

Universidad de las Ciencias Informáticas

Facultad 2



Título: *“Portal Web para Empresas petroleras.”*

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autora: Dailenis Zamora Zorrilla

Tutores: Ing. Yeilin Pérez Martínez
Ing. Joan Martínez Herrera

Consultante: Ing. Michel Arias Arias

La Habana, 2011

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Dailenis Zamora Zorrilla

Ing. Yeilin Pérez Martínez.

Ing. Joan Martínez Herrera.

AGRADECIMIENTOS

A mi mamá, por se la luz de mi vida.

A mis amigas, en especial a Mari, por su paciencia.

A mis amigos, los de la etapa del pre universitario como los que conocí aquí, por siempre tener siempre una mano amiga.

A mis compañeros, todos, les agradezco los momentos inolvidables que pasamos juntos.

A mis tutores. A Yeilin por ver donde yo no podía y por su comprensión, a Joan por su paciencia.

A todos los profesores que han influido en mi formación y a los que me han ayudado.

DEDICADO A:

Mi Maitica...

Por quién un día programaré en Francés.

RESUMEN

La utilización de las tecnologías de la información dentro del campo de la industria petrolera representa una excelente alternativa para la gestión de sus procesos. En una refinería es posible la ocurrencia de incidentes que ponen en riesgo tanto la vida de sus trabajadores, como el desarrollo eficiente de sus actividades. Durante las jornadas de trabajo se hace necesario gestionar los incidentes ocurridos en tiempo real, pero muchas refinerías no cuentan con un sistema que automatice estos procesos.

TETRA es un estándar global de comunicación que define un sistema digital de radio móvil. El mismo satisface las necesidades de los clientes facilitando una conexión punto a punto y punto a múltiples puntos vía radio de forma rápida.

El presente trabajo es una aplicación Web que mediante el uso de servicios Web, RMI y mensajes WAP Push, a través de redes TETRA, gestiona la información relacionada con los incidentes de una refinería.

Para el desarrollo de la solución se utilizó la metodología FDD (*Feature Driven Development*) y tecnología Java, luego del análisis de herramientas de administración de incidentes disponibles.

Palabras Clave: Refinería petrolera, servicio Web, RMI, mensajes WAP Push, TETRA.

ÍNDICE

INTRODUCCIÓN.....	6
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.....	10
1.1 Introducción.....	10
1.2 Estudio del Arte	10
1.3 Principales Conceptos:	11
1.3.2 WSDL.....	12
1.3.3 SOAP	12
1.4 Principales Servicios	13
1.4.1 Servicios Web.....	13
1.4.2 Servicio Java RMI.....	14
1.4.3 Mensajes WAP Push.....	14
1.5 Tecnologías utilizadas.....	15
1.5.1 Lenguaje de programación.....	15
1.5.2 Framework utilizado. Spring Framework	16
1.5.3 Spring Security	18
1.5.4 Dojo Framework	19
1.6 Metodología de desarrollo	19
FDD (Feature Driven Development)	20
1.7 Herramientas	22
1.7.1 Entorno Integrado de Desarrollo. Eclipse	22
1.7.2 Servidor Web. Apache Tomcat 6.0.24	22
1.7.4 Herramienta de modelado. StarUML	23
1.8 Conclusiones Parciales	24
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.....	25
2.1 Introducción.....	25
2.2 Problemática	25
2.3 Objeto de automatización.	26
2.4 Propuesta del sistema.....	26
2.5 Modelo de Dominio	27
2.6 Relación de los requerimientos.	28

2.6.1	Requerimientos funcionales.....	28
2.6.2.	Requerimientos no funcionales.....	29
2.7	Definición de los casos de uso.....	29
2.7.1	Definición de los actores	30
	Tabla 1: Definición de los actores	30
2.7.2	Diagrama de casos de uso.....	30
2.7.3	Casos de uso expandidos	30
2.8	Conclusiones Parciales.....	36
CAPÍTULO 3: DISEÑO DEL SISTEMA.....		37
3.1	Introducción	37
3.2	Diseño.....	37
3.3	Diagrama de Paquetes	37
3.4	Diagrama de Clases del Diseño.....	38
3.5	Diagramas de interacción.....	42
3.5.1	Diagrama de secuencia.....	42
3.6	Diseño de la base de datos.....	43
3.7	Arquitectura	44
3.8	Patrones de Diseño	47
3.9	Conclusiones Parciales	51
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA		52
4.1	Introducción.....	52
4.2	Implementación	52
4.2.1	Diagrama de despliegue.....	52
4.2.2	Diagrama de componentes.....	53
4.3.1	Modelo de prueba. Pruebas de Funcionalidad	56
4.3.2	Pruebas de Seguridad	60
4.4	Conclusiones Parciales.....	60
CONCLUSIONES		61
RECOMENDACIONES		62
BIBLIOGRAFÍA.....		63
GLOSARIO		65

ÍNDICE DE FIGURAS

FIGURA 1: ELEMENTOS FUNDAMENTALES DE SPRING SECURITY.....	18
FIGURA 3: MODELO DE DOMINIO.....	28
FIGURA 4: DIAGRAMA DE CASO DE USO DEL SISTEMA.....	30
FIGURA 5: DIAGRAMA DE PAQUETES DE LA APLICACIÓN.....	37
FIGURA 6: DIAGRAMA DE PAQUETES DE ACCESO A DATOS.....	38
FIGURA 7: DIAGRAMA DE CLASES DEL PORTAL WEB DE LA FUNCIONALIDAD SAVEINJURED.....	39
FIGURA 8: DIAGRAMA DE CLASES DEL PAQUETE INJURED JAR WS.....	39
FIGURA 9: DIAGRAMA DE CLASES DEL PAQUETE INJURED JAR RMI.....	40
FIGURA 10: DIAGRAMA DE CLASES DEL PAQUETE INJURED JAR JDBC.....	40
FIGURA 11: DIAGRAMA DE CLASES DE SERVICIO RMI DEL PAQUETE SERVICES DE INJURED.....	40
FIGURA 12: DIAGRAMA DE CLASES DEL SERVICIO WEB DEL PAQUETE SERVICES DE INJURED...	41
FIGURA 13: DIAGRAMA DE CLASES DE LOS SERVICIOS WEB Y RMI PAQUETES BUSINESS Y DATAACCESS DE INJURED.....	42
FIGURA 14: DIAGRAMA DE SECUENCIA DEL ESCENARIO SALVAR LESIONADO, CU GESTIONAR LESIONADO.....	43
FIGURA 15: MODELO DE DATOS.....	44
FIGURA 16: ARQUITECTURA 3 CAPAS.....	45
FIGURA 17: ARQUITECTURA MVC.....	45
FIGURA 18: ARQUITECTURA DEL SISTEMA.....	46
FIGURA 19: MUESTRA DEL PATRÓN EXPERTO EN EL SISTEMA.....	47
FIGURA 20: MUESTRA DEL PATRÓN CREADOR EN EL SISTEMA.....	48
FIGURA 21: MUESTRA DEL PATRÓN CONTROLADOR EN EL SISTEMA.....	48
FIGURA 22: MUESTRA DEL PATRÓN ALTA COHESIÓN EN EL SISTEMA.....	49
FIGURA 23: MUESTRA DEL PATRÓN BAJO ACOPLAMIENTO EN EL SISTEMA.....	50
FIGURA 24: MUESTRA DEL PATRÓN INYECCIÓN DE DEPENDENCIAS EN EL SISTEMA.....	51
FIGURA 25: DIAGRAMA DE DESPLIEGUE.....	52
FIGURA 26: DIAGRAMA DE SUBSISTEMAS DE IMPLEMENTACIÓN.....	53
FIGURA 27: DIAGRAMA DE COMPONENTES DE LESIONADO.....	54
FIGURA 28: COMPONENTES DE LAS LIBRERÍAS DEL SISTEMA.....	55

ÍNDICE DE TABLAS

TABLA 1: DEFINICIÓN DE LOS ACTORES.....	30
TABLA 2: CASO DE USO GESTIONAR INCIDENTE	36
TABLA 3: CASO DE PRUEBA DE LISTAR LESIONADO.....	59
TABLA 4: RESUMEN DE NO CONFORMIDADES	59

INTRODUCCIÓN

El mundo cuenta hoy con el vertiginoso avance de las aplicaciones Web como un potente medio de comunicación. En consecuencia, disímiles instituciones logran verter todo, o parte de sus procesos de trabajo diario, hacia el mundo digital e Internet. Cada día la sociedad es más dependiente de la comunicación y la transmisión de datos mediante la red, y utiliza tecnologías de punta que reflejan los productos y servicios que resultan de los procesos de negocios.

La transformación de las corporaciones en la búsqueda de una mayor eficiencia, tiene como herramienta fundamental el uso de la tecnología de información que abarca la Informática, las Telecomunicaciones y la Automatización Industrial, con el objetivo de ponerlas a disposición del usuario capacitado para la toma de decisiones.

En el ascenso de los índices de eficiencia y calidad juegan un papel significativo los procesos de gestión de la información y los conocimientos. Producto del auge constante de las tecnologías de la Información y las Comunicaciones (TICs), estos procesos de gestión son automatizados tanto en el campo de la investigación, como de la formación y/o la producción, tributando a mejoras en el ámbito económico, social y cultural.

Las empresas petroleras no están al margen de dicho avance y sus beneficios. En las mismas, la utilización de las tecnologías de la información, constituye una excelente herramienta que permite potenciar el desarrollo de sus capacidades competitivas a nivel internacional. Con el auge de los sistemas de comunicación, muchas de estas entidades han invertido en tecnologías basadas en el estándar TETRA (*TErrestrial Trunked RAdio*) para gestionar sus comunicaciones. TETRA es un estándar global de comunicación de radio desarrollado para satisfacer las necesidades de los clientes que necesitan rápida conexión punto a punto y punto a múltiples puntos vía radio. El mismo define un sistema digital de radio móvil que aporta mayor privacidad y confidencialidad, además de la capacidad de acceso a otras redes como Internet, red telefónica fija o móvil. Este tipo de tecnología permite recibir y enviar información por medio de los radios profesionales. Gracias a este progreso, los agentes de campo se mantienen comunicados entre sí y las aplicaciones informáticas pueden interactuar con los trabajadores mediante sus radios profesionales. Su uso permite un incremento y calidad de los servicios que deben ofrecerse al usuario, con la posibilidad de alcanzar un liderazgo tecnológico, con énfasis en la eficiencia.

Teltronic es una empresa que suministra soluciones completas de comunicaciones inalámbricas, específicamente redes TETRA y servicios para ellas, a sectores profesionales como la seguridad pública, el

transporte masivo de pasajeros, y sectores industriales como la energía, la minería, la siderurgia, el petróleo y gas, entre otros. Dentro de la Universidad de las Ciencias Informáticas, específicamente en el Centro de Telemática (TLM), desde hace dos años se desarrollan soluciones y servicios para esta compañía. Uno de los sistemas a desarrollar involucra una solución centrada en la gestión de información para empresas petroleras.

Las industrias petroleras constan de varias unidades de trabajo, entre ellas, la unidad de refinación, donde se transforma el crudo de petróleo en sus productos derivados mediante una serie de procesos físicos y químicos, que adicionan una mayor calidad al mínimo costo posible.

En el área donde se trabaja directamente con el crudo, es posible la ocurrencia de incidentes que pongan en riesgo tanto la vida de sus miembros, como el desarrollo eficiente del trabajo. Los eventos que pueden surgir durante una jornada laboral podrían ser fugas de gases, escapes, derrames, explosiones o incendios. Estos incidentes pueden tener lugar en diferentes elementos dentro de la instalación como: tuberías, conexiones flexibles, filtros, válvulas, recipientes, bombas, compresores, tanques, chimeneas; y lo acontecido puede provocar daños por sustancias que pueden ser tóxicas, muy reactivas, explosivas e inflamables.

Una parte cuantiosa de estas empresas petroleras carecen de un sistema de avisos y registros de incidentes, que les permita obtener detalles respecto a los mismos, tanto durante como después de lo ocurrido. Por tal motivo, la transmisión de la información que el personal debe conocer referente al estado de la institución puede verse afectada, y el almacenamiento de los datos puede excluir elementos relevantes. Al mismo tiempo, estas no poseen un sistema que exponga la información que reportan los agentes de campo, referente al estado del equipamiento de la empresa.

La actividad de una refinería está dirigida por el abastecimiento de su mercado y el aprovechamiento de las oportunidades de negocio que este le permita, por tanto, un sistema para la gestión de información, facilitará en gran medida la mitigación de accidentes que contrarresten dicha actividad, y en el mejor de los casos, que los mismos no se produzcan. Su uso permitirá el acceso a la información de una forma rápida y segura, agilizando la toma de decisiones ante estos eventos, y economizando el tiempo. Además, aumentará el control de las situaciones, viabilizará respuestas inmediatas por parte del personal encargado y mejorará la productividad de las operaciones diarias incrementando el nivel de preparación de cara a cualquier desastre. La aplicación posibilitará que los responsables del mantenimiento de la instrumentación queden informados, y los trabajadores encargados de usar dichos equipos conozcan previamente las

condiciones en que estos se encuentran.

Luego del análisis de la situación actual en las refinerías de las empresas petroleras, surge el siguiente **problema a resolver**: ¿Cómo gestionar en empresas petroleras que utilicen tecnología TETRA, la información concerniente a sus incidentes?

A partir del problema a resolver se puede definir como **objeto de estudio**: Los procesos que tienen lugar dentro de las refinerías petroleras, el mismo se encuentra enmarcado en el **campo de acción**: La gestión de la información en las refinerías petroleras que utilicen tecnología TETRA.

El **objetivo general**: Desarrollar un sistema que gestione la información de los incidentes ocurridos en refinerías petroleras que utilicen tecnología TETRA.

Para profundizar el objetivo general, se persiguen como **objetivos específicos**:

- ✓ Diseñar un portal Web para la gestión de información de incidentes provenientes de refinerías petroleras.
- ✓ Diseñar las bases de datos necesarias a utilizar en el sistema.
- ✓ Implementar el portal Web para la gestión de información de incidentes provenientes de refinerías petroleras.
- ✓ Utilizar la implementación del envío de mensajes WAP Push en la aplicación.

Con la finalidad de un desarrollo óptimo de la aplicación surgen las siguientes **preguntas científicas**:

- ✓ ¿Cómo funcionan los procesos dentro de una refinería?
- ✓ ¿Cómo se informan los trabajadores de un incidente en la refinería?
- ✓ ¿Qué información puede ser útil para los trabajadores en caso de un incidente en la refinería?
- ✓ ¿Cómo implementar los mecanismos de acceso a datos mediante JDBC, servicios Web y servicios RMI?
- ✓ ¿Qué es un mensaje de tipo WAP Push?
- ✓ ¿Cómo adaptar la implementación del envío de mensajes WAP Push a la futura aplicación?

Para guiar el cumplimiento de los objetivos se definen como **tareas de investigación**:

- ✓ Análisis del negocio existente en las refinerías petroleras y las posibles funcionalidades que puedan implementarse.
- ✓ Búsqueda de información en diversas fuentes sobre aplicaciones Web que gestionen la información que proviene de los incidentes de una refinería petrolera.
- ✓ Selección de metodología, tecnologías, lenguajes y herramientas que tributen al desarrollo del sistema.
- ✓ Definición de los patrones de diseño y arquitectura a utilizar para la posible solución.

- ✓ Análisis de los mecanismos de acceso a datos mediante tres vías: JDBC, servicios Web y servicios RMI.
- ✓ Utilización de la implementación del envío de mensajes WAP Push en el sistema.

La realización de las tareas estuvo sustentada por un conjunto de métodos de investigación:

Métodos Teóricos:

- ✓ Analítico – Sintético: El uso de este método se puso de manifiesto en la investigación durante el análisis de servicios Web, las diferentes tecnologías y herramientas, para luego definir una síntesis específica y concreta de estas.
- ✓ Histórico lógico: Este método ha sido usado con el fin de estudiar la trayectoria y evolución de los productos informáticos existentes en las empresas petroleras.

Métodos Empíricos:

- ✓ Entrevista: Con el objetivo de recopilar información sobre los incidentes que pueden ocurrir dentro de una refinería y las respuestas ante estos eventos.

El presente estudio, está constituido por 4 capítulos, a continuación se muestra una breve descripción de cada uno de ellos:

Capítulo 1 “*Fundamentación Teórica*”: Se expone un análisis sobre los sistemas informáticos inmersos en la gestión de la información, específicamente en las empresas petroleras que utilicen tecnología TETRA. Se fundamentan los principales conceptos que se manejan a lo largo del tema y contiene la investigación de las metodologías, herramientas y lenguajes de programación utilizados para el desarrollo de la solución.

Capítulo 2 “*Descripción de la Solución Propuesta*”: Se describe el modelo del negocio y la descripción de los procesos que serán objetos de automatización. Se presenta la propuesta del sistema, los requerimientos funcionales y no funcionales. Se desarrolla el modelo de dominio y se definen los casos de uso.

Capítulo 3 “*Diseño del Sistema*”: Incluye el diseño del sistema, obteniendo como resultado principal, los diagramas de clases del diseño y diagramas de interacción para cada sección de los casos de uso. Muestra la descripción de la arquitectura y patrones de diseño empleados. Incluye el diseño de la base de datos.

Capítulo 4 “*Implementación y Prueba*”: Expone los diagramas de componentes y la distribución física del sistema a través del diagrama de despliegue conformando así la implementación, además de las pruebas realizadas al sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se realiza un análisis de las aplicaciones Web y de las aplicaciones informáticas que gestionan la información en las refinerías petroleras que hacen uso de la tecnología TETRA. Se fundamentan las herramientas, tecnologías, metodología y lenguajes usados con el fin de lograr la solución del problema planteado.

1.2 Estudio del Arte

Con el desarrollo tecnológico e industrial, empresas de todo tipo hacen uso de los avances más notorios del mundo de la Informática y las comunicaciones. El auge y veracidad que han alcanzado las aplicaciones informáticas, han propiciado a las empresas petroleras unir a su desarrollo y productividad estos adelantos.

Las empresas petroleras por lo general se dividen en unidades de trabajo como la unidad de exploración y producción. La misma es la encargada de la exploración, certificación y perforación de los yacimientos de petróleo. La unidad encargada de la separación, mejoramiento y obtención de productos a través de las refinerías y las plantas de procesamiento, se llama unidad de refinación. Otras unidades que se pueden encontrar son además, la unidad de distribución y comercialización. Todas las unidades de las empresas petroleras se benefician de una forma u otra con aplicaciones informáticas.

Las expectativas de solución de comunicaciones varían según el área dentro de las empresas en cuestión. Las refinerías petroleras se han visto beneficiadas por las aplicaciones para la gestión de información.

La gestión de la información, potente medio de control en estas empresas, facilita la toma de decisiones eficientemente, mejorando las respuestas ante eventos que ponen en riesgo la seguridad e integridad de las instalaciones. Durante el uso de aplicaciones para la gestión de la información en las refinerías, se puede disponer de información necesaria en el momento oportuno y colocarla a disposición de los miembros de la empresa, beneficiando el desarrollo de las actividades cotidianas.

Mundialmente se han desarrollado servicios y aplicaciones para las distintas áreas de una refinería, con la finalidad de mejorar la gestión de la información que se maneja en cada una de estas. Se han desarrollado aplicaciones como servicios de controles remotos en las áreas de producción, soluciones de monitoreo y tránsito, soluciones de mapeo y soluciones de ubicación de personas, individuos y otros activos. [1]

Motorola es una de las empresas líderes en brindar servicios y aplicaciones para gestionar la información proveniente de las áreas de una refinería. Esta desarrolla aplicaciones encaminadas a optimizar la eficacia de los recursos y aumentar la seguridad de los trabajadores en el campo. Las aplicaciones de Motorola para refinerías petroleras son soluciones que conectan Personas, Recursos y Activos. [1]

Entre las soluciones de Motorola se encuentran las soluciones flexibles y ampliables que otorgan una amplia cobertura y las soluciones de comunicación de datos para controlar las áreas de trabajo, localizar personas, vehículos, etc. Entre estas se encuentran el Sistema Base Transmisor Receptor Mejorado (EBTS, *Enhanced Base Transceiver System*) que provee conectividad inalámbrica y un sistema de alarma, también el Envío de mensajería en red que permite enviar comunicaciones de voz entre PC clientes y dispositivos inalámbricos y el Subsistema de consola que provee interfaces con envío de mensajería en red, que permite controlar los grupos de trabajo y brinda soporte para situaciones de emergencia desde un punto de operaciones central. [1]

Las propuestas mencionadas pueden contribuir a solucionar muchos de los problemas de gestión de la información, pero no pueden adaptarse a un problema específico debido a que están distribuidas con licencias privativas, además las mismas están vinculadas a infraestructuras específicas que no son adaptables a una red de tipo TETRA de Teltronic, sobre todo, en el envío de mensajería de tipo Push, donde se utilizan protocolos de la empresa Teltronic para hacerlos llegar a los terminales. Al mismo tiempo adquirirlas requiere de un alto capital por parte de la empresa que no siempre puede cubrir.

1.3 Principales Conceptos:

La creación de servicios Web se realiza mediante una serie de elementos que permiten el intercambio de información. Entre los elementos fundamentales que componen un servicio Web se encuentran XML, WSDL, y SOAP.

1.3.1 XML

XML (*Extensive Markup Language*) fue creado por World Wide Web Consortium (*W3C*) y constituye un subconjunto de SGML (*Standard Generalized Markup Language*) especializado en la gestión de información para la Web. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Es un Lenguaje de Etiquetado Extensible muy simple, y a su vez estricto, que juega un papel fundamental en el intercambio de una gran variedad de datos. Es muy similar a HTML (*HyperText Markup Language*) pero su función principal es describir datos y no mostrarlos como es el caso de HTML.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. En resumen, XML sirve para estructurar, almacenar e intercambiar información. [2]

Características de XML [2]:

- ✓ No pertenece a ninguna compañía y su utilización es libre.
- ✓ Es fácilmente procesable tanto por humanos como por software.
- ✓ Separa la información o el contenido de su presentación o formato.
- ✓ Permite la utilización efectiva de Internet en diferentes hardwares (teléfonos celulares, PDAs (*Personal Digital Assistant*), terminales Braille, entre otros).

1.3.2 WSDL

WSDL (*Web Service Description Language*) es el lenguaje definido por el W3C para describir un servicio Web. El estándar WSDL está basado en XML, describiendo la interfaz pública a los servicios Web. Se utiliza para definir los elementos claves de un servicio Web, así como los formatos de mensajes previstos. De modo similar, describe detalles técnicos, pero no la funcionalidad del servicio Web. [3]

Describe además los requisitos de protocolos y los formatos de los mensajes. Cuando un programa cliente se conecta a un servicio Web puede leer el WSDL, determinando qué funciones están disponibles en el programa servidor.

Características del estándar WSDL [3]:

- ✓ Facilita escribir y mantener servicios; reduce el código que hay que escribir para hacer un cliente.
- ✓ Facilita hacer cambios para ampliar los servicios, reduciendo la posibilidad de que los clientes dejen de funcionar al llamar a esos servicios.

1.3.3 SOAP

SOAP (*Simple Object Access Protocol*) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo se basa en el estándar HTTP (*Hypertext Transfer Protocol*) para la transmisión y XML para la codificación de datos. Su funcionamiento es independiente de la plataforma, lenguaje de desarrollo e implementación. [4]

SOAP es un marco extensible y descentralizado que permite trabajar sobre múltiples protocolos de redes informáticas. Los procedimientos de llamadas remotas pueden ser modelados en la forma de varios mensajes SOAP interactuando entre sí.

Estos mensajes constan de 3 secciones: envoltura, encabezado y cuerpo. La envoltura es el elemento raíz del mensaje para describir su contenido y la forma de procesarlo. El encabezado es la información de identificación del contenido, un grupo de reglas de codificación para expresar las instancias de tipos de datos definidos por la aplicación. El cuerpo es el contenido del mensaje, una convención para representar las llamadas y las respuestas a procedimientos remotos. [4]

Presenta las siguientes características [4]:

- ✓ No está asociado con ningún lenguaje.
- ✓ No se encuentra fuertemente asociado a ningún protocolo de transporte.
- ✓ Aprovecha los estándares existentes en la industria. Permite la interoperabilidad entre múltiples entornos.

1.4 Principales Servicios

Muchos son los servicios que pueden estar presentes e integrar las aplicaciones Web. Su integración con el sistema aporta calidad del servicio, solución más completa y posibilitan la comunicación con este desde cualquier lugar. A continuación se exponen los principales servicios presentes en el sistema para dar solución a la problemática expuesta.

1.4.1 Servicios Web

La programación para Internet cambia continuamente. Los usuarios necesitan acceder a una serie de servicios que se encuentran disponibles en la red de redes, motivando un giro hacia la construcción de aplicaciones orientadas a servicios. En el contexto actual, las aplicaciones antes mencionadas, necesitan comunicarse entre sí, estar integradas y funcionar como un todo.

Los servicios Web permiten que los servidores de Internet interactúen entre ellos, y el desarrollo alcanzado, ha favorecido que cada día sean más independientes del programador. La tecnología de servicios Web proporciona funcionalidades de sistema de aplicación por red, independientemente de la plataforma. [5]

Estos no tienen una definición única, pero en general son un conjunto de aplicaciones o tecnologías con capacidad de operar en la Web, que intercambian datos entre sí para ofrecer servicios. Los usuarios solicitan mediante la Web los procedimientos remotos que los proveedores brindan como servicio. Dichos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, las cuales interactúan entre sí para presentar información dinámica al usuario.

Resumiendo, un servicio Web es un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red, que tiene una interfaz descrita en un formato procesable por una máquina, específicamente WSDL. Otros sistemas interactúan con el servicio Web a través de los mensajes SOAP, usando el protocolo de comunicación HTTP para la conexión sobre Internet y una serialización XML en relación con otros estándares relacionados con la Web. [6]

1.4.2 Servicio Java RMI

RMI (*Remote Method Invocation*) es un sistema de programación para la distribución e intercambio de datos entre las distintas aplicaciones existentes en un entorno distribuido. [7]

El intercambio se realiza de manera transparente de un espacio de dirección a otro. Su uso permite la llamada de métodos remotos sin necesidad de tener los métodos localmente. Para su creación es necesario que tanto el cliente como el servidor sean desarrollados en Java.

En RMI, cuando los objetos remotos ya han sido referenciados, el programador los puede utilizar igual que si estuviesen en la máquina local, accediendo a sus métodos y manejándolos de forma normal. La capa RMI permanece oculta al programador proporcionando mayor claridad al código y mayor comodidad a la hora de programar la aplicación. [7]

Ventajas del uso de RMI [7]:

- ✓ Facilidad de uso.
- ✓ Facilidad para programar.
- ✓ Permite realizar sistemas distribuidos orientados a objetos.
- ✓ Independencia del protocolo de comunicación.

1.4.3 Mensajes WAP Push

Las redes TETRA ofrecen servicios principalmente a clientes del mercado de radio profesional y el Protocolo de Aplicaciones Inalámbricas WAP (*Wireless Application Protocol*) permite la mejora de los servicios en las mismas. El estándar WAP soporta la mayoría de las redes inalámbricas y permite que los usuarios accedan a información de forma instantánea a través de dispositivos inalámbricos.

El protocolo WAP contiene el modelo de envío de mensajes tipo Push. Los mensajes WAP Push están basados en el modelo cliente/servidor, pero en el cual no existe una solicitud específica para la misma por parte del cliente antes del proceso de envío de información desde el servidor. [8]

Entre las clasificaciones para estos mensajes se encuentran: SI (*Service Indication*) y SL (*Service Loading*). Los mensajes de tipo SI contienen un pequeño texto para el usuario del terminal, así como una URI (*Uniform Resource Identifier*) mediante la cual se puede acceder a un servicio y es responsabilidad del usuario decidir si visitar el servicio que se ha enviado en el momento que llega el mensaje o en otro instante. Los SL por su parte, no requieren acciones por parte de los usuarios, estos mensajes contienen una URI mediante la cual se puede acceder al servicio que puede ser cargado sin la confirmación del usuario final. El contenido puede ser ejecutado o cargado en la memoria caché del navegador utilizado en el dispositivo móvil. [8]

El envío de mensajes WAP Push permite a los usuarios acceder de forma sencilla a aplicaciones donde se encuentre información útil para ellos, optimizando el acceso a la información.

1.5 Tecnologías utilizadas

En los últimos años la informática y las comunicaciones han dado pasos de gigante en su desarrollo gracias al uso de la tecnología más reciente. Durante el desarrollo de aplicaciones Web es necesario el uso de tecnologías estables, sólidas y extendidas; que aunque no en todos los casos asegure la creación de un mejor sistema, sí garantice una elaboración más simple de este.

1.5.1 Lenguaje de programación.

Los lenguajes de programación de hoy han estado influenciados por la experiencia de los lenguajes iniciales. Varios conceptos han sido inventados, examinados e implementados para incrementar la facilidad de uso y la naturaleza dinámica e interactiva de la red.

Java es un lenguaje orientado a objeto que cuenta con una tecnología madura, fuerte y variable, con el cual se pueden desarrollar diferentes tipos de programas.

Características [10]:

✓ Simple:

Elimina las características menos utilizadas y borrosas de otros lenguajes potentes, logrando un aprendizaje rápido y fácil. Reduce las especificaciones del lenguaje y añade otras características muy útiles, como el reciclador de memoria dinámica más conocido como recolector de basura.

✓ Orientado a Objetos:

Java fue diseñado como un lenguaje orientado a objetos desde su creación.

Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos que los manipulan. En este lenguaje se incorporan funcionalidades como la resolución dinámica de métodos.

✓ Distribuido:

Cuando se habla de que Java es un lenguaje distribuido, significa que este proporciona las librerías y herramientas para que los programas puedan ser distribuidos, que corran en varias máquinas, interactuando entre sí.

✓ Robusto:

Obliga a la declaración explícita de métodos para reducir la posibilidad de error. Maneja la memoria, erradicando problemas al programador, de liberación o corrupción de memoria mediante la implementación de arreglos auténticos con comprobación de límites. Además realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Este lenguaje reduce el tiempo de desarrollo de las aplicaciones.

✓ Portable:

El lenguaje fue desarrollado para que las aplicaciones pudiesen ser inmediatamente ejecutables en cualquier ordenador, y en cualquier sistema operativo; esto es posible debido a que Java trabaja con una máquina virtual que genera su propio código máquina. De esta forma con tener una máquina virtual de Java para una determinada plataforma dicho programa será completamente operativo.

✓ Dinámico:

Este lenguaje no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las clases solo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la red.

1.5.2 Framework utilizado. Spring Framework

Un framework puede definirse como un esqueleto o un patrón para el desarrollo y/o la implementación de una aplicación. En el contexto de la programación, es un código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicación, como la creación de objetos, conexión a base de datos, limpieza de cadenas, etc. La utilización de un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, además de que estructura el código fuente. En general, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Spring es un framework de aplicaciones Java/J2EE (*Java 2 Enterprise Edition*) de código abierto. Es potente en cuanto a la gestión del ciclo de vida de los componentes y fácilmente ampliable. Las características de este pueden emplearse en cualquier aplicación desarrollada en Java,

teniendo en cuenta que ofrece muchas libertades a los desarrolladores, y soluciones bien documentadas. Es un framework de peso ligero que interviene en todas las capas de una aplicación JEE (*Java Enterprise Edition*). [7]

Su meta es lograr separar los accesos a datos y los aspectos relacionados con las transacciones, permitiendo que objetos de la capa de negocio reutilizables no dependan de ninguna estrategia de acceso a datos o transacciones. Está diseñado además para facilitar una flexibilidad arquitectónica. [11]

En general, estas son algunas de las características de Spring [11]:

- ✓ Simplifica el acceso a datos.
- ✓ Soporte para acceso a componentes remotos.
- ✓ Su propio framework MVC (Modelo Vista Controlador).
- ✓ Manejo simplificado de excepciones.

Spring Web Services

Spring Web Services (*Spring-WS*) es un producto de Spring Framework para facilitar la creación de servicios Web basados en el intercambio de documentos. Permite la creación de servicios Web flexibles, aportando numerosas ventajas a la hora de desarrollar y mantener los servicios Web.

Su desarrollo puede ser mediante dos enfoques: *contract first*, donde primeramente se crea el contrato del servicio y luego se crea su implementación o *contract last*, donde se crea antes la implementación del servicio y luego se genera el contrato del servicio a partir del código. Se ha decidido realizar los servicios Web a través del enfoque *contract last* ya que su desarrollo es de manera más fácil y rápida. [7]

Las principales características de Spring-WS son [11]:

- ✓ Mejores prácticas para la creación de servicios Web.
- ✓ Facilidad para distribuir los pedidos XML a través de diferentes tipos de mapeos.
- ✓ Soporte para mapeo de XML a objetos.
- ✓ Integración con Spring Framework.

Spring RMI

La implementación de RMI haciendo uso del framework Spring facilita en gran medida el trabajo de los programadores. Este simplifica los pasos para la creación y consumo de servicios RMI, ya que solo es necesario especificar cuales servicios exportan o importan servicios RMI.

Vale destacar que cualquier componente desarrollado en la capa de servicio puede ser expuesto como un servicio remoto sin que fuese desarrollado para tal fin. Ante significativas facilidades, presenta las desventajas de que el cliente y el servidor deben estar desarrollados en Java, y que no trabaja bien con firewalls. [7]

1.5.3 Spring Security

Spring Security es un framework escrito en Java que brinda autenticación avanzada, autorización y otros elementos de seguridad para aplicaciones mediante el uso de Spring. La autenticación y autorización son necesarias ya que una permite verificar la identidad de los agentes solicitantes y proveedores y la otra permite controlar el acceso a los recursos.

Con el objetivo de asegurar las aplicaciones Web, Spring Security usa filtros de servlets, dichos filtros interceptan las peticiones hechas a los servlets para realizar la autenticación e implementar la seguridad mediante el uso de los componentes de Spring Security. [5]

Spring Security está conformado por cinco componentes claves [11]:

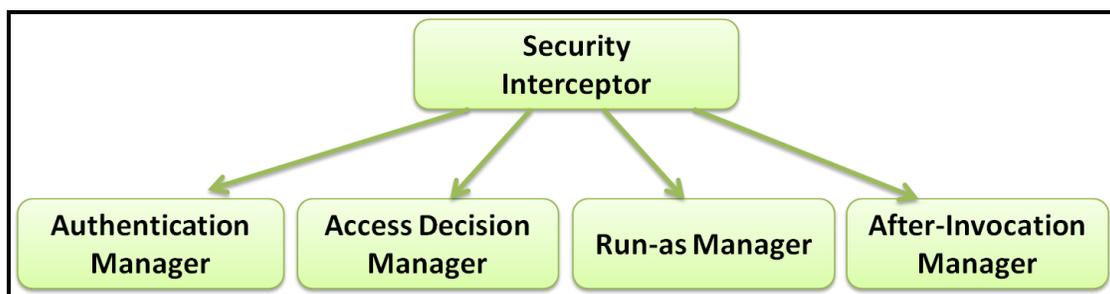


Figura 1: Elementos fundamentales de Spring Security.

Security Interceptor realmente no aplica reglas de seguridad. Intercepta el acceso a los recursos para implementar la seguridad y delega la autoridad de administrarla a los demás managers.

Authentication Manager es responsable de determinar quién es quién. Realiza esto al considerar el *principal* (generalmente un usuario) y las *credenciales* (generalmente una contraseña).

Una vez que se ha determinado quién es quién, entra en funcionamiento el **Access Decision Manager**. Este es el que realiza el proceso de autorización; decide si se puede acceder a un recurso, en dependencia de la información de la autenticación y de los atributos de seguridad asociados a cada recurso.

Cuando hay recursos con diferentes requerimientos de seguridad y, una vez autenticado y garantizado el acceso, se usa **Run-as Manager** para reemplazar la autenticación existente por una que da completo acceso a estos recursos.

After-Invocation Manager es el componente que asegura de que el usuario que ha realizado la petición tenga permitido ver los datos que están siendo retornados de un recurso con seguridad.

1.5.4 Dojo Framework

Dojo es un Framework de código abierto que soporta el desarrollo de interfaces de aplicaciones Web dinámicas. Resuelve asuntos de usabilidad comunes como la navegación, soporta cambios de URL (*Uniform Resource Locator*) en la barra de URLs para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando JavaScript no es completamente soportado en el cliente. [12]

Características [12]:

- ✓ Contiene una completa librería de componentes de interfaz de usuario.
- ✓ Contiene temas CSS (*Cascading Style Sheets*).
- ✓ Contiene formularios y rutinas de validación para los parámetros.
- ✓ Ahorra el tiempo de desarrollo y diseño de interfaces complicadas y manejos de eventos complejos.

1.6 Metodología de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Detallan la información que se debe producir como resultado de una actividad y la información necesaria para iniciar otras.

Hoy frente a un mundo donde evolucionan con mayor rapidez las tecnologías, y los problemas son cada vez más complicados, no es sencillo hacer proyectos de software. Para lograr el desarrollo de un software de manera exitosa es muy importante escoger la metodología de desarrollo que más se ajuste a las necesidades del producto y adaptarla para beneficiar el desarrollo. Actualmente existen varias metodologías, muchas realmente buenas y útiles, corresponde al equipo de trabajo escoger la más adaptable para desarrollar el proyecto.

Las metodologías de desarrollo de software se dividen en dos tipos: las tradicionales o "pesadas" y las ágiles. Las metodologías tradicionales están centradas en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán. [13]

Dentro de las metodologías tradicionales sobresale RUP (*Rational Unified Process*), que presenta como característica principal la organización que brinda y la documentación que genera. Está basado en componentes interconectados a través de interfaces y utiliza el UML (*Unified Modeling Language*) como lenguaje de modelado de procesos, es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Teniendo en cuenta que esta metodología genera una gran cantidad de documentación que es prescindible para el desarrollo de este proyecto, y que además es necesario involucrar mayor número de personas en el equipo de desarrollo, no se pretende hacer uso de ella. [14]

Por su parte, las metodologías ágiles cuentan con la ventaja de ser sencillas, tanto en su proceso de aprendizaje como en su aplicación. Son aplicables a proyectos que necesitan solución rápida y son metodologías que no generan mucha documentación. Estas cuentan con inconvenientes y restricciones que están dirigidos a pequeños y medianos equipos, además que el entorno físico permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo de desarrollo. [15]

Entre las metodologías ágiles más usadas se encuentran XP y SCRUM:

La metodología XP (*eXtreme Programming*) tiene como objetivos principales la satisfacción al cliente y potenciar al máximo el trabajo en equipo. Pero XP define que el cliente forma parte del equipo de desarrollo, lo cual no es posible durante la realización de este trabajo [13].

La metodología SCRUM está indicada para proyectos con cambios rápidos de requisitos, pero define como verdaderas protagonistas del desarrollo, reuniones diarias durante 15 minutos para coordinación e integración. Considerando que durante el proceso de desarrollo los requisitos no presentarán cambios constantes y las reuniones diarias propuestas constituyen una ineficiencia y desvío de tiempo de implementación, no se tendrá en cuenta esta metodología para el desarrollo de la aplicación.

Otra de las metodologías es la de Desarrollo basado en funcionalidades (FDD por sus siglas en inglés). Como las otras metodologías adaptables, se enfoca en iteraciones cortas que entregan funcionalidades tangibles. En el caso de FDD las iteraciones duran dos semanas. [15]

FDD (*Feature Driven Development*)

La metodología FDD se enfoca en iteraciones cortas que entregan funcionalidades tangibles y está pensada para proyectos relativamente cortos, menos de un año. Este es un enfoque ágil para el desarrollo de sistemas y su énfasis está en cómo se realizan las fases de diseño y construcción.

Acentúa el tema de calidad durante todo el proceso y permite un monitoreo permanente del avance del proyecto.

El proyecto que utilice esta metodología será dividido en cinco fases, donde las primeras tres fases ocupan parte del tiempo en las primeras iteraciones y son las dos últimas las que absorben la mayor parte del tiempo según va avanzando el proyecto, limitándose las primeras a un proceso de refinamiento.

Fases secuenciales [15]:

1. Desarrollo de un modelo general: Se obtiene una idea del contexto y del alcance del sistema.
2. Construcción de la lista de funcionalidades: El equipo define las primeras funcionalidades, las prioriza, las agrupa y las pondera.
3. Plan de releases en base a las funcionalidades a implementar: En esta fase el administrador de desarrollo, el jefe de los programadores y el administrador de proyectos establecen hitos y diseñan cronogramas de diseño y construcción, de acuerdo a las funcionalidades de la etapa anterior.
4. Diseñar en base a las funcionalidades: Definidas ya las funcionalidades y el cronograma de diseño y construcción, esta fase es donde el jefe de los programadores escoge la característica a diseñar, identifica las clases existentes. A continuación el equipo realiza el diagrama de secuencia correspondiente y el programador hace una descripción de la clase y los métodos.
5. Implementar en base a las funcionalidades: En esta etapa es donde se construyen los métodos para cada clase y luego se realizan las pruebas para cada clase e inspección de código. Después se realiza un principio de construcción, en el cual se hace la integración con las funcionalidades antes realizadas.

Esta metodología se basa en un proceso iterativo con iteraciones cortas de 2 semanas que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar. Las iteraciones se deciden en base a funcionalidades, que son pequeñas partes del software con significado para el cliente. Su uso propone tener etapas de cierre cada dos semanas, lo cual implica que los desarrolladores tendrán nuevas actividades que realizar en dicho período de tiempo. [15]

Teniendo en cuenta todo lo anterior, la mejor opción sería FDD ya que es un proceso intermedio entre XP y RUP. Esta metodología genera más documentación que XP el cual enfatiza en el código fuente; y menos documentación que RUP que garantiza una extensa documentación.

Resumiendo, se puede definir que FDD [16]:

- ✓ Ayuda al equipo a producir resultados periódicos y tangibles.
- ✓ Hace énfasis en la obtención de resultados cada dos semanas.
- ✓ Asegura en gran parte la calidad del software entregado.
- ✓ Es adaptable, pues permite realizar cambios de último momento debido a nuevos requerimientos y a las necesidades del negocio.

1.7 Herramientas

El avance de los sistemas Web se ha patentizado gracias al aporte que han recibido de las nuevas herramientas. La velocidad de creación, la calidad y la ayuda para la concepción de un servicio Web se pueden alcanzar gracias al uso de herramientas mejores y más potentes.

1.7.1 Entorno Integrado de Desarrollo. Eclipse

Un Entorno Integrado de Desarrollo (*IDE*), es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica o GUI (*Graphical User Interface*). Pueden utilizarse por uno o varios lenguajes y pueden ser aplicaciones independientes o formar parte de aplicaciones ya existentes. Proveen además un marco de trabajo amigable para la mayoría de los lenguajes.

Eclipse es un IDE de código abierto y multiplataforma. Esta es una plataforma potente y completa para la programación, desarrollo y compilación tanto de aplicaciones Java, sitios Web, programas en C++, entre otros. Este entorno de desarrollo permite trabajar con varias tecnologías asociadas. [17]

Características [17]:

- ✓ La compilación es en tiempo real.
- ✓ La programación utilizando Frameworks provee un desarrollo amplio de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones Web, entre otros.

1.7.2 Servidor Web. Apache Tomcat 6.0.24

Apache Tomcat es un servidor Web y de aplicaciones que gestiona solicitudes y respuestas HTTP. Es además el servidor de aplicaciones o contenedor de Servlets/JSP (*JavaServer Pages*). Tomcat puede funcionar como servidor Web por sí mismo. Como fue desarrollado usando el lenguaje de programación Java, puede funcionar en cualquier sistema operativo que disponga de la máquina virtual de Java. [18]

Características del servidor de aplicaciones Web Tomcat [18]:

- ✓ Es gratis, fácil de instalar, es compatible con las APIs (*Application Programming Interface*) más recientes de Java, ocupa muy poco espacio y es bastante rápido.
- ✓ Es fiable, su solidez se basa en que miles de desarrolladores contribuyen con su código.
- ✓ Tomcat tiene seguridad de nivel de aplicación a partir de la versión 4.0.

1.7.3 Sistema Gestor de Base de Datos. MySQL 5.0

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar la definición de los elementos, mantenimiento de la integridad de estos dentro de la base de datos, control de la seguridad, privacidad y manipulación de los datos. [19]

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos.

MySQL es un gestor de base de datos de código abierto. Constituye un sistema de administración de base de datos relacional, multihilo, multiusuario y multiplataforma. Tiene la posibilidad de almacenar gran capacidad y variedad de datos, y distribuirlos cubriendo la necesidad de cualquier tipo de organización. [19]

1.7.4 Herramienta de modelado. StarUML

Las herramientas CASE son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de desarrollo de un software.

Entre las ventajas de usar una herramienta CASE se encuentran que mejora la calidad de los desarrollos realizados y aumenta la productividad.

StarUML es una herramienta de modelado de código abierto para un desarrollo rápido, flexible y extensible que aunque no genera código SQL cuenta con las siguientes características. [20]

- ✓ Proporciona una arquitectura sencilla y potente.
- ✓ No está ligado a ningún lenguaje de programación específico, por lo que cualquier lenguaje puede ser utilizado para modelar sobre StarUML.
- ✓ Soporta la mayoría de los tipos de diagramas especificados en UML 2.0.

1.8 Conclusiones Parciales

Durante este capítulo se abordaron las principales tecnologías que estarán presentes en el desarrollo del portal Web. Quedan definidos las herramientas, metodologías, lenguajes y conceptos usados a lo largo del desarrollo del sistema. Se realizó durante este capítulo un estudio sobre aplicaciones existentes a nivel internacional que responden a las necesidades de las refinerías de petróleo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se realiza un análisis de cómo se llevan a cabo los procesos implícitos en el desarrollo de la aplicación. Además se describen los objetos definidos como: objetos de automatización. Se muestra la propuesta del sistema, la especificación de los requerimientos funcionales y los no funcionales. Se presenta el modelo de dominio y la definición de los casos de uso donde se definen los actores y se presenta el diagrama de casos de uso del sistema.

2.2 Problemática

Las refinерías petroleras son las encargadas de transformar el crudo de petróleo en sus productos derivados, esto se logra mediante una serie de procesos físicos y químicos ante los cuales se hace necesaria una extrema protección del personal de trabajo y del funcionamiento de la industria.

Durante las jornadas de trabajo existentes en una refinería es posible que ocurran determinados incidentes que provoquen afectaciones en la instalación, daño en sus trabajadores, o la paralización completa de la industria.

Varias de estas empresas no cuentan con sistemas de avisos y registros de incidentes, resultando difícil mantener informado a todo el personal y garantizar un almacenamiento de los datos de manera exacta. Carecen al mismo tiempo de un sistema que exponga la información referente al estado del equipamiento de la empresa.

Cuando se desencadena un incidente en un área específica dentro de la industria, es posible que otras áreas no sepan de la ocurrencia de este, poniendo en riesgo la vida de los trabajadores que no están al tanto de la situación y la integridad de la industria. Es posible que los directivos y/o personas involucradas en la toma de decisiones para salvaguardar al personal y los activos de la empresa, no estén al corriente.

Además, es muy probable que todos los trabajadores no estén al tanto de la información, haciendo más difícil la correcta toma de decisiones y mantener la integridad de los agentes de campo, al mismo tiempo que se dificulta el control de los principales eventos que ponen en riesgo la industria y sus causas.

2.3 Objeto de automatización.

La automatización persigue mejorar la productividad, mejorar las condiciones de trabajo personal, la calidad y eficiencia en las empresas e instituciones que se beneficien de procesos automatizados.

Con el objetivo de cumplir con las ventajas que esta propone, se procederá a la automatización de los procesos de gestión de la información sobre Incidentes, Lesionados, Noticias y el Estado de los componentes de una refinería petrolera. La información relacionada está compuesta por:

- ✓ Incidentes: tipo de incidente, tipo de sustancia, descripción, área, zona de ocurrencia, fecha, estado del incidente y si existen lesionados.
- ✓ Lesionados: causa, cantidad de lesionados, descripción.
- ✓ Boletines: título, noticias y fecha.
- ✓ Reportes de equipos: estado, situación y la fecha del reporte.

2.4 Propuesta del sistema.

Se propone el desarrollo de un sistema PetroWeb que podrá ser accedido por el usuario mediante una interfaz sencilla y amigable. El usuario deberá registrarse y autenticarse para acceder a las funcionalidades del sistema a las cuales esté autorizado. Mediante este, puede gestionar la información de los incidentes que surjan dentro de la refinería. Puede asimismo gestionar, en caso de existir, los lesionados que pertenecen a un incidente, almacenando datos notables del suceso. También tiene la posibilidad de listar el estado de los equipamientos reportados por los agentes de campo y gestionar las noticias relevantes para la empresa. Conjuntamente la propuesta incluye el envío de mensajes WAP Push, desde la aplicación hacia los terminales.

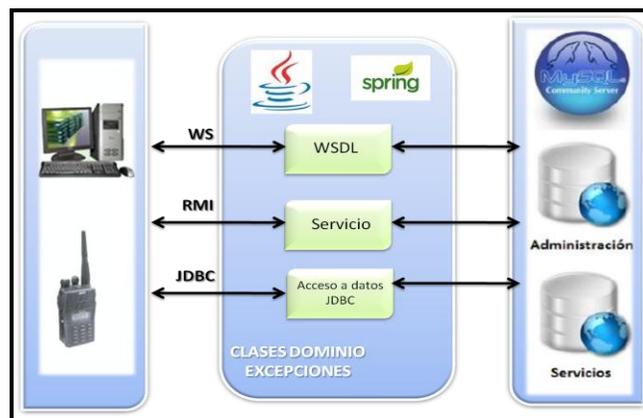


Figura 3: Propuesta del sistema

Cuando el acceso a los datos se realiza mediante servicios Web la aplicación se comunica mediante la interfaz pública de este y accede a la base de datos. Si la comunicación con los datos se realiza mediante RMI, entonces el sistema se comunica con la interfaz del servicio RMI y luego accede a la base de datos. En caso que la aplicación no necesite acceder a los datos de una manera distribuida, puede realizarlo mediante JDBC, accediendo desde la misma aplicación a la base de datos. El protocolo de comunicación SOAP está presente durante el acceso a los datos por servicios Web.

2.5 Modelo de Dominio

Para desarrollar cualquier sistema, independientemente de su complejidad, es necesario dividirlo en secciones, estas se pueden representar mediante modelos que permitan abstraer sus características esenciales. Como paso previo al análisis y diseño del sistema es preciso realizar un Modelo de Negocio o Modelo de Dominio para capturar y enunciar la visión del proceso, encaminado a satisfacer las necesidades durante estas fases. [14]

Para la realización de un Modelo de Negocio es necesario previamente haber realizado una identificación de los procesos existentes; que estos no tengan un bajo nivel de estructuración y además la existencia de flujos de información bien interconectados y definidos. Mientras tanto, el Modelo de Dominio es un concepto menos amplio que el Modelo de Negocio, que se centra en una parte del negocio relacionada con el ámbito del proyecto. Considerando que durante el estudio de los procesos presentes en una refinería petrolera no ha sido posible definir de manera clara los procesos que se desarrollan dentro de esta, se realizó para la comprensión común que ayude a usuarios, clientes, desarrolladores e interesados a entender el contexto en que se ubica la aplicación, el Modelo de Dominio.

El Modelo de Dominio posibilita la comprensión de los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. La representación de este mediante un diagrama de clases no contiene los conceptos propios de un sistema de software, sino de la propia realidad física y mediante este se pueden obtener correctamente los requerimientos. En el siguiente Modelo de Dominio se especifican las relaciones que existen entre los principales conceptos que interactúan en el sistema:

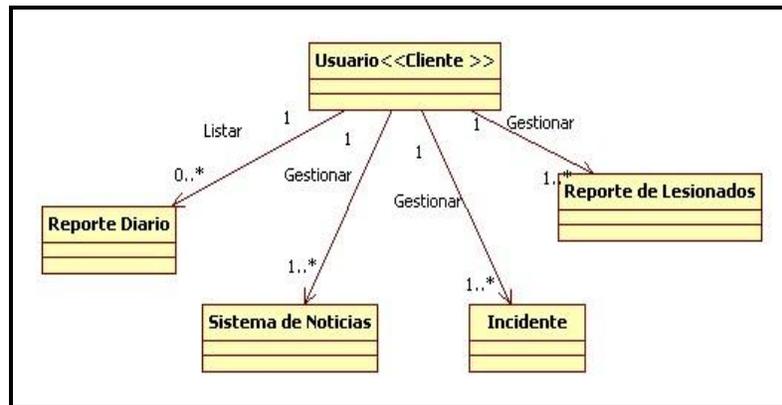


Figura 2: Modelo de Dominio.

Descripción de los objetos del dominio

- ✓ Cliente: actor que gestiona y lista todos los objetos.
- ✓ Incidente: objeto que representa los datos de los incidentes.
- ✓ Reporte de Lesionados: objeto que representa los datos de los lesionados.
- ✓ Sistema de Noticias: objeto que representa los datos de las noticias.
- ✓ Reporte Diario: objeto que representa los datos de los equipos de la refinería.

2.6 Relación de los requerimientos.

Los requisitos del software sustentan la tarea del diseño, convirtiéndose este en la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema finalizado. [14]

2.6.1 Requerimientos funcionales.

Los requerimientos funcionales son las capacidades o condiciones que el sistema debe cumplir.

RF 1: Autenticar Usuario.

RF 2: Gestionar Incidente.

- 2.1 Permitir a los usuarios reportar Incidente.
- 2.2 Permitir a los usuarios actualizar Incidente.
- 2.3 Permitir a los usuarios eliminar Incidente.
- 2.4 Permitir a los usuarios listar Incidentes.

RF 3: Gestionar Reporte de Lesionados

3. 1 Permitir a los usuarios actualizar Reporte de Lesionados.
3. 2 Permitir a los usuarios eliminar Reporte de Lesionados.

3. 3 Permitir a los usuarios listar Reportes de Lesionados.

RF 4: Gestionar Sistema de Noticias

- 4. 1 Permitir a los usuarios reportar Noticia.
- 4. 2 Permitir a los usuarios actualizar Noticia.
- 4. 3 Permitir a los usuarios eliminar Noticia.
- 4. 4 Permitir a los usuarios listar Noticias.

RF 5: Listar Reporte Diario.

RF 6: Enviar Mensajes WAP.

- 6. 1 Enviar un Mensaje *Service Indication*.
- 6. 2 Enviar un Mensaje *Service Loading*.

RF 7: Registrar eventos y errores.

RF 8: Registrar las trazas de las operaciones de los usuarios.

2.6.2. Requerimientos no funcionales.

Los requerimientos no funcionales son cualidades con las que debe contar el producto a fin de hacerlo atractivo, usable, rápido y/o confiable; esto podría marcar la diferencia entre un producto bien aceptado y otro no tan aceptado. Los requerimientos funcionales presentes son:

RNF 1. Portabilidad:

El sistema será multiplataforma.

RNF 2. Apariencia o Interfaz Externa:

La aplicación propuesta poseerá una interfaz sencilla, amigable y fácil de utilizar, dirigida directamente al usuario del sistema.

RNF 3. Soporte:

La Universidad de las Ciencias Informáticas debe brindar soporte tanto a los clientes como a los administradores que necesiten capacitación para operar con el sistema.

RNF 4. Software:

El sistema requiere del servidor Apache Tomcat 6.0.24 instalado y la Máquina Virtual de Java 1.6.0 para su funcionamiento. Es necesaria la instalación y configuración del servidor de base de datos MySQL 5.0.

2.7 Definición de los casos de uso.

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

2.7.1 Definición de los actores

Actores	Justificación
Usuario	Representa a un usuario que va a realizar las operaciones en el sistema.

Tabla 1: Definición de los actores

2.7.2 Diagrama de casos de uso

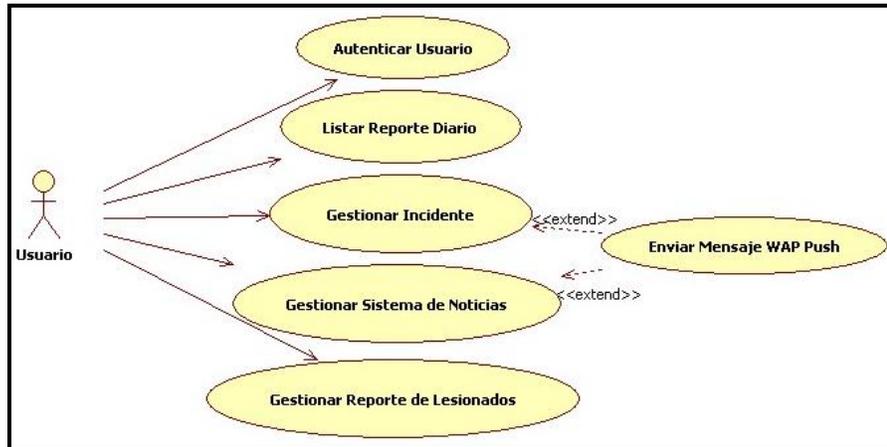


Figura 3: Diagrama de caso de uso del sistema

2.7.3 Casos de uso expandidos

Caso de Uso	
CU_2	Gestionar Reporte de Lesionados
Propósito	Actualizar, Eliminar y Listar los datos de los lesionados.

Actores: El Usuario.	
Resumen: El caso de uso se inicia cuando el usuario accede a la opción gestionar los datos de los Reportes de Lesionados. El sistema permite Actualizar, Listar y Eliminar un reporte de lesionados. Si selecciona la opción Actualizar un reporte de lesionados introduce los datos del reporte a actualizar y el sistema lo actualiza. En el caso de Eliminar, el usuario escoge el reporte de lesionados y el sistema lo elimina. Si selecciona la opción de Listar Lesionados el sistema consulta los datos y muestra la lista de coincidencias. Si desea Listar por algún criterio en específico el sistema permite filtrar los resultados mostrados por diferentes parámetros. El caso de uso termina.	
Prototipo:	
Referencias	RF 7, RF 8.
Acción del Usuario	Respuesta del Sistema
1. El usuario selecciona la opción: "Gestionar Lesionados".	2. Muestra las opciones: <ul style="list-style-type: none"> ▪ Listar Lesionados ▪ Actualizar Lesionados ▪ Eliminar Lesionados
Sección: "Actualizar Lesionados"	
Prototipo	
	
3. El usuario selecciona la opción "Actualizar Lesionados".	4. Muestra los parámetros de búsqueda y los botones "Buscar" y "Cancelar": <ul style="list-style-type: none"> ▪ Fecha ▪ Causa

<p>5. El usuario selecciona el criterio de búsqueda y da clic en el botón “Buscar”.</p>	<p>6. Muestra los datos de los reportes de lesionados que coincidan con los datos introducidos por el usuario:</p> <ul style="list-style-type: none"> ▪ Fecha ▪ Causa ▪ Cantidad de lesionados ▪ Incidente ▪ Descripción
<p>7. El usuario selecciona el reporte de lesionados y modifica los datos que desee.</p>	
<p>8. El usuario da clic en el botón “Actualizar”.</p>	<p>9. Verifica la estructura de los cambios.</p>
	<p>10. Actualiza los datos del reporte de lesionado.</p>
	<p>11. Se registran eventos y errores y las trazas de las operaciones de los usuarios.</p>
	<p>12. Se muestra un mensaje al usuario confirmando que la operación se realizó correctamente.</p>
<p>Flujo Alternativo 5a Opción Campos Vacíos en la búsqueda</p>	
<p>5a1. El usuario no selecciona ningún campo para realizar la búsqueda y da clic en el botón “Buscar”.</p>	<p>5a2. El sistema muestra un mensaje de error “Debe llenar al menos un campo para realizar la búsqueda.”. Regresa a la Acción 4.</p>
<p>Flujo Alternativo 7a Opción Estructura de algún campo incorrecta.</p>	
<p>7a1. El usuario introduce algún dato erróneo y da clic en el botón “Actualizar”.</p>	<p>7a2. Se muestra el mensaje de error “Debe llenar los campos correctamente.”. Regresa a la Acción 6.</p>

Flujo Alternativo 7b Opción Campos Vacíos	
7b1. El usuario no llena todos los campos obligatorios y da clic en el botón "Actualizar".	7b2. El sistema muestra un mensaje de error "Debe llenar todos los campos.". Regresa a la Acción 6.
Flujo Alternativo 9a Opción Fallo en la actualización de los datos.	
	9a1. Se le envía un mensaje de error notificándole que hubo un fallo en la actualización de datos en la base de datos.
Flujo Alternativo *a Opción Cancelar	
*a1. El usuario selecciona la opción "Cancelar".	a2. Redirecciona al usuario a la Página Listar Lesionados.
Sección: "Eliminar Lesionados"	
Prototipo	
	
3. El usuario selecciona la opción "Eliminar Lesionados".	4. Muestra los parámetros de búsqueda y los botones "Buscar" y "Cancelar" : <ul style="list-style-type: none"> ▪ Fecha ▪ Causa
5. El usuario selecciona el criterio de búsqueda y da clic en el botón "Buscar".	6. Muestra los datos de los reportes de lesionados que coincidan con los datos introducidos por el usuario: <ul style="list-style-type: none"> ▪ Fecha ▪ Causa ▪ Cantidad de lesionados ▪ Incidente ▪ Descripción

7. El usuario selecciona el reporte de lesionados y da clic en el botón “Eliminar”.	8. Muestra un mensaje verificando si el usuario desea eliminar el reporte de lesionados seleccionado.
9. El usuario acepta eliminar el reporte de lesionados.	10. Se elimina el reporte de lesionados.
	11. Se registran eventos y errores y las trazas de las operaciones de los usuarios.
	12. Se muestra un mensaje al usuario confirmando que la operación se realizó correctamente.
Flujo Alterno 5a Opción Campos Vacíos en la búsqueda	
5a1. El usuario no selecciona ningún campo para realizar la búsqueda y da clic en el botón “Buscar”.	5a2. El sistema muestra un mensaje de error “Debe llenar al menos un campo para realizar la búsqueda.”. Regresa a la Acción 4.
Flujo Alterno 7a Opción Eliminar sin seleccionar un reporte de lesionados.	
7a1. El usuario da clic en el botón “Eliminar” sin seleccionar ningún reporte de lesionado.	7a2. El sistema muestra un mensaje “Debe seleccionar algún resultado.”. Regresa a la acción 6.
Flujo Alterno 9a Opción Cancelar Eliminar	
9a1.El usuario selecciona la opción “Cancelar”.	9a2.Regresa a la acción 7.
Flujo Alterno 10a Opción Fallo en la eliminación de los datos.	
	10a1. Se le envía un mensaje de error notificándole que hubo un fallo en la eliminación de los datos en la base de datos.

Flujo Alternativo a Opción Cancelar	
*a1. El usuario selecciona la opción "Cancelar".	*a2. Redirecciona al usuario a la Página Listar Lesionados.
Sección: "Listar Lesionados"	
Prototipo: 	
3. El usuario selecciona la opción "Listar Lesionados".	4. Muestra una lista con todos los reportes de lesionados insertados en la base de datos y además los parámetros de búsqueda y el botón "Buscar": <ul style="list-style-type: none"> ▪ Fecha ▪ Causa
5. El usuario selecciona el criterio de búsqueda y da clic en el botón "Buscar".	6. Muestra los datos de los reportes de lesionados que coincidan con los datos introducidos por el usuario: <ul style="list-style-type: none"> ▪ Fecha ▪ Causa ▪ Cantidad de lesionados ▪ Incidente ▪ Descripción
	7. Se registran eventos y errores y las trazas de las operaciones de los usuarios.
	8. Se muestra un mensaje al usuario confirmando que la operación se realizó correctamente.

Flujo Alternativo 5a Opción Campos Vacíos en la búsqueda	
5a1. El usuario no selecciona ningún campo para realizar la búsqueda y da clic en el botón "Buscar".	5a2. El sistema muestra un mensaje de error "Debe llenar al menos un campo para realizar la búsqueda." Regresa a la Acción 4.
Flujo Alternativo 6a Opción Fallo en la búsqueda de los datos.	
	6a1. Se le envía un mensaje de error notificándole que hubo un fallo para acceder a los datos en la base de datos.
Prioridad	Crítico.

Tabla 2: Caso de uso Gestionar Reporte de Lesionados

2.8 Conclusiones Parciales

Durante este capítulo se realizó un análisis de los objetivos estratégicos de la organización y de los procesos de negocio que lo soportan. Se expuso la descripción de la propuesta del sistema, se definieron los requisitos funcionales y los no funcionales y se mostró el modelo de dominio, el diagrama de casos de uso del sistema, la descripción de los actores y una descripción de uno de los casos de uso más significativos.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

Durante este capítulo se describen los procesos que se llevan a cabo durante la fase Diseñar en base a las funcionalidades, por lo que se exponen el diagrama de diseño de clases, el diagrama de secuencia por funcionalidad y diagramas de paquetes. También se muestra el Diagrama Entidad Relación, la arquitectura del sistema y los patrones de diseño.

3.2 Diseño

El diseño es la representación significativa de la ingeniería del sistema que se va a construir. Va cambiando continuamente a medida que se desarrollan nuevos métodos, mejores análisis y se amplía el conocimiento del proceso. [14]

3.3 Diagrama de Paquetes

Los paquetes de diseño son usados fundamentalmente como herramienta organizacional del modelo para agrupar elementos, ya sea una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén relacionados de alguna forma. El diagrama de paquetes está destinado a estructurar el modelo de diseño dividiéndolo en partes más pequeñas sin una interfaz formal. A continuación se presenta el diagrama de paquetes de la futura aplicación.

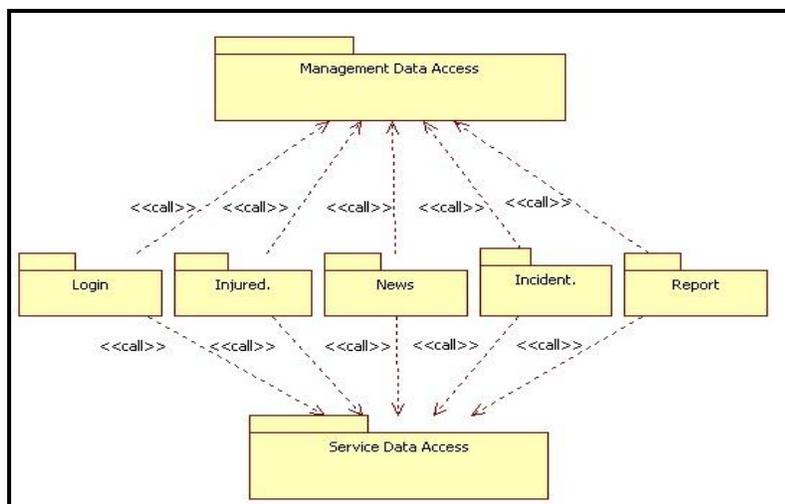


Figura 4: Diagrama de paquetes de la aplicación

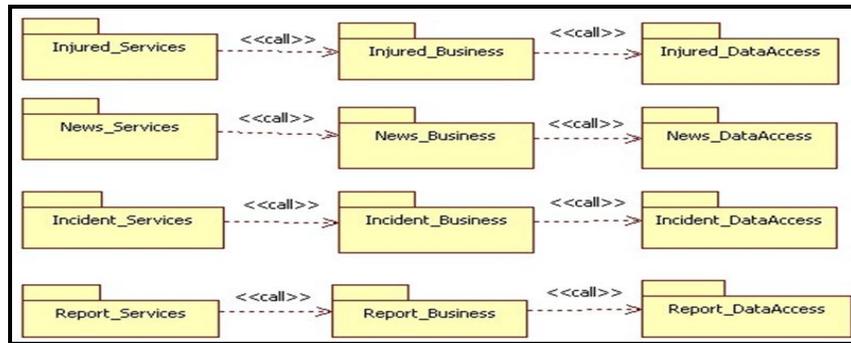


Figura 5: Diagrama de paquetes de acceso a datos

3.4 Diagrama de Clases del Diseño

Durante el diseño el Diagrama de Clases se elabora para tener en cuenta los detalles concretos de la implementación del sistema. Este describe gráficamente las especificaciones de las clases del software y las interfaces en una aplicación, conteniendo definiciones de las entidades del software en vez de conceptos del mundo real. Su uso expresa la definición de las clases como componentes del software. [14]

A continuación se muestran los diagramas presentes en la funcionalidad Salvar Reporte de Lesionados y la navegabilidad entre sus objetos:

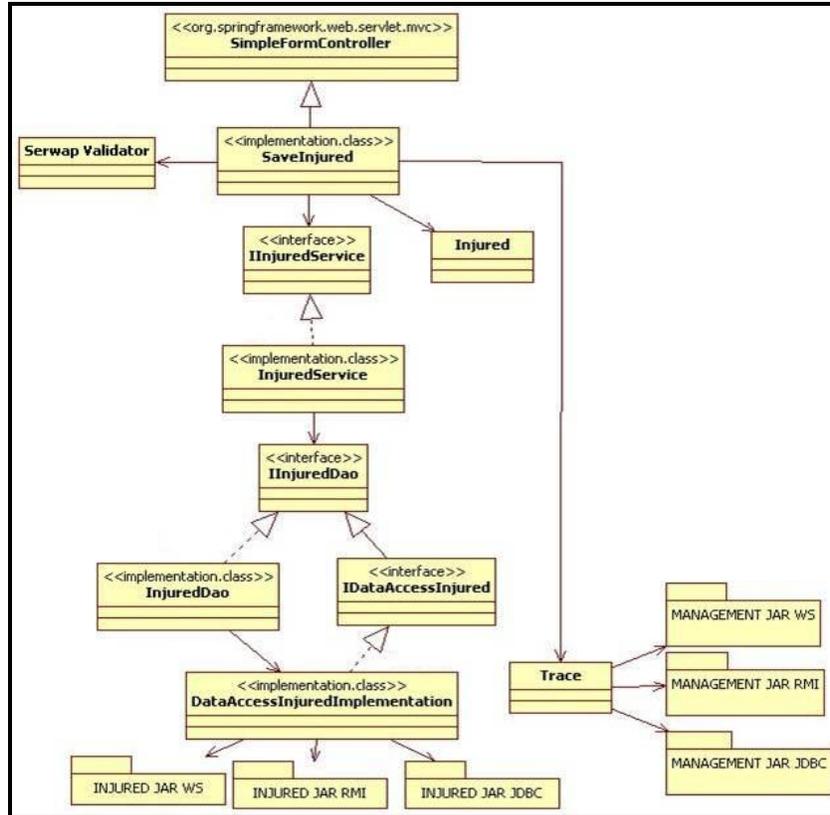


Figura 6: Diagrama de clases del portal Web de la funcionalidad SaveInjured

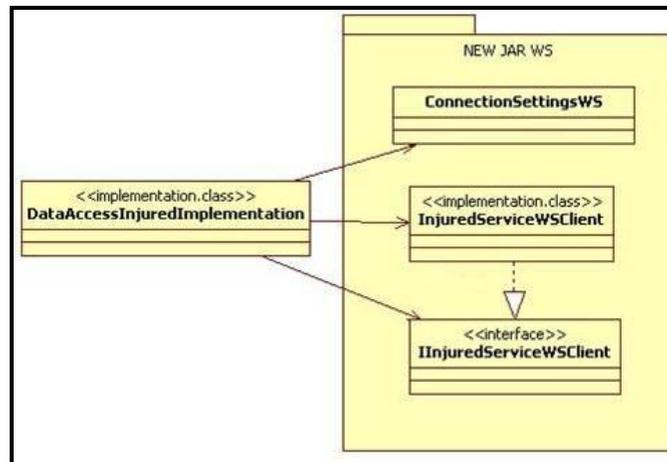


Figura 7: Diagrama de clases del paquete INJURED JAR WS

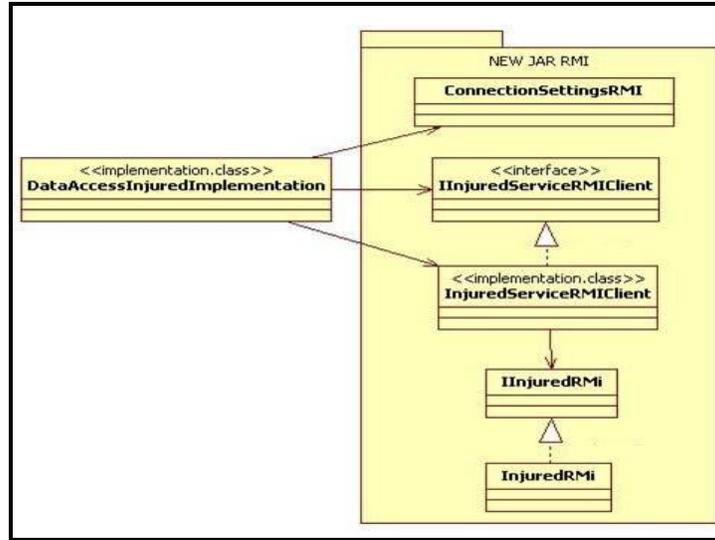


Figura 8: Diagrama de clases del paquete INJURED JAR RMI

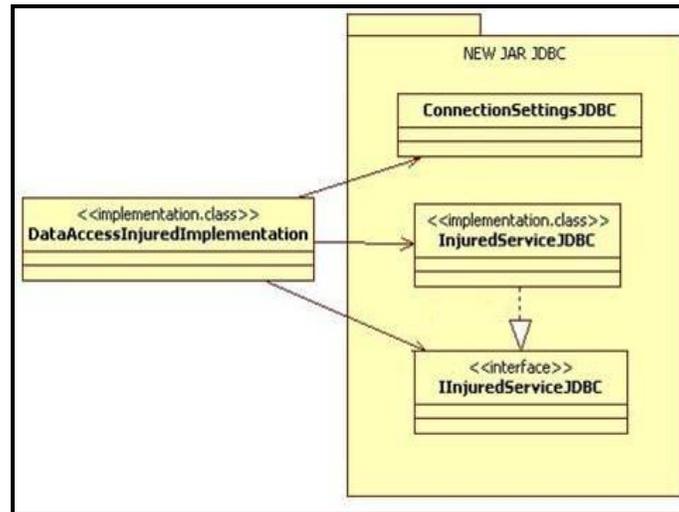


Figura 9: Diagrama de clases del paquete INJURED JAR JDBC

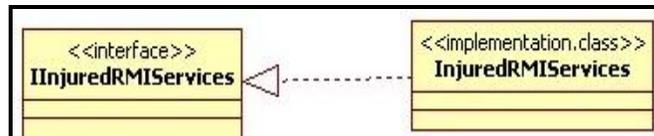


Figura 10: Diagrama del paquete Services del servicio RMI de Injured

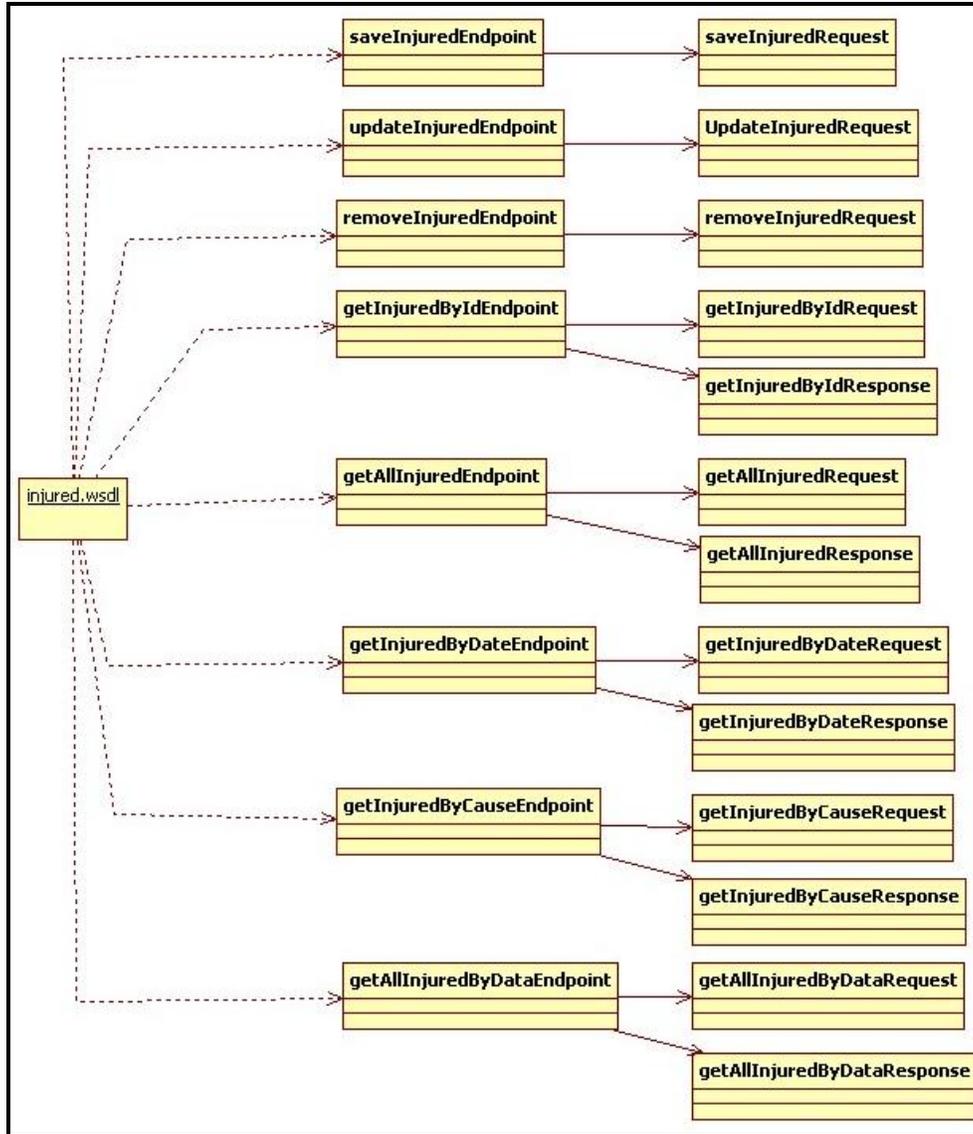


Figura 11: Diagrama del paquete Services del servicio Web de Injured

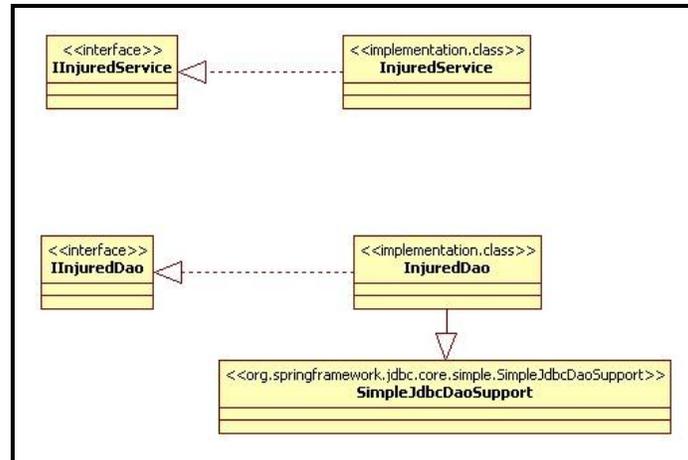


Figura 12: Diagrama de clases de los servicios Web y RMI paquetes Business y Dataaccess de Injured

3.5 Diagramas de interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. Por lo general abarcan todo en el contexto de un escenario que ilustra un comportamiento, como por ejemplo modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, conjuntamente con los mensajes enviados entre ellos. [14]

Útiles para construir sistemas ejecutables por medio de la ingeniería directa o inversa, consisten en un conjunto de objetos y sus relaciones. Los diagramas de secuencia y de colaboración son diagramas de interacción; el primero destaca la ordenación de los mensajes y el segundo destaca la organización estructural de los objetos que envían y reciben mensajes. [14]

3.5.1 Diagrama de secuencia

En este diagrama se muestran los objetos, sus relaciones y los mensajes enviados entre ellos. A continuación se muestra el diagrama de secuencia de un escenario de uno de los casos de usos más significativos, Salvar Lesionado accediendo a los datos mediante JDBC.

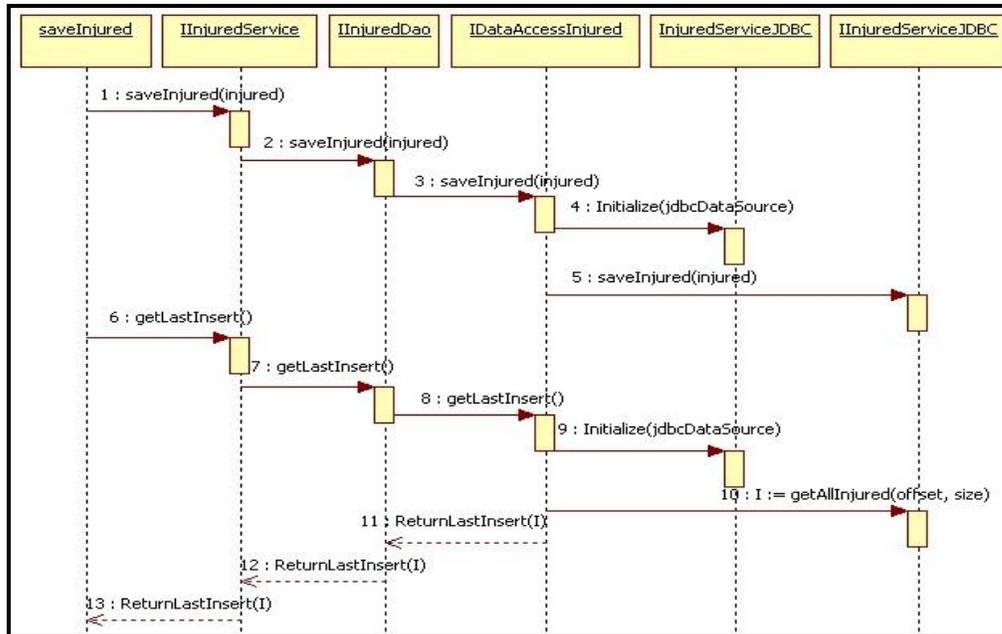


Figura 13: Diagrama de secuencia del escenario Salvar Lesionado, CU Gestionar Lesionado

3.6 Diseño de la base de datos.

El diseño de la base de datos es una de las tareas más importantes en la construcción de un sistema que utilice una base de datos. Se construye con el fin de poder acceder a la información de una manera eficiente y con la menor redundancia posible. Es un diseño para facilitar el acceso y mantenimiento a la información de manera estándar. Su buen diseño garantiza un software eficiente en el acceso a la información.

Modelo de Datos.

Un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan estos entre sí.

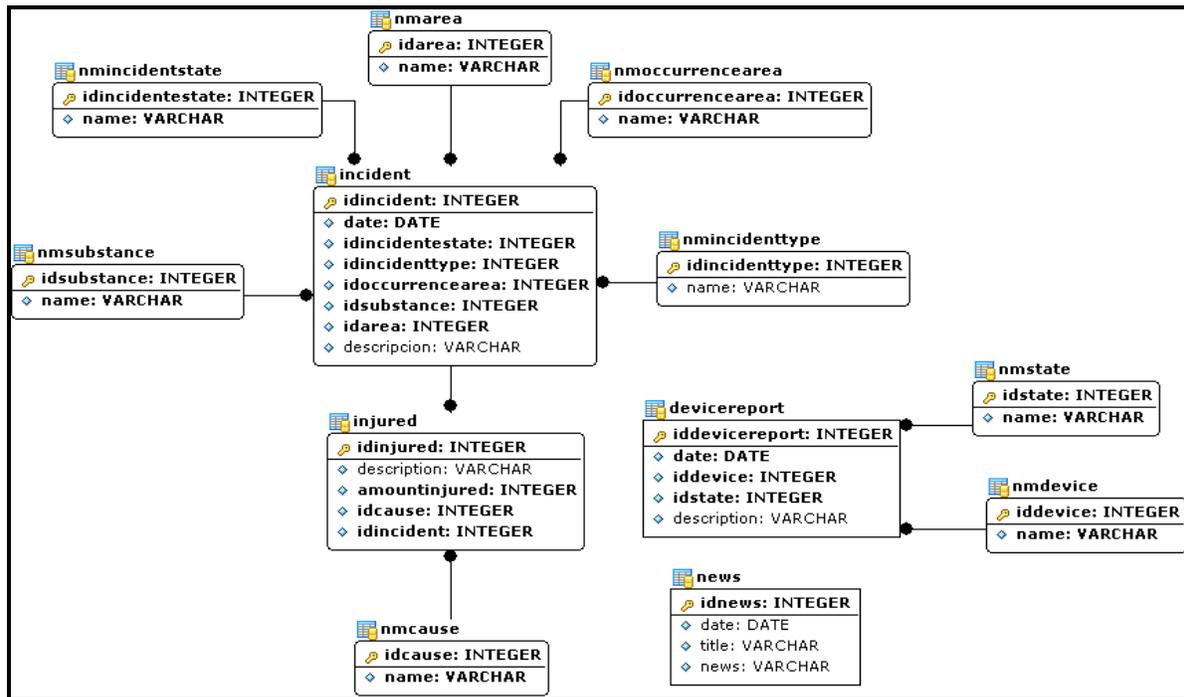


Figura 14: Modelo de datos

3.7 Arquitectura

La Arquitectura es el esqueleto o base de una aplicación, que contiene estilos y patrones arquitectónicos. Un estilo arquitectónico encapsula decisiones esenciales sobre los elementos arquitectónicos y enfatiza restricciones importantes de los elementos y sus relaciones posibles. Se afirma que son artefactos de la ingeniería porque definen clases de diseño junto con las propiedades conocidas asociadas a ellos. Por su parte, los patrones arquitectónicos expresan un esquema organizativo estructural fundamental para sistemas de software.

Uno de los estilos de arquitectura es el conocido Estilo de Llamada y Retorno que permite al diseñador del software construir una estructura relativamente fácil de modificar y ajustar a escala. [14]

Dentro del Estilo de Llamada y Retorno está presente el Patrón de Arquitectura en Capas que permite a los implementadores fraccionar un problema complejo en una secuencia de pasos incrementales, además que admite muy naturalmente optimizaciones y refinamientos, y proporciona amplia reutilización. Define una organización jerárquica, donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. [21]

Una especialización muy usada de la arquitectura en capas es la arquitectura de tres capas donde se observan muy bien delimitadas las responsabilidades de cada funcionalidad en la aplicación: Presentación, Negocio y Datos.



Figura 15: Arquitectura 3 Capas.

La capa de Presentación es la que ve el usuario, comunica y captura la información de este. Esta capa se comunica únicamente con la capa de Negocio. La capa Negocio es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso de negocio. Esta capa se comunica con la capa de Presentación para recibir las solicitudes y presentar los resultados y con la capa de Datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. Por su parte la capa de Datos es donde residen los datos y es la encargada de acceder a los mismos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de Negocio. [21]

El patrón de arquitectura Modelo – Vista – Controlador (MVC) separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

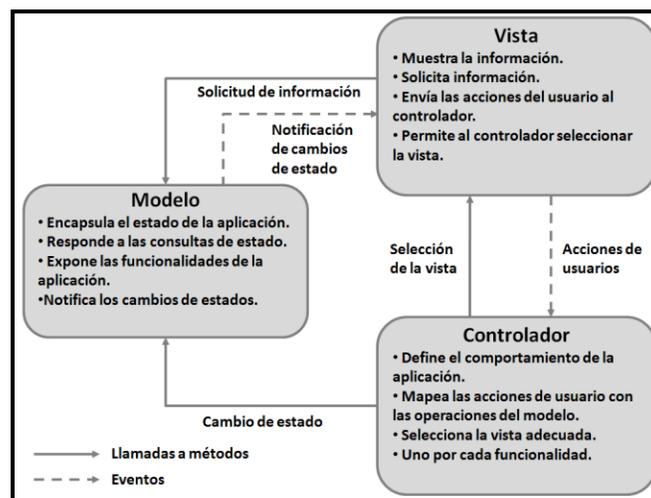


Figura 16: Arquitectura MVC

El modelo representa los datos y las reglas de negocio que rigen su acceso y actualización; puede verse como una modelación de los procesos del mundo real. Las vistas se encargan de presentar los datos obtenidos del modelo y mantener la información actualizada. El controlador actúa como un traductor de las acciones que se realizan en las vistas en las operaciones que ocurren en el modelo. Las acciones realizadas por el modelo desencadenan la activación de procesos de negocio o cambian el estado del modelo. Sobre la base de las acciones del usuario y los resultados del modelo el controlador responde mediante la selección de la vista apropiada. [21]

La arquitectura del sistema cuenta con el patrón de Arquitectura MVC de manera general, y en su componente modelo, se encuentra la Arquitectura 3 capas.

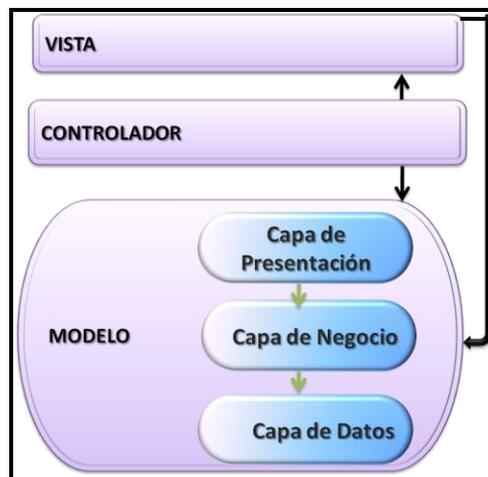


Figura 17: Arquitectura del sistema

El uso de ambas arquitecturas en el desarrollo de la aplicación brinda las siguientes ventajas:

- ✓ Facilidad en la evolución del sistema, ya que los cambios solo deben afectar a la capa donde se encuentre la modificación sin afectar al resto.
- ✓ Mantenimiento y soporte más sencillo.
- ✓ Mayor flexibilidad al añadir nuevos módulos para dotar al sistema de nuevas funcionalidades.
- ✓ Aplicaciones más robustas debido al encapsulamiento.
- ✓ Diseño basado en niveles de abstracción creciente.
- ✓ Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista.
- ✓ Adaptación al cambio y soporte para nuevos dispositivos.

3.8 Patrones de Diseño

Una forma de reutilizar las soluciones a los problemas de diseño comúnmente presentes en el desarrollo de software es mediante el uso de los patrones de diseño. Estos describen un problema que ocurre en múltiples ocasiones en un mismo entorno, así como la solución al mismo, de tal modo que se puede utilizar esta solución varias veces sin tener que volver a pensarla.

El uso de los patrones de diseño beneficia en gran medida el desarrollo y la calidad del software. Permite evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, permite además formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño. [14]

Durante el desarrollo de la aplicación se han usado los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns*), traducido al español como Patrones de Principios Generales para Asignar Responsabilidades:

Experto: Indica el principio básico de asignación de responsabilidad, usado durante la implementación del sistema para asignar responsabilidades a las clases que cuentan con la información necesaria para cumplirlas. Su uso proporciona un sistema más fácil de mantener y ampliar, además ofrece la posibilidad de reutilizar los componentes en futuras aplicaciones.

La Figura 19 expone como se presenta el patrón Experto en el sistema, la clase `DataAccessInjuredImplementation`, es la responsable de implementar la clase `IDataAccessInjured`, cuenta además con la responsabilidad de verificar los permisos del usuario, para luego acceder a los métodos solicitados.

```
public class DataAccessInjuredImplementation implements IDataAccessInjured {

    private IInjuredService injuredService;

    @Override
    public List<Injured> getAllInjured() throws Exception {
        try {
            User user=(User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();

            initializeInjured(user.getUsername(), user.getPassword());

            return injuredService.getAllInjured();
        } catch (Exception e) {
            throw e;
        }
    }
}
```

Figura 18: Muestra del patrón Experto en el sistema

Creador: Indica quién debe ser el responsable de la creación de una nueva instancia de alguna clase. Su uso en el desarrollo de este sistema brinda la posibilidad de mantenimiento, mayor claridad y reutilización.

La Figura 20 presenta un ejemplo del patrón Creador en el sistema. La clase SaveIncident es la clase encargada de crear las instancias que serán necesarias en el transcurso de la implementación.

```
public class SaveIncident extends SimpleFormController {

    private IIncidentServices incidentsServices;
    private IUserService userService;
    private static final Log logger = LogFactory.getLog(SaveNews.class);
    private Internacionalization internacionalization;
    private Trace trace;
    private IWAPPushService wapPushService;
    private WAPMessageOperationsImplementations wapPushOperations;
```

Figura 19: Muestra del patrón Creador en el sistema

Controlador: Asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Este funciona como intermediario entre los datos del usuario y las clases según los métodos necesarios. Ha sido usado en la implementación del sistema separado por capas, aislando las funciones: lógica del servicio, lógica del negocio y lógica de datos.

La figura 21 presenta un ejemplo del patrón Controlador en el sistema. La clase SaveInjured, es la encargada de controlar el evento que atiende una petición del usuario que luego responde mediante clases y métodos.

```
public class SaveInjured extends SimpleFormController {

    private IInjuredService injuredService;
    private Trace trace;
    private static final Log logger = LogFactory.getLog(ListNews.class);

    public void setTrace(Trace trace) {
        this.trace = trace;
    }

    public void setInjuredService(IInjuredService injuredService) {
        this.injuredService = injuredService;
    }

    public SaveInjured() {}

    protected void initBinder(HttpServletRequest request, ServletRequestData

    protected void onBind(HttpServletRequest request, Object command, Bind

    protected ModelAndView onSubmit(HttpServletRequest request, HttpServlet
```

FIGURA 20: MUESTRA DEL PATRÓN CONTROLADOR EN EL SISTEMA

Alta Cohesión: Indica la fuerza relativa funcional del módulo, donde cada elemento debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. El uso de interfaces en Java permite que el sistema cuente con los principios de alta cohesión, además de agrupar funcionalidades fácilmente reutilizables.

La figura 22 expone la presencia del patrón Alta Cohesión, donde las clases relacionadas con los incidentes responden a los procesos relacionados con incidentes, lo mismo sucede con las clases que representan las labores de los lesionados. Cada una de estas clases realiza una labor específica que responde solo a las necesidades que debe resolver.

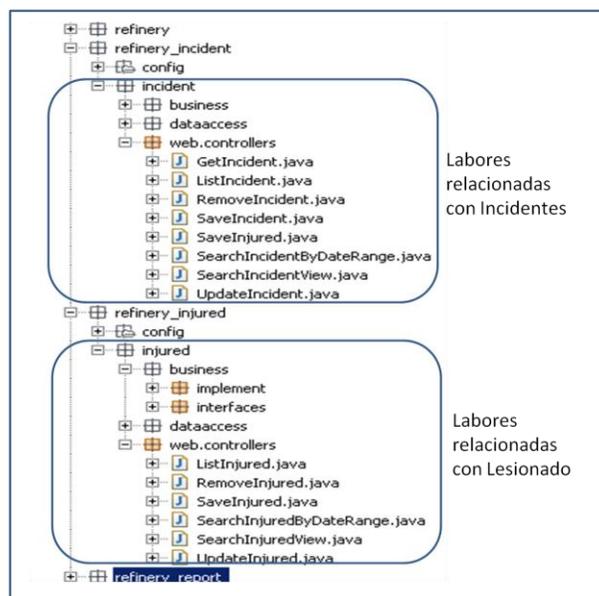


Figura 21: Muestra del patrón Alta Cohesión en el sistema

Bajo Acoplamiento: Indica la independencia existente entre módulos, es decir, tener las clases lo menos ligadas posible entre sí. Su uso garantiza que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

La figura 23 expone la presencia del patrón Bajo Acoplamiento, donde se muestra la independencia de los paquetes services, bussines y dataaccess. La relación entre los paquetes se realiza mediante un fichero de configuración, donde se relacionan primero las clases services con las del bussines y luego las del bussines con las del dataaccess. En caso de alguna modificación en alguna de las clases, las demás no se verán afectadas, esto sucede gracias a que las clases de la capas inferiores, no son consientes de ningún detalle de clases superiores.

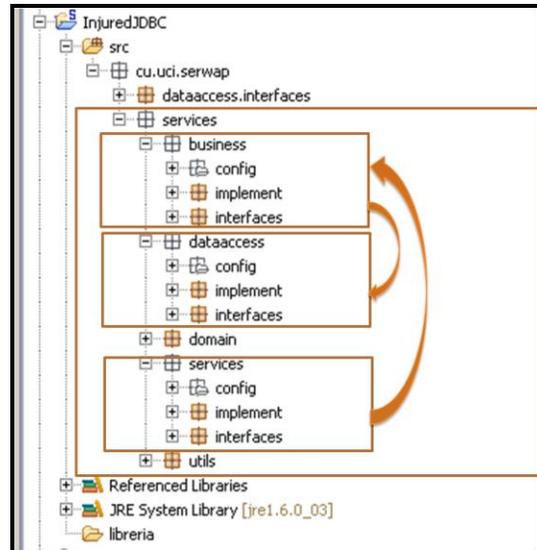


Figura 22: Muestra del patrón Bajo Acoplamiento en el sistema

También se ha hecho uso de un patrón que brinda útiles ventajas durante el desarrollo de la aplicación, el patrón de Inyección de dependencias, que consiste en inyectar objetos a una clase en vez de hacerlo la misma clase. Presente en el sistema por el uso de un agente externo, un contenedor con la responsabilidad de inyectar a cada objeto los objetos necesarios según las relaciones plasmadas en un fichero de configuración e implementado mediante Spring.

La figura 24 muestra la presencia del patrón Inyección de dependencias. La clase `InjuredServices` del paquete `business` recibe un objeto, denominado `injuredService` desde la clase `InjuredServiceJDBC` del paquete `service`. La clase `JdbcInjured` del paquete `dataaccess` recibe un objeto denominado `injuredDao` de la clase `InjuredService` del paquete `business`. De esta forma se crea una instancia de una clase en otra.

```
<!-- Service beans declarations -->
<bean id="InjuredServiceJDBC" class="cu.uci.serwap.services.services.implement.InjuredServiceJDBC">
<property name="injuredService" ref="injuredService"></property>
</bean>

<!-- Business beans declarations -->

<bean id="injuredService" class="cu.uci.serwap.services.business.implement.InjuredServices">
  <property name="injuredDao" ref="injuredDao" />
</bean>

<!-- DAO beans declaration -->
<bean id="injuredDao" class="cu.uci.serwap.services.dataaccess.implement.JdbcInjured">
<property name="dataSource" ref="dataSource"></property>
</bean>
```

Figura 23: Muestra del patrón Inyección de dependencias en el sistema

3.9 Conclusiones Parciales

En este capítulo se elaboraron los diagramas de paquetes, además los diagramas de clases del diseño y de secuencia de uno de los casos de uso más significativos, lo que ha permitido obtener una visión del sistema, y mostrar el flujo de operación presente durante cada acción solicitada. Se mostró además el modelo de datos, el estilo y patrón arquitectónico presente en la aplicación y los patrones de diseño usados durante el desarrollo del sistema.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

Con el objetivo de establecer una línea base de la arquitectura, clarificar los requisitos restantes y completar el desarrollo de la aplicación se exponen mediante este capítulo las características de la fase de Implementación en base a las funcionalidades. Se describen cómo los elementos del Modelo de Diseño se implementan en términos de componentes y su organización de acuerdo a los nodos específicos mediante el Modelo de Despliegue, además de las pruebas realizadas a todas las funcionalidades necesarias.

4.2 Implementación

Durante la implementación del sistema se define la organización del código, se implementan los elementos del diseño, se integran los resultados producidos en un sistema ejecutable y se distribuyen físicamente asignando componentes ejecutables a nodos.

4.2.1 Diagrama de despliegue

El diagrama de despliegue indica la situación física de los componentes lógicos desarrollados, mostrando cómo y dónde se desplegará el sistema. Se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Cada hardware se representa como un nodo, la ubicación es guiada por el uso de las especificaciones de despliegue.

A continuación se presenta el diagrama de despliegue del sistema:

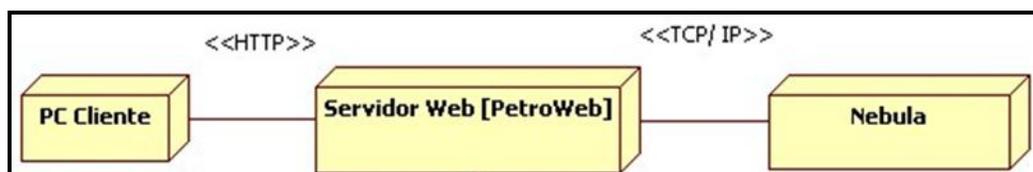


Figura 24: Diagrama de Despliegue

PC Cliente:

Es la PC Cliente la que se conecta a la Plataforma PETROWEB para visualizar las aplicaciones Web desarrolladas.

Plataforma PETROWEB:

Es el servidor que contiene a las aplicaciones y software del proyecto, a los cuales se accederán desde las máquinas clientes: Portal Web, servicios Web, RMI y las bases de datos.

Infraestructura NEBULA:

NEBULA es una Infraestructura TETRA de 2ª generación desarrollada por la empresa española Teltronic. Su estructura interna está basada en Ethernet / IP, la cual permite una conexión directa vía Ethernet de las estaciones base al nodo central y una mayor integración con otras soluciones IP y de comunicaciones.

4.2.2 Diagrama de componentes.

Los diagramas de componentes muestran la organización de los componentes del sistema. Modelan los aspectos físicos de este, es decir, un conjunto de elementos físicos y sus relaciones. Un componente se corresponde con una o varias clases, interfaces o colaboraciones. Exponen los subsistemas de implementación y sus dependencias de importación y los subsistemas de implementación organizados en capas.

A continuación se expone el diagrama de subsistemas de implementación del sistema PetroWeb, el diagrama de componente de una de las funcionalidades más significativas la funcionalidad de Lesionado, y algunos de los componentes presentes en las librerías del sistema.

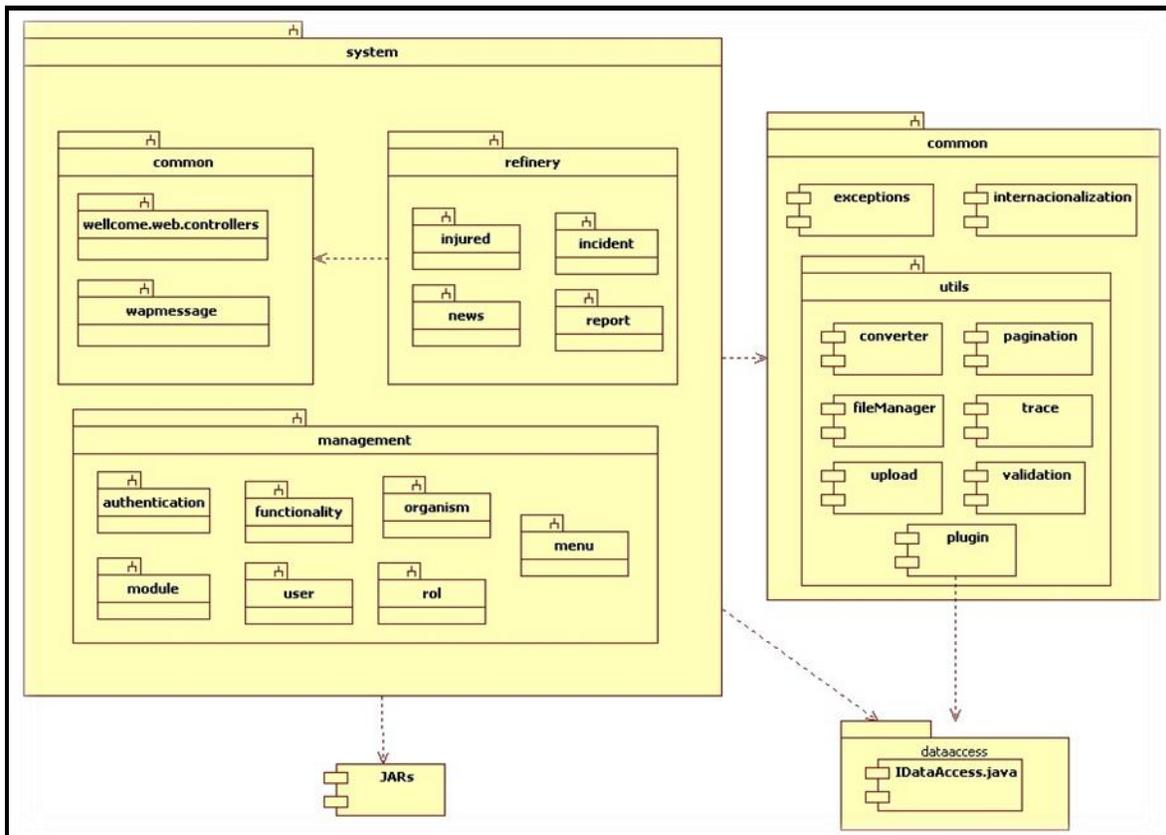


Figura 25: Diagrama de subsistemas de implementación

Descripción de los elementos:

system: contiene los artefactos del sistema.

common: contiene los artefactos relacionados con el envío de mensajes WAP Push.

refinery: contiene los artefactos relacionados con las funcionalidades del sistema.

management: contiene los artefactos relacionados con la administración del sistema.

common: contiene los artefactos generales para todo el sistema, como las validaciones, la internacionalización y paginación.

dataaccess: contiene como artefactos a la interfaz IDataAccess que permite el acceso a los datos independientemente de la vía de comunicación(JDBC, servicios Web o RMI).

JARs: contiene las librerías usadas, tanto las de Spring como las creadas durante el desarrollo del sistema.

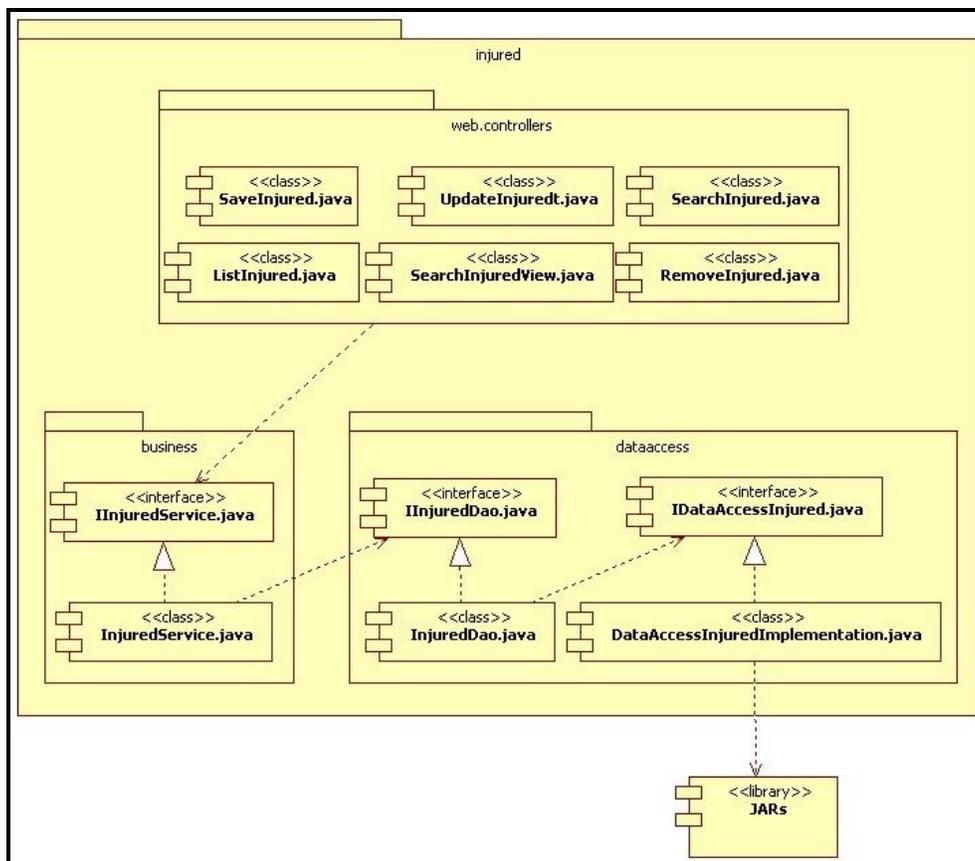


Figura 26: Diagrama de Componentes de Lesionado

Descripción de los elementos:

web.controllers: contiene las clases controladoras encargadas de atender las peticiones de los usuarios.

business: contiene las clases que permiten relacionar las clases controladoras con las clases de acceso a datos.

dataaccess: contiene las clases encargadas de acceder a los datos ya sea mediante JDBC,RMI o servicio Web.

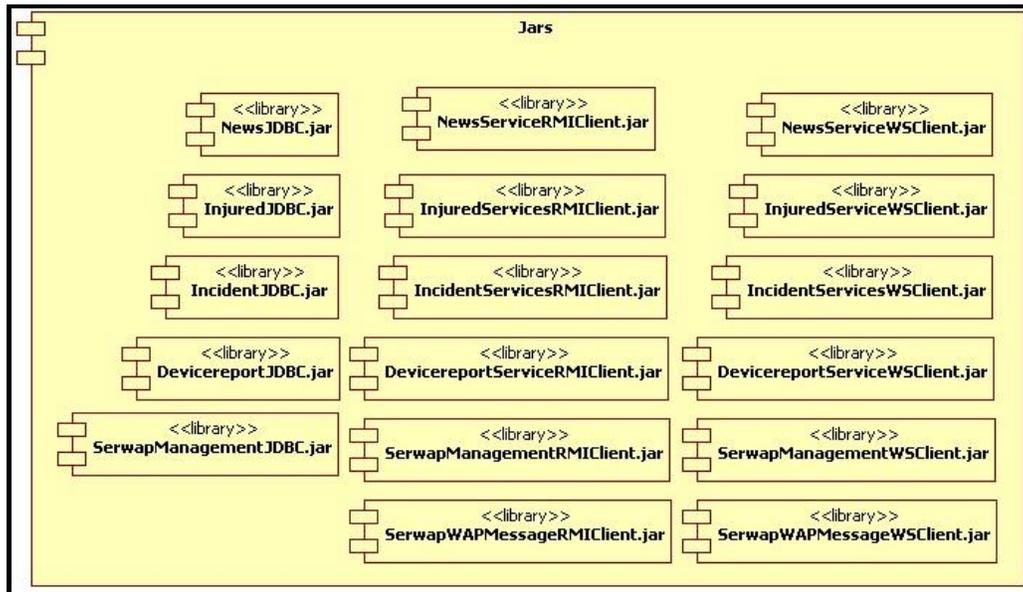


Figura 27: Componentes de las librerías del Sistema

Jars: contiene las APIs para acceder a los datos mediante servicios Web, RMI y JDBC.

4.3 Pruebas de software

Las pruebas de software son un elemento crítico para la validez de la calidad del software, representan una inspección final de las especificaciones del diseño y la implementación.

Una prueba no es más que una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones específicas, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. [22]

Las pruebas de software implican ejecutar una implementación del software con datos de prueba. Se examinan las salidas del software y su entorno operacional para comprobar que funcionan tal y como se requiere. Son una técnica dinámica de verificación y validación. [13]

Como objetivos fundamentales pretenden demostrar al desarrollador y al cliente que el software satisface sus requerimientos, y descubrir defectos en el software de comportamiento incorrecto, no deseable o que no cumple su especificación.

4.3.1 Modelo de prueba. Pruebas de Funcionalidad

Entre las pruebas existentes para comprobar el cumplimiento de los requerimientos, se encuentra la prueba por funcionalidad. Esta pretende evaluar el cumplimiento de los requisitos funcionales, incluye la navegación, entrada de los datos, procesamiento y la obtención de resultados. Se ejecuta en cada funcionalidad de caso de uso, corroborando la implementación adecuada de las reglas del negocio. Su uso permite conocer si mediante la introducción de datos válidos, se obtienen los resultados esperados, además de que sean exhibidos los mensajes apropiados de error y precaución cuando se usan datos inválidos. [22]

Descripción de los casos de prueba

La intensidad de los casos de prueba es probar el sistema de una forma detallada, incluyendo las entradas con las que se experimentarán, las condiciones bajo las cuales se realizan y los resultados esperados.

Para comprobar el funcionamiento de PetroWeb se precisaron un conjunto de casos de pruebas que agrupan todas las situaciones posibles. Las pruebas han sido realizadas sobre el propio sistema PetroWeb, luego de haber realizado un plan de pruebas, detallar los casos de pruebas y posteriormente especificar los resultados de dichas pruebas.

Plan de pruebas

1.1. <u>LISTAR DATOS DE LESIONADOS</u>
OBJETIVO
Verificar que se muestren los datos de los lesionados.
CONDICIONES PREVIAS

El usuario debe estar autenticado previamente.			
El usuario debe tener en el rol que presente la funcionalidad que le permite Listar Lesionados.			
PROCEDIMIENTO			
Acción	Resultado esperado	Gravedad	
1.	<p>Seleccionar la opción “Listar Lesionados”.</p>	<p>- Se muestran las características de todos los lesionados registrados:</p> <ul style="list-style-type: none"> • Fecha • Causa • Cantidad de Lesionados • Incidente • Descripción <p>- Se muestran los campos “Fecha”, “Causa” y el botón “Buscar”.</p>	crítico
2.	<p>Pulsar el botón “Buscar” sin introducir el criterio de búsqueda.</p>	<p>- Se muestra el mensaje: “Debe llenar al menos un campo para realizar la búsqueda.”.</p> <p>- Se muestra la misma pantalla.</p>	moderado
3.	<p>- Introducir en el campo “Fecha” caracteres extraños, por ejemplo: \$, %.</p> <p>- Pulsar el botón “Buscar”.</p>	<p>- Se muestra el mensaje: “Debe llenar los campos correctamente.”.</p> <p>- Muestra la misma pantalla.</p>	moderado

4.	<ul style="list-style-type: none"> - Introducir en el campo “Causa” caracteres extraños, por ejemplo: \$, %. - Pulsar el botón “Buscar”. 	<ul style="list-style-type: none"> - Se muestra el mensaje: “Debe llenar los campos correctamente.”. - Muestra la misma pantalla. 	moderado
5.	<ul style="list-style-type: none"> - Introducir un criterio de búsqueda (el filtrado puede ser por uno o varios parámetros) por el que no se encuentre ningún resultado. - Pulsar el botón “Buscar”. 	<ul style="list-style-type: none"> - Muestra el mensaje “No se encontraron resultados acorde con el criterio de búsqueda”. - Se muestran la misma pantalla. 	moderado
6.	<ul style="list-style-type: none"> - Introducir correctamente el criterio de búsqueda (el filtrado puede ser por uno o varios parámetros). - Pulsar el botón “Buscar”. 	<ul style="list-style-type: none"> - Se muestra los campos “Fecha”, “Causa” y el botón “Buscar” y además una Lista de los lesionados como resultado de la búsqueda con los siguientes datos: <ul style="list-style-type: none"> • Fecha • Causa • Cantidad de Lesionados • Incidente • Descripción - Se guarda la traza en la base de datos. 	crítico
7.	<ul style="list-style-type: none"> - Si se muestra más de diez resultados se muestran los vínculos para la navegación entre las páginas. - Pulsar el vínculo “Siguiete”. 	<p>Se muestra la siguiente página del resultado de la Lista de Incidentes.</p>	moderado

8.	Pulsar el vínculo “Anterior”.	Se muestra la página anterior a la actual.	moderado
9.	Pulsar cualquier número correspondiente a la página del resultado deseada.	- Se muestra la página correspondiente al número seleccionado. - Se muestra el número de la página activa en color negro fuerte.	moderado
10.	Seleccionar la opción “Inicio”.	Se redirecciona a la pantalla inicial donde se muestra la opción “Lesionado”.	crítico
Comentarios			
<ul style="list-style-type: none"> - Si ocurre algún error se registra en el fichero “petroWeb_ErrorWar.log” el error ocurrido. - En el fichero “petroWeb _InfoDebug.log” se registran los eventos ocurridos en este tipo de acción. - Cuando se guardan las trazas en la base de datos, se registran los siguientes datos: la acción, la fecha en que se produce, una breve descripción de la misma y el usuario que la realiza. 			

Tabla 3: Caso de prueba de Listar Lesionado

Errores

A continuación se muestran la cantidad de errores que han sido detectados durante la revisión de la documentación del sistema, separando cuáles de estos errores proceden, cuáles no proceden y los que ya fueron resueltos; todos fueron no significativas.

Sistema	Total de errores	# de errores que proceden	# de errores que no proceden	# de errores Resueltos
PetroWeb	6	5	1	5

Tabla 4: Resumen de no conformidades

4.3.2 Pruebas de Seguridad

El uso de esta prueba garantiza que los usuarios estén restringidos a funciones específicas o su acceso esté limitado únicamente a los datos que están autorizados a acceder, y que solo aquellos usuarios autorizados a acceder al sistema sean capaces de ejecutar las funciones del sistema. [22]

Para llevar a cabo dicha prueba se ha identificado a cada tipo de usuario y las funciones y datos a los que se debe autorizar. Se ha hecho uso de las pruebas encaminadas a cada tipo de usuario para verificar sus permisos.

Las pruebas arrojaron como resultado que para cada tipo de usuario conocido, las funciones, datos apropiados y todas las transacciones funcionan como se esperaba.

4.4 Conclusiones Parciales

En este capítulo se han expuesto los elementos que indican cómo ha sido implementado el sistema. Se realizaron los diagramas de componentes para modelar la vista estática del sistema, y el diagrama de despliegue para indicar la situación física de los componentes lógicos desarrollados. Mediante este capítulo se definió la organización del código, se implementaron todas las funcionalidades definidas para el sistema y se integraron los resultados en un sistema ejecutable. Con el objetivo de comprobar el correcto funcionamiento de los requerimientos del sistema se han realizado las pruebas que determinaron la validez del software.

CONCLUSIONES

El desarrollo del sistema PetroWeb permitió arribar a las siguientes conclusiones:

- ✓ Se realizó un estudio de los sistemas similares existentes a nivel internacional y sobre las herramientas y tecnologías, lo que permitió escoger las adecuadas para la solución.
- ✓ Se identificaron requerimientos y especificaron casos de uso, modelando la solución y describiendo las funcionalidades del sistema.
- ✓ Se elaboró el modelo de diseño y de las bases de datos, lo que permitió obtener una abstracción más cercana a la implementación del sistema.
- ✓ Se obtuvo una aplicación funcional que responde a los requisitos planteados, gestionar los incidentes, lesionados, noticias y listar reportes de equipamiento, dándole agilidad a la toma de decisiones; aumentando el control y el nivel de información de los agentes de campo.
- ✓ El sistema dispone del envío de mensajes WAP Push, a través del cual los trabajadores que se encuentren en distintas áreas podrán estar al tanto de los eventos, evitando así su traslado a zonas de potencial riesgo para sus vidas.
- ✓ Se realizaron pruebas a la aplicación verificando que los requisitos funcionales estaban libres de ambigüedades y eran correctos. Además se validó el cumplimiento de las especificaciones del diseño y la implementación.

RECOMENDACIONES

Se recomienda:

- ✓ Agregar un módulo al sistema para gestionar los datos de las tablas nomencladoras de la base de datos.
- ✓ Realizar un monitoreo del rendimiento del sistema para verificar el límite donde colapsa la solución propuesta.
- ✓ Aumentar la seguridad en el sistema mediante el uso de certificados digitales.

BIBLIOGRAFÍA

- [1] Motorola(2010). La solución TETRA de Motorola.
Disponible:<http://www.motorola.com/Business/XUEN/Product+Lines/Dimetra+TETRA/TETRA+Services>
- [2] W3C. (2010). Extensible Markup Language(XML). [Online]. Disponible: <http://www.w3.org/XML/>
- [3] W3C. (2010). Web Services Description Language (WSDL). [Online]. Disponible: <http://www.w3.org/TR/wsdl>
- [4] EcuRed. (2010). SOAP. [Online]. Disponible: [http://www.ecured.cu/index.php/Protocolo_simple_de_acceso_a_objetos_\(SOAP\)](http://www.ecured.cu/index.php/Protocolo_simple_de_acceso_a_objetos_(SOAP)).
- [5] Barriento. Módulo de acceso a datos con servicios Web. Habana, 2010 .
- [6] W3C. (2010). Guía Breve de Servicios Web(WS). [Online]. Disponible: <http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb>
- [7] C. Walls. Spring in Action (2da ed.). Greenwick: Manning Publications, 2001.
- [8] J.Martínez. SERWAP para redes TETRA. API para la codificación de mensajes WAP. Habana, 2010.
- [9] Empresa TELTRONIC. Una realidad en comunicaciones profesionales. España, 2008.
- [10] IEC(2010). Características del lenguaje Java. [Online]. Disponible: <http://www.iec.csic.es/criptonomicon/java/quesjava.html>
- [11] Spring. (2010). Spring Web Services. [Online]. Disponible: <http://static.springsource.org/spring-ws/sites/1.5/>
- [12] Dojo. (2010). Dojotoolkit. . [Online]. Disponible: <http://dojotoolkit.org/>
- [13] I. Somerville. Software Engineering. 2006.
- [14] R. Pressman. Ingeniería del Software: Un enfoque práctico. Madrid: McGraw Hill, 2001.
- [15] FDD. (2010) La Nueva Metodología. [Online]. Disponible: www.featuredrivendevelopment.com
- [16] L E. Coad. (1999). Java Modeling In Color With UML: Enterprise Components and Process. Prentice Hall.
- [17] EcuRed. (2010). Eclipse. [Online]. Disponible: http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado#Ventajas_en_la_utilizaci.C3.B3n_de_Eclipse

- [18] Tomcat. (2010). Elección del servidor de aplicaciones web. [Online]. Disponible: <http://tomcat.apache.org/>
- [19] Rubi. (2010). MySQL, la base de datos de gran uso. [Online]. Disponible: <http://mysql.ruby.com.ve/>
- [20] StarUML. (2010). The Open Source UML/MDA Platform. [Online]. Disponible: <http://staruml.sourceforge.net/en/>
- [21] F. Verdecia. Sistema de Gestión Policial. Especificación de la Arquitectura. Habana. 2009.
- [22] pruebasdesoftware. (2011). Gestión de Calidad y Pruebas de Software. [Online]. Disponible: <http://www.pruebasdesoftware.com/laspruebasdesoftware.htm>
- [23] DesarrolloWeb. (2010). Sistemas gestores de bases de datos.[Online]. Disponible: <http://www.desarrolloweb.com//articulos/sistemas-gestores-bases-datos.html>

GLOSARIO

API (*Application Programming Interface*): Es un conjunto de funciones que facilitan el intercambio de mensajes o datos entre dos aplicaciones.

C#: Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.

C++: Es un lenguaje de programación. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos.

CVS (*Concurrent Versioning System*): Es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros que forman un proyecto y permite que distintos desarrolladores colaboren.

FTP (*File Transfer Protocol* en español Protocolo de Transferencia de Archivos) es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

GUI (*Graphical User Interface*, en español Interfaz Gráfica de Usuario): Es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

HTML (*HyperText Markup Language*, en español Lenguaje de Marcado de Hipertexto): Es el lenguaje de marcado predominante para la elaboración de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTTP (*HyperText Transfer Protocol*, en español Protocolo de Transferencia de Hipertexto): Es el protocolo usado en cada transacción de la World Wide Web.

IDE (*Integrated Development Environment*, en español Entorno de Desarrollo Integrado): Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor

de código, un compilador, un depurador y un constructor de interfaz gráfica (*GUI*). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Inversión de Control (IoC) (en inglés *Inversion of Control*): Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (*procedure calls*) o funciones.

JDBC (*Java Database Connectivity*): Es un API para trabajar con bases de datos desde Java, independientemente de la base de datos a la que accedemos.

JEE (*Java Platform, Enterprise Edition o Java EE*): Es una plataforma de programación (parte de la Plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

Orientación a Aspectos (AOP): Derivado de Programación Orientada a Aspectos (POA o AOP), es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

PUSH: Describe un estilo de comunicaciones donde la petición de transacción se origina en el servidor.

RMI (*Java Remote Method Invocation*): Es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y provee de un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.

RUP (*Rational Unified Process*, en español Proceso Unificado de Rational): Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

SGML (*Standard Generalized Markup Language*, en español Lenguaje de Marcado Generalizado): Consiste en un sistema para la organización y etiquetado de documentos. El lenguaje SGML sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.

SOAP (*Simple Object Access Protocol*): Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

TCP/IP (*Transmission Control Protocol / Internet Protocol*, en español Protocolo de Control de Transmisión / Protocolo de Internet) es uno de los protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

TETRA (*Terrestrial Trunked Radio*): Es un estándar definido por el Instituto Europeo de Estándares de Telecomunicación. Este estándar define un sistema móvil digital de radio.

UML (*Unified Modeling Language*, en español Lenguaje Unificado de Modelado): Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

URI (*Uniform Resource Identifier* en español Identificador Uniforme de Recurso) es una cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema. Los URI pueden ser localizadores uniformes de recursos, Uniform Resource Name, o ambos.

WAP (*Wireless Application Protocol*): Es un estándar seguro que permite que los usuarios accedan a información de forma instantánea a través de dispositivos inalámbricos.

WSDL (*Web Services Description Language*): Es un formato XML que se utiliza para describir Servicios Web.

XML (*Extensible Markup Language*, en español Lenguaje de Marcas Extensible): Es un metalenguaje extensible de etiquetas. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML).

ANEXOS

Entrevista sobre las refinerías petroleras

Realizada a la Especialista en Refinación de Crudo: Ing. Patricia Toja.

Objetivos:

1. Cómo funciona una empresa petrolera.
2. Cuáles son los principales incidentes que ocurren dentro de una empresa petrolera.
- 3.Cuál es la actitud ante los eventos que ocurren dentro de una Empresa Petrolera.

Preguntas

Áreas de trabajo y trabajadores:

¿Cuáles son las áreas de trabajo (con el crudo) con las que cuenta una refinería?

¿Quiénes son los trabajadores que laboran en cada una de estas áreas?

¿Se comunican entre ellos en algún momento? ¿Cómo?

¿Cómo se mantienen los trabajadores informados de lo que ocurre dentro del área de trabajo?

¿Pueden los trabajadores presentar algún problema de salud? ¿Cuáles? ¿Cómo se solucionan estos problemas?

Sucesos negativos:

¿Cuáles son los problemas que pueden surgir durante una jornada laboral dentro de la refinería?

¿En qué áreas ocurren estos problemas?

¿Existe alguien encargado de informar los incidentes?

¿Existe algún responsable de solucionar los incidentes?

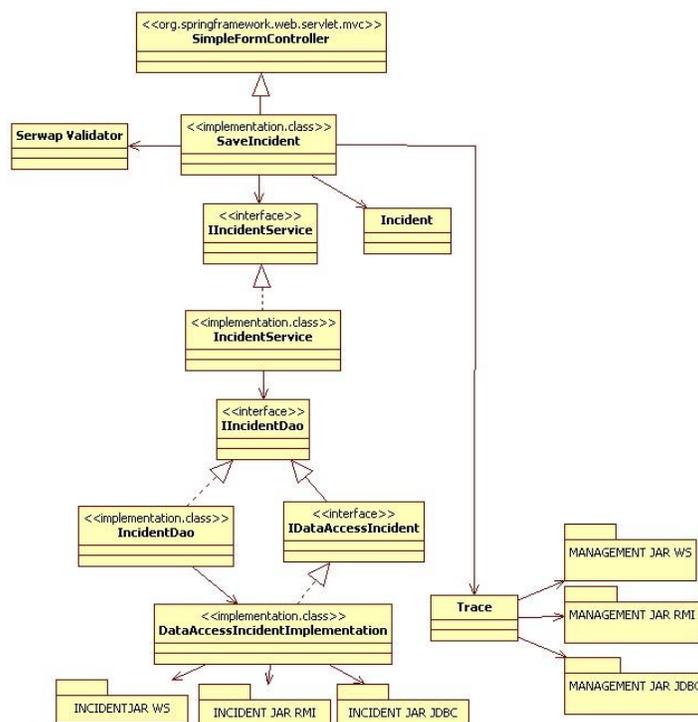
¿Si ocurre un incidente en un área específica, los trabajadores de otra área aledaña o que necesiten conocer de lo que ocurre, están actualizados? ¿Cómo?

Sobre la refinería:

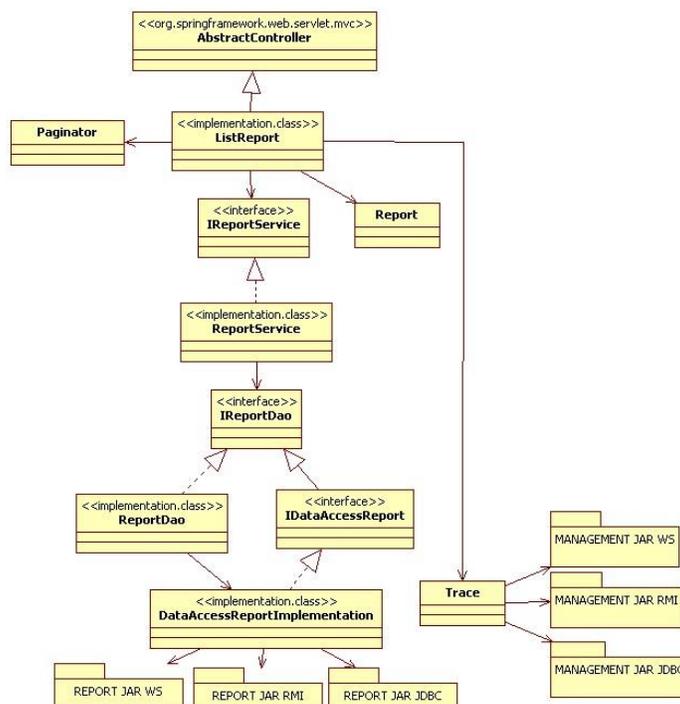
¿De qué manera se informan los trabajadores cuando es necesario que todos conozcan acerca de algún tema?

¿Saben los trabajadores en que estados se encuentran las áreas de trabajo?

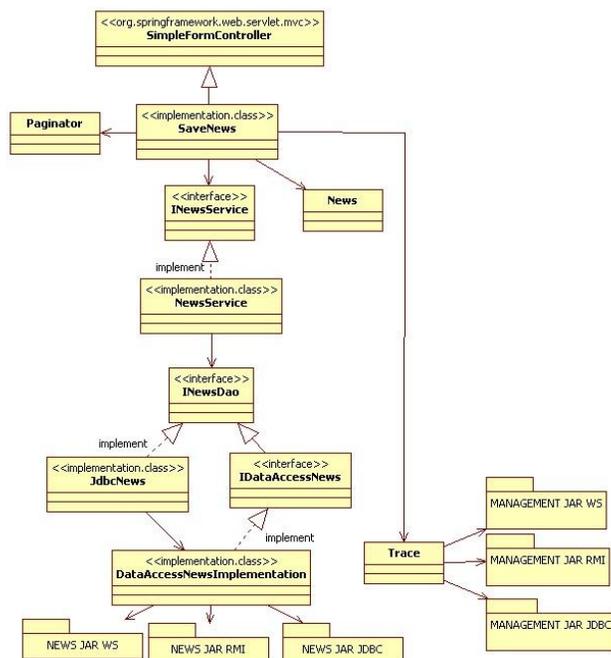
Anexo 1: Entrevista realizada.



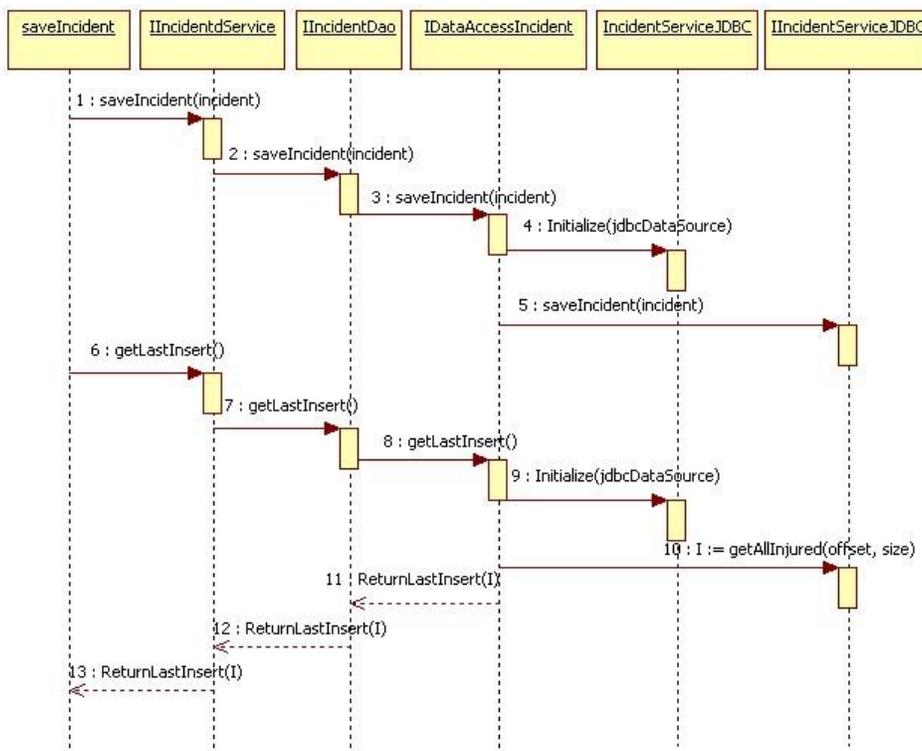
Anexo 2: Diagrama de clases del portal Web de la funcionalidad Save Incident



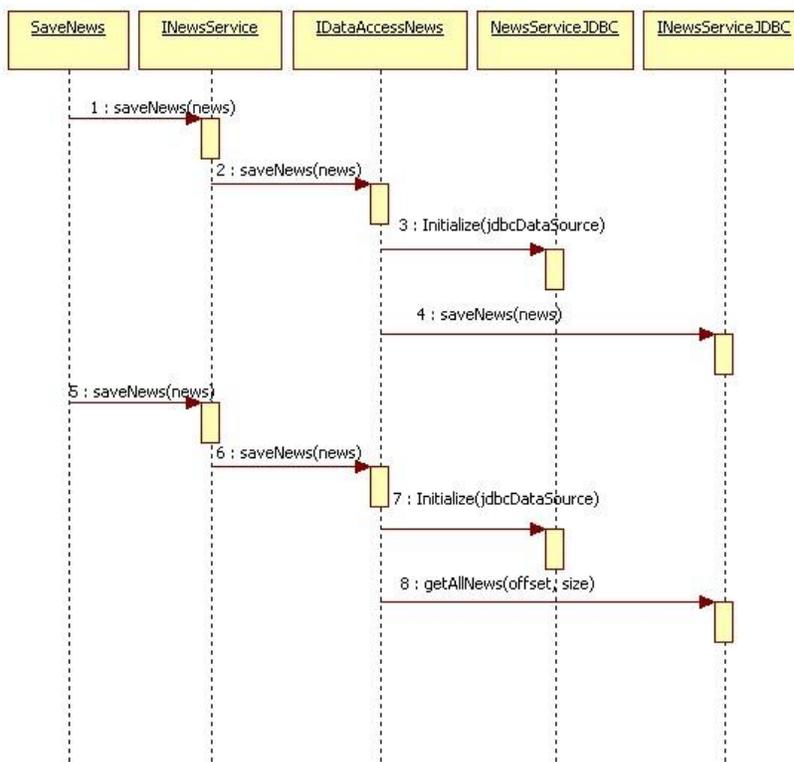
Anexo 3: Diagrama de clases del portal Web de la funcionalidad ListReport



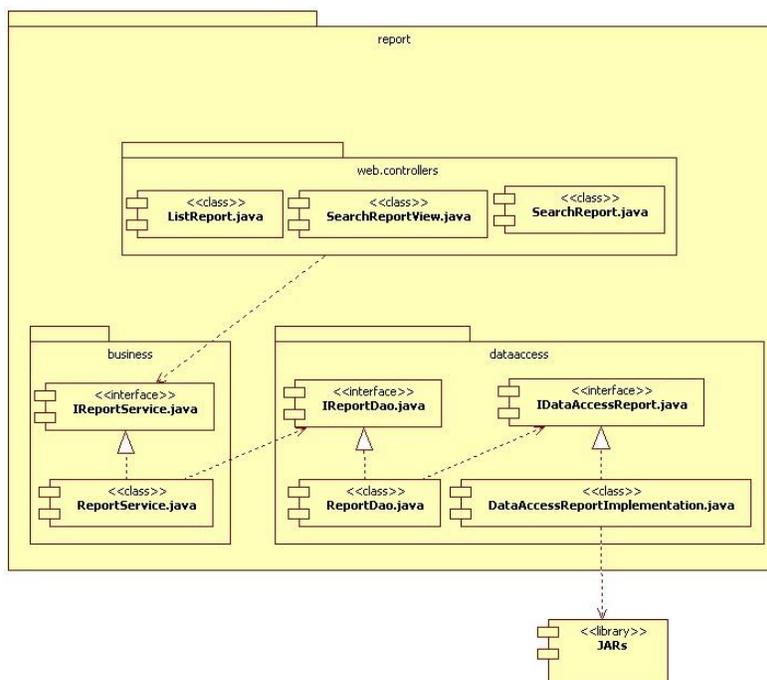
Anexo 4: Diagrama de clases del portal Web de la funcionalidad SaveNews



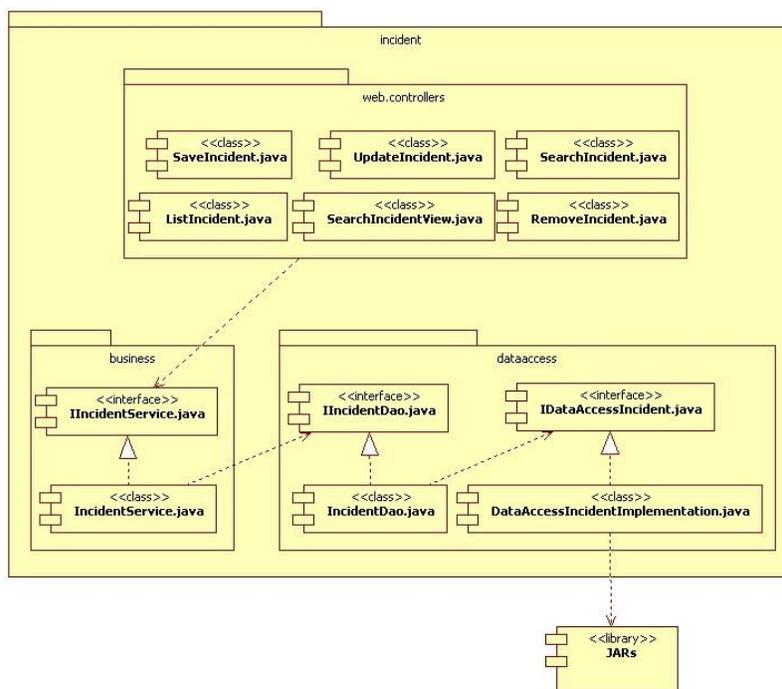
Anexo 5: Diagrama de secuencia del escenario Salvar Incidente, CU Gestionar Incidente



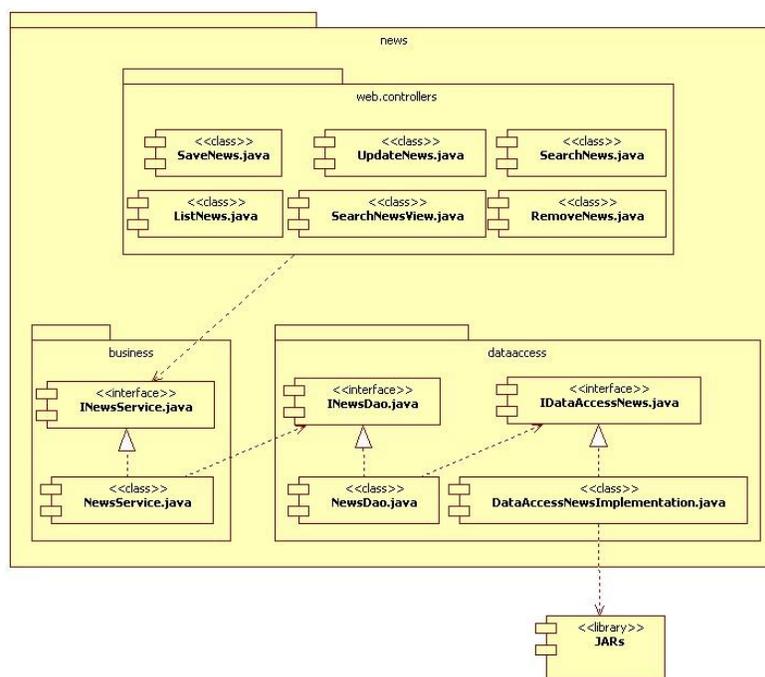
Anexo 6: Diagrama de secuencia del escenario Salvar Noticia, Cu Gestionar Noticia



Anexo 7: Diagrama de Componentes de Reporte de equipamiento



Anexo 8: Diagrama de Componentes de Incidente



Anexo 9: Diagrama de Componentes de Noticia