Desarrollo del Programario. También me ha ayudado el haber abordado un proyecto cuyo contenido y requerimientos ya estaban conceptualmente definidos en mi mente.

7. Líneas de desarrollo futuro

La aplicación está sujeta a muchas posibles ampliaciones y mejoras, muy probablemente el propio uso que se haga de ella por parte de los usuarios finales muestren otras que inicialmente no se habían contemplado.
Una vez realizado el desarrollo aparecen las siguientes posibilidades:

- **Gestión de menús:** crear una gestión dinámica del menú a partir del contenido de un fichero identificado por perfil de usuario. Además, que pueda ser configurable a través del fichero de estilos CSS.

- **Notificación de incidencia:** generar un correo destinado al administrador cuando el empleado cree una nueva incidencia, notificando de esa manera que hay incidencias pendientes de asignar.

- **Descripción de la incidencia:** sustituir los campos de descripción de la incidencia por un área de texto para poder incrustarle ficheros adjuntos que complementen o amplíen la descripción.

- **Encriptar la contraseña:** desarrollar un módulo de seguridad para encriptar la contraseña en el formulario de entrada y en el fichero de la base de datos.

- **Lectura de correos:** permitir leer el contenido del cuerpo de un correo que se haya enviado a una dirección concreta de correo. A partir del remitente y con los datos del correo generar la incidencia automáticamente.

- **Conectar con aplicación externa:** con el fin de poder profundizar más exhaustivamente en el conocimiento que nos pueda aportar el histórico de incidencias creadas, se podría conectar con alguna otra aplicación de minería de datos, tipo Weka, para poder predecir comportamientos a partir de los datos extraídos de las incidencias.

8. Glosario

CSS: Son las hojas de estilo en cascada (Cascading Style Sheets, CSS). Es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

ER: Diagramas de Entidad Relación. Es un modelo conceptual para la representación gráfica del diseño de las bases de datos.

HSSF: Es la API de POI para implantar ficheros con formato Excel.

MVC: Modelo, Vista y Controlador. Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz gráfica y la lógica de control.

OO: Orientación a Objetos. Es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan *estado*, *comportamiento e identidad.*

Open source: Código abierto (open source en inglés) es el término por el que se conoce al software distribuido y desarrollado en una determinada forma.

POI: Es un conjunto de APIs para manipular diferentes tipos de formatos de ficheros basados en formatos de Documento Compuesto OLE 2 de Microsoft.

SMTP: Simple Mail Transfer Protocol (SMTP), o protocolo simple de transferencia de correo electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras.

Weka: Es un programa que permite realizar minería de datos efectuando combinaciones de algoritmos para poder extraer conocimiento. Dispone de una interfaz gráfica y de un conjunto de librerías con sus propias APIs para la utilización de sus funciones.

9. Bibliografía y referencias

En la bibliografía se incluye, además del material de consulta utilizado, referencias a artículos de Internet que son de inclusión necesaria ya que se han utilizado en el desarrollo del proyecto.

Falkner Jayson, Galbraith Ben, Irani Romin, Kochmer Casey, Narayan Sathya, Perrumal Krishnaraj, Timmey John, Moidoo Meeraj. (2001). *Fundamentos Desarrollo Web con JSP*. Madrid: Ediciones Anaya Multimedia

Eckel Bruce, (2002) *Piensa en Java*.(segunda edición) Madrid: Prentice Hall

Bernaus Albert, Blanco Jaume. (1996). *Diseño de Paginas Web*. Barcelona: Inforbook's

Hall Marty (1998). *Core Servlets and JavaServer Pages.* Recuperado el 3 Marzo 2006, desde
http://csajsp-chapters.corewebprogramming.com/Core-Servlets-and-JSP.pdf

MpCon. *Instalación y configuración Tomcat y MySQL.* Recuperado el 6 Marzo 2006, desde
http://mpcon.org/apacheguide/

MySQL AB. *Tutorial básico de MySQL*. Recuperado el 6 Marzo 2006, desde
http://www.mysql-hispano.org/page.php?id=6&pag=1

Universidad de las Palmas de Gran Canaria. *Tutorial de JavaScript*. Recuperado el 9 Marzo 2006, desde
http://www.ulpgc.es/otros/tutoriales/JavaScript/index.htm

Canales Roberto, (2003). *TagLibs y JSPs*. Recuperado el 9 Marzo 2006, desde
http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=taglibs

DesarrolloWeb.com "*CSS, hojas de estilos*". Recuperado el 16 Marzo 2006, desde
http://www.desarrolloweb.com/manuales/2/

Jakarta.apache.org. *The Apache Jakarta Tomcat 5 Servlet/JSP Container*. Recuperado el 25 Marzo 2006, desde
http://tomcat.apache.org/tomcat-5.0-doc/jndi-datasource-examples-howto.html

Canales Roberto, (2003). *Instalación de Tomcat 5 y Pool de Conexiones*. Recuperado el 25 Marzo 2006, desde
http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=tomcat5

Programacion.com. *J2EE.* Recuperado el 25 Marzo 2006, desde
http://www.programacion.com/java/articulos/J2EE/

Merelo J., *Programando con JSP*. Recuperado el 1 Abril 2006, desde http://kal-el.ugr.es/~jmerelo/JSP/

Acción Estudiantil, (2003). *Introducción a JSP*. Recuperado el 1 Abril 2006, desde http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSP.pdf

Canales Roberto, (2003). *Control de navegación en Servlets*. Recuperado el 1 Abril 2006, desde http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=servletsbasico

Queiruga Claudia, (2005). *Servlet Context*. Recuperado el 26 Abril 2006, desde http://www.linti.unlp.edu.ar/tiki-download_file.php?fileId=347.

Jakarta.apache.org. *POI-HSSF- Java API to Access Microsoft Excel format Files*. Recuperado el 10 Abril 2006, desde http://jakarta.apache.org/poi/hssf/index.html

Jakarta.apache.org. *Commons email*. Recuperado el 10 Abril 2006, desde http://jakarta.apache.org/commons/email/