HELPER. Aplicación de gestión de flota

Santiago Castillo Lozano
Ingeniería informática de gestión

José Juan Rodríguez

25/06/08

A Dolors, mi esposa, por todos estos años de apoyo incondicional.

Y a mi hijo Alejandro por las muchas horas de juego que le debo.

Resumen.

Helper es una aplicación de gestión de flota distribuida en internet y desarrollada dentro del área J2EE. La principal característica que justifica la elección de esta tecnología es la posibilidad de que cada cliente pueda acceder a la aplicación desde cualquier punto de acceso a la red.

J2EE facilita la aplicación de modelos multicapa donde podemos separar las diferentes interfaces para cada tipo de usuario, la lógica de negocio de nuestra aplicación y toda la base de datos. Las estructuras multicapa proporcionan enormes ventajas en reutilización de componentes, facilidad en los mantenimientos y mejoras de la aplicación.

Helper proporciona la gestión de la flota del sistema de emergencias de Cataluña que utiliza un modelo sanitario centralizado en cuanto a la recepción de las llamadas y a la logística pero con bases sanitarias distribuidas por todo el territorio catalán.

Cada base se compone de recursos humanos y materiales suficientes para dar cobertura sanitaria al territorio asignado. Es responsabilidad de cada base asignar los vehículos apropiados a cada demanda asistencial transmitida por la base central y a notificar las incidencias producidas en su flota adscrita a los servicios centrales de logística. Los servicios centrales de logística deberán gestionar todas las incidencias producidas en los vehículos de todas las bases del territorio catalán.

Helper proporciona diferentes interfaces para cada tipo de usuario y diferentes accesos a la lógica de negocio y datos en función del usuario en concreto.

Las necesidades descritas en el modelos sanitario del sistema de emergencias médicas de Cataluña y las características que ofrece la tecnología J2EE hace que esta sea la más indicada para el desarrollo de la aplicación.

Índice de contenidos.

Índice de figuras.

Capitulo 1.

1.1. Introducción.

El proyecto esta basado en un hipotético sistema de emergencias sanitarias que da cobertura a toda Cataluña mediante recursos y personal propio gestionado desde una central.

SEM (Sistema de emergencias médicas) dispone de varias bases distribuidas por todo el territorio catalán, cada base se compone de un determinado número de vehículos, de personal y de instalaciones suficientes para guardar la flota, descanso del personal, central de radiocomunicaciones, etc.…

Existe una base central en la que se encuentran la central de coordinación que está encargada de la recepción de llamadas, la logística que gestiona la flota de vehículos y las oficinas.

El personal que compone una base son: 1 médico, 2 enfermeros, 3 técnicos en transporte sanitario (TTS) y un operador de central.

La flota que posee cada base es de 4 ambulancias, una de ellas de reserva, y un vehículo ligero sin camilla pero equipado con material sanitario. En función de la asistencia sanitaria requerida se asignará un recurso u otro, siendo estas las posibles combinaciones:

- ◊ Medico + enfermero + TTS = Ambulancias Mike.
- ◊ Enfermero + TTS = Ambulancia Alfa.
- ◊ TTS + TTS = Ambulancia Tango.
- ◊ Médico + TTS = Vehículo Víctor.

El funcionamiento de SEM es simple, cuando un usuario avisa al teléfono de emergencias su llamada será atendida en el centro coordinador, como resultado de la llamada se determinará que base deberá realizar la asistencia y que recursos deberán intervenir, es decir, Mike, Alfa, Víctor o Tango. Esta información será transmitida al operador de la base que corresponda y será este quien active los recursos necesarios asignando los vehículos apropiados en función de la disponibilidad de los mismos.

Una vez realizada la asistencia sanitaria y trasladado el paciente al hospital, si corresponde, los vehículos regresan a su base en espera de una nueva asignación de servicio. Si durante la asistencia el TTS ha detectado alguna incidencia mecánica en el vehículo deberá notificarlo inmediatamente al operador de su base que avisará a logística que gestionará la incidencia.

Debido al modelo de atención sanitaria que ofrece SEM mediante bases distribuidas por todo el territorio catalán pero con un servicio de logística centralizado, se precisará de una herramienta capaz de gestionar la flota desde las diferentes bases y desde la propia logística central. Por este motivo se ha pensado en una aplicación distribuida en internet.

Helper, como así será llamada la aplicación de gestión de flota, constará de dos subsistemas:

- ◊ Subsistema gestión de asignaciones: Es la parte de Helper que controlará y registrará cualquier uso de un vehículo. El operador de cualquier base podrá asignar alguno o varios de sus vehículos a un servicio concreto. Todo uso de vehículo quedará registrado y podrá ser consultado desde logística.

- ◊ Subsistema gestión de incidencias: Es la parte de Helper que gestionará las incidencias mecánicas de la flota. El operador de cualquier base podrá abrir una incidencia o consultar las de su propia base mientras que logística se encargará de la reparación de las averías pudiendo dar de baja y alta vehículos, cerrar incidencias o realizar consultas.

Podría considerarse un tercer subsistema de gestión de usuarios que se encargaría del alta o baja de los operadores, las bases y los usuarios mecánicos. Este subsistema quedará fuera del ámbito del proyecto y se considerará que la gestión de usuarios se realizará directamente sobre la base de datos, aunque la aplicación deberá distinguir entre operadores de distintas bases y entre el servicio central de logística para ofrecer a cada usuario las funcionalidades adecuadas y los resultados específicos en las consultas de cada base.

1.2. Justificación.

La empresa actual es una empresa extremadamente dinámica, en constante crecimiento, modificación de sus políticas y con una enorme movilidad geográfica. Esto hace que se produzca una elevada demanda de profesionales con capacidad de diseñar aplicaciones que satisfagan las necesidades de este tipo de empresas. La tecnología J2EE sin duda es una de las más indicadas.

J2EE permite la constante modificación y ampliación de la aplicación gracias a la construcción de software mediante capas en la que podemos cambiar partes de la aplicación sin afectar al resto, además se dan ventajas de reutilización.

La distribución de la aplicación en internet nos proporciona ventajas de accesibilidad a la lógica de la empresa desde cualquier acceso a la red. Así como la posibilidad de mantener los datos de la empresa en servidores dedicados a ello fuera de la empresa por lo que esta no deberá hacer grandes inversiones en tecnología y protección de datos.

Otra de las características que justifica el uso de esta tecnología es la posibilidad de llevar la tienda a casa del cliente y que además todas las acciones del cliente son automatizadas inmediatamente en la lógica de empresa, con todas las ventajas en cuanto a comercio que ello significa.

Estos son los rasgos más importantes que justifican la elección del área J2EE, aunque el punto de partida se limitaba a los conocimientos adquiridos en la plataforma estándar de Java, ingeniería del software y bases de datos adquiridos en las diferentes asignaturas cursadas.

Sin duda el TFC aporta enormes conocimiento sobre la plataforma J2EE y sobre la compatibilidad en el uso de otras tecnologías lo que abre muchísimas posibilidades para el desarrollo de aplicaciones muy completas.

1.3. Objetivos.

Objetivos principales:

- ◊ Desarrollar un proyecto informático mediante las fases de plan de trabajo, análisis y diseño, implementación y memoria.
- ◊ Aprender desde cero y ser capaz de llevar a cabo el proyecto mediante el uso de la tecnología J2EE.

Objetivos específicos:

- ◊ Aprender a instalar y desplegar la aplicación en el servidor Tomcat.
- ◊ Aprender y usar el framework Strut.
- ◊ Aprender y usar los patrones de diseño MVC, composición de vistas y DAO.
- ◊ Aprender y usar las páginas JSP.

1.4. Enfoque y método seguido.

El desarrollo del proyecto se hará siguiendo el clico de vida en cascada del software sin profundizar en la fase de testing y obviando el mantenimiento.

Debido a que el punto de partida en cuanto a conocimientos previos sobre J2EE es cero el estudio de la tecnología necesaria para el desarrollo del proyecto se llevará de manera paralela durante cada fase del proyecto.

1.5. Planificación del proyecto.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Estudio de J2EE	82 días	jue 28/02/08	lun 19/05/08
2	**FASE1. Plan de trabajo**	**16 días**	**jue 28/02/08**	**vie 14/03/08**
3	Introducción	3 días	jue 28/02/08	sáb 01/03/08
4	Descripción del proyecto y objetivos	4 días	dom 02/03/08	mié 05/03/08
5	Requerimientos funcionales y técnicos	4 días	jue 06/03/08	dom 09/03/08
6	Planificación temporal	5 días	lun 10/03/08	vie 14/03/08
7	PAC1. Plan de trabajo	0 días	vie 14/03/08	vie 14/03/08
8	**FASE2. Analisis y diseño**	**25 días**	**vie 21/03/08**	**lun 14/04/08**
9	Analisi requerimientos y subsistemas	3 días	vie 21/03/08	dom 23/03/08
10	Casos de uso	5 días	lun 24/03/08	vie 28/03/08
11	Descripción tecnologica	4 días	sáb 29/03/08	mar 01/04/08
12	Arquitectura de la aplicación	3 días	mié 02/04/08	vie 04/04/08
13	Diseño de la persistencia	2 días	sáb 05/04/08	dom 06/04/08
14	Diseño de la lógica de negocio	3 días	lun 07/04/08	mié 09/04/08
15	Diseño de la interface de usuario	5 días	jue 10/04/08	lun 14/04/08
16	PAC2. Analisis y diseño	0 días	lun 14/04/08	lun 14/04/08
17	**FASE3. Implementación.**	**35 días**	**mar 15/04/08**	**lun 19/05/08**
18	Configuración punto de trabajo	2 días	mar 15/04/08	mié 16/04/08
19	Implementar base de datos	3 días	jue 17/04/08	sáb 19/04/08
20	Implementar lógica de negocio	10 días	dom 20/04/08	mar 29/04/08
21	Implementar interfaces de usuario	14 días	mié 30/04/08	mar 13/05/08
22	Integrar según patrón MVC	3 días	mié 14/05/08	vie 16/05/08
24	Pruebas y puesta en marcha	2 días	dom 18/05/08	lun 19/05/08
25	PAC3. Aplicación	0 días	lun 19/05/08	lun 19/05/08
26	**FASE4. Memoria**	**37 días**	**mar 20/05/08**	**mié 25/06/08**
27	Documento memoria	22 días	mar 20/05/08	mar 10/06/08
28	Presentación	15 días	mié 11/06/08	mié 25/06/08
29	PAC4. Memoria y presentación	0 días	mié 25/06/08	mié 25/06/08

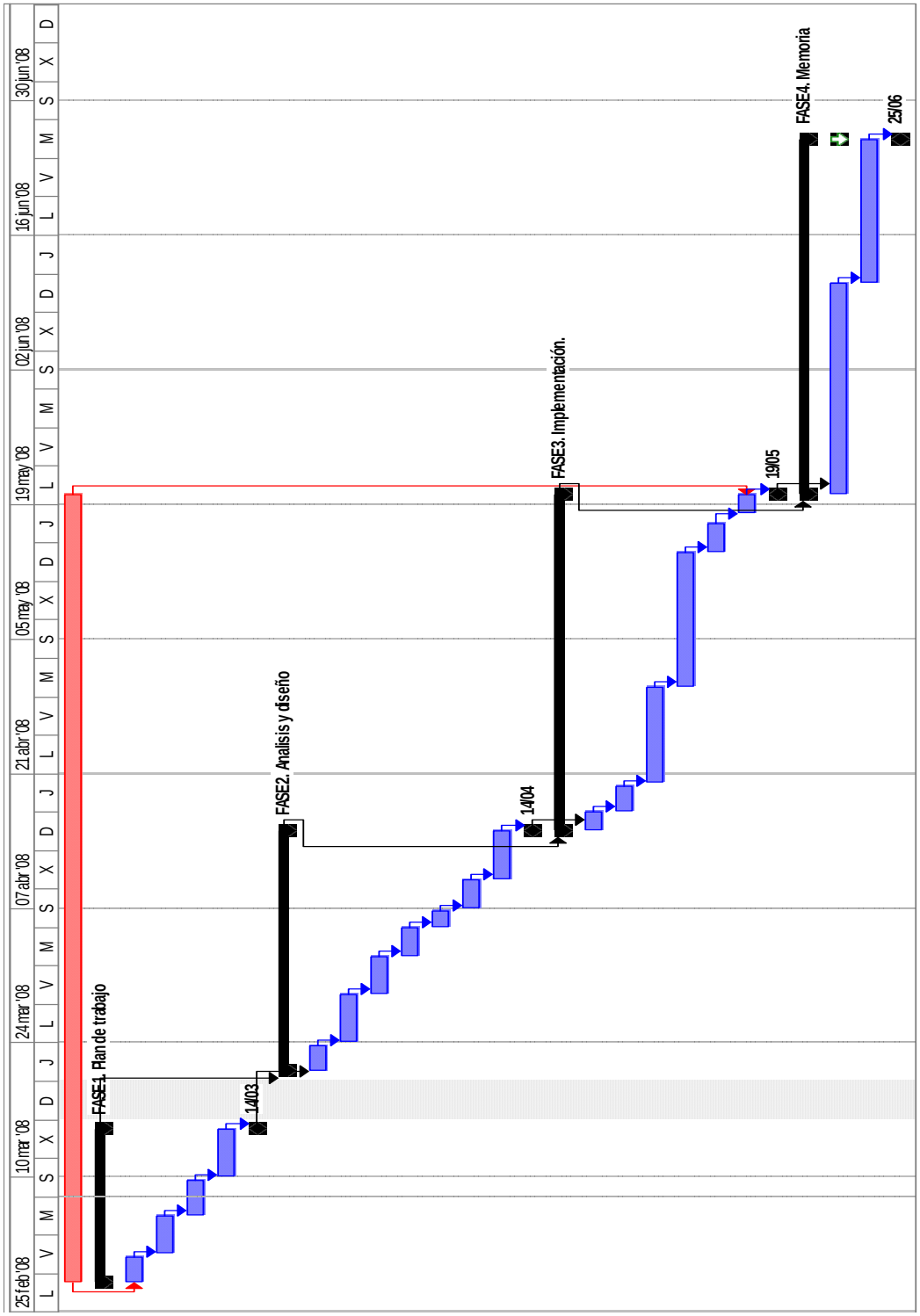


Figura 1. Diagrama de Grantt.

1.6. Productos obtenidos.

Título	
PAC1. Plan de trabajo	Documento donde se planificó todo el proyecto y se entregó como PAC1.
PAC2. Análisis y diseño	Documento en el que se desarrollo en análisis de la aplicación y se diseñó, fue entregado como PAC2.
PAC3. Aplicación	Archivo .WAR y código de la totalidad de la aplicación. Versión 1 sujeta a modificaciones. Archivo sql para la base de datos que contiene las tablas y algunos insert necesarios. Manual del usuario donde se describen la totalidad de las funcionalidades de la aplicación y se detalla en proceso de instalación. Fue entregado en la PAC3.
PAC4. Memoria y presentación	Documento de memoria. Documento de presentación de la aplicación. Archivo .WAR y código de la totalidad de la aplicación en su versión última con las correcciones indicadas durante la fase anterior. Archivo sql de la base de datos con las tablas y los insert necesarios.

1.7. Descripción de otros capítulos de la memoria.

Título	
Análisis	Capitulo 2. En el que se describen los requerimientos funcionales de la aplicación, la composición en subsistemas y los casos de uso.
Diseño	Capitulo 3. Se inicia con una descripción de la tecnología a usar, se diseña la persistencia, las interfaces del usuario y la arquitectura de la aplicación.
Implementación	Capitulo 4. Diagrama del funcionamiento de la arquitectura, decisiones de diseño e implementación.
Conclusiones	Capitulo 5. Conclusiones extraídas del proceso de desarrollo del TFC.

Capitulo 2.

2.1. Análisis de requerimientos.

Login: La primera pantalla que visualizará cualquier usuario será la de entrada al sistema, en ella se pedirá que se introduzca nombre de usuario y contraseña, los datos serán contrastados con la información almacenada. Si el login es incorrecto se mostrará un mensaje de error y se volverá a facilitar la posibilidad de loginarse, si el login es correcto se dará acceso a la pantalla principal de la aplicación según el rol que tenga asignado el usuario.

Existen dos tipos de usuarios con acceso a funcionalidades distintas, por un lado esta el usuario logística y por otro el operador de cada base. Cuando un usuario se logina, el sistema verifica la información y comprueba que rol tiene asignado el usuario, en caso de ser operador también mira a que base pertenece.

El alta, baja y modificación de los usuarios registrados en el sistema no será objeto de este aplicativo por lo que se considerará que se efectúan de manera directa sobre la base de datos por el administrador del sistema.

Deslogin: Cualquier tipo de usuario y desde cualquier pantalla de la aplicación dispone de la opción salir, en tal caso se volverá a la pantalla que ofrece loginarse. También será posible regresar a la pantalla principal desde cualquier otra pantalla.

Asignar vehículo a servicio: El operador de cada base recibe una petición de vehículo para un número de asistencia concreta, mediante el aplicativo de central de coordinación que está fuera del ámbito de este proyecto, entonces deberá asignar un vehículo al servicio solicitado. El operador accederá a la pantalla asignaciones donde podrá introducir el número de asistencia y el indicativo del vehículo asignado, el sistema deberá avisar de los siguientes casos: El vehículo ya tiene un servicio asignado, el vehículo esta dado de baja por logística o el vehículo indicado no pertenece a la base del operador por tanto no puede hacer uso de él. En caso de error se muestra un mensaje y se regresa a la pantalla principal.
Si la asignación de vehículo ha tenido existo se registrará la información en la base de datos y se regresará a la pantalla principal del operador.

Un mismo servicio puede tener varios vehículos asignados ya que pueden existir varios heridos en un mismo accidente o la complejidad del servicio requiere de ayuda.

Cerrar asignación de vehículo a servicio: El operador de cada base dispone de la opción cerrar asignación, al ser utilizada mostrará por pantalla todas las asignaciones que

tenga el operador abiertas pudiendo marcar para ser cerradas. Una vez efectuada la operación se actualizarán los datos y se regresará a la pantalla principal.

Consulta asignaciones de vehículos a servicio: El operador de cada base dispone de la funcionalidad consulta de vehículos asignados, al acceder se preguntará la fecha y se mostrará por pantalla todas las asignaciones que se han producido en ese día para la base a la que esta adscrita el operador. Desde dicha pantalla se podrá regresar a la principal otra vez. Esta información no puede ser manipulada y tendrá la consideración de hoja de servicios de la base.

Alta de incidencia: El operador de cada base puede informar de una incidencia producida en alguno de los vehículos asignados a su base, para ello seleccionará la opción de menú incidencias y rellenará la información, vehículo y descripción. Si el vehículo no forma parte de la base del operador el sistema avisará de ello y si no se produce ningún error la incidencia será registrada en el sistema con un número de referencia y con el estado pendiente. Se volverá a la pantalla principal.

Listado de incidencias pendientes: Cada operador de base puede listar las incidencias que tiene pendientes de resolución, para ello clicará la opción que corresponde y la información será mostrada en pantalla. Esta información no puede ser manipulada por el operador. Se volverá a la pantalla principal.

Operatividad recursos: El usuario logística tiene potestad para retirar del servicio cualquier vehículo de cualquier base, ya sea por incidencia notificada, mantenimiento preventivo o cualquier otra opción, además no todas las incidencias notificadas por cualquier operador deberán concluir con la retirada del servicio del recurso, es por este motivo es por lo que existe una funcionalidad especifica e independiente para este fin.

Al clicar sobre esta opción se abrirá una pantalla donde el logístico podrá indicar vehículo y elegir entre estado operativo o no operativo, la información será actualizada en la base de datos. Esta opción puede realizarse en cualquier momento incluso si el vehículo se encuentra asignado a algún servicio, la repercusión que este supuesto tendría es que el operador no podrá asignarlo al siguiente servicio una vez finalizado el que esta en curso. Se volverá a la pantalla principal.

Baja de incidencia: El usuario logística puede dar de baja cualquier incidencia de cualquier vehículo una vez que dicha incidencia ha sido reparada o se considera nula, para ello accederá a la opción de menú que corresponde y visualizará todas las incidencias pendientes, pudiendo cambiar el estado de reparada de esta manera la incidencia se considera dada de baja. Se actualizará la información y se regresará a la pantalla principal.

Consulta de incidencia: El usuario logística puede acceder a los datos de cualquier incidencia mediante una opción de menú que le solicitará el número de incidencia, una vez introducido mostrará la información que no podrá ser modificada. Se regresará a la pantalla principal.

Listar incidencias pendientes: El usuario logística puede listar todas las incidencias que tenga pendientes de resolución accediendo a la opción del menú que corresponde, se mostrará por pantalla, esta información tiene la consideración de tareas a realizar por el servicio mecánico de logística. Se regresará a la pantalla principal.

Listar incidencias por vehículo: Otra útil opción que requiere el usuario de logística es la posibilidad de listar todas las incidencias tanto pendientes como resueltas de un vehículo concreto, esta información tiene la consideración de histórico de un vehículo. Al acceder a esta opción el sistema preguntará que vehículo, si el indicativo no pertenece a la flota el sistema avisará de ello, en caso contrario se mostrará la información por pantalla y se regresará a la pantalla principal.

Los casos en los que se deba contemplar las opciones de incorporar nuevos vehículos o asignarlos a nuevas bases quedan fuera del ámbito de este proyecto por lo que se considerará que son opciones realizadas por el administrador del sistema directamente sobre la base de datos.

2.2. Composición subsistemas.

Helper, como así será llamada la aplicación, consta de 2 subsistemas. Por un lado el subsistema asignaciones está destinado a gestionar y controlar las asignaciones de vehículos que se hacen a cada servicio. Por otro lado el subsistema incidencias está destinado a la gestión de las incidencias producidas en la flota.

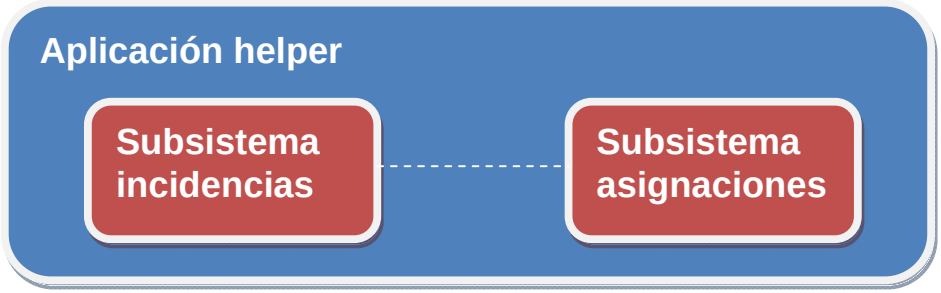


Figura 2. Diagrama de composición subsistemas.

2.3. Casos de uso.

2.3.1. Casos.

Login	
Funcionalidad general	Permitir acceso a la aplicación.
Actor	Logística, operador.
Casos de uso relacionados	Deslogin.
Pre condición	Ninguna.
Flujo normal	1. El usuario introduce nombre usuario y contraseña. 2- Pulsa aceptar. 2. Sistema verifica datos. a. Si los datos son correctos se mostrará la pantalla principal que corresponde a cada usuario.
Flujo alternativo	b. Si los datos no son correctos se muestra un mensaje y se vuelve a ofrecer al usuario la posibilidad de introducir de nuevo sus datos.
Pos condición	El sistema muestra la pantalla principal al usuario que corresponde.
Deslogin	
Funcionalidad general	Salir de la aplicación.
Actor	Logística, operador.
Casos de uso relacionados	Login
Pre condición	El usuario se encuentra dentro del sistema correctamente identificado.
Flujo normal	1. El usuario pulsa la opción deslogin. 2. El sistema muestra la pantalla de login.
Flujo alternativo	
Pos condición	El usuario se encuentra fuera de la aplicación.
Asignar vehículo	
Funcionalidad general	Se asigna un vehículo a una asistencia.
Actor	Operador.
Casos de uso relacionados	Relativos a asignaciones de vehículos.
Pre condición	El operador esta correctamente loginado y se ha recibido una petición de asistencia mediante otro aplicativo.
Flujo normal	1. El operador pulsa asignación de vehículo. 2. El sistema muestra una pantalla donde se

	solicita número de servicio e indicativo de vehículo. 3. El operador rellena datos y pulsa aceptar. 4. El sistema valida datos. a. Si los datos son validos el sistema registra uso del vehículo. b. El sistema informa del registro. 5. El operador acepta. 6. El sistema regresa a la pantalla principal.
Flujo alternativo	a. Si el sistema no valida datos por causas: vehículo ya estaba asignado a otro servicio, el vehículo que se pretende asignar pertenece a otra base o el vehículo está en situación de no operatividad, se mostrará un mensaje de error.
Pos condición	La asignación del vehículo al servicio ha sido registrada en la base de datos.
Cerrar asignación de vehículo	
Funcionalidad general	Se cerrará la asignación de un vehículo a un servicio de manera que el vehículo vuelve a estar libre.
Actor	Operador.
Casos de uso relacionados	Relativos a asignaciones de vehículos.
Pre condición	El operador está correctamente loginado.
Flujo normal	1. El operador pulsa cerrar asignación de vehículo a servicio. 2. El sistema muestra por pantalla todas las asignaciones en curso para esa base, si no hay se indica en un mensaje. a. El operador marca las asignaciones que desea cerrar. b. El operador pulsa aceptar. c. El sistema actualiza los datos y lo indica en un mensaje. 3. El operador pulsa aceptar. 4. El sistema vuelve a la pantalla principal.
Flujo alternativo	a. Al aceptar sin marcar ninguna asignación el sistema lo indicará mediante un mensaje.
Pos condición	Las asignaciones marcadas han sido actualizadas como cerradas en la base de datos.

Consultar asignaciones	
Funcionalidad general	Muestra las asignaciones de vehículo a un servicio en una fecha concreta.
Actor	Operador.
Casos de uso relacionados	Relativos a asignaciones de vehículos.
Pre condición	El operador está correctamente loginado.
Flujo normal	1. El operador pulsa consulta de asignaciones. 2. El sistema pide la fecha. 3. El operador introduce la fecha y pulsa aceptar. a. El sistema valida formato fecha. 4. El sistema muestra las asignaciones que corresponden o un mensaje en caso de no haber. 5. El operador pulsa aceptar. 6. El sistema regresa a la pantalla principal.
Flujo alternativo	a. El sistema no valida formato fecha y muestra un mensaje por pantalla. b. El operador pulsa aceptar. c. El sistema vuelve a ofrecer para introducir fecha. d. El operador pulsa aceptar.
Pos condición	La información solicitada ha sido mostrada por pantalla
Alta incidencia	
Funcionalidad general	El operador notifica una incidencia en algún vehículo de su base.
Actor	Operador.
Casos de uso relacionados	Relativos a incidencias en vehículos.
Pre condición	El operador está correctamente loginado.
Flujo normal	1. El operador pulsa sobre alta incidencias. 2. El sistema muestra para introducir vehículo y descripción de la incidencia. a. El sistema valida vehículo. b. El sistema registra la incidencia y muestra mensaje. 3. El operador acepta. 4. El sistema regresa a la pantalla principal.
Flujo alternativo	a. El sistema no valida vehículo porque este no está adscrito a la base y muestra un mensaje. b. El operador acepta. c. El sistema vuelve a permitir introducir

	datos.
Pos condición	La incidencia ha sido registrada junto a un identificador automático y con estado pendiente.
Listar incidencias pendientes	
Funcionalidad general	Muestra las incidencias pendientes de resolución.
Actor	Operador.
Casos de uso relacionados	Relativos a incidencias en vehículos.
Pre condición	El operador está correctamente loginado.
Flujo normal	1. El operador pulsa listar incidencias pendientes. 2. El sistema muestra las incidencias pendientes para los vehículos de la base o un mensaje en caso de no haber ninguna. 3. El operador pulsa aceptar. 4. El sistema regresa a la pantalla principal.
Flujo alternativo	
Pos condición	La información ha sido mostrada por pantalla.
Operatividad recurso	
Funcionalidad general	Permite cambiar de estado un vehículo entre operativo y no operativo.
Actor	Logística.
Casos de uso relacionados	
Pre condición	El usuario logística está correctamente loginado.
Flujo normal	1. Logística pulsa operatividad de recurso. 2. El sistema permite introducir vehículo y estado. 3. Logística introduce datos y pulsa aceptar. a. El sistema valida el vehículo. b. El sistema actualiza los datos de vehículo y muestra un mensaje. 4. Logística pulsa aceptar. 5. El sistema regresa a la pantalla principal.
Flujo alternativo	a. El sistema no valida porque el vehículo no existe y muestra un mensaje. b. Logística acepta. c. El sistema permite volver a introducir datos.
Pos condición	Los datos sobre la operatividad del vehículo han sido actualizados.
Baja de incidencia	
Funcionalidad general	Permite cambiar el estado de una incidencia de pendiente a reparada simulando así una baja.

Actor	Logística.
Casos de uso relacionados	Relativos a incidencias en vehículos.
Pre condición	El usuario logística está correctamente loginado.
Flujo normal	1. El usuario logística pulsa sobre baja incidencia. 2. El sistema muestra todas las incidencias pendientes de todos los vehículos de cualquier base. a. Logística marca aquellas incidencias que desea dar de baja y pulsa aceptar. b. El sistema actualiza los datos y muestra un mensaje. 3. Logística acepta. 4. El sistema regresa a la pantalla principal.
Flujo alternativo	a. Al aceptar sin marcar ninguna incidencia el sistema mostrará un mensaje.
Pos condición	Los datos sobre el estado de la incidencia han sido actualizados.
Consultar incidencias	
Funcionalidad general	Permite visualizar los datos relativos a una incidencia concreta.
Actor	Logística.
Casos de uso relacionados	Relativos a incidencias en vehículos.
Pre condición	El usuario logística está correctamente loginado.
Flujo normal	1. El usuario logística pulsa consultar incidencia. 2. El sistema muestra pantalla para introducir número de identificador incidencia. 3. Logística introduce identificar y acepta. a. El sistema válida identificador. b. El sistema muestra la información relativa a la incidencia. 4. Logística acepta. 5. El sistema muestra la pantalla principal.
Flujo alternativo	a. El sistema no valida el identificador de la incidencia porque no existe y muestra un mensaje. b. Logística acepta. c. El sistema vuelve a mostrar para introducir identificador.
Pos condición	La información solicitada ha sido mostrada por pantalla.
Incidencias pendientes	
Funcionalidad general	Lista todas las incidencias pendientes.

Actor	Logística.
Casos de uso relacionados	Relativos a incidencias en vehículos.
Pre condición	El usuario logística está correctamente loginado.
Flujo normal	1. El usuario logística pulsa incidencias pendientes. 2. El sistema muestra las incidencias pendientes de resolución de todos los vehículos de todas las bases. 3. Logística acepta. 4. El sistema regresa a la pantalla principal.
Flujo alternativo	
Pos condición	La información solicitada ha sido mostrada por pantalla.
Incidencias por vehículo	
Funcionalidad general	Muestra las incidencias pendientes o resueltas de un vehículo concreto.
Actor	Logística.
Casos de uso relacionados	Relativos a incidencias en vehículos.
Pre condición	El usuario logística está correctamente loginado.
Flujo normal	1. El usuario logística pulsa incidencias por vehículo. 2. El sistema pide el vehículo. 3. Logística rellena dato y acepta. a. El sistema valida vehículo. b. El sistema muestra las incidencias del vehículo seleccionado. 4. Logística acepta. 5. El sistema regresa a la pantalla principal.
Flujo alternativo	a. El sistema no valida el vehículo porque no existe y muestra un mensaje. b. Logística acepta. c. El sistema vuelve a mostrar para introducir vehículo.
Pos condición	La información solicitada ha sido mostrada por pantalla.

2.3.2. Diagrama casos de uso.

Para reflejar la parte en común que tienen los casos cerrar asignación vehículo y consultar asignaciones base se crea el caso buscar asignaciones según criterio que tendrá relación de inclusión con ambos. No es posible establecer una relación de extensión entre cerrar asignación vehículo y consulta asignaciones ya que exactamente no realizan el mismo proceso, cerrar asignación lista por asignaciones de esa base que están abiertas y consulta

asignaciones lista por fecha. Así se opta por generalizar en buscar asignaciones según criterios.

Se da la misma situación con el caso buscar incidencias según criterios pudiendo reflejar esta parte común a los 4 casos relativos a incidencias y establecer con ellos relaciones de inclusión.

Sin embargo si hay una relación de extensión entre incidencias pendientes y baja de incidencia ya que en ambos casos pueden ser casos independientes con el mismo actor y baja incidencias reproduce exactamente incidencias pendientes cuando da de baja incidencias.

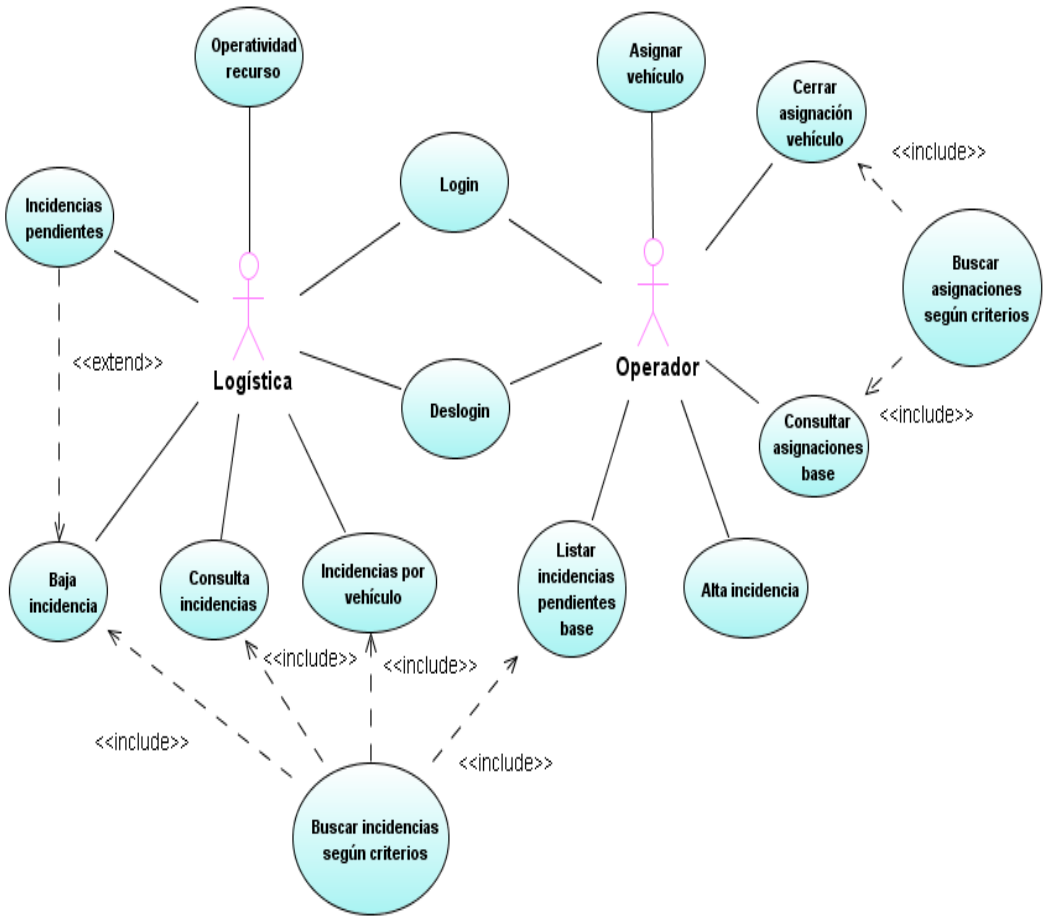


Figura 3. Diagrama de casos de uso.

Capitulo 3.

3.1. Descripción tecnológica.

3.1.1. La plataforma J2EE.

Conjunto de estándares para el desarrollo de aplicaciones multicapa basado en contenedores, componentes y servicios.

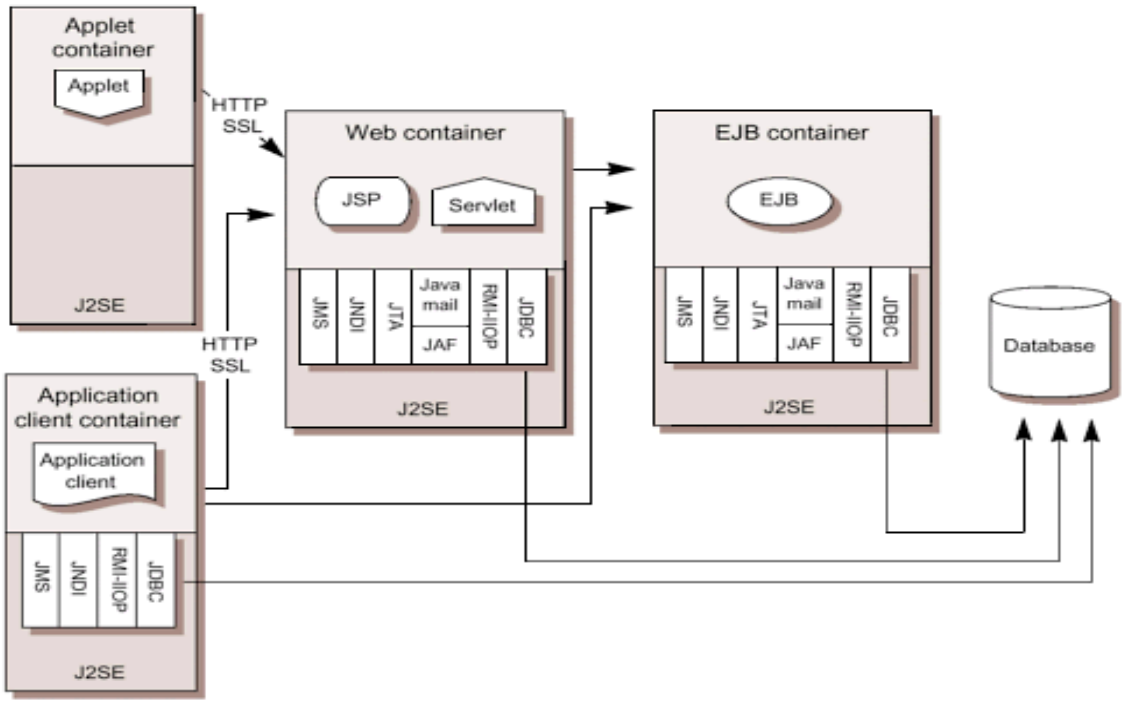


Figura 4. Plataforma J2EE.

El programador desarrolla los componentes de la aplicación, como por ejemplo Servlets y páginas JSP, junto con los descriptores de implementación, archivos XML que describen el componente.

Por otro lado, la plataforma aporta los contenedores que proporcionan un entorno de ejecución para los componentes, la implementación del conjunto de API de la especificación J2EE, APIs de servicios adicionales que pueda proporcionar el contenedor y los servicios declarativos como seguridad o transacciones.

La plataforma facilita el desarrollo de aplicaciones distribuidas, escalables, por capas e independientes de la plataforma.

3.1.2. Patrón DAO.

En la aplicación planteada será necesario el acceso a una base de datos para escribir, leer y modificar información, por lo que nos plantearemos el uso del lenguaje SQL en la lógica de negocio mediante el API JDBC.

Bajo este planteamiento podemos prever que ante un cambio de base de datos la lógica de negocio se vería afectada debido al alto grado de acoplamiento con el acceso a datos. Además el código resultante es más difícil de mantener provocando que la aplicación sea menos escalable.

Para solucionar este inconveniente aplicaremos el patrón DAO mediante el cual encapsularemos el acceso a base de datos en clases DAO donde residirá todo el código del API JDBC. Cuando la lógica de negocio necesite acceder a los datos deberá hacerlo mediante el objeto transfer que es donde DAO colocará los datos.

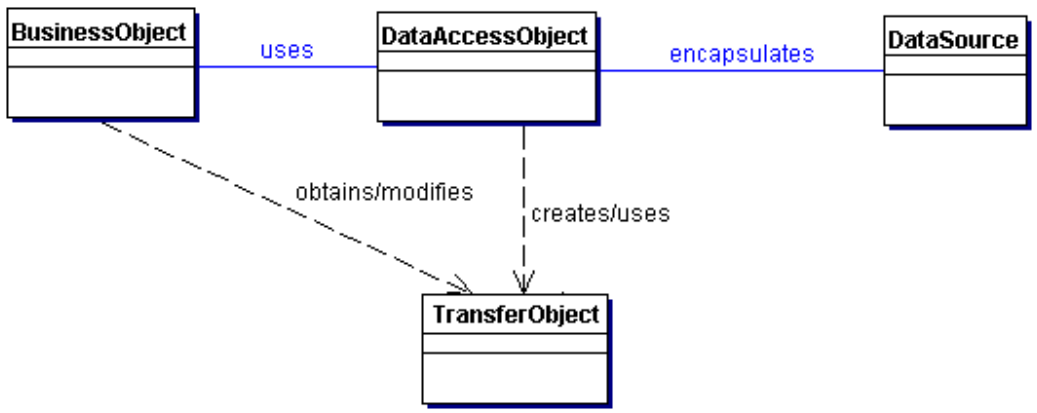


Figura 5. Patrón DAO.

Gracias al uso de este patrón obtenemos las siguientes características:

- ◊ La aplicación es más fácil de mantener y es más escalable debido a la separación del código de acceso a datos de la lógica de negocio.
- ◊ La aplicación podría adaptarse fácilmente a una migración de base se datos.
- ◊ El diseño e implementación de la aplicación ha subido en dificultad debido al incremento de clases y forma de manejar los datos.

3.1.3. Patrón composición de vistas.

En las especificaciones de funcionalidades y en los casos de uso se ha hecho referencia a que el sistema mostrará múltiples pantallas, algunas de ellas mantendrán algunos elementos siempre visibles como el encabezado o las opciones de menú. En estos casos sería de gran utilidad poder reutilizar las vistas en varias pantallas para ello aplicaremos el patrón composición de vistas.

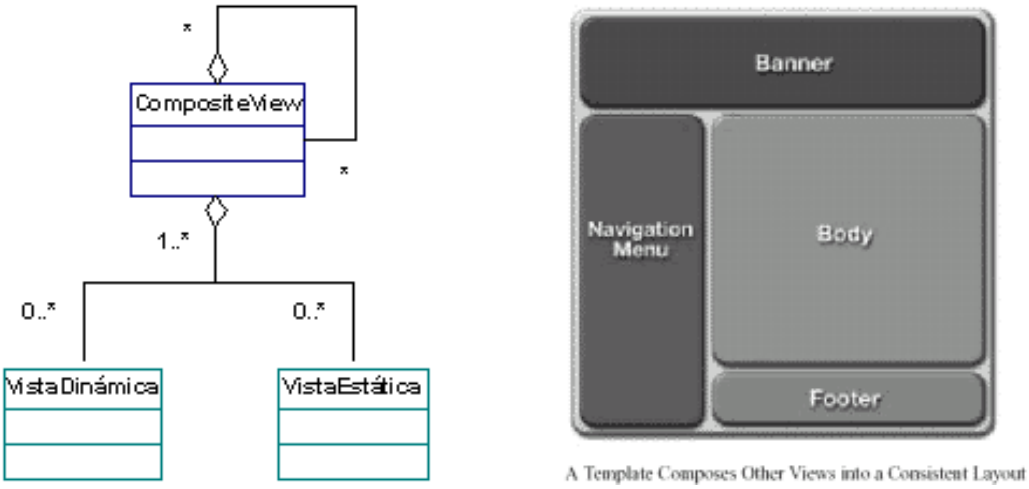


Figura 6. Patrón composición de vistas.

El patrón composición de vistas utiliza la etiqueta <jsp:include> mediante la cual podemos incluir tanto otras vistas dinámico como estáticas en cualquier vista que utilicemos.

Mediante la utilización del patrón composición de vistas obtenemos las siguientes características:

- ◊ Favorecemos la reutilización de vistas reduciendo la dificultad de diseño de las vistas y facilitando posibles cambios en algunos elementos de las vistas.

- ◊ Para hacer posible la aplicación del patrón será necesario que el diseño de las vistas de la aplicación se mantenga homogéneo.

3.1.4. Patrón MVC.

El patrón de diseño MVC divide la aplicación en tres capas:

- ◊ Capa controlador: Es la encargada de redirigir las peticiones que recibe a la lógica de negocio donde se ofrece la respuesta adecuada, que volverá al controlador que a su vez la servirá a la vista que corresponda. Para ello el controlador dispondrá de un mapa donde se le indicará que modelo responderá a cada petición y a que vista deberá servir la respuesta.

- ◊ Capa modelo: Es la lógica de negocio que genera la respuesta a las peticiones que el controlador le pasa. Una vez generada, la respuesta es devuelta al controlador.

- ◊ Capa vistas: Es la presentación de los datos que el controlador ha servido y que han sido generados por el modelo.

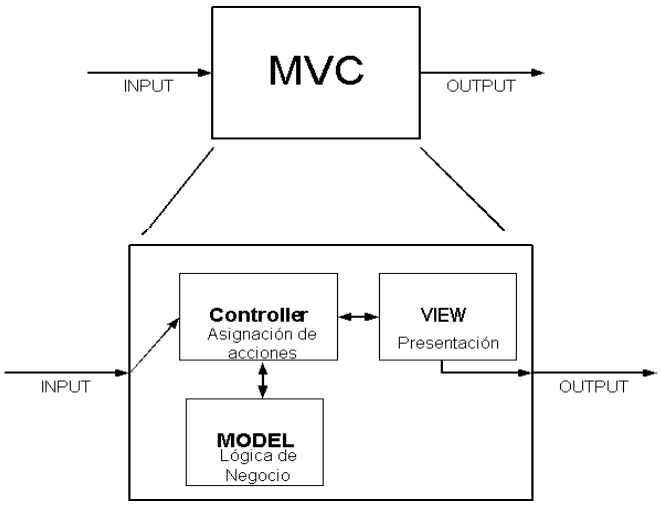


Figura 7. Patrón MVC.

Mediante la utilización del patrón MVC obtenemos las siguientes características:

- ◊ Separación efectiva de la capa de presentación y de la lógica de negocio lo que permitirá aplicar distintos cambios de presentación sin afectar al modelo.

- ◊ Dotar a la aplicación de robustez, facilidad de mantenimiento y reutilización.

3.1.5. Framework Struts.

Struts es un framework que facilita la aplicación del patrón de diseño MVC, para ello struts constará de tres componentes:

- ◊ Un Servlet controlador proporcionado por el propio Struts que se encarga de enrutar las peticiones hacia páginas JSP o clases Action según se halla especificado en el archivo de mapeo struts-config.xml. Las clases Action pueden responder al cliente o puede indicar a que control de deberá reenviar.

- ◊ El modelo se encarga de generar todas las clases JavaBeans necesarias para satisfacer los requerimientos de la aplicación. Aquí se desarrolla toda la lógica de negocio.

- ◊ La vista agrupa todas las páginas JSP necesarias para mostrar la información y para capturar la información de los formularios. Por cada formulario habrá un ActionForm con los mismos atributos que campos tenga el formulario para poder almacenar la información.

Un ejemplo de funcionamiento sería el siguiente:

- ◊ El controlador de Struts recibe una petición del cliente, consulta el archivo de mapeo y se enruta la petición hacia un Action.

- ◊ El Action actuará cargando un formulario, sobre este formulario se ha definido un ActionForm que se encargará de recoger los datos introducidos.

- ◊ Struts redirige a otro Action que se encargará de invocar la lógica de negocio que recuperará los datos del ActionForm y los trabaja generando un resultado.

- ◊ Por último struts cargará la JSP que mostrará el resultado.

3.1.6. CSS.

Las hojas de estilo en cascada se usan para determinar la presentación que tendrá un documento escrito en HTML o en XML. La idea principal del lenguaje CSS es poder separar la presentación del documento de su estructura.

Las ventajas principales derivadas del uso de la tecnología CSS son:

- ◊ Facilidad de actualización de páginas gracias a la centralización de la presentación.
- ◊ Una misma página puede tener varios estilos a escoger.
- ◊ Simplifica el código HTML de la página.

3.1.7. Java Scripts.

Lenguaje de programación que no requiere ser compilado, es interpretado. Se inserta dentro del código HTML de la página que el cliente se descarga y se ejecuta directamente.

La principal ventaja de este lenguaje es la facilidad con la que estamos ejecutando código directamente en el cliente. En este proyecto se ha usado en validaciones del lado cliente y en mejorar la presentación en algunas pantallas de la interface.

Capitulo 4.

4.1. Diseño de la persistencia.

4.1.1. Diagrama E/R.

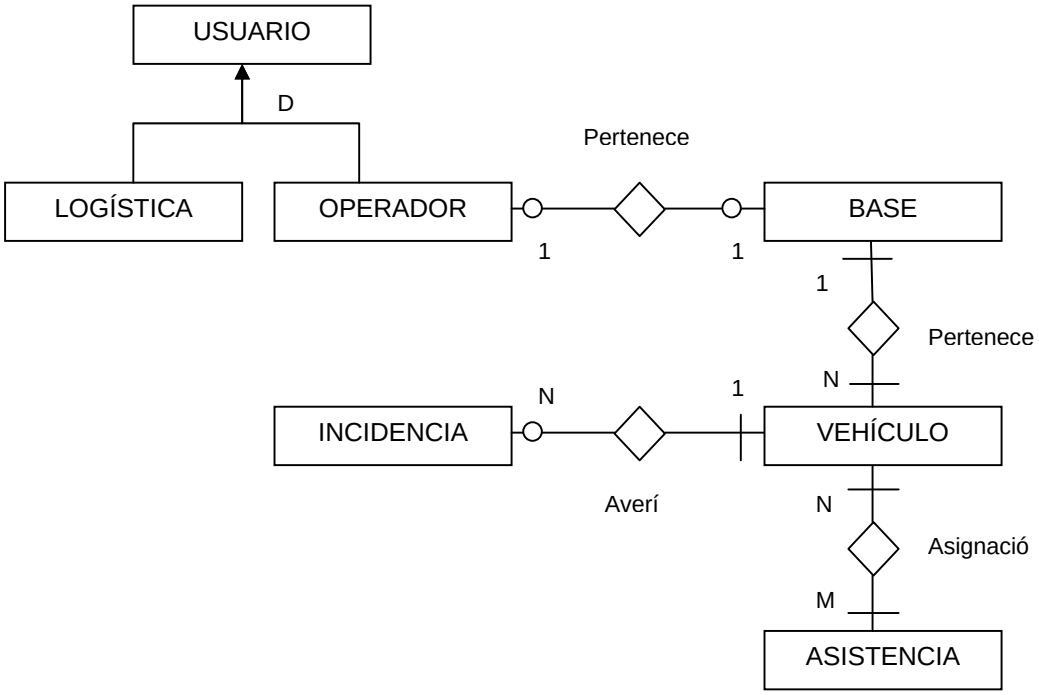


Figura 8. Diagrama E/R.

4.1.2. Tablas.

Tabla Usuario

Contiene el nombre y contraseña de los usuarios registrados en la aplicación, facilitará el acceso y el rol que cada usuario tenga asignado (logística o operador). Esta tabla se mantiene directamente por el administrador de la aplicación, es decir no existen funcionalidades de eliminación, modificación o escritura de datos desde Helper.

Nombre Campo	Tipo	Clave	Descripción	Null
Id_Usuario	Varchar(20)	Primaria	Identificador del usuario	NO
Clau	Varchar(20)		Contraseña de acceso	NO
Rol	Varchar(20)		Rol del usuario (logística o operador)	NO

Tabla logística.

Esta tabla contiene a los usuarios registrados con rol logística, aunque no será necesaria a la hora de implementar la aplicación se ha decidido mantenerla en el diseño para posibles ampliaciones de información sobre usuarios. Será mantenida por el administrador.

Nombre Campo	Tipo	Clave	Descripción	Null
Id_Logística	Varchar(20)	Primaria y foránea a Usuario(Id_Usuario)	Identificador del usuario logística	No

Tabla operador.

Esta tabla contiene a los usuarios registrados con el rol de operador, además proporcionará el código de la base a la que cada operador está adscrito. Mantenida por administrador.

Nombre Campo	Tipo	Clave	Descripción	Null
Id_Operador	Varchar(20)	Primaria y foránea a Usuario(Id_Usuario)	Identificador del usuario	NO
Id_Base	Integer	Foránea Base(Id_Base)	Identificador de base a la que pertenece el operador	NO

Tabla base.

Esta tabla contiene todas las bases que componen el sistema de emergencias sanitarias. Mantenida por administrador.

Nombre Campo	Tipo	Clave	Descripción	Null
Id_Base	Integer	Primaria	Identificador de la base	NO
Nombre	Varchar(50)		Nombre de la base	NO

Tabla vehículo.

Esta tabla contiene la totalidad de la flota de vehículos que posee el sistema de emergencias sanitarias. Cada vehículo está adscrito a una base, es de un tipo determinado y puede estar operativo o fuera de servicio. La tabla está mantenida por el administrador aunque la aplicación podrá cambiar el estado del vehículo.

Nombre Campo	Tipo	Clave	Descripción	Null
Indicativo	Varchar(20)	Primaria	Identifica al vehículo	NO
Id_Base	Integer	Foránea Base(Id_Base)	Identifica la base a la que pertenece el vehículo	NO
Tipo	Varchar(20)		Indica el tipo del vehículo (Alfa, Mike, Tango o Víctor)	NO
Estado	Varchar(20)		Indica si el vehículo está operativo o fuera de servicio	NO

Tabla incidencia.

Esta tabla contiene las incidencias que se producen en los vehículos registrando la fecha, descripción y permitiendo la gestión del estado. Será mantenida por la aplicación.

Nombre Campo	Tipo	Clave	Descripción	Null
Id_Incidencia	Integer	Primaria	Identificador de la incidencia	NO
Id_Vehiculo	Varchar(20)	Foránea Vehiculo (indicativo)	Identificador del vehículo que sufre la incidencia	NO
Fecha	Data		Fecha en la que se produce la incidencia	NO
Descripción	Varchar(200)		Descripción de la incidencia	NO
Estado	Varchar(20)		Indica en que estado de reparación se encuentra la incidencia (pendiente, en curso o reparada)	NO

Tabla asignación.

Esta tabla contiene todos los usos que se hacen del vehículo, cada vez que se solicita un vehículo para realizar una asistencia quedará registrado el uso con su correspondiente rango de tiempo. La tabla será mantenida por la aplicación.

Nombre Campo	Tipo	Clave	Descripción	Null
Id_Asistencia	Integer	Primaria	Identificador de la asistencia	NO
Id_Vehiculo	Varchar(20)	Primaria y foránea Vehículo(Indicativo)	Identificador del vehículo asignado a la asistencia	NO
Fecha inicio	Data		Fecha de inicio de la asignación	NO
Hora inicio	Data		Hora de inicio de la asignación	NO
Fecha final	Data		Fecha de finalización de la asignación	
Hora final	Data		Hora de finalización de la asignación	
Id_Asignación	Integer	Primaria	Identificador de la asignación	NO

4.2. Diseño interface de usuario.

Para el diseño de la interface se aplicará el patrón de diseño de composición de vistas de manera que se podrán definir cuatro partes bien diferenciadas en la pantalla.

Por un lado y constante a todas las pantallas aparecerá en el margen superior una cabecera institucional. Inmediatamente debajo y en forma de menú desplegable aparecerán las opciones que cada tipo de usuario dispone. De manera que el usuario logística dispondrá de sus opciones concretas y diferentes de las del usuario con rol operador.

En la parte central de la pantalla aparecerán los distintos formularios en función de las opciones elegidas y los resultados a las consultas de información que los usuarios requieran.

Por último, cada pantalla finaliza con un pie de página institucional y común a todas las pantallas.

Figura 9. Interface usuario logística.

Figura 10. Interface usuario operador.

4.3. Diseño arquitectura aplicación.

4.3.1. Clases Action.

Las clases action están destinadas a procesar solicitudes del navegador mediante su método perform(), como resultado del método obtenemos un ActionForward que identifica donde deberá dirigirse el control.

Será el fichero struts-config.xml quien designará la clase Action que deberá atender a una solicitud. Todas las clases action que vamos a definir heredarán del Action de Struts.

Será necesario definir una Action por cada solicitud lógica que pueda recibirse. No se contempla en el diagrama las Action para realizar la carga de la jsp a falta de valorar la posibilidad de usar la misma Action.

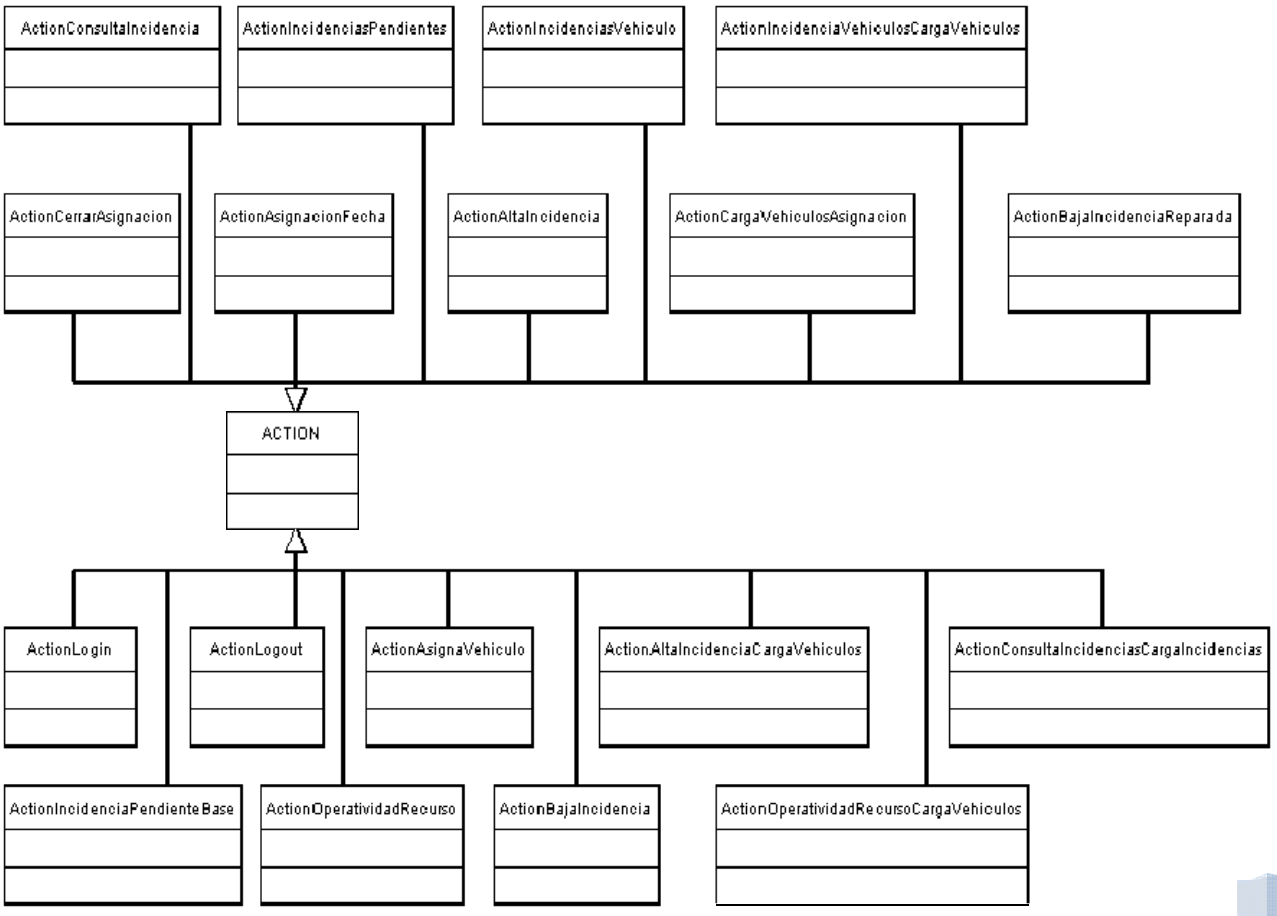


Figura 11. Diagrama clases Action.

4.3.2. Clases DynaActionForm.

Las clases ActionForm nos permiten el intercambio de datos entre el navegador y la lógica de negocio. Existirá un ActionForm por cada formulario que tengamos, los atributos del ActionForm serán los campos del formulario, de esta manera la lógica de negocio podrá acceder a los métodos get del ActionForm y extraer la información del usuario.

Todos los ActionForm que definiremos heredan del ActionForm de Struts.

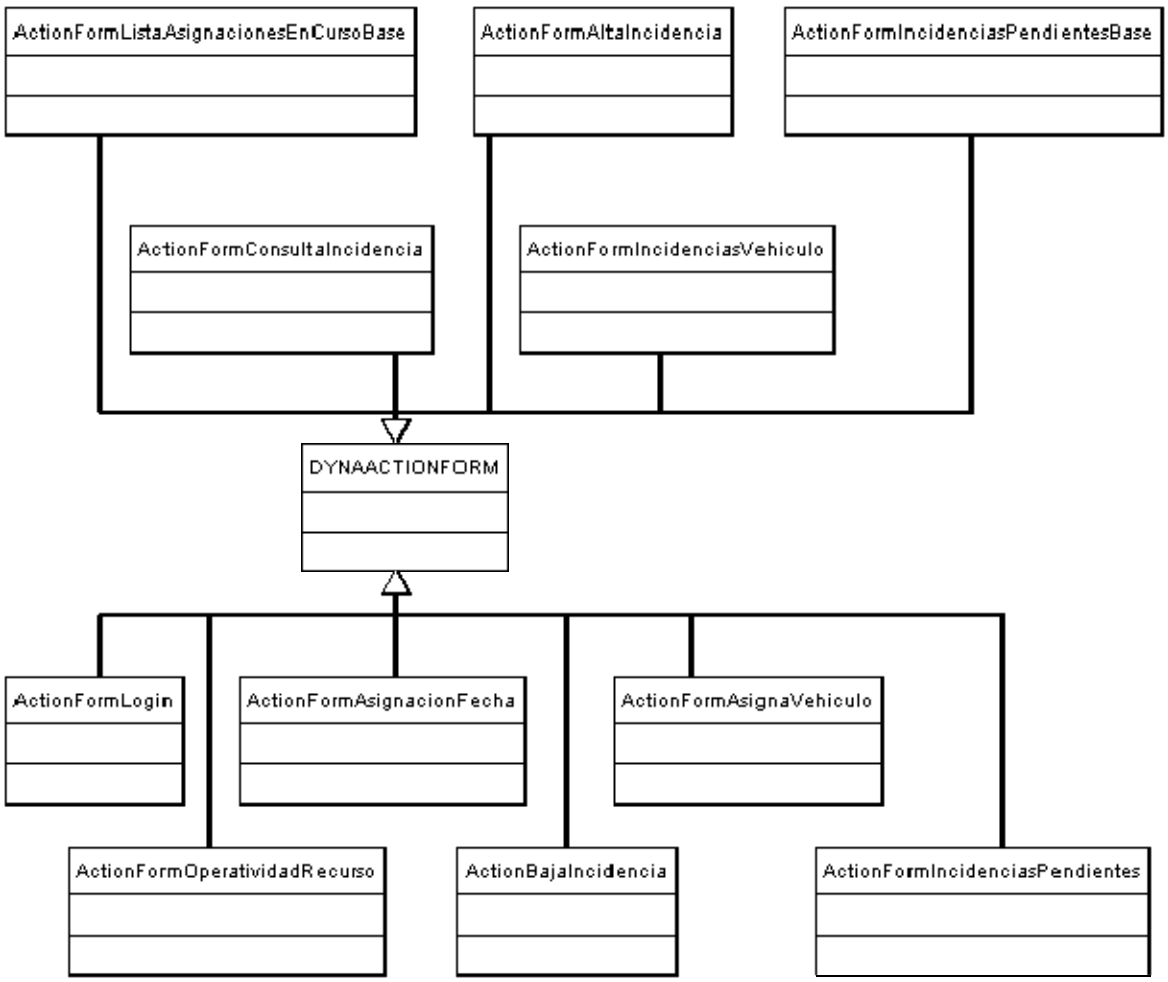


Figura 12. Diagrama clases DynaActionForm

4.3.3. Cajas UML clases DynaActionForm.

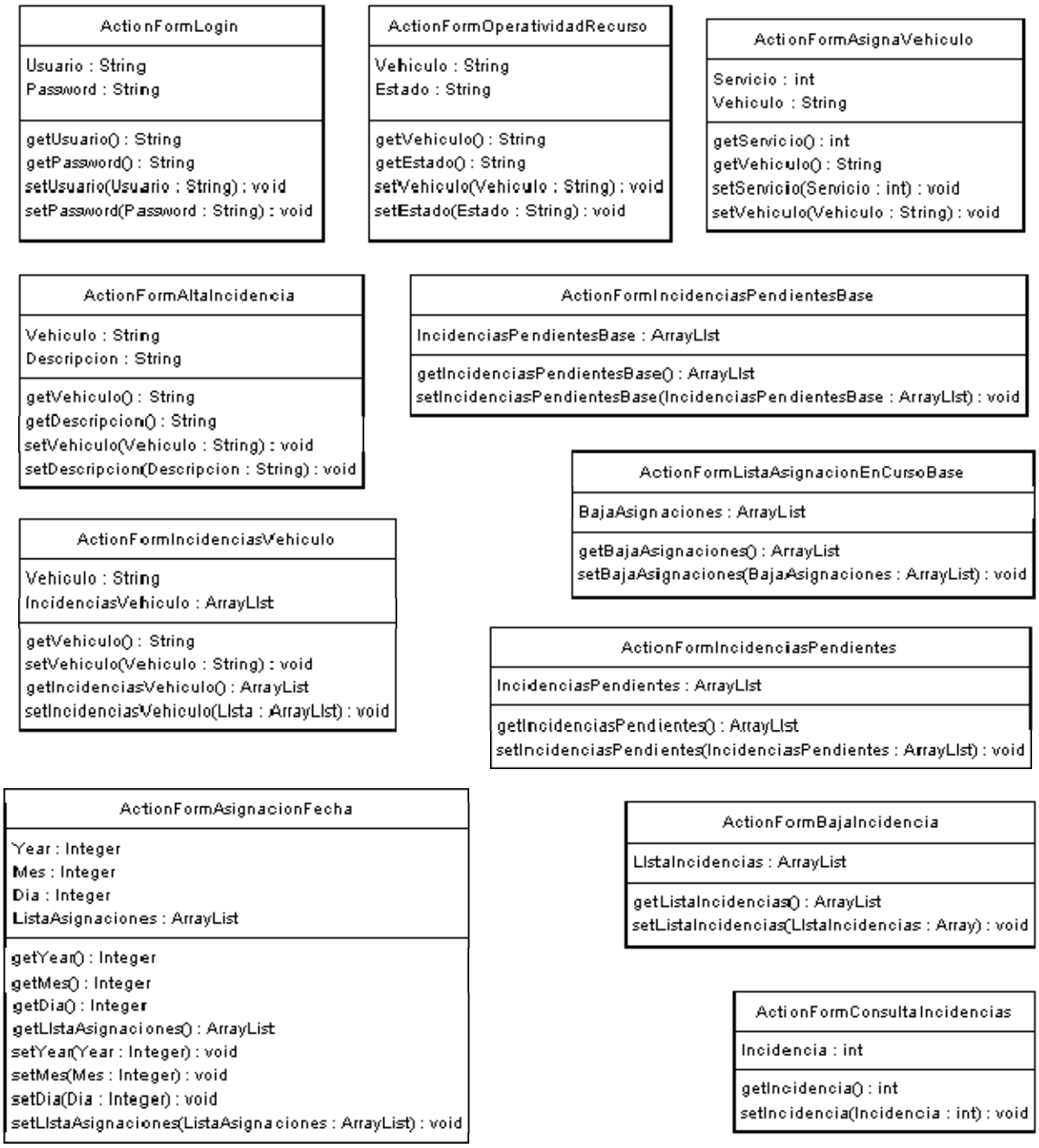


Figura 13. Cajas UML clases DynaActionForm

4.3.4. Clases de la lógica de negocio.

Las clases que componen la lógica de negocio se han diseñado a partir del patrón de diseño DAO por lo que tendremos unas clases beans que modelan las entidades y unas clases gestoras de los accesos a la base de datos, serán las llamadas DAOs. Todas las clases DAOs harán uso de una clase gestora que se ocupará de establecer y gestionar la conexión con la base de datos.

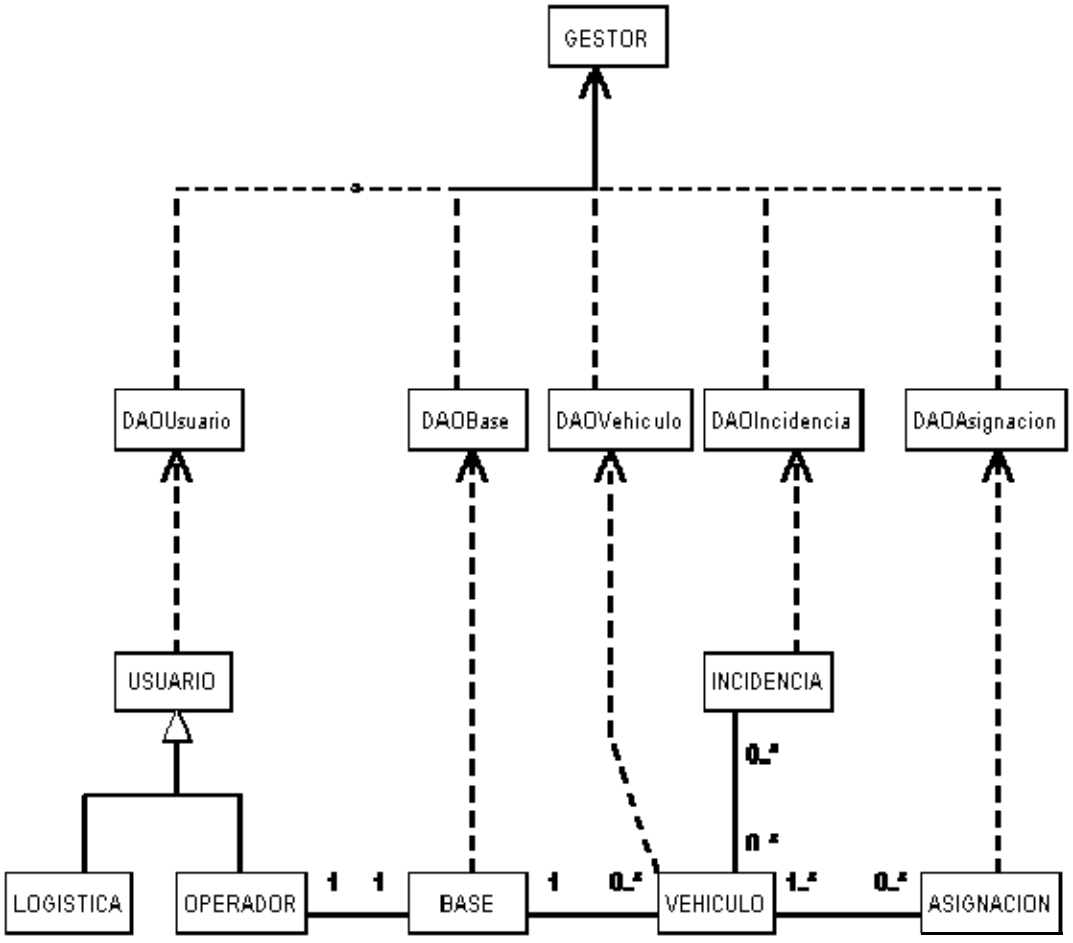


Figura 14. Diagrama clases de negocio.

4.3.5. Cajas UML Beans.

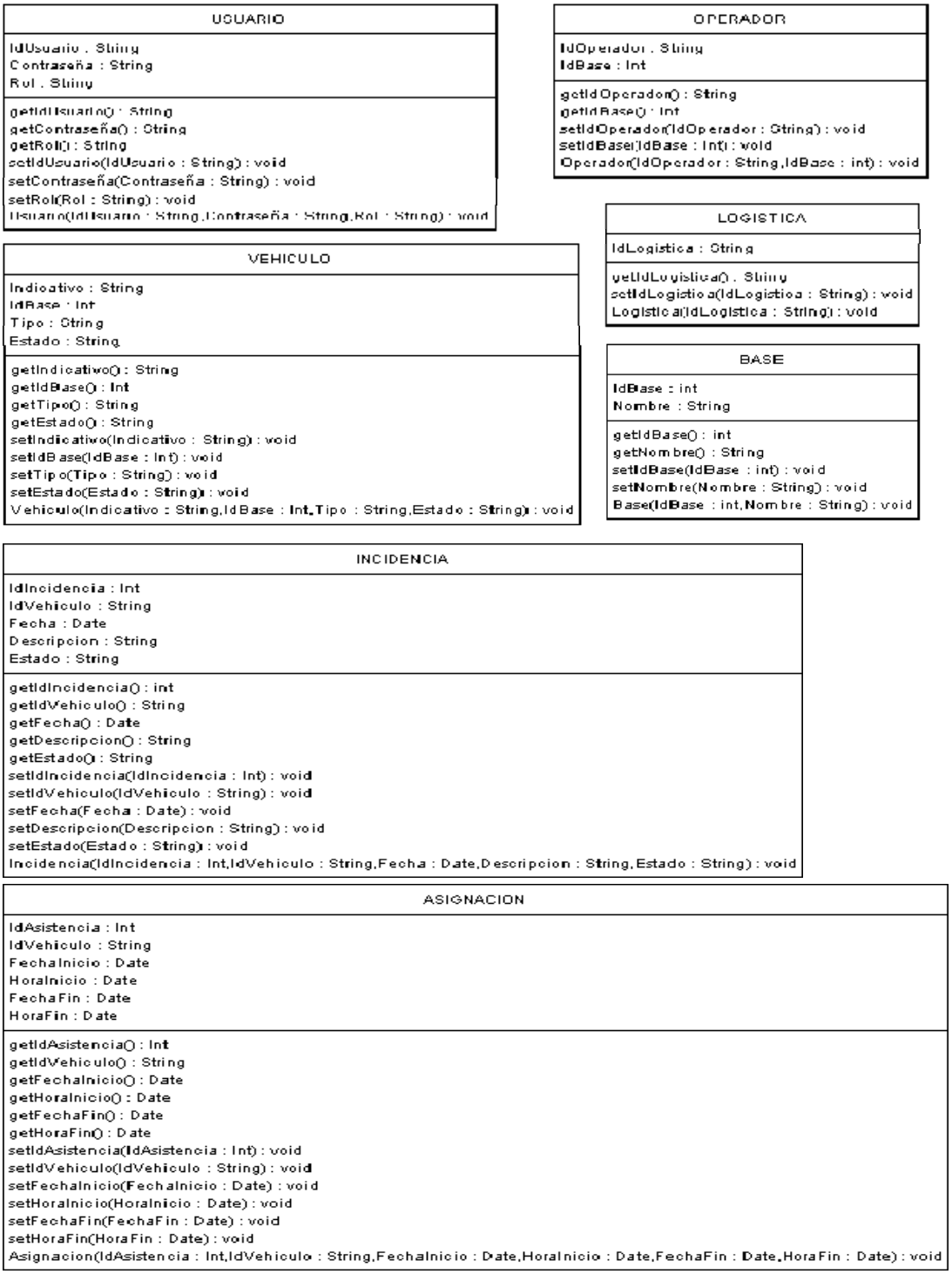


Figura 15. Cajas UML Beans.

4.3.6. Cajas UML DAOs.

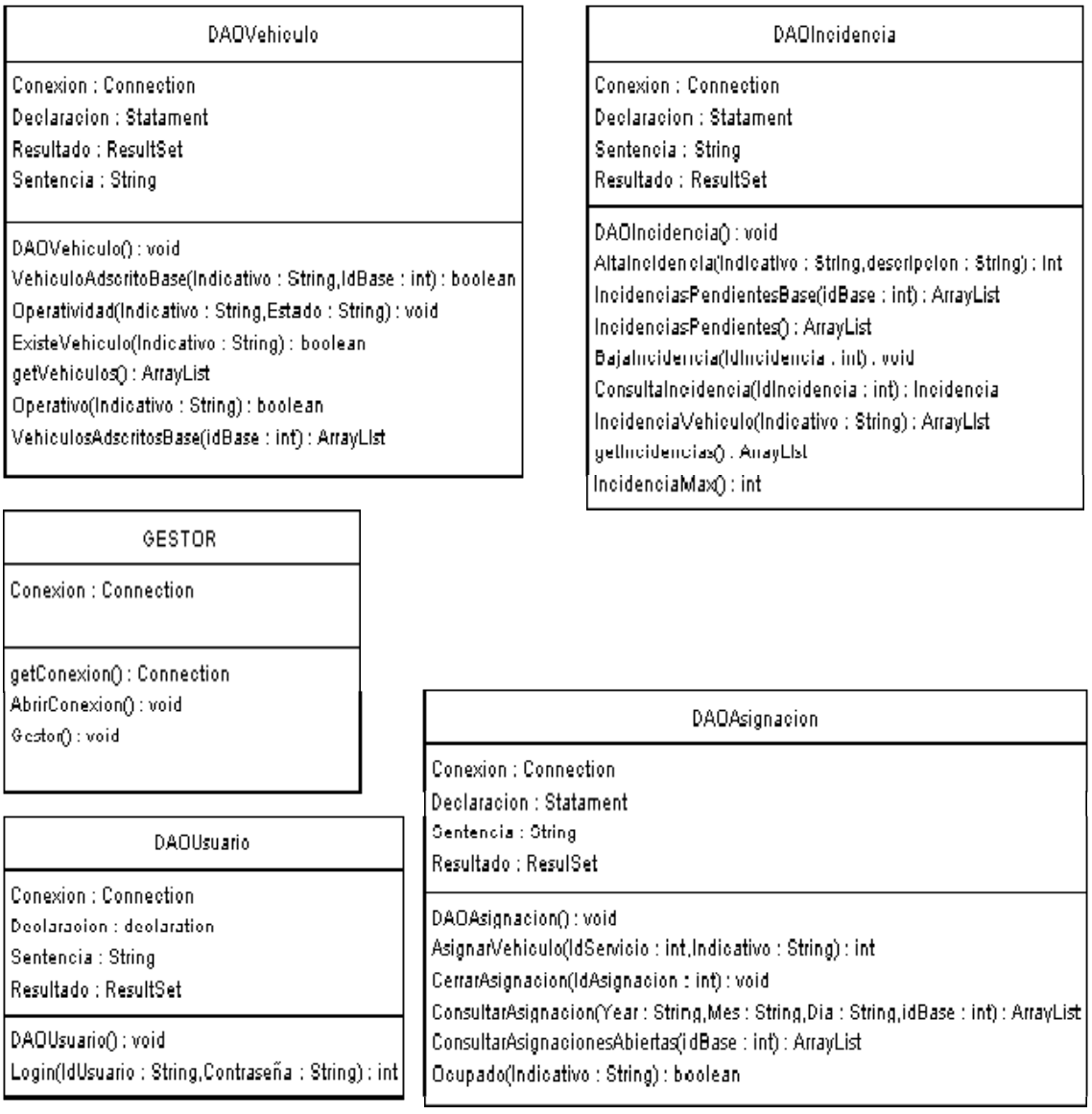


Figura 16. Cajas UML Daos.

4.3.7. Vistas JSP.

Vista	Descripción
VistaLogin.jsp	Formulario para loginarse en la aplicación.
VistaLogistica.jsp	Pantalla principal para el usuario logística una vez loginado correctamente.
VistaOperador.jsp	Pantalla principal para el usuario operador una vez loginado correctamente.
VistaLoginError.jsp	Mensaje de error si login no tiene éxito.
VistaAsignaVehiculo.jsp	Formulario para la asignación de vehículos.
VistaVehiculoAsignadoOK.jsp	Mensaje de confirmación de que el vehículo ha sido asignado a un servicio.
VistaVehiculoAsignadoError.jsp	Mensaje error si la asignación del vehículo no ha tenido existo. Diversas causas.
VistaListaAsignacionEnCursoBase.jsp	Listado de las asignaciones de vehículos de una base, permite marcar para cerrar asignaciones.
VistaListaAsignacionEnCursoBaseVacio.jsp	Mensaje que indica que la base no tiene vehículos asignados a servicios en este momento.
VistaListaAsignacionEnCursoBaseActualizado.jsp	Mensaje que indica que las asignaciones marcadas han sido cerradas.
VistaListaAsignacionBaseError.jsp	Mensaje de error al marcar asignaciones para cerrar. Diversas causas.
VistaAsignacionFecha.jsp	Formulario para indicar fecha de consulta de asignaciones de una base.
VistaAsignacionFechaError.jsp	Mensaje de error al hacer consulta de asignaciones por fecha en una base.
VistaAsignacionFechaVacia.jsp	Mensaje de no vehículos asignados como resultado de la consulta asignaciones por fecha base.
VistaAsignacionFechaDatos.jsp	Lista de vehículos resultantes de la consulta asignaciones por fecha base.
VistaAltaIncidencia.jsp	Formulario para dar de alta una incidencia en un vehículo.
VistaAltaIncidenciaVehiculoNoAdscrito.jsp	Indicará que el vehículo sobre el cual se quiere dar de alta una incidencia no está adscrito a la base del operador que realiza la acción.
VistaAltaIncidenciaOK.jsp	Mensaje que confirma el alta de la incidencia en un vehículo.
VistaAltaIncidenciaError.jsp	Mensaje que indica que se ha producido un error al dar de alta una incidencia de un vehículo.
VistaListaIncidenciasPendientesBase.jsp	Lista las incidencias de vehículos pendientes de resolución en una base.

VistaListaIncidenciasPendientesBaseVacio.jsp	Mensaje que indica que no hay incidencias de vehículos pendientes de resolución en una base
VistaListaIncidenciasPendientesBaseError.jsp	Mensaje que indica que se ha producido un error al realizar la consulta de incidencias pendientes en vehículos de una base.
VistaOperatividadRecurso.jsp	Formulario para cambiar la operatividad de un vehículo.
VistaOperatividadRecursoOK.jsp	Mensaje que confirma el cambio de operatividad de un recurso.
VistaOperatividadRecursoVehiculoInexistente.jsp	Mensaje que indica que se ha intentado efectuar un cambio de operatividad en un vehículo que no existe.
VistaOperatividadRecursoError.jsp	Mensaje que indica que ha ocurrido un error al intentar hacer cambio de operatividad en un vehículo.
VistaListaIncidenciasPendientesBaja.jsp	Lista las incidencias pendientes de resolución y permite marcarlas para darlas de baja.
VistaListaIncidenciasPendientesBajaOK.jsp	Mensaje que confirma que las incidencias pendientes de resolución marcadas han sido dadas de baja.
VistaListaIncidenciasPendientesBajaVacio.jsp	Mensaje que indica que no hay incidencias pendientes de resolución en ese momento.
VistaListaIncidenciasPendientesBajaError.jsp	Mensaje que indica que se ha producido un error al realizar la consulta de incidencias pendientes de resolución.
VistaConsultaIncidencia.jsp	Formulario para consultar incidencias a partir de su identificador.
VistaConsultaIncidenciaDatos.jsp	Muestra los datos relativos a la incidencia solicitada.
VistaConsultaIncidenciaNoExistente.jsp	Mensaje que indica que la incidencia solicitada no existe.
VistaConsultaIncidenciaError.jsp	Mensaje que indica que se ha producido un error al realizar la consulta de la incidencia solicitada.
VistaListaIncidenciasPendientes.jsp	Lista las incidencias pendientes de resolución.
VistaListaIncidenciasPendientesVacio.jsp	Mensaje que indica que no hay incidencias pendientes de resolución.
VistaListaIncidenciasPendientesError.jsp	Mensaje que indica que se ha producido un error al realizar la consulta de incidencias pendientes de resolución.
VistaListaIncidenciasVehiculo.jsp	Formulario que permite indicar el vehículo sobre el que se desea saber las incidencias.
VistaListaIncidenciasVehiculoDatos.jsp	Lista de incidencias del vehículo indicado.
VistaListaIncidenciasVehiculoNoExiste.jsp	Mensaje que indica que el vehículo indicado no existe.
VistaListaIncidenciasVehiculoVacio.jsp	Mensaje que indica que el vehículo indicado no tiene ninguna incidencia en el historial.
VistaListaIncidenciasVehiculoError.jsp	Mensaje que indica que se ha producido un error al

	realizar la consulta de incidencias por vehículo.
Cabecera.jsp	Cabecera institucional de la empresa
SubcabeceraLogistica.jsp	Menú del usuario con rol logística
SubcabeceraOperador.jsp	Menú del usuario con rol operador
VistaLogout.jsp	Salida de la aplicación
PiePagina.jsp	Pie de página institucional de la empresa

5. Capitulo 5.

5.1. Decisiones de diseño e implementación.

◊ Se ha decidido usar el patrón de diseño DAO por la ventaja que supone ante una posible migración de base de datos y poder obtener así una aplicación fácilmente adaptable a la base de datos que ofrezca el servidor contratado. Para apaliar el inconveniente que supone el uso del patrón DAO que hace incrementar la dificultad de la aplicación como consecuencia al incremento de clases y teniendo en cuenta que esta aplicación no tiene una lógica de negocio extremadamente complicada se experimentará con la posibilidad de asumir parte de la lógica entre el DAO y el Action que actué. Por lo que es posible que se sacrifique un poco de escalabilidad de la aplicación.

◊ Para que el usuario solo mantenga una conexión abierta con la base de datos durante su sesión y para seguir la filosofía de JAVA se ha encapsulado todo el código de conexión en la clase GESTOR que será usada por todos los DAOs para obtener la conexión y ejecutar las consultas sobre ella.

◊ Debido a que se ha identificado en el diseño de la interface del usuario partes de la pantalla que se mantienen idénticas a lo largo de la aplicación y que existirá la zona de menú que se adapta a cada tipo de usuario de ha optado por la aplicación del patrón de diseño de composición de vistas.

◊ Se decide aplicar el modelo MVC porque hoy en día no se concibe una aplicación que no pueda ser fácilmente actualizable, ampliable y modificable, en especial la parte de interface del usuario. Mediante el modelo MVC se ofrece un particular aislamiento de las vistas de la lógica de negocio. Aunque la dificultad de diseño e implementación se ve aumentada el esfuerzo queda compensado en la fase de mantenimiento de la aplicación.

◊ Para compensar la dificultad de aplicación del modelo de diseño MVC se ha pensado en aplicar un framework, después de valorar Spring se llega a la

conclusión de que quizás sea más completo que Struts al incluir la lógica de negocio y posiblemente más robusto gracias al uso de interfaces. Aún así se opta por Struts por estar mucho más difundido en el mercado y ser más sencillo de aplicar que precisamente era el objetivo del uso del framework en el modelo MVC.

◊ Aunque no se especifica en los diagramas por motivo de espacio y tras valorar la posibilidad de que el mismo Action que contiene el ActioForm sea el que inicialmente cargue la JSP que muestra el formulario se ha decidido separa los Action y generar una serie de Action destinados únicamente a la carga de las diferentes JSP facilitando así el seguimiento del funcionamiento de la aplicación.

◊ Se ha decidido no tratar las excepciones producidas mediante una jerarquía de clases exception y dejar que sea el Action quien al detectar un resultado con exception cargue la página que se ha diseñado para cada caso.

◊ Se escoge el servidor de aplicaciones Tomcat por su masiva comercialización y por estar disponible en numerosos alojamientos. Aunque en esta aplicación no se precisa y su uso está bastante cuestionado se ha valorado negativamente que Tomcat no ofrezca contenedor para los EJB como el JBoss. Un punto a destacar a favor de Tomcat es su fácil instalación y configuración con el IDE Netbeans 6 aunque en el servidor de aplicaciones que realmente esta integrado con Netbeans y ofrece contenedor EJB es el GlassFish pero con relativa poca difusión.

◊ Como criterio principal para la elección del IDE adecuado ha prevalecido la experiencia personal en Eclipse y Netbeans siendo este último el elegido gracias al respaldo de SUN, su creador. Otro IDE destacable es myEclipse que queda descartado por no ser gratuito.

◊ La elección de la base de datos PostgreSQL versus MySQL responde a su extraordinaria interface gráfica que facilita enormemente su manejo, aunque la difusión respecto a MySQL es menor se ha verificado que muchos de los alojamientos que ofrecen MySQL también dan soporte a PostgreSQL.

◊ Se ha optado en utilizar los DynaActionForm en lugar de los ActionForm a pesar de no obtener mejoras especialmente significativas pero se a considerado positivamente el hecho de no tener que generar manualmente los métodos getter y setter de los atributos, disminución del número de clases del proyecto por tanto menor complejidad y mayor manejo de clases y en especial, la fácil definición de ActionForm agrupados y muy legibles.

- ◊ Debido a que las características de la aplicación hacen que el código correspondiente no sea excesivamente largo se ha optado por hacer que las clases Action contengan el código correspondiente a los accesos a las clases DAOs. De esta manera se está considerando a la clase Action como parte de la lógica de negocio y simplificando el número de clases de la aplicación por tanto la complejidad. Para que esta decisión no suponga un aumento de la dificultad del mantenimiento de la aplicación se ha delimitado mediante comentarios el espacio correspondiente a dicho código dentro de las clases Action.

- ◊ Para mejorar la usabilidad de la aplicación se han incluido desplegables en todas las interfaces donde los campos requeridos sean un número de valores conocidos, de manera que cuando un operador debe seleccionar un vehículo en su desplegable solo tendrá aquellos vehículos destinados a su base, si es el caso del usuario logística en su desplegable tendrá la totalidad de los vehículos dados de alta en la aplicación. Esta filosofía se ha llevado a la totalidad de los desplegables.

- ◊ La estrategia seguida en la implementación de la carga de valores en los desplegables en función del usuario que utiliza la aplicación ha sido implementar unas clases Action destinadas a la carga de valores.

- ◊ Como consecuencia de la mejora del punto anterior se ha reducido considerablemente las necesidades de validaciones en el lado servidor ya que los datos no son escritos por el usuario en su mayor parte sino que son seleccionados de un rango de valores ya validados.

- ◊ Para aquellos campos de los formularios donde no ha sido posible aplicar desplegables se han realizado validaciones en el lado cliente mediante java scripts por su facilidad de programación y el poco código requerido para hacer una validación ya que este código será descargado por el cliente junto con la vista.

- ◊ Para facilitar la distribución de las clases dentro del proyecto se han colocado las vistas en una carpeta aparte y se ha separado en dos pakages distintos los DAOs de los Action y Beans.

- ◊ Se gestionan los log para errores mediante servlet.log un mensaje y el error para facilitar la traza de la aplicación en caso de problemas.

- ◊ Se sustituyen todas las referencias de la clase Vector en la versión 1 de la aplicación por ArrayList en esta nueva versión.

- ◊ Se garantiza en cierre de la conexión con la base de datos usando la clausula finally después del try y catch.

5.2. Software usado en la implementación.

El software necesario para el desarrollo del proyecto es:

- ◊ Microsoft Office incluido Project 2003 y Visio 2003.
- ◊ Java EE 5 SDK.
- ◊ Apache Tomcat 6.
- ◊ NetBeans 6.
- ◊ Framework Struts .
- ◊ Postgress + herramienta administración.
- ◊ Driver Postgress jdbc.

5.3. Funcionamiento de la arquitectura.

Para apreciar el funcionamiento de la arquitectura se usa el diagrama de colaboración de uno de los casos, en concreto, el alta de una incidencia en un vehículo.

Cuando Struts recibe la petición de alta de incidencia en vehículo entonces carga el formulario correspondiente que una vez rellenado será guardado en el ActionForm. Struts asigna el Action que se encargará de invocar a la lógica de negocio, esta recuperará la información del ActioForm y procesará los datos, el resultado será devuelto a Struts que cargará la vista que corresponda en función del resultado.

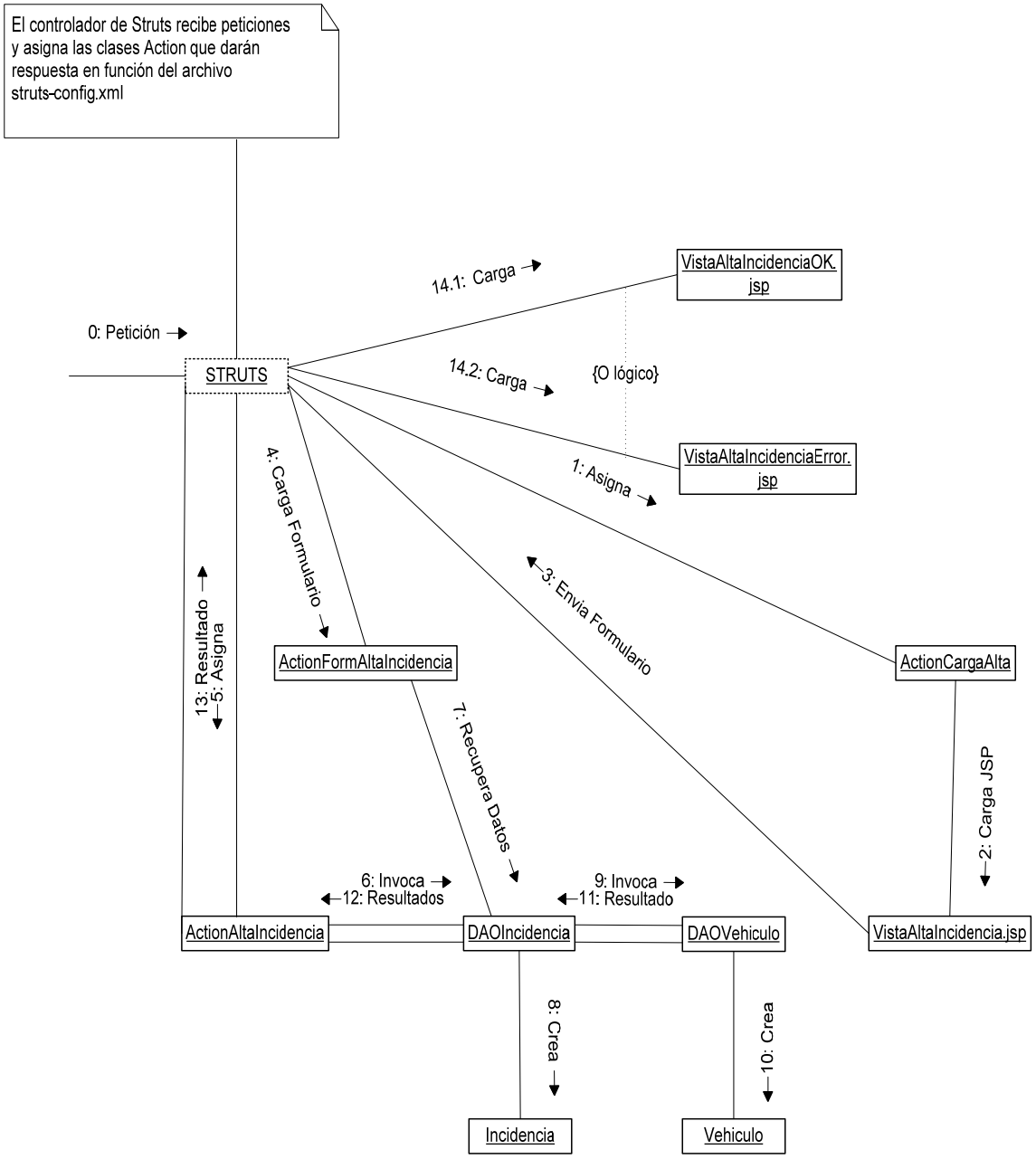


Figura 17. Diagrama de colaboración de la arquitectura.

5.4. Instrucciones de instalación.

Requisitos de software para un correcto funcionamiento de la aplicación Helper será necesario tener instalado el siguiente software:

- ◊ Base de datos Postgres 8.1.4.
- ◊ Apache Tomcat 6.0
- ◊ Maquina java, jre 1.6.

Para un correcto funcionamiento de la aplicación será necesario seguir unos mínimos pasos de configuración:

- ◊ Cree una base de datos asignándole un nombre y una contraseña (puede usar la BD por defecto que se crea al instalar postgres) y cargue en ella el script proporcionado. Además de la creación de las tablas también se cargaran los inserts mínimos necesarios con la información que quedaba fuera del ámbito de la aplicación. Por ejemplo, el mantenimiento de usuarios no está descrito entre las funcionalidades de la aplicación por lo que encontrará los usuarios necesarios ya introducidos en las tablas.

- ◊ Despliegue el archivo Helper.war proporcionado en el servidor Tomcat.

- ◊ Puede configurar la conexión con la base de datos mediante el archivo configuration.properties que encontrará dentro de Helper.war, puede editar el archivo con Winrar antes de desplegar la aplicación en la ruta /WEB-INF/clases/com/myapp/struts o si lo prefiere una vez desplegada la aplicación en la ruta del Apache webapps/Helper/WEB-INF/clases/com/myapp/struts. En cualquier caso deberá ajustar el username y password al que utilizó en el momento de la instalación y carga de datos en postgres.

- ◊ Para hacer uso de la aplicación solo deberá acceder mediante el navegador usando la dirección: http://localhost:8080/Helper/

6. Capitulo 6.

6.1. Conclusiones.

Durante el desarrollo del proyecto se han alcanzado tanto los objetivos principales como los secundarios con creces.

Resulta un tanto complejo comenzar con el proyecto debido a la diferencia docente entre las asignaturas de la carrera en las que hay una programación y temporalización determinada y sobre todo un temario definido y concreto. Y la asignatura TFC en la que cada uno deberá buscar su propio temario y realizar su propio plan de estudio. Este cambio de filosofía es tremendamente positivo porque es la filosofía que cada uno encontrará en munchos ámbitos laborales y que ahora es capaz de afrontar.

Respecto al componente docente tecnológico debo señalar que en el caso de J2EE me he sentido muy satisfecho con las posibilidades y con el estilo de programación. Pienso que es una tecnología que aún va a desarrollarse mucho más en los próximos años y que copará gran parte del mercado laboral.

La curva de aprendizaje de la asignatura es muy elevada y requiere muchísimo trabajo si el punto de partida de los conocimientos de cada uno respecto a la tecnología es nulo o muy bajo, por lo que en ese caso no hay una correspondencia con el número de créditos de la asignatura. Sin embargo, si el objetivo real es aprender una magnifica tecnología como J2EE merece la pena dedicarle tanto esfuerzo y tiempo.

Glosario.

Asignación: Funcionalidad del operador que permite asignar un vehículo a un servicio.

Base: Lugar ubicado en el territorio catalán compuesto de recursos materiales y humanos capaces de dar cobertura sanitaria al territorio asignado. Todas las bases dependen de la central.

Central coordinación: Centro de referencia del sistema de emergencias donde se reciben todas las llamadas y se traspasa la demanda asistencial a la base que corresponde.

Central logística: Centro de referencia del sistema de emergencias donde se gestiona la totalidad de la flota.

Helper: Nombre de la aplicación de gestión de flota.
Incidencia: Avería o problema detectado en algún vehículo de la flota.

Indicativo: Identificador de los vehículos de la flota.

Operatividad: Denota si un vehículo está disponible para trabajar o no en función de las incidencias detectadas.

Servicio: Asistencia sanitaria realizada o demandada.

Usuario logística: Personal ubicado en la central de logística que se encarga de la gestión de la flota.

Usuario Operador: Personal ubicado en cada base y se encarga de la asignación de vehículos a servicios y de la notificación de incidencias en vehículos.

Bibliografía.

Programación Java Server con J2EE Edición 1.3. Ed. Anaya. ISBN 978-84-415-1358-7.

http://www.sun.com/

http://www.netbeans.org/

http://tomcat.apache.org/

http://struts.apache.org/

http://www.postgresql.org/

http://www.javaworld.com/

http://www.programacion.net/java/

http://www.adictosaltrabajo.com/

http://www.javahispano.org/

http://es.wikipedia.org/

http://javascripts.astalaweb.com/_inicio/Presentacion.asp

http://www.mundojavascript.com/

http://www.w3c.es

http://www.webestilo.com/

http://www.javapassion.com/j2ee/

Anexo 1. Manual del usuario.

1. Manual de uso.

1.1. Introducción.

El modelo de asistencia sanitaria pre-hospitalaria de Cataluña se configura mediante unas bases distribuidas por todo el territorio catalán. Cada base proporciona asistencia sanitaria a una zona determinada de la región mediante una dotación de vehículos una serie de personal y unos recursos determinados. Sin embargo, todas las bases dependen de una base central encargada de gestionar los servicios comunes a todas las bases.

La base central es donde se alberga la central de coordinación encargada de recibir las llamadas de los usuarios y transmitir las demandas asistenciales a la zona correspondiente que procederá con la asignación de los recursos necesarios a cada asistencia sanitaria.

Logística también es un servicio común ubicado en la base central, desde aquí se gestionarán las averías e incidencias en las unidades de la flota.

Este modelo de dispersión geográfica para dar cobertura sanitaria a todo el territorio pero con centralización de servicios comunes requiere una herramienta de comunicación y de gestión de flota accesible desde cualquier punto del territorio. Helper ofrece esta característica porque trata a la totalidad de las bases como clientes ligeros e implementa toda la lógica y la base de datos en un servidor aparte donde se procesará toda la información.

1.2. Usuarios.

1.2.1. Usuario logística.

El usuario logística es un usuario que pertenece a la base central, por tanto sus funcionalidades irán encaminadas a la gestión de las incidencias y averías de la totalidad de la flota.

En el script de la base de datos se da de alta al usuario: **User1 con password: Pass1** que tiene el rol de usuario de logística, entre las funcionalidades asignadas a este tipo de usuario tenemos:

Modificar la operatividad del vehículo. Cada vehículo de la totalidad de la flota podrá ser retirado del servicio para su reparación o por cualquier otro motivo si el usuario de logística lo considera necesario.

Dar de baja una avería: La baja es lógica en realidad lo que se hace es marcar como reparada una avería o incidencia cuando el usuario de logística a tramitado su reparación o nulidad.

Consultar averías por número de identificación: El usuario logística podrá acceder a los datos de cualquier avería a través del número identificador que se le asignó al darla de alta.

Listar averías pendientes de resolución: El usuario de logística accede a un listado de todas aquellas averías que están pendientes de resolución.

Listar averías por vehículo: El usuario logística puede acceder a un listado de las averías que haya tenido un vehículo.

1.2.2 Usuario operador.

Cada base tiene asignado un operador que tiene las funciones de mini central de coordinación haciendo de nexo entre los servicios centrales y los recursos asistenciales de la base.

En el script de la base de datos ya hay algunos operadores dados de alta.

Ciudad	Usuario	Password
Barcelona	User2	Pass2
Gerona	User3	Pass3
Lérida	User4	Pass4
Tarragona	User5	Pass5

Las funcionalidades que dispone un usuario de tipo operador son:

Asignar vehículo a servicio: El operador recibe los datos de la central de coordinación referentes a una demanda sanitaria, dicha demanda tiene un número de identificador que el operador introduce y asigna un vehículo de la flota adscrita a la base.

Cerrar asignación: Una vez finalizado el servicio que tenía asignado el vehículo se procede al cierre de la asignación.

Consultar asignaciones por fecha: Un usuario operador puede acceder a las asignaciones que se han producido en una fecha y con los vehículos adscritos a su base.

Alta de avería o incidencia: Los operadores tienen como responsabilidad notificar las incidencias o averías producidas en la flota de vehículos adscrita a su base.

Listar averías pendientes de resolución: Cada operador puede listar cuales son las averías que tiene pendientes de resolución en sus vehículos.

1.3. Funcionalidades.

1.3.1. Login.

Funcionalidad de inicio de la aplicación, se demanda un nombre de usuario y un password, según la entrada de datos se dará acceso al menú logística o menú operador. En caso de no correspondencia con los datos almacenados se procederá a informar y volver a ofrecer la posibilidad de login.

1.3.2. Modificar la operatividad del vehículo.

Cuando logística decide que debe impedir la utilización de un vehículo a raíz de una notificación de incidencia, avería o a causa de mantenimiento o cualquier otra índole podrá hacer uso de esta funcionalidad.

El usuario de logística puede acceder a esta funcionalidad mediante la parte del menú que corresponde a la operatividad de vehículos. En la pantalla podrá desplegar el campo indicativo accediendo a la totalidad de los vehículos registrados en Cataluña y podrá marcar la situación de no operativo si desea retirar el vehículo del servicio ordinario u operativo si desea devolverlo a la operativa diaria.

Observe el cambio de estado entre operativo o no operativo se producirá con independencia de cualquier otra situación que tenga el vehículo. Por ejemplo, si el vehículo al que se le pretende retirar del servicio estuviese en uso en ese momento se permitirá igualmente el cambio de estado, así cuando el vehículo finalice su servicio el operador de la base comprobará que no puede asignarle ningún otro servicio.

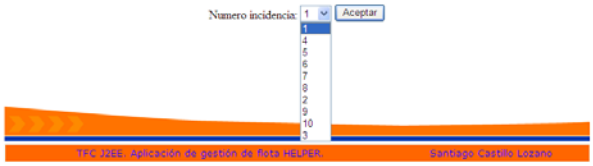
1.3.3. Dar de baja una avería.

Una vez que la incidencia o avería notificada por el operador de la base correspondiente ha sido gestionada por el servicio de logística es necesario quitar la avería de la lista pendientes para pasarla a la lista de reparadas. Para ello el usuario logística accederá a la parte del menú averías donde encontrará un listado de todas las averías que están pendientes de resolución y junto a cada avería un icono que podrá presionar para marcar la avería como reparada.

Una vez marcada una incidencia o avería como reparada el usuario logística recibirá una confirmación del cambio de estado mostrando los datos de la incidencia seleccionada.

En caso de no existir averías pendientes el sistema informará de ello.

1.3.4. Consultar avería por número de identificador.



El usuario logística podrá consultar los datos de cualquier incidencia o avería registrada, tenga el estado que tenga, mediante el número identificador. Para ello accederá a la parte del menú averías y entrará en la pantalla donde encontrará un desplegable con todos los números de referencias disponibles.

Observe que los valores del desplegable se irán actualizando automáticamente en función de las incidencias que se vayan dando de alta en la base de datos.

Una vez marcada la incidencia los datos serán mostrados por pantalla.

1.3.5. Listar averías pendientes de resolución.

El usuario logística podrá listar todas aquellas incidencias o averías que estén dadas de alta y que tengan un estado pendiente de resolución. Para ello accederá a la parte del menú averías y seleccionando la opción correspondiente se le mostrará por pantalla la información.

Si no hay incidencias pendientes el sistema informará de ello.

1.3.6. Listar averías por vehículo.

El usuario logística podrá listar las averías con independencia de si tienen estado pendiente o reparado según el vehículo en el que se produjo la incidencia. Para ello accederá a la parte del menú avería y seleccionará la opción que corresponde.

Se le mostrará un cuadro desplegable con la totalidad de los vehículos introducidos en la base de datos.

Una vez seleccionado el vehículo deseado se le mostrará la información solicitada por pantalla. O se avisará en caso de no haber incidencias para ese vehículo.

1.3.7. Asignar vehículo a servicio.

El operador deberá asignar uno o varios vehículos a una demanda asistencial notificada por la central de coordinación. Dicha demanda queda identificada por un número. El operador accede al menú de asignaciones y selecciona dicha opción.

En la pantalla que se muestra el operador marcará el número identificador de asistencia y seleccionará el vehículo asignado

del desplegable. Observe que el desplegable solo contiene los vehículos adscritos a la base donde pertenece el operador y que quedó registrada al loginarse.

En caso de intentar seleccionar un vehículo que figure con estado no operativo o dicho vehículo se encuentra ya asignado a algún servicio el sistema avisará de ello. Si la asignación a tenido éxito se mostrará los datos de la asignación incluyendo el registro de fecha y hora.

1.3.8 Cerrar asignación.

Después de que un operador ha asignado un vehículo a una asistencia y esta finaliza se deberá proceder al cierre de la asignación para volver a dejar al vehículo libre a la espera de una nueva asignación.

El operador accede a esta funcionalidad a través del menú asignaciones de vehículos y al seleccionar cierre de asignaciones se le mostrará la lista de todas las asignaciones abiertas sobre los vehículos adscritos a su base.

En caso de no existir asignaciones abiertas en esa base el sistema mostraría un mensaje informando de ello.

Observe que las casillas de fecha fin y hora fin no tienen valor ya que se trata de asignaciones aún no finalizadas. El operador podrá seleccionar la asignación que desea cerrar mediante el icono de candado y el sistema procederá a cerrar la asignación registrando fecha y hora de fin y mostrando los datos de la asignación ya cerrada.

1.3.9. Consultar asignaciones por fecha.

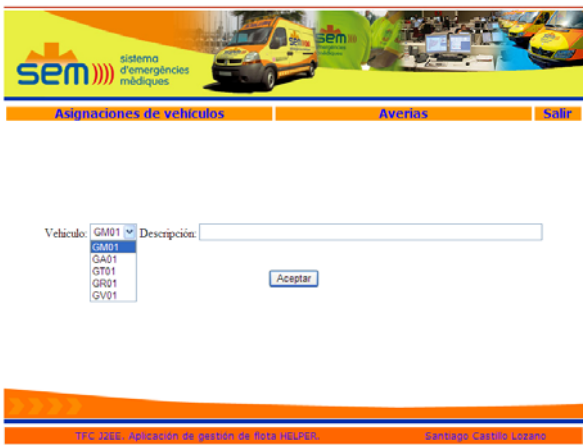
Otra funcionalidad que el operador de una base puede requerir es la consulta de todas las asignaciones que se han producido en un día concreto. Para ello accederá al menú de asignaciones y seleccionando la opción correspondiente se le mostrará una pantalla para la selección de la fecha.

Una vez marcada la fecha se le mostrarán todas las asignaciones del día sobre los vehículos adscritos a su base, tanto si están aún abiertas como cerradas. Si no hubiese asignaciones en ese día el sistema avisaría mediante un mensaje.

1.3.10. Alta de avería o incidencia.

Cada operador tiene la obligación de registrar todas las incidencias o averías producidas en la flota de vehículos adscrita a su base. Para ello deberá acceder a la parte del menú averías y seleccionando alta de averías se le muestra una pantalla donde puede desplegar sus vehículos y rellenar un campo donde describirá la incidencia.

Para mostrar que la incidencia ha sido registrada de manera satisfactoria el sistema mostrará los datos de la misma por pantalla. Será el usuario de logística el encargado de gestionar la reparación y determinar la operatividad del vehículo.



1.3.11. Listar averías pendientes de resolución.

El operador seguro necesitará saber cuales son las incidencias que están aún pendientes de resolución por parte del servicio de logística. Para ello accederá al menú averías y seleccionando la opción listar averías se le mostrará un listado con los datos de aquellas averías o incidencias que aún están pendientes de resolución.

1.3.12. Salir.

Tanto el usuario operador como el usuario logística podrán abandonar la aplicación en cualquier momento si seleccionan la opción salir del menú, en tal caso se les mostrará una pantalla en scroll vertical dejando como única opción el cierre del navegador.