UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TEMA

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A."

APLICATIVO

"IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A. CON LOS FRAMEWORKS ZK Y HIBERNATE"

Autor: DAVID ALEJANDRO BOLAÑOS PUENTE

Director: ING. HÉCTOR RENÉ BROWN

Ibarra – Ecuador

2012

CERTIFICACIÓN

Certifico que la Tesis **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A."** con el aplicativo **"IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A. CON LOS FRAMEWORKS ZK Y HIBERNATE"** ha sido realizada en su totalidad por el señor: David Alejandro Bolaños Puente portador de la cédula de identidad número: 1003267786.

……

Ing. René Brown

Director de Tesis

CERTIFICACIÓN

Ibarra, 14 de mayo del 2012

Señores
UNIVERSIDAD TÉCNICA DEL NORTE
Presente

De mis consideraciones.-

Siendo auspiciantes del proyecto de tesis del Egresado David Alejandro Bolaños Puente con CI: 1003267786 quien desarrolló su trabajo con el tema "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A." con el aplicativo "IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A. CON LOS FRAMEWORKS ZK Y HIBERNATE", me es grato informar que se han superado con satisfacción las pruebas técnicas y la revisión de cumplimiento de los requerimientos funcionales, por lo que se recibe el proyecto como culminado y realizado por parte del egresado(a) David Alejandro Bolaños Puente. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra empresa.

El egresado David Alejandro Bolaños Puente puede hacer uso de este documento para los fines pertinentes en la Universidad Técnica del Norte.

Atentamente,

_____
Ing. René Brown
Director del Centro de Cómputo
EMPRESA ELECTRICA REGIONAL NORTE

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN A FAVOR DE LA UNIVERSIDAD TECNICA DEL NORTE

Yo, David Alejandro Bolaños Puente, con cedula de identidad Nro. 1003267786, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, articulo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A."** con el aplicativo **"IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A. CON LOS FRAMEWORKS ZK Y HIBERNATE",** que ha sido desarrollada para optar por el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes mencionada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte

……..

Firma

Nombre: David Alejandro Bolaños Puente

Cédula: 1003267786

Ibarra a los 14 días del mes de mayo del 2012

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente investigación:

DATOS DE CONTACTO	
CEDULA DE IDENTIDAD	1003267786
APELLIDOS Y NOMBRES	DAVID ALEJANDRO BOLAÑOS PUENTE
DIRECCIÓN	IBARRA CDLA. LA VICTORIA ROSA ANDRADE 646
EMAIL	davidbopu@yahoo.com
TELÉFONO FIJO	062951109
TELÉFONO MOVIL	093618695
DATOS DE LA OBRA	
TITULO	"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A. APLICATIVO: IMPLEMENTACIÓN DE UN SISTEMA WEB DE ATENCIÓN DE RECLAMOS PARA LA EMPRESA ELÉCTRICA REGIONAL NORTE S.A. CON LOS FRAMEWORKS ZK Y HIBERNATE"
AUTOR	DAVID ALEJANDRO BOLAÑOS PUENTE
FECHA	14 DE MAYO DEL 2012
PROGRAMA	PREGRADO
TITULO POR EL QUE	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. HECTOR RENE BROWN

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, David Alejandro Bolaños Puente, con cedula de identidad Nro. 1003267786, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 143.

……..
Nombre: David Alejandro Bolaños Puente
Cédula: 1003267786
Ibarra a los 14 días del mes de mayo del 2012

Dedicatoria

A mis padres, quienes me han apoyado incondicionalmente a lo largo de mi carrera para terminar esta nueva etapa de mi vida, espero que se sientan honrados de su hijo que con mucho sacrificio culmina este trabajo de grado.

Agradecimiento

A Dios, por ser el creador de todas las cosas y por brindarme la sabiduría necesaria para cumplir las metas que me he propuesto a lo largo de la vida.

A mis padres y hermanos, quienes siempre están pendientes de mi desenvolvimiento como persona y profesional.

A todos mis docentes, en especial al Ing. René Brown, por guiarme a lo largo de este proyecto.

Índice

Índice de Figuras

Índice de Tablas

Índice de Ejemplos de Código

RESUMEN

La investigación se encuentra focalizada en la regulación Nro. 012/08 que impuso el Consejo Nacional de Electricidad (CONELEC), a las empresas eléctricas del país, para que sistematicen lo procesos relacionados a la atención de reclamos, el principal objetivo de la mencionada normativa es crear un sistema web de atención de reclamos (SAR); es por ello que se implementó el SAR en toda el área de concesión de la Empresa Eléctrica Regional Norte (EMELNORTE).

El SAR, brinda las facilidades necesarias para realizar un correcto seguimiento a los reclamos registrados por los clientes de EMELNORTE, además está apoyado por una plataforma tecnológica de última generación, que permite el cumplimiento de la regulación impuesta por el CONELEC.

Para desarrollar el sistema se utilizó frameworks que soporten la madurez de la plataforma JEE, centrándose en el manejo de la persistencia se aprovechó de las bondades que ofrece Hibernate y para obtener una aplicación web basada en clientes ricos se utilizó el framework ZK.

Gracias a los frameworks antes mencionados se logró una aplicación altamente escalable con los estándares que ofrece una metodología de desarrollo como lo es Rational Unified Process (RUP).

SUMMARY

The investigation is focused on the regulation nº 012/08 imposed by the National Electricity Council (CONELEC), the electricity companies in the country, to systematize the processes related to health claims, the main purpose of that legislation is to create a web of care claims (SAR), which is why the SAR was implemented throughout the concession area of the Northern Regional Electricity Company (EMELNORTE).

The SAR provides the necessary facilities for proper monitoring of claims filed by customers of EMELNORTE also is supported by a platform-edge technology, which enables compliance with the regulation imposed by CONELEC.

To develop the system was used frameworks that support the maturity of the JEE platform, focusing on the management of persistent took advantage of the benefits offered by Hibernate and for a web application based on rich clients used the ZK framework.

Thanks to the aforementioned frameworks was achieved with a highly scalable application that provides a standard development methodology such as Rational Unified Process (RUP).

CAPÍTULO I INTRODUCCIÓN AL SISTEMA DE ATENCIÓN DE RECLAMOS

1.1 Antecedentes

El 25 de noviembre de 1975 se constituye la EMPRESA ELECTRICA REGIONAL NORTE S.A. EMELNORTE, como principal accionista el INECEL y las empresas eléctricas de Ibarra, Montufar y Tulcán, incluyendo todos sus activos y trabajadores, como gerente encargado se nombra al Dr. José Albuja Chávez.

Posteriormente fueron ingresando como accionistas todos los municipios y consejos provinciales del norte del país.

En 1973 y 1975 mediante decreto supremo se creó el Programa FERUM, con el objeto de electrificar las zonas rurales y urbano marginales del país con fondos del estado, el cual se mantiene hasta ahora y ha sido fundamental en el desarrollo y servicio de energía eléctrica para todas las poblaciones del país.

Con la creación de INECEL en 1.961, el sistema eléctrico ecuatoriano toma un giro protagónico en el desarrollo económico y social de la nación; se estructura el primer plan maestro de energía eléctrica, cuyo objetivo fundamental era: integrar, normalizar y masificar la cobertura de este servicio.

Durante los años setenta y parte de los ochenta, con el “boom” petrolero que vivió el país y el consecuente acceso al crédito internacional, se ejecutaron megaproyectos de equipamiento en las áreas de generación, transmisión y distribución.

Esta inversión a la postre, es la que mantiene actualmente con energía al país y ha permitido que el índice de población servida alcance aproximadamente un 80%.

La Empresa Regional Norte S.A "EMELNORTE S.A." está constituida por accionistas ecuatorianos. Los accionistas de conformidad con la Ley, son todos y cada uno de los organismos seccionales; esto es, los H. Consejos Provinciales y los I. Municipios de la zona de servicio. Además, la empresa tiene como accionistas particulares a ciudadanos Ibarreños. Su principal domicilio está en la ciudad de Ibarra y tiene establecidas sucursales, agencias y oficinas en todos los cantones de las provincias de Imbabura, Carchi y Norte de Pichincha.

EMELNORTE de acuerdo a los estatutos de 1.984, contó como accionistas al INECEL, Consejo Provincial del Carchi, Los Municipios de: Ibarra, Tulcán, Montufar, Espejo, Mira, Otavalo, Cotacachi, Antonio Ante, Pimampiro, Cayambe, Pedro Moncayo, Sucumbíos, Centro Agrícola de Imbabura, Cooperativa Unión de la Cruz, Sindicato de Choferes y personas particulares a los señores: Eduardo Almeida, Hernán Daza, Alberto Enríquez, Jorge Guerrón, Luis Iturralde, Renato Portilla, Hugo Jáuregui, Joaquín Sandoval, Elías Castelo y Amador Dávila.

La empresa Eléctrica Regional Norte, es una de las 19 empresas eléctricas nacionales, cuya misión fundamental consiste en la distribución y comercialización de energía eléctrica, en un mercado cautivo, conformado por consumidores industriales, comerciales y residenciales, asentados en las áreas urbanas y rurales de las provincias de Imbabura y Carchi, así como en los cantones de Cayambe y Pedro Moncayo de la provincia de Pichincha y en el cantón Sucumbíos de la provincia del mismo nombre.

En 1999 las acciones del ex INECEL, pasaron a nombre de la entidad estatal Fondo de Solidaridad, que era el accionista mayoritario de todas las empresas eléctricas del país, se incorporaron como accionistas a los Consejos Provinciales de Imbabura, Pichincha, Sucumbíos, Municipios de Bolívar, Urcuquí y otras personas particulares

Desde el año 2008 empieza la transformación del país, se inicia con la nueva Constitución de la República del Ecuador, aprobada mediante referéndum por el pueblo ecuatoriano y en vigencia desde el 20 de octubre del 2008.

El 16 de octubre del 2009 se aprueba la Ley Orgánica de Empresas Públicas, mediante la cual se dispone la transferencia de las acciones de todas las empresas públicas que eran del Fondo de Solidaridad al Ministerio de Electricidad y Energía Renovable. El 2 de diciembre del 2009 se formalizó el traspaso de la acciones por el Ab. Patricio Vintimilla Loor, Liquidador del Fondo de Solidaridad.

En el numeral 2.2.1.5 de la Ley de Empresas Públicas, determina que EMELNORTE S.A., hasta que se expida el nuevo marco jurídico del sector eléctrico, seguirá operando como compañía anónima regulada por la Ley de Compañías, exclusivamente para los asuntos de orden societario. Para los demás aspectos tales como el régimen tributario, fiscal, laboral, contractual, de control y de funcionamiento de las empresas se observarán las disposiciones contenidas en esta Ley.[1]

1.2 Misión y Visión de Emelnorte s.a.

1.2.1 Misión de Emelnorte s.a.

Generar, distribuir y comercializar energía eléctrica bajo estándares de calidad para satisfacer las necesidades de sus clientes, con servicios de excelencia, personal calificado y comprometido, contribuyendo al desarrollo del país.

[1] http://www.emelnorte.com/eern/index.php?option=com_content&view=article&id=50&Itemid=56

1.2.2 Visión de Emelnorte s.a.

EMELNORTE, será una empresa competitiva, técnica, moderna, modelo y referente del sector eléctrico; por la calidad de sus productos y servicios, gestión transparente y por su efectiva contribución al desarrollo del país.

1.3 Situación Tecnológica Actual de Emelnorte s.a.

En la parte tecnológica, la empresa EMELNORTE cuenta al momento con alrededor de 20 servidores instalados en su data center, una red WAN que comunica a todas sus agencias distribuidas en 5 provincias del norte del país y aproximadamente 35 sistemas informáticos que dan soporte a las actividades diarias de los usuarios internos.

Muchos de estos sistemas son aplicaciones legadas, mientras que los nuevos requerimientos de sistemas se están desarrollando sobre arquitecturas distintas (Java y Visual Basic). El mayor sistema con el que cuenta la empresa es el sistema comercial que está implementado sobre un esquema cliente/servidor con herramientas de la casa comercial Oracle.

1.4 Problema

Para mejorar la atención al cliente de todas las empresas eléctricas del país, el directorio del Consejo Nacional de Electricidad (CONELEC) expidió la regulación No. “CONELEC – 012/08”, en la cual en uno de sus numerales establece la elaboración de un Sistema de Atención de Reclamos (SAR) vía web.

Actualmente en EMELNORTE existe la carencia de un sistema web que recolecte los reclamos de los clientes de la empresa, además el manejo de reclamos que existe no cumple con los procesos establecidos en la regulación del CONELEC No. “CONELEC – 012/08”.

Por la inexistencia de un sistema de atención de reclamos, los responsables de la empresa realizan el registro de los reclamos manualmente, es decir llevan un libro diario de los reclamos presentados por los clientes de la empresa; es por eso que no se posee información actualizada y en línea de los reclamos registrados en las trece agencias que son parte del área de concesión de la empresa, de la misma manera el cliente no puede realizar un seguimiento eficiente y eficaz a su reclamo registrado.

1.5 Objetivos

1.5.1 Objetivo General

Diseñar e implementar un sistema web de atención de reclamos para la empresa Eléctrica Regional Norte S.A. EMELNORTE con los frameworks ZK y Hibernate.

1.5.2 Objetivos Específicos

- Implementar una herramienta web de calidad basándose en una correcta metodología de desarrollo de software.
- Permitir a los usuarios el acceso al software desde cualquier punto geográfico siempre y cuando se tenga acceso a internet, teniendo la información actualizada y en línea.
- Definir los diferentes roles de usuario que tendrá el Sistema de Atención de Reclamos (SAR).
- Capacitar a los usuarios en la manipulación constante del software para que este se familiarice con el funcionamiento del mismo.
- Desarrollar el software por lo menos en dos idiomas, que serán en español e inglés.
- Generar reportes de análisis de los reclamos registrados en el Sistema de Atención de Reclamos (SAR).
- Investigar el manejo de los framework's ZK e HIBERNATE.
- Integrar el mapeador objeto relacional HIBERNATE con el framework ZK.
- Manejar el framework ZK con la arquitectura Modelo Vista Controlador (MVC).

1.6 Justificación

La empresa eléctrica regional norte EMELNORTE tiene la necesidad de mejorar la atención de reclamos de sus clientes y dar cumplimiento a la regulación del CONELEC No. "CONELEC-012/08", además de sistematizar las técnicas empleadas para el registro de reclamos, analizando las pautas que establece dicha regulación.

De tal forma que se optimizara los procesos de registro de reclamos, se tendrá una base de datos actualizada con los distintos trámites y además este sistema servirá para enviar los distintos reportes solicitados por el CONELEC.

La utilización del framework ZK se aplicará en el "Front - End", debido a que se basa en la web y se integra perfectamente con la plataforma Java EE, este framework posee características basadas en clientes ricos y su implementación es bastante similar a una aplicación de escritorio.

Para interactuar con el “Back - End” se usará el mapeador objeto relacional Hibernate, debido a que implementa una persistencia de objetos en un soporte relacional, además satisface la necesidad de no tener separado la estructura relacional y la objetual.

La implementación del sistema se encuentra amparada por una metodología de desarrollo de software, en este caso se usará Rational Unified Process (RUP), ya que éste se caracteriza por ser iterativo e incremental, está centrado en la arquitectura y guiado por los casos de uso, indicando que entregables generar y como desarrollarlos; además de delegar trabajadores, es decir, los papeles que una persona puede desempeñar en el proceso de desarrollo.

1.7 Alcance

El Sistema de Atención de Reclamos (SAR), beneficiará a la Empresa Eléctrica Regional Norte y a sus clientes. El diseño del sistema se concentra en el análisis y seguimiento de cada reclamo registrado por los clientes de la empresa.

La óptima implementación del software permitirá la emisión de reportes para realizar una correcta gestión mejorando la atención al cliente apegándose a lo que establece la regulación del CONELEC No. **“CONELEC-012/08”**.

Las delimitaciones bajo las cuales estará el desarrollo del producto de software, son a través de módulos globales.

CAPÍTULO II ESTUDIO DE LAS HERRAMIENTAS PARA DESARROLLAR EL SOFTWARE

2.1 Modelo Vista Controlador (MVC)

2.1.1 Introducción a MVC

Los orígenes de MVC fueron descritos por primera vez en el año 1979 por Trygve Reenskaug quien trabajaba en los laboratorios de investigación de Xerox.[2]

Modelo-Vista-Controlador (MVC) es un patrón de diseño de software, el cual pretende lo siguiente:[3]

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.
- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

La estructura MVC es un paradigma utilizado en diversos desarrollos de software, a través de este patrón de diseño se logra una división de las diferentes partes que conforman una aplicación, siendo su principal razón de ser: mantenimiento del código fuente.[4]

2.1.2 Arquitectura MVC

MVC, se acopla a las arquitecturas de software de tres niveles, lo que pretende es de ejecutar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.[5]

Los elementos de la arquitectura MVC son los siguientes:

- **Modelo:** datos y reglas de negocio.
- **Vista:** muestra la información del modelo al usuario.
- **Controlador:** gestiona las entradas del usuario.

[2] http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
[3] http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o
[4] http://javaweb.osmosislatina.com/curso/mvc.htm
[5] http://www.proactiva-calidad.com/java/patrones/mvc.html

En la siguiente figura se ilustra de mejor manera los elementos que componen la arquitectura MVC.

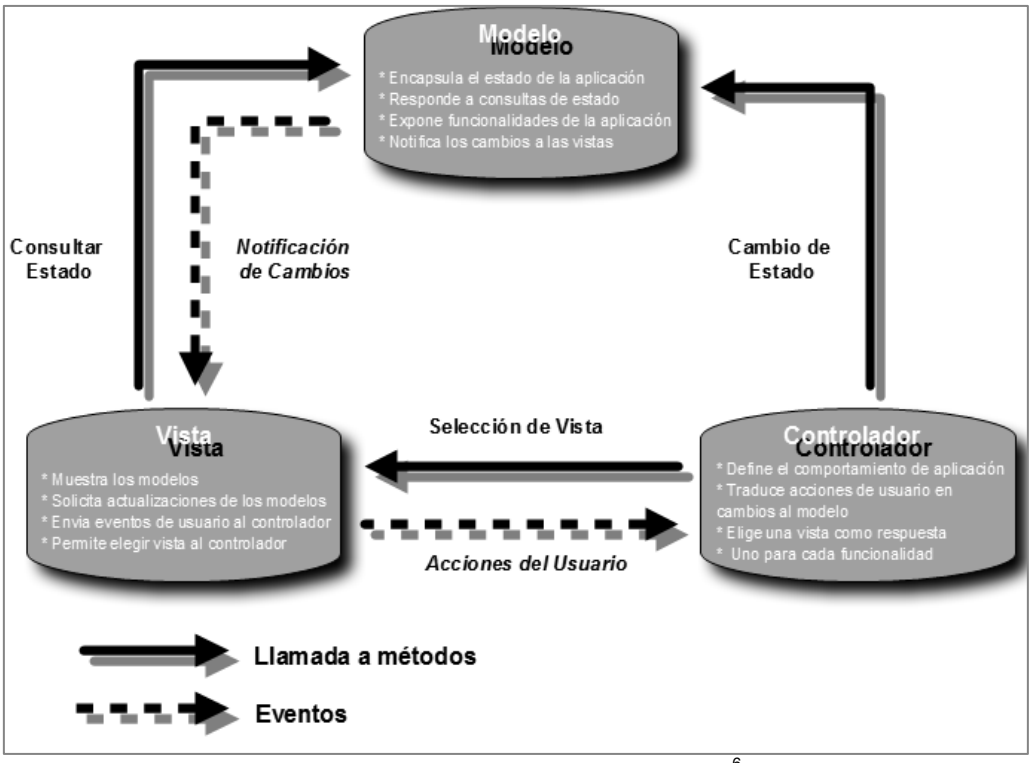

Figura 2 - 1 Modelo Vista Controlador [6]

El siguiente grafico ejemplifica el flujo que realiza MVC en la web.

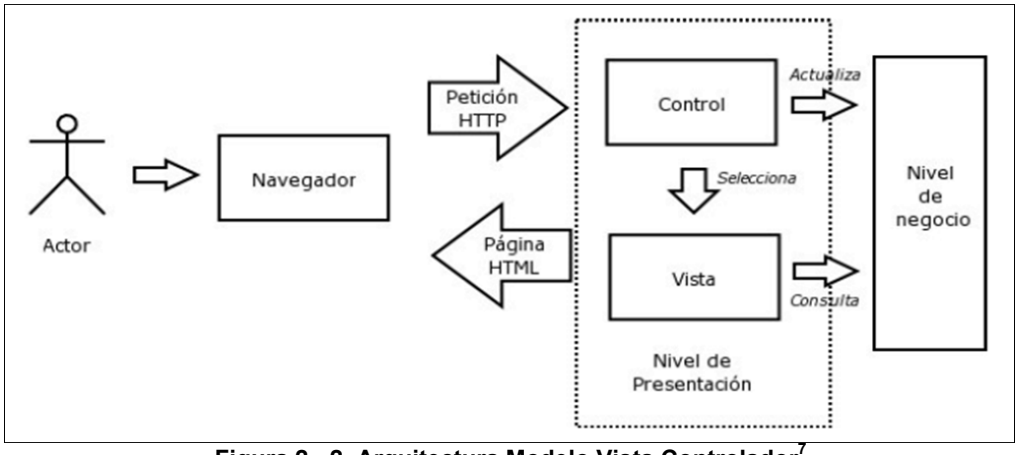

Figura 2 - 2 Arquitectura Modelo Vista Controlador[7]

[6] http://doviedo.rianetworks.net/PLANTILLA%20WEB/design/resumen_MVC.png
[7] http://www.cii-murcia.es/informas/ene05/articulos/Arquitectura_y_disenyo_de_sistemas_web_modernos.html

2.1.2.1 El Modelo

El Modelo comprende la implementación de la lógica de negocio y donde se debe soportar todos los requisitos funcionales del sistema sin mezclarlo con partes correspondientes al workflow que corresponden al Controlador.[8]

2.1.2.2 La Vista

Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

- Las vistas son las porciones de la aplicación MVC que presentan salida al usuario.
- La salida más común para aplicaciones web es el HTML. Podrían ser otras.
- Las vistas generalmente crean instancias de los modelos y utilizan métodos de esos modelos para conseguir los datos que presentan a los usuarios.

2.1.2.3 El Controlador

Dependiendo de la acción solicitada por el usuario, es el que pide al modelo la información necesaria e invoca a la plantilla (de la vista) que corresponda para que la información sea presentada.[9]

2.1.3 Frameworks MVC

En la actualidad, existe gran variedad de frameworks que trabajan bajo el patrón de diseño MVC, es por ello que el desarrollador debe plantearse las siguientes interrogantes antes de seleccionar su framework:

- ¿Mi aplicación es orientada a la web o no lo es?
- ¿Cuál es la curva de aprendizaje del framework?

Si el framework que se ha elegido satisface las incógnitas planteadas no se debe dejar pasar por alto que se acoplen al manejo de tecnologías estandarizadas.
A continuación se presenta diversas implementaciones para utilizar un "Framework MVC":

[8] http://es.scribd.com/doc/97147/introduccion-al-framework-struts
[9] http://www.jourmoly.com.ar/introduccion-a-mvc-con-php-primera-parte/

Lenguaje	Nombre
Ruby	Ruby on Rails
Ruby	Merb
Ruby	Ramaze
Java / J2ee	Grails
Java / J2ee	Interface Java Objects
Java / J2ee	ZK Framework
Java / J2ee	Struts
Java / J2ee	JavaServerFaces
JavaScript	ExtJS 4
Perl	CGI::Builder
PHP	ZanPHP
PHP	Tlalokes
PHP	SiaMVC
PHP	Agavi
PHP	Zend Framework
PHP	Kohana
PHP	PHP4ECore
PHP	PRADO
PHP	Zope3
PHP	InterJinn
PHP	Qcodo
PHP	rwfphp
PHP	Seagull
PHP	Sitellite
PHP	SolarPHP
PHP	sQeletor
PHP	Studs
PHP	struts4php
PHP	symfony
PHP	TaniPHP
Python	Zope3
Python	Turbogears

Tabla 2 - 1 Frameworks MVC[10]

[10] http://questchile.wordpress.com/2007/09/14/lista-de-framework-mvc-para-php/

2.2 Mapeador Objeto Relacional (Hibernate)

2.2.1 Introducción a Hibernate

Hibernate se inició en el año 2001 por Gavin King como una alternativa al uso de EJB2, su misión en ese entonces era simplemente ofrecer una mejor capacidad de persistencia que brinda EJB2 mediante la simplificación de la complejidad que este ofrecía.

A principios del 2003, el equipo de desarrollo de Hibernate comenzó Hibernate2 que ofreció muchas mejoras significativas a la primera versión y que catapultaron a Hibernate como el "de facto" para la persistencia en Java. [11]

Hibernate es un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos.[12]

Las características que Hibernate Ofrece son:

- **Modelo de programaciones naturales:** Hibernate permite desarrollar clases persistentes en lenguajes orientados a objetos que permiten: herencia, polimorfismo, la asociación, la composición y el marco de colecciones de Java.
- **Persistencia transparente:** Hibernate no requiere de interfaces o clases base para las clases persistentes y permite a toda la estructura de clases o los datos a ser persistentes. Además, Hibernate permite una rápida construcción de los procedimientos, ya que no presenta acumulación de tiempo de origen o de generación de código de bytes o la transformación.
- **Alto rendimiento:** Hibernate soporta la inicialización perezosa, muchas estrategias de ir a buscar, y el bloqueo optimista con versiones automáticas y selladas de tiempo. Hibernate no requiere de tablas o campos especiales en las bases de datos, además genera la mayor parte del SQL en el momento de inicialización del sistema en lugar de tiempo de ejecución. Hibernate siempre ofrece un rendimiento superior en rectas de codificación JDBC.
- **Fiabilidad y escalabilidad:** Hibernate es bien conocido por su excelente estabilidad y calidad, demostrada por la aceptación y uso por parte de decenas de miles de desarrolladores de Java. Hibernate fue diseñado para trabajar en un clúster de servidor de aplicaciones y proporcionar una arquitectura altamente escalable.

[11] http://www.hibernate.org/about/history
[12] http://www.hibernate.org/about/why-hibernate

Hibernate funciona bien en cualquier ambiente: Se utiliza para manejar dentro de una Intranet que sirve a cientos de usuarios o para aplicaciones de misión crítica que atienden a cientos de miles de personas.

- **:** Hibernate es altamente personalizable y extensible.

- **Instalaciones integrales de consulta:** Incluye soporte para Hibernate Query Language (HQL), Java Persistence Query Language (JPAQL), las consultas de los criterios, y consultas nativas de SQL, todos los cuales se puede desplazar y paginar para satisfacer sus necesidades de rendimiento exacto.

2.2.2 Configuración De Archivos Hibernate

Como hemos visto, Hibernate unifica el mundo objetual con el relacional. Ahora bien, para realizar dicha unión hay que presentar una serie de documentación que es la que posibilita la alianza.[13]

¿Qué ha de saber Hibernate para unificar los dos modelos?

1. **Quién está detrás del modelo relacional**:
   - Qué gestor de bases de datos está detrás.
   - A qué base de datos me conecto.
   - Cómo me conecto.
2. **Cómo se emparejan propiedades y campos de tablas**:
   - Clave primaria:
     - ¿Cuál es la propiedad que se corresponde con la clave primaria de la tabla correspondiente?
     - ¿Qué método deberé utilizar para generar un nuevo valor de la clave primaria?
     - Etc.
   - Otras propiedades:
     - Cómo empareja una propiedad con un campo de una tabla de la base de datos
       - ¿Cómo se llama la columna correspondiente?
       - ¿Qué tipo tiene?
       - ¿Es obligatoria?

[13] http://www.javahispano.org/contenidos/es/introduccrion_a_hibernate/

- Cómo gestiona las relaciones entre tablas
    - ¿Es una relación “uno-a-uno” ?
    - “uno-a-muchos”
    - “muchos-a-muchos”

Para responder a las dos grandes preguntas, se utilizan dos archivos distintos:

- El archivo de configuración de Hibernate (hibernate.cfg.xml) es el encargado de determinar los aspectos relacionados con el gestor de bases de datos y las conexiones con él.
- Los archivos que definen el emparejamiento (mapping) de propiedades con tablas y columnas (*.hbm.xml)

2.2.2.1 Archivo De Configuración De Hibernate (Hibernate.cfg.xml)

En la siguiente figura se detallan los componentes básicos que posee el archivo “hibernate.cfg.xml”:

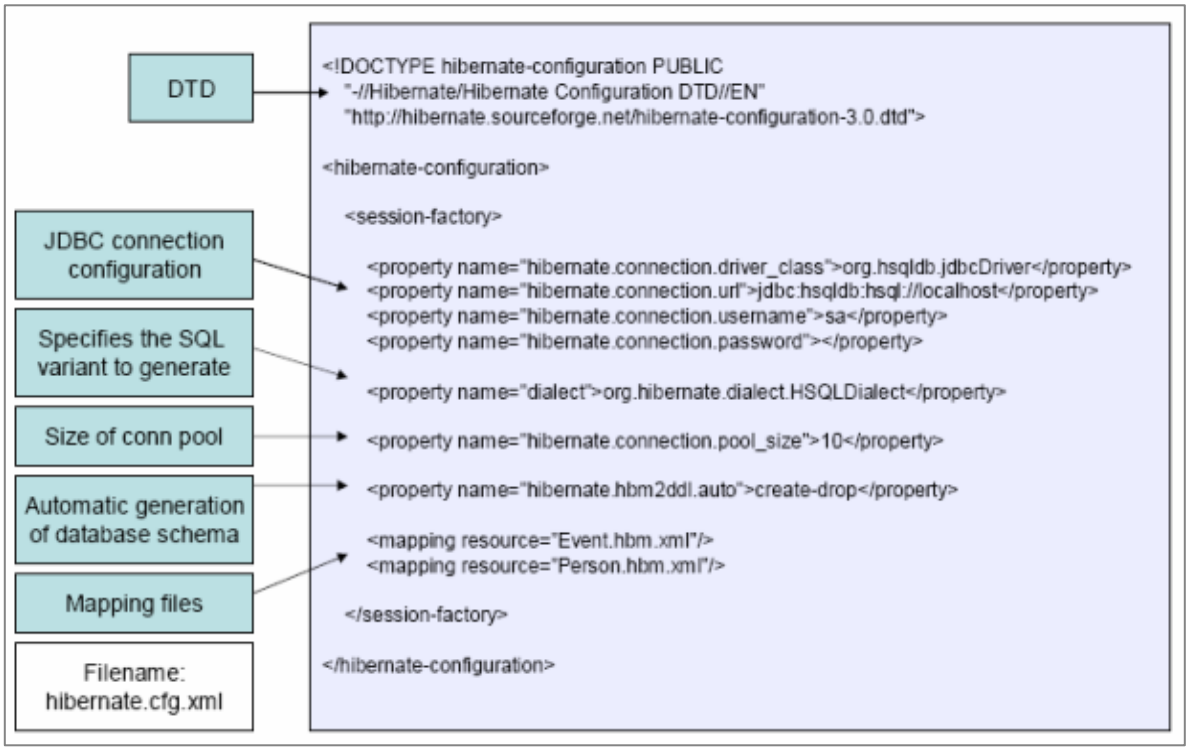

Figura 2 - 3 Archivo de configuración (hibernate.cfg.xml)[14]

[14] http://www.developersbook.com/hibernate/interview-questions/hibernate-interview-questions-faqs.php

A continuación, se especifican las propiedades más utilizadas a manejarse en el archivo hibernate.cfg.xml.

- hibernate.dialect

Permite a Hibernate generar SQL optimizado para una base de datos relacional en particular.

- hibernate.show_sql

Escribe todas las declaraciones SQL a la consola. Esta es una alternativa para establecer la categoría de registro org.hibernate.SQL a debug.

- hibernate.format_sql

Imprime el SQL en el registro y la consola.

- hibernate.default_schema

Califica los nombres de tabla sin calificar con el esquema/espacio de tabla dado en el SQL generado.

- hibernate.default_catalog

Califica los nombres de tabla sin calificar con el catálogo dado en el SQL generado.

- hibernate.session_factory_name

Automáticamente se vinculará el org.hibernate.SessionFactory a este nombre en JNDI después de que se ha creado.

- hibernate.cache.provider_class

El nombre de clase de un CacheProvider personalizado.

- hibernate.cache.use_minimal_puts

Optimiza la operación del caché de segundo nivel para minimizar escrituras, con el costo de lecturas más frecuentes. Esta configuración es más útil para cachés en clúster y en Hibernate3, está habilitado por defecto para implementaciones de caché en clúster.

- hibernate.cache.use_query_cache

Habilita el caché de consultas. Las consultas individuales todavía tienen que establecerse con cachés.

- hibernate.cache.use_second_level_cache

Se puede utilizar para deshabilitar por completo el caché de segundo nivel, que está habilitado por defecto para clases que especifican un mapeo <cache>.[15]

[15] http://docs.jboss.org/hibernate/core/3.6/reference/es-ES/html/tutorial.html

2.2.2.2 Archivo De Emparejamiento (Mapping)

Este es el archivo que une los mundos relacional y objetual. Para ejemplificar de mejor manera vamos a trabajar con una base de datos para una escuela la cual posee las siguientes tablas: Personas, Niveles, Clase, Categorias y Clase_Personas. El siguiente modelo relacional y modelo objetual, va usarse en los siguientes ejemplos, que serán tratados en este capítulo sobre Hibernate:

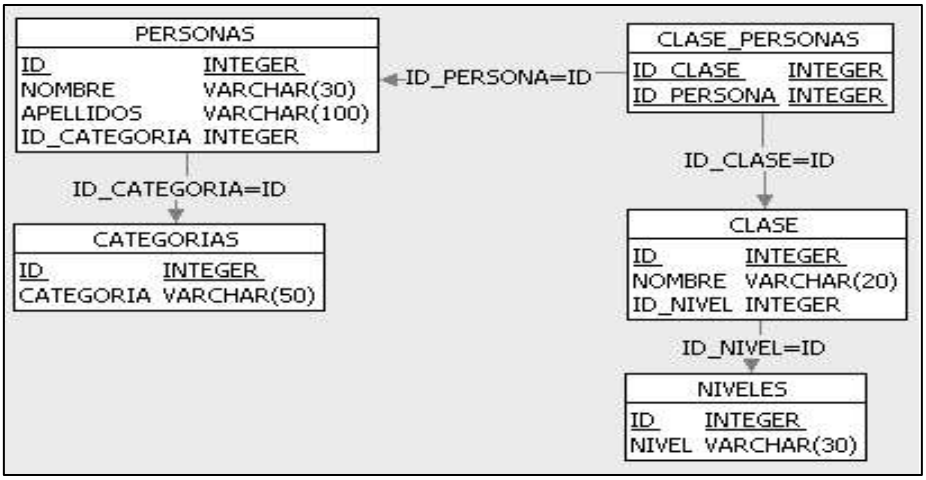

Figura 2 - 4 Modelo relacional para usar en el archivo de emparejamiento

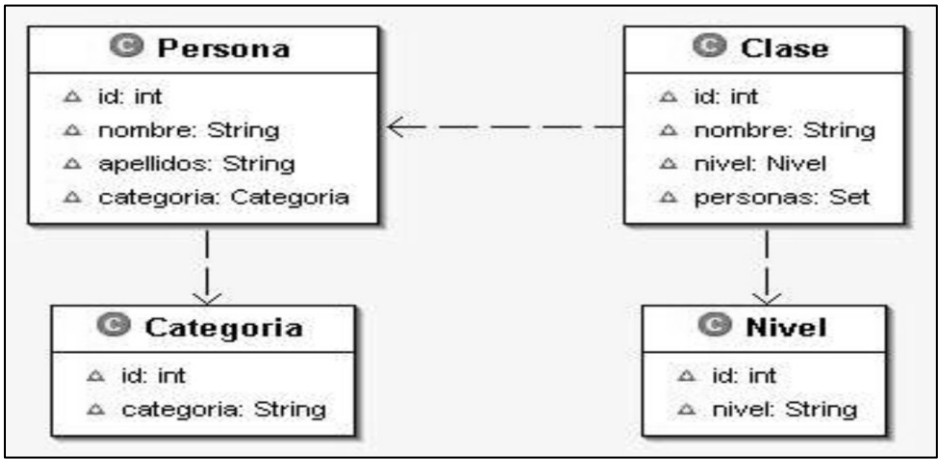

Figura 2 - 5 Modelo objetual para usar en el archivo de emparejamiento

El ejemplo, sencillísimo, que presentamos a continuación es el que se corresponde con la clase “Categoria”.

```
<?xml version="1.0"?>
```

```
<! DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>
<class name="dto.Categoria" table="categorias">
<id name="id" type="integer" column="ID" unsaved-value="-1">
<generator class="identity"/>
</id>

<property name="categoria" column="CATEGORIA"
type="string"
unique=”true”
not-null="true"/>
</class>
</hibernate-mapping>
```

Ejemplo de código 2 - 1 Archivo de emparejamiento (Categoria.hbm.xml)

Como vemos, para definir los emparejamientos se usa un documento XML.

- Cabecera XML

Éste tiene una cabecera fija, típica de todos los documentos XML, que no describiremos.

- El elemento <hibernate-mapping>

Después de la cabecera, entramos en la parte de descripción de los emparejamientos, delimitada por el elemento <hibernate-mapping>. Este elemento permite especificar, mediante atributos, diversas características como el esquema de la base de datos.

- El elemento <class>

Como estamos emparejando una clase con una tabla, hemos de especificar cuáles son. Lo hacemos usando el elemento <class>:

```
<class name="dto.Categoria" table="categorias">
```

Ejemplo de código 2 - 2 Elemento “<class>” del archivo de mapeo de Hibernate

Indicamos el nombre de la clase (name=dto.Categoria) y el nombre de la tabla con la que se empareja (table=”categorias”).

- El elemento <id>

Una vez emparejados objeto y tabla, hay que emparejar las propiedades del objeto con las columnas de la tabla. En nuestro ejemplo, observamos que para describir este emparejamiento se utilizan dos elementos XML distintos: **<**id**>** y **<**property**>.**
El primero, <id>, empareja una de las propiedades con la clave primaria de la tabla:

```
<id name="id" type="integer" column="ID" unsaved-value="-1">
```

Ejemplo de código 2 - 3 Elemento "<id>" del archivo de mapeo de Hibernate

Identificamos la propiedad que se empareja con la clave primaria mediante el atributo XML name. En nuestro caso, la propiedad a relacionar con la clave primaria es id.
Esta propiedad se empareja con la columna ID de la tabla: column="ID". Como el nombre de la propiedad y el de la columna coinciden, en este caso, nos podríamos ahorrar el atributo column.
También, y opcionalmente, especificamos el tipo de datos con el que estamos trabajando. En este caso, con enteros. Como es un tipo básico, no necesitaríamos usar el atributo type. Hibernate sabe deducir el tipo de datos que estamos utilizando a partir de la introspección. Hibernate analiza en tiempo de ejecución las clases Java y obtiene la información que necesita. Aquí, a guisa de ejemplo, especificamos el tipo de datos Hibernate: type = "integer".

En nuestro ejemplo, hemos vuelto a utilizar un atributo opcional: unsaved**-**value. Este atributo permite especificar qué valor tiene el campo clave cuando aún no ha sido guardado en la base de datos. En nuestro caso, -1. Así, de entrada, parece un atributo poco útil, pero no es así. La clase Session de Hibernate dispone de un método muy práctico: saveOrUpdate(Object). Este método distingue si un objeto debe ser guardado por primera vez en la base de datos o bien modificado porque ya está en ella, en función de lo que especifiquemos en el atributo unsaved-value.
En nuestro caso, la propiedad id de una Categoria que aún no haya sido guardada en la base de datos tendrá un valor -1. Esto indicará a Hibernate que hay que guardarla; y al hacerlo, la propiedad id de esta Categoria pasará a tener un valor diferente de -1, por ejemplo, 25.Si recuperamos de la base de datos esta categoría, tendremos un objeto Categoria con un id igual a 25. Si modificamos la propiedad categoria y después utilizamos el método saveOrUpdate(), Hibernate comprobará que la propiedad id no vale -1 y deducirá que en vez de guardar, ha de modificar.

El atributo unsaved-value puede tener los siguientes valores predefinidos: any, none y null. Por defecto, null.

- El subelemento <generator>

Este elemento nos permite definir cómo se generan los valores de las claves primarias. Hibernate nos ofrece diversos métodos de generación de valores para las claves primarias.

```
<generator class="identity"/>
```

Ejemplo de código 2 - 4 Elemento "<generator>" del archivo de mapeo de Hibernate

El método se especifica mediante el atributo class. En nuestro ejemplo, usamos el método identity.

Hay más muchos métodos predefinidos y, si fuera necesario, los podríamos definir nosotros. Algunos, como identity, generan claves directamente y otros necesitan la ayuda de tablas auxiliares para generarlas. Sería demasiado largo comentarlos todos, así que nos centraremos en un tres de ellos.

El método native es bastante interesante. Deja que Hibernate escoja entre los métodos identity, sequence o hilo, en función de las características del gestor de bases de datos con el que trabajemos.

El método assigned deja que sea nuestra aplicación la que asigne un identificador al objeto antes de guardarlo en la base de datos.

El método increment genera identificadores de tipo long, short o int que son únicos sólo cuando no hay otro proceso que esté insertando datos en la misma tabla. Por lo tanto, no es recomendable usarlo, por ejemplo, en un *cluster*.

- El elemento <property>

Sirve para emparejar aquellas propiedades que no forman parte de la clave primaria con las correspondientes columnas de una (o más) tablas.

Aquí tenemos el emparejamiento propuesto para la propiedad categoria de nuestro ejemplo.

```
<property name="categoria" column="CATEGORIA" type="string"  unique="true" not-null="true" />
```

Ejemplo de código 2 - 5 Elemento "<property>" del archivo de mapeo de Hibernate

Como en el caso del elemento id, el atributo name indica el nombre de la propiedad, el atributo column, el nombre de la columna de la tabla “categorias” y el atributo type nos indica el tipo de dato.
El atributo unique, opcional, nos permite especificar si se admiten valores duplicados o no. En nuestro caso, no.
El atributo opcional not-null, nos permite especificar la obligación de que la propiedad tenga un valor no nulo.

2.2.3 Manejo De Sesiones

Para comunicarnos con Hibernate desde nuestra aplicación java, necesitamos un objeto de la interface **“**Session**”**, y para crearla, necesitamos obtener el objeto de la interface **“**SessionFactory**”**:

```
sessionFactory = getConfiguration().buildSessionFactory();
```

Ejemplo de código 2 - 6 Obtención del SessionFactory en Hibernate

Una vez que tenemos el “SessionFactory”, ya podemos crear el objeto Session:

```
session = getSessionFactory().openSession();
```

Ejemplo de código 2 - 7 Obtención de “Session” en Hibernate

- Insertar datos en nuestra base de datos relacional

Añadir una nueva Categoria a la base de datos es sencillo:

```
public void anadir(String categ) {
// Construimos una nueva Categoria
Categoria categoria = new Categoria();
categoria.setCategoria(categ);

// Creamos una Session
Session session = getSession();
if (session == null) {
System.out.println(“Error abriendo sesión.”);
return;
}
```

```
Transaction tx = null;
try {
//  Iniciamos una unidad de trabajo
tx = session.beginTransaction();
//  Guardamos la nueva Categoria en la base de datos
session.save(categoria);
// Confirmamos la transacción
tx.commit();
} catch (HibernateException e) {
if (tx != null) {
try {
tx.rollback();
} catch (HibernateException e) {
}
}
}
finally {
try {
 Cerramos la sesión
session.close();
} catch (HibernateException e) {
}
}
} // anadir()
```

Ejemplo de código 2 - 8 Proceso de insertar datos en el lenguaje Java mediante Hibernate

[1] Para añadir una nueva Categoria, primero hay que crearla. En nuestro ejemplo, la creamos a partir del parámetro “categ” del método.

[2] Hemos visto que la Session es el puente entre nuestra aplicación e Hibernate. Usamos el método getSession(), descrito más arriba, para obtener una Session.

[3] Iniciamos una unidad de trabajo antes de añadir la Categoria:

```
Transaction tx = getSession().beginTransaction();
```

[4] Guardamos categoria en la base de datos.

```
session.save(categoria);
```

Observamos que no hay ninguna instrucción SQL. Simplemente, estamos persistiendo un objeto.

[5] Si no ha habido ningún problema, confirma la transacción a la base de datos, haciendo un commit() y cierra la unidad de trabajo. En definitiva, hace un SQL COMMIT pero además, sincroniza la sesión con la base de datos ejecutando implícitamente el método flush() de la clase Session.

[6] Si ha habido algún problema, tira para atrás la transacción y cierra la unidad de trabajo. En definitiva, hace un ROLLBACK.

[7] Para acabar, pase lo que pase, cerramos la sesión.

- Eliminar datos en nuestra base de datos relacional

El esquema general para dar de baja una categoría es básicamente el mismo que el de darla de alta. Nos centraremos, pues, en la parte específica de darla de baja:

```
public void borrar(int idCategoria) {
[...]
Categoria unaCategoria = new Categoria();
unaCategoria.setId(idCategoria);
Transaction tx = session.beginTransaction();
session.delete(unaCategoria);
tx.commit();
[...]
}
```

Ejemplo de código 2 - 9 Proceso de borrar datos en el lenguaje Java

En el ejemplo anterior, creamos una Categoria y le asignamos el identificador que se nos pasa como parámetro, Este identificador es el que usará Hibernate para eliminar la Categoria de la base de datos.

Hay tres estrategias para borrar registros. Por ejemplo, podemos usar una consulta basándonos en el identificador de clave primaria:

```
public void borrar(int idCategoria) {
[...]
```

```
session.beginTransaction();
Categoria unaCategoria = (Categoria)session.load(Categoria.class,
new Integer(id));
session.delete(unaCategoria);
tx.commit();
[...]
}
```

Ejemplo de código 2 - 10 Proceso de borrar datos en el lenguaje Java método load()

Fijémonos en el método load() de Session. Este método nos devuelve un objeto a partir de un identificador de clave primaria, Como es lógico, le hemos de pasar la clase (Categoria.class) para que sepa en qué tabla ha de buscar. Recordemos que en el archivo de emparejamiento (Categoria.hbm.xml) hemos emparejado la clase Categoria con la tabla CATEGORIAS. Con esta información, Hibernate genera una instrucción SQL similar a esta:

```
SELECT * FROM CATEGORIAS WHERE ID = miID
```

Ejemplo de código 2 - 11 Sentencia sencilla SQL mediante un criterio de búsqueda

Hibernate nos proporciona un lenguaje de consulta orientado a objetos bastante avanzado. Se llama HQL y hablaremos de él con más detalle, más adelante.
El método delete() admite una consulta en HQL como parámetro. En este caso, eliminaría todos los objetos devueltos por la consulta:

```
[...]
String sel = "FROM Categoria AS C WHERE C.id =" + idCategoria;
session.delete(sel);
[...]
```

Ejemplo de código 2 - 12 Proceso de borrar datos en el lenguaje Java mediante HQL caso 1

En este caso, sólo borraría un registro. Si, por ejemplo, nos interesase borrar todos los registros que tuvieran "r" en el campo CATEGORIA, sólo tendríamos que cambiar la consulta:

```
[...]
String sel = "FROM Categoria AS C WHERE C.categoria LIKE '%r%'";
session.delete(sel);
```

```
[...]
```

Ejemplo de código 2 - 13 Proceso de borrar datos en el lenguaje Java mediante HQL caso 2

Fijémonos un poco en la sintaxis de la consulta. Se parece bastante a la sintaxis SQL habitual, pero está adaptada al mundo de los objetos:

- No hay SELECT ya que no devuelve valores de columnas, sino objetos.
- No especificamos ninguna tabla en el FROM. Especificamos una clase. Es a partir de esta clase y de las definiciones que hemos hecho en el archivo de emparejamientos que Hibernate deducirá qué tablas están implicadas.
- No usamos nombres de columnas en el WHERE. Usamos propiedades de objetos.

- Modificar datos en nuestra base de datos relacional

La estrategia va a ser la misma que para las bajas:

```
public void cambiar(int idCategoria) {
[...]
Transaction tx = session.beginTransaction();
Categoria unaCategoria = (Categoria)session.load(Categoria.class,
new Integer(id));
// Cambiamos una propiedad del objeto
unaCategoria.setCategoria("Una nueva categoría");

// Lo modificamos en la base de datos
session.update(unaCategoria);
tx.commit();
[...]
}
```

Ejemplo de código 2 - 14 Proceso de actualizar datos en el lenguaje Java mediante Hibernate

2.2.4 HQL

El HQL (Hibernate Query Language) es un lenguaje de interrogación. En el mundo relacional disponemos del SQL (Structured Query Language) que nos permite obtener información haciendo preguntas basadas en las tablas y sus columnas. El equivalente en el mundo

objetual es el HQL, que nos permite hacer preguntas basadas en los objetos y sus propiedades.

Una vez más, Hibernate se encarga de casar los dos mundos. Traduce las consultas que hacemos desde el mundo objetual en HQL al lenguaje de interrogación del mundo relacional, el SQL, y transforma los resultados obtenidos en el mundo relacional (filas y columnas) en aquello que tiene sentido en el mundo objetual: objetos.

El concepto de "traducción" es importante para entender qué hace Hibernate y uno de los sentidos de HQL. Un ejemplo sencillo de una consulta:

```
FROM Persona P WHERE P.categoria.id = 3
```

Ejemplo de código 2 - 15 Ejemplo básico de la sentencia "select" de HQL

Se podría "traducir" a:

```
SELECT ID, NOMBRE, APELLIDOS FROM PERSONAS WHERE CATEGORIA_ID = 3
```

Ejemplo de código 2 - 16 Ejemplo básico de la sentencia "select" de SQL

El ejemplo que acabamos de ver es sumamente sencillo y la traducción también lo es. Fijémonos que se usa una sintaxis SQL estándar. Pero si hemos tenido que trabajar con diversos gestores de bases de datos, habremos observado que hay tantas sintaxis SQL como gestores. Es decir, cada gestor tiene su "dialecto SQL" propio.

En una aplicación JDBC tradicional, si cambiamos el gestor de bases de datos y no hemos ido con cuidado con las instrucciones SQL, nos podemos ver obligados a tener que adaptar las instrucciones SQL a las peculiaridades del dialecto del nuevo gestor.

En una aplicación con Hibernate, el problema desaparece. Sólo hay que cambiar el "dialecto" en el archivo hibernate.cfg.xml e Hibernate se encarga de traducir el HQL al dialecto SQL que toque. Tenemos, pues, una relación "uno a muchos": lo escribimos una vez en HQL y lo podemos aplicar a muchos gestores/dialectos distintos.

Intentemos describir los pasos que hace Hibernate para resolver una consulta sencilla como se cita en el ejemplo de código 2-15.

- Determina el dialecto SQL que estamos utilizando (en nuestro caso, Oracle) a partir de la información contenida en el archivo hibernate.cfg.xml
- Traduce nuestra consulta al dialecto SQL que toque:
- Ejecuta la consulta vía JDBC y obtiene un ResultSet:
- Construye un objeto de tipo Persona a partir de los elementos del ResultSet:

- Devuelve el objeto Persona creado a partir del ResultSet.

Si disponemos de un potente lenguaje de interrogación es para poder hacer preguntas complejas. El lenguaje SQL nos permite hacer preguntas muy, muy complejas, y su equivalente en el mundo objetual, el HQL, también.

Es necesario separar desde un principio lo que es el lenguaje de interrogación (HQL) de los métodos que Hibernate nos proporciona para ejecutar la pregunta expresada en este lenguaje.

Así, la consulta:

```
String consulta = “FROM Categorias C WHERE C.categoria LIKE 'A%';
```

Ejemplo de código 2 - 17 Ejemplo básico de la sentencia “select” y “like” en HQL

Se puede ejecutar de diversas maneras:

```
List categorias = sesion.find(consulta);
Iterator iterador = sesion.iterate(consulta);
Query q = sesion.createQuery(consulta); List categs = q.list();
```

Ejemplo de código 2 - 18 Ejemplos para obtener lista de colecciones en Hibernate

Fijémonos que en los tres casos estamos ejecutando la misma consulta.

En primer lugar, nos centraremos en el lenguaje de interrogación HQL y después hablaremos de los diversos métodos que Hibernate nos ofrece para ejecutar las consultas.

Se han citado visto diversos ejemplos de HQL. Normalmente, han sido consultas simples que no ilustran todas las posibilidades del lenguaje de interrogación. Vamos a plantear algunas que muestren algunas posibilidades más del HQL.

- Funciones escalares

HQL nos permite usar funciones escalares en nuestras consultas. Así, por ejemplo,

```
COUNT(*)
AVG (id)
SUM (id)
MAX(id), MIN(id)
```

Ejemplo de código 2 - 19 Ejemplos de funciones escalares en Hibernate.

- Listado de colecciones

Listar los niveles de las diferentes Clases es sencillo:

```
SELECT C.nivel FROM Clase C
```

Ejemplo de código 2 - 20 Ejemplo de colecciones en HQL caso 1

También es fácil obtener la lista de clases:

```
FROM Clase
```

Ejemplo de código 2 - 21 Ejemplo de colecciones en HQL caso 2

Ahora bien, nos puede interesar listar todas las personas de una clase. Es decir, la colección de personas que pertenecen a una clase. Fijémonos que Clase tiene una propiedad personas que es un Set. Para obtener la lista de personas, tendremos que utilizar la función elements().

```
SELECT elements(C.personas) FROM Clase C
```

Ejemplo de código 2 - 22 Ejemplo de colecciones en HQL caso 3

- Contar Colecciones: size

Imaginémonos que nos interesa saber qué clases tienen más de 20 personas relacionadas. Hibernate nos ofrece la posibilidad de contar colecciones mediante una propiedad especial, size, o una función especial, size(). Así, podremos formular nuestra pregunta de dos maneras distintas:

```
FROM Clase C WHERE C.personas.size > 20
```

Ejemplo de código 2 - 23 Ejemplo para contar colecciones en HQL(size)

- La cláusula Group By

Esta cláusula tiene el mismo sentido que en SQL, así que no nos entretendremos demasiado. Un ejemplo sencillo que nos cuente el número de personas por clase nos puede servir:

```
SELECT C.nivel, C.nombre, count(elements(C.personas))
FROM Clase C
GROUP BY C
```

Ejemplo de código 2 - 24 Ejemplos para contar colecciones en HQL(group by)

- Ejecución de consultas

Existen diversos métodos para ejecutar consultas. Hasta ahora, todas las que hemos visto en los ejemplos utilizaban el método Session.load(), si la búsqueda se hacía a partir del id, o Session.find(), si ejecutábamos una consulta compleja expresada en HQL. Hay, sin embargo, más maneras de ejecutar una consulta.
En este capítulo, veremos algunos de estos métodos y cómo se pueden pasar parámetros a una consulta.

- El método “Session.find()”

Este método, lo hemos visto ya en la mayor parte de los ejemplos. Nos devuelve el resultado de la consulta en una java.util.List. Es bastante práctico si se devuelven pocos resultados, ya que los tiene que mantener en memoria:

```
List personas = sesion.find(“FROM Persona P WHERE P.nombre = 'Berta'”);
```

Ejemplo de código 2 - 25 Ejemplo del método “sesion.find()”

- El método “Session.iterate()”

Este método nos devuelve un java.util.Iterator y es práctico si la consulta nos proporciona un gran número de resultados.
El iterador se encarga de cargar los objetos resultantes de la consulta uno a uno, a media que los vamos pidiendo:

```
Iterator iterador = sesion.iterate(“FROM Personas AS P
ORDER BY P.apellidos, P.nombre”);
while (iterador.hasNext()) {
// Obtiene el objeto
Persona unaPersona = (Persona)iterador.next();
// Alguna cosa no expresable en la consulta
if (algoritmoComplicado(unaPersona)) {
// Borramos la instancia actual
iterador.remove();
// No hace falta que sigamos
break; } }
```

Ejemplo de código 2 - 26 Ejemplo del método “sesion.iterate()”

- Parámetros posicionales

Todas las consultas que hemos visto hasta ahora eran fijas; es decir, sabíamos todo lo que necesitábamos para hacer la consulta. Por ejemplo:

```
List personas = sesion.find("FROM Persona P WHERE P.nombre = 'Jaime'");
```

Ejemplo de código 2 - 27 Ejemplo del método "sesion.find()" caso parámetros posicionales

Buscamos aquellas personas que se llaman 'Jaime'.
Pero, a veces, nos puede interesar pasar el nombre como un parámetro. Por ejemplo, si tenemos la consulta ya definida y la ejecutamos diversas veces variando el nombre que buscamos.
El método Session.find() nos permite especificar parámetros:

```
String consulta = "FROM Persona P WHERE P.nombre = ?";
List personas = sesion.find(consulta, unNombre, Hibernate.STRING);
[...]
List personas = sesion.find(consulta, otroNombre, Hibernate.STRING);
```

Ejemplo de código 2 - 28 Ejemplo del método "sesion.find()" especificación de un solo parámetro

Fijémonos que en las dos ejecuciones de find() usamos el mismo enunciado de la consulta, pero en el primer caso preguntamos por un Nombre y en el segundo por otro Nombre.
En el enunciado de la consulta, usamos un parámetro por posición, igual que lo haríamos en JDBC, usando un signo de interrogación.
El ejemplo que hemos puesto es sencillo porque sólo tiene un parámetro. El método find() nos permite especificar el valor de este parámetro (unNombre, otroNombre) y nos obliga a especificar el tipo Hibernate del parámetro (Hibernate.STRING). La clase Hibernate define constantes estáticas que nos permiten acceder a instancias de net.sf.hibernate.type.Type, lo cual nos facilita la vida.
Si la consulta tuviera más de un parámetro, tendríamos que especificar los valores y los tipos mediante matrices de valores y de tipos:

```
String cons = "FROM Persona P WHERE P.nombre = ? AND P.apellidos LIKE ?"
List personas = Sesion.list(cons,
new Object[] {"Jaime", "R%"},
new Type[] {Hibernate.STRING,
```

```
Hibernate.STRING});
```

Ejemplo de código 2 - 29 Ejemplo del método "sesion.find()" especificación de varios parámetros

- Parámetros con nombre

Todos los que hemos trabajado con JDBC sabemos qué quiere decir tener que contar interrogantes en una consulta con 25 parámetros: acertar a la quinta.

La interface Query nos permite especificar parámetros con nombre, además de los posicionales. Esto tiene varias ventajas:

- El orden en que aparezcan dentro de la tira de la consulta es indiferente
- Un mismo parámetro puede aparecer diversas veces en la misma consulta
- Si el gestor de bases de datos soporta *Scrollable Result Sets*, nos permite movernos arriba y abajo por los resultados de la consulta
- Son auto explicativos

Para especificar un parámetro con nombre, sólo hay que poner dos puntos (:) seguidos del nombre del parámetro:

```
// Parámetro con nombre (preferido)
Query q = sesion.createQuery("FROM CATEGORIAS C " +
WHERE C.categoria = :nombreCategoria";
q.setString("nombreCategoria", "Profesor");
Iterator categorias = q.find();
// Parámetro posicional
Query q = sesion.createQuery("FROM CATEGORIAS C " +
WHERE C.categoria = ?";
q.setString(0, "Profesor");
Iterator categorias = q.find();
```

Ejemplo de código 2 - 30 Ejemplo de parámetros con nombre en Hibernate caso 1

También podemos pasar una lista de parámetros con nombre:

```
List nombresCateg = new ArrayList();
nombresCateg.add("Profesor");
nombresCateg.add("Alumno");
Query q = sesion.createQuery("FROM CATEGORIAS C " +
```

```
WHERE C.categoria IN (:listaNombresCateg);
q.setParameterList("listaNombresCateg", nombresCateg);
```

Ejemplo de código 2 - 31 Ejemplo de parámetros con nombre en Hibernate caso 2

En este caso, estamos buscando todas las categorías cuyos nombres (C.categoria) estén en la lista de nombres nombresCateg.

- Lectura de resultados

Hemos visto ya cómo escribir consultas en HQL y cómo ejecutarlas, pero aún no hemos visto cómo leer los resultados de la ejecución de la consulta. Veámoslo.
En algunos de los ejemplos que hemos visto, ya leíamos resultados. Es el caso de aquéllos que utilizan el método load(), ya que este método siempre devuelve un solo objeto:

```
Persona p = sesion.load(Persona.class, new Integer(1));
```

Ejemplo de código 2 - 32 Ejemplo de lectura de resultados ("sesion.load()")

También hemos visto aquellos casos de find() que nos devuelven una java.util.List con la lista de objetos resultante, todos del mismo tipo:

```
List personas = sesion.find("FROM Persona P WHERE p.nombre LIKE 'A%');
```

Ejemplo de código 2 - 33 Ejemplo de lectura de resultados ("sesion.find()")

Los objetos resultantes están en la List personas. Sólo hay que recorrer la lista para acceder a ellos.
A veces, sin embargo, la consulta nos devuelve diversos objetos de diversos tipos. Por ejemplo, la consulta:

```
SELECT C.nivel , C.nombre, count(elements(C.personas))
FROM Clase C
GROUP BY C
```

Ejemplo de código 2 - 34 Ejemplo de lectura de resultados con varios tipos de objetos

Nos devolvería un objeto de tipo Nivel, otro de tipo String y un Integer. Podríamos leer los resultados de la consulta usando el método iterate():

```
String consulta = "SELECT C.nivel , C.nombre, " +
"count(elements(C.personas)) " +
```

```
"FROM Clase C " +
"GROUP BY C";
Iterator i = sesion.iterate(consulta);
while (i.hasNext()) {
Object[] registro = (Object[])i.next();
Nivell nivel = (Nivell)registro[0];
String nombre = (String)registro[1];
Integer personas = (Integer)registro[2];
}
```

Ejemplo de código 2 - 35 Ejemplo de lectura de resultados usando "iterate()"

Como podemos ver, el iterador nos devuelve un registro por iteración y este registro es una matriz de objetos: uno por cada una de las columnas de la tabla de resultados. Sólo tenemos que hacer la conversión al tipo esperado (Nivel, String e Integer en nuestro ejemplo) y ya tendríamos los resultados.

2.2.5 Paginación Con Hibernate

Navegar por multitudes de datos es una de las principales tareas a las que deben hacer frente las aplicaciones, saber encontrar lo que se quiere, y mostrar la información de una manera ordenada y rápida es una tarea que no por elemental suele estar bien resuelta. Uno de los casos más comunes es tener miles de registros almacenados en una base de datos y querer mostrarlos a los usuarios de una manera tabular ordenados por algún criterio, en este caso, enseñar todos los registros de golpe es un disparate, tanto desde el punto de vista de utilidad, es difícil encontrar una aguja en un pajar , como técnico, cargar miles de objetos que no se van a usar solo ralentiza la aplicación, y ocupa un espacio muy valioso cuando de una aplicación multiusuario se está hablando. La solución con la que se suele enfrentar a este problema una aplicación es seguir el viejo axioma romano "divide y vencerás", mostrar solo una parte de los datos y dar la posibilidad y los recursos necesarios a el usuario para que pueda acceder al resto de los datos mediante alguna interacción con la pantalla. Así se consiguen solucionar lo antes mencionado, al usuario se le muestran unos pocos datos, en los que puede centrarse y analizar, y el sistema no se recarga de manera innecesaria. Bueno, esto es en teoría, porque la manera "tradicional" sigue siendo cargar todos los objetos en memoria y paginarlos usando las api de la interfaz List, con lo que el sistema se

sigue recargado. La solución correcta sería dividir los datos en origen, en la base de datos, y cargar únicamente en el entorno java los elementos que se necesitan mostrar, pero eso implica, conocer las extensiones que cada motor de base de datos aporta para estas misiones, al no ser una de las facilidades incorporadas al estándar SQL.

- Consideraciones Generales

La solución que aporta hibernate es bella y simple a la vez, como casi todo en hibernate. Hibernate ofrece, principalmente, la posibilidad de realizar consultas a la base de datos de dos maneras distintas mediante sentencias HQL o mediante el api de criterios, en un caso se emplea la interfaz Query y en el otro la interfaz Criteria, pero ambas interfaces han sido dotadas de los mismos dos métodos:

- setFirstResult. Nos permite posicionarnos en un registro determinado de la tabla. Empezando por cero.
- setMaxResults. Establece cuantos registros, a partir del especificado con el método setFirstResult, recuperaremos de la base de datos.

La ventaja de la combinación de los anteriores métodos es que hibernate traduce nuestra petición y construye la sentencia sql correcta para cada motor, recuperando solo los registros necesarios, sin que sepamos las particularidades de cada base de datos.
Como un ejemplo vale más que mil palabras, hagamos un caso práctico. En una tabla llamada ABECEDARIO, hemos guardado todas las letras que forman parte del alfabeto español, si quisiéramos recuperar todas las letras con hibernate, escribiríamos con HQL algo parecido a:

```
Query query = session.createQuery("from Abecedario order by letra asc");
List  resultados  = query.list();
```

Ejemplo de código 2 - 36 Ejemplo sencillo de la interface Query en Hibernate

Con el código anterior obtendríamos una lista con todas las 26 letras , 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L' ,'M' ,'N' ,'O' ,'P' ,'Q' , 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'.
Hagamos una pequeña modificación al código anterior y empleemos los métodos que me aporta hibernate para paginar, con ellos quisiéramos recuperar únicamente las letras comprendidas entre las posiciones 6 y 10, 'F', 'G', 'H', 'I', 'J', la expresión quedaría como:

```
Query query = session.createQuery("from Abecedario a order by a.letra
asc").setFirstResult(5).setMaxResults(5);
List resultados = query.list();
```

Ejemplo de código 2 - 37 Paginación en Hibernate

Como se ve en el código superior, la sentencia HQL no ha cambiado, sigue siendo la misma, lo único que hemos hecho es añadir información al objeto que implementa la interfaz Query para que sepa el rango de registros que estamos buscando.

Con estos dos métodos de paginación nos garantizamos que nuestra aplicación va a rendir mejor y ser más estable, al no cargar toneladas de datos basura como se hacía antes.

Hibernate solo construye la sentencia SQL adecuada para cada motor de búsqueda, nada más, y nada menos, pero el rendimiento de una sentencia con paginación puede ser muy distinto al rendimiento de una sin paginar, sobre todo cuando hay uniones de tablas involucradas. Así que antes de usar la paginación, se debe probar las distintas combinaciones que ofrece hibernate para asegurarse de que se ha escogido el mejor sistema para recuperar los datos. Reiterando, el rendimiento de las paginaciones no depende de hibernate, sino del motor de base de datos, asegurarse y realizar pruebas para escoger siempre el mejor sistema de acceso a sus recursos.

2.3 Framework ZK

2.3.1 Introducción a ZK Ajax Framework

El mundo de desarrollo de aplicaciones web crece y crece. Podemos leer muchos artículos y libros que explican cómo desarrollar una aplicación web. Casi cada semana surge un nuevo framework en el ámbito del desarrollo. Sin embargo, cada nueva tecnología significa que tenemos que aprender.

Sin aprendizaje, no es posible aprovechar la potencia completa de la técnica elegida. Hay algunas razones para el surgimiento de una amplia gama de posibilidades para el desarrollo de una aplicación web. La evolución de la tecnología es una de las razones, y otra es la demanda de una manera más rápida y más eficiente para desarrollar una aplicación web. Cada desarrollador tiene su preferencia propia por el camino del desarrollo. Sin embargo, no sólo los problemas en el desarrollo de la demanda de nuevas formas de construir una aplicación web. El panorama general es de la web 2.0 y las técnicas subyacentes AJAX (Asynchronous JavaScript and XML).

Las aplicaciones web han evolucionado a partir de páginas HTML estáticas, a páginas HTML dinámico (DHTML), a páginas con applets y Flash, y, finalmente, a los que incorporan tecnologías como Ajax. Dos grandes ejemplos de Ajax son Google Maps y Google Suggest.
Ajax da nueva vida a las aplicaciones web mediante la entrega del mismo nivel de interactividad y capacidad de respuesta en las aplicaciones de escritorio. Sin embargo, a diferencia de applets o Flash, Ajax está basado en el navegador y JavaScript, y no es necesaria la utilización de un plug-in.
Ajax es una especie de nueva generación DHTML, por lo que depende en gran medida de JavaScript, escuchar a los eventos provocados por la actividad del usuario y manipular la representación visual de una página en el navegador de manera dinámica es la función principal de JavaScript.
Como hemos mencionado, el éxito de una aplicación web dinámica depende de la capacidad del desarrollador en conocer correctamente el funcionamiento de Ajax y JavaScript, es por eso que existe la necesidad de un framework, que este basado en Ajax y no se necesite mayor conocimiento en JavaScript ya que implementar estas tecnologías resulta muy tedioso para el desarrollador.
Para satisfacer las necesidades antes mencionadas ha nacido el framework basado en Ajax **"ZK"** ya que este no necesita tener ningún conocimiento de JavaScript para desarrollar aplicaciones basadas en la web, ya que el motor ZK genera automáticamente el código JavaScript.
Para desarrollar una aplicación web con ZK, lo que se necesita es saber sólo un poco de HTML. Para simplificar el desarrollo de aplicaciones web, el equipo ZK ha definido la interfaz de usuario mediante el lenguaje de marcado ZUML para proveer una manera intuitiva para crear componentes, simplemente declarando una etiqueta, que es similar a las etiquetas HTML.

- **Características del Framework ZK**
  - Experiencia de usuario, la accesibilidad Web obedece a los componentes Ajax con componentes que ofrecen unas atractivas ventajas para crear y maximizar la satisfacción y la experiencia del usuario y la eficiencia del trabajo.
  - RIA Directo, ZK fomenta la productividad de los desarrolladores permitiendo una manipulación directa de la capa UI, base de datos y recursos empresariales. Con

programación directa, desarrollando aplicaciones web de una forma directa e intuitiva al igual que las aplicaciones de escritorio.

- Open Source, ZK es framework de código abierto líder en el desarrollo de Ajax la comunidad de ZK es extremadamente activa con 20 traducciones, 100 artículos/blogs, 100,000 líneas de código, 1,200,000 descargas desde más de 190 países.
- Basado en estándares, ZK es una solución estándar. Con elementos compilados XUL y HTML, ZK te protege de las tecnologías de puertas cerradas y propietarias. ZK también compila con JSP, JSF, Portlet y tecnologías JEE incluyendo la habilidad de integrarse con entornos e IDEs java ya existentes.
- Marcas y Lenguajes de Script, Las aplicaciones ZK pueden ser construidas en Java puro, con marcas o con lenguaje de script. Con XUL/XHTML, diseñando interfaces de usuario ricas tan sencillas como diseñar un HTML, Groovy y otros lenguajes de scripting están soportados, desarrollar aplicaciones es tan rápido como hacer el prototipo.
- Direct Push, ZK proporciona de manera excepcional, una tecnología robusta e intuitiva denominada Direct Push que permite actualizar espontáneamente la información enviada por el servidor al navegador Ajax y a los dispositivos móviles. Con la tecnología Direct Push de ZK, las soluciones empresariales dinámicas pueden ser creadas a un coste mínimo.
- Personalización y extensibilidad, ZK es completamente personalizable y extensible con una arquitectura modular plug-and-play, CSS, plantillas y componentes macro, la imagen y el comportamiento de los componentes puede ser cambiado sin modificar la aplicación. Con una factoría de Interfaces de Usuario (UI) "arrastrables", cada usuario puede tener una UI personal cargada desde la base de datos o mediante otros recursos.
- Seguridad, ZK está diseñado desde el inicio para ser seguro. ZK protege a las aplicaciones empresariales de inyecciones de JavaScript/SQL maliciosas, la exposición a la capa cliente de la lógica de negocio y la exposición de los datos en tránsito.
- Escalabilidad, Clustering y fallos, ZK soporta alta escalabilidad y disponibilidad con Interfaces de Usuario serializables, y un gestor de errores arrastrable. ZK es

también compatible con el clustering y balanceo de carga soportados por los servidores de aplicaciones modernos.

- Arquitectura “Server-client-fusion” de ZK

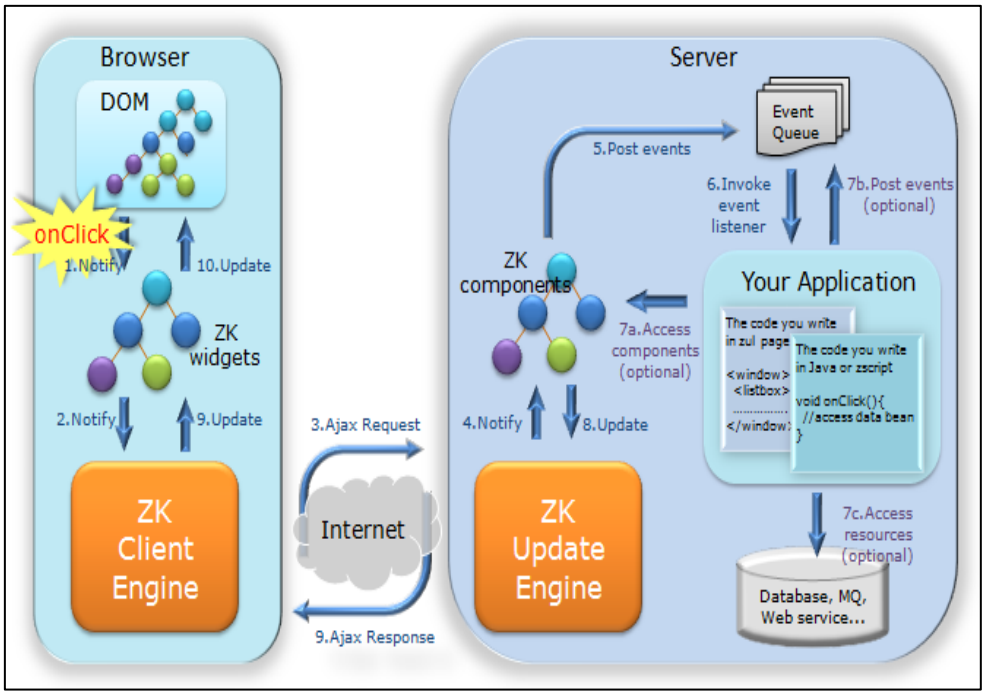

Figura 2 - 6 Arquitectura Sever-client-fusion de ZK[16]

ZK se ejecuta en el servidor, puede tener acceso a los recursos del servidor, montar la interfaz de usuario con los componentes, escuchar a la actividad del usuario, y luego manipular los componentes para actualizar la interfaz de usuario. Todo se lleva a cabo en el servidor. La sincronización de los estados de los elementos comprendidos entre el navegador y el servidor se realiza automáticamente por ZK y transparente para la aplicación. Cuando se ejecuta en el servidor, la aplicación puede acceder a la pila completa de la tecnología Java. Las actividades del usuario, incluyendo Ajax y el empuje del servidor, se abstraen de los objetos de evento. La interfaz de usuario se compone como un componente POJO. ZK es el enfoque más productivo para desarrollar una aplicación web moderna. Con la arquitectura “Server-client-fusion”, la aplicación nunca va a dejar de correr en el servidor. La aplicación podría mejorar la interactividad mediante la adición opcional de la funcionalidad del cliente, tales como el manejo de eventos del lado del cliente, la personalización de efecto visual, incluso la interfaz de usuario componer sin necesidad de

[16] http://books.zkoss.org/wiki/ZK_Developer's_Reference/Overture/Architecture_Overview

programación del lado del servidor. ZK permite la fusión perfecta en el servidor y el cliente. Así obtenemos lo mejor de dos mundos: la productividad y la flexibilidad.

2.3.2 Componentes ZK

El lenguaje que utilizamos para declarar los componentes es ZUML, una abreviatura de lenguaje de marcado de la interfaz de usuario ZK. ZUML sigue la sintaxis de XML. Aquí detallamos un par de notas básicas:

- Los elementos deben estar bien formados
  - declaración de cierre con una etiqueta de cierre:

```
<window></window>
```

Ejemplo de código 2 - 38 Ejemplo de declaración de cierre con una etiqueta de cierre en ZUML

  - declaración de cierre sin una etiqueta de cierre (equivalente a la afirmación anterior):

```
<window/>
```

Ejemplo de código 2 - 39 Ejemplo de declaración de cierre sin una etiqueta de cierre en ZUML

- Los elementos deben estar anidados correctamente:
  - Correcto:

```
<window>
        <groupbox>
           Hello World!
        </groupbox>
</window>
```

Ejemplo de código 2 - 40 Ejemplo de uso correcto de declaración componentes en ZUML

  - Incorrecto

```
<window>
        <groupbox>
           Hello World!
        </window>
```

```
</groupbox>
```

Ejemplo de código 2 - 41 Ejemplo de uso incorrecto de declaración componentes en ZUML

- **Sólo una única "raíz" de componentes se permite:**
  - una de las raíces – legal

```
<button />
```

Ejemplo de código 2 - 42 Ejemplo de uso correcto de declaración de una única raíz sin componentes en ZUML

  - dos raíces – ilegal

```
<button />
<button/>
```

Ejemplo de código 2 - 43 Ejemplo de uso incorrecto de declaración con más de una raíz en ZUML

  - una raíz que contiene todos los otros componentes – legal

```
<window>
        <button/>
        <button/>
</window>
```

Ejemplo de código 2 - 44 Ejemplo de uso correcto de declaración de una única raíz con más de un componente en ZUML

- **Valor del atributo debe ser citado**
  - Correcto:

```
<window width="600px"/>
```

Ejemplo de código 2 - 45 Ejemplo de uso correcto de declaración un componente con atributos en ZUML

  - Incorrecto:

```
<window width=600px/>
```

Ejemplo de código 2 - 46 Ejemplo de uso inccorrecto de declaración de un componente con atributos en ZUML

En ZK, se trabaja con componentes de interfaz de usuario, para ello podemos declarar los componentes utilizando el lenguaje de marcado o Java.

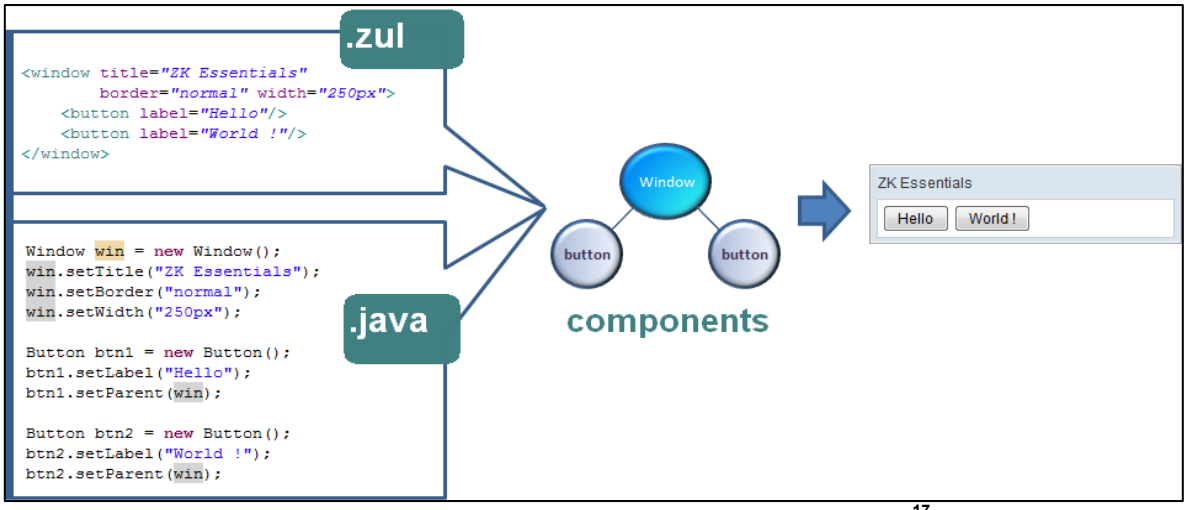

Figura 2 - 7 Componentes de interfaz de usuario en ZK[17]

Los componentes se declaran en archivos con la extensión ".zul". Una página ZUL se interpreta de forma dinámica en el servidor. Podríamos pensar en él como una JSP con capacidad Ajax.

Por ejemplo, aquí creamos un nuevo archivo ZUL, ajax.zul, y poner en práctica una muestra, en la que la entrada de los usuarios en el cuadro de texto se refleja en la etiqueta debajo de forma instantánea cuando el cuadro de texto pierde el foco:

```
<window title="ZK Essentials" border="normal" width="250px">
   <vlayout>
   <textbox id="txtbx" onChange="lbl.value = txtbx.value"/>
   <label id="lbl"/>
   </vlayout>
</window>
```

Ejemplo de código 2 - 47 Ejemplo de archivo zul en ZK

La declaración de marcas anterior, realiza un programa de ejemplo, como se ve a continuación:

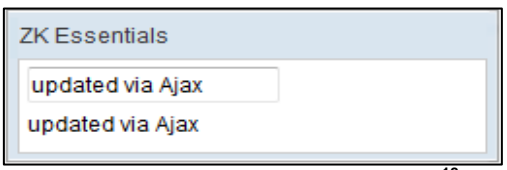

Figura 2 - 8 Programa ZK (ajax.zul)[18]

[17] http://books.zkoss.org/wiki/File:ZKEssentials_Intro_Hello.png

Los componentes de ZK pueden ser fácilmente declarados en los códigos fuente de Java como se indica a continuación:

```
package org.zkoss.zkdemo;
import org.zkoss.zk.ui.Page;
import org.zkoss.zk.ui.GenericRichlet;
import org.zkoss.zul.*;
 public class TestRichlet extends GenericRichlet {
   //Richlet//
   public void service(Page page) {
      final Window win = new Window("ZK Essentials", "normal", false);
      win.setWidth("250px");
      Vlayout vl = new Vlayout();
      vl.setParent(win);
      final Textbox txtbx = new Textbox();
      txtbx.setParent(vl);
      final Label lbl = new Label();
      lbl.setParent(vl);
      txtbx.addEventListener("onChange", new EventListener(){
        @Override
        public void onEvent(Event event) throws Exception {
           lbl.setValue(txtbx.getValue());
        }
      });
      win.setPage(page);
   }
}
```

Ejemplo de código 2 - 48 Ejemplo de componentes ZK declarados desde Java

En la siguiente figura nos muestra la manera en que los componentes ZK, se convierten a instrucciones que pueden ser entendidas por el cliente:

[18] http://books.zkoss.org/wiki/File:ZKEssentials_Intro_AjaxZUL.png

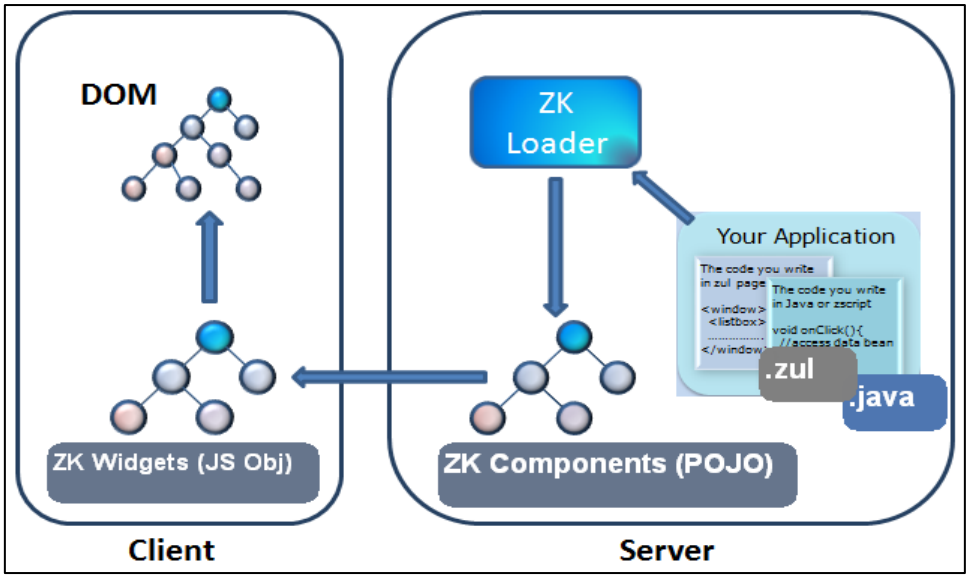

Figura 2 - 9 Componentes ZK interpretados por el cliente[19]

Los componentes que son declarados en un archivo con extensión .zul como se indica en el ejemplo de código "2-44", posteriormente son analizados por "ZK – loader", luego estos son creados como POJO en la máquina virtual de java del servidor, para de ahí, convertirse en instrucciones (JSON) para realizar la creación de widget (JavaScript) y finalmente ser enviados al cliente.

- **Acceso a un componente especifico en lenguaje de marcado (ZUML)**

Con los componentes anidados y apilados juntos para darnos nuestra interfaz de usuario de la aplicación, necesitamos una forma de identificar a los necesarios para la procesión. Por ejemplo, es posible que necesitemos añadir dinámicamente un componente de un componente existente, o que está programado que el comportamiento de uno de los componentes depende de la de otro. El siguiente ejemplo ilustra el caso:

Figura 2 - 10 Acceso a un componente en ZK[20]

[19] http://books.zkoss.org/wiki/File:ZKEssentials_Intro_ZULtoPOJO.png

La fuente de marcado es:

```
<window title="ZK Essentials" mode="overlapped" border="normal" width="250px">
   <label id="lbl"/>World !
   <button label="Hello " onClick="lbl.value = self.label"/>
   <button label="Good-bye " onClick="lbl.value = self.label"/>
</window>
```

Ejemplo de código 2 - 49 Ejemplo de acceso a un componente en ZUML

El valor de la etiqueta con la identificación "lbl" depende de que el usuario hace clic en el botón: "Hola", o "Adiós". Cuando se hace clic en un botón, el valor de la etiqueta del botón se le asigna el valor de la etiqueta. Tenga en cuenta que la cadena "Mundo!" se convierte automáticamente en una etiqueta. Sin asignar en primer lugar "lbl" al componente de etiqueta deseada, sería imposible ir a buscar "lbl" y dotarla de un valor. De hecho, ZK asigna a cada componente con un UUID (Identificador Único Universal) para realizar un seguimiento de su árbol de los componentes internos. Este UUID es anulado cuando el desarrollador le proporciona una identificación más legible.

- **Acceso a un componente especifico en JAVA**

Supongamos que tenemos una declaración POJO para una interfaz de usuario simple que se parece a esto:

```
Window outerWin = new Window();
 Button outerBtn = new Button();
btn.setParent(outerWin);
 Window innerWin = new Window();
innerWin.setParent(outerWin);
 Button innerBtn = new Button();
innerBtn.setParent(innerWin);
```

Ejemplo de código 2 - 50 Ejemplo de acceso a un componente en JAVA (código java)

Para una mejor legibilidad, la declaración equivalente en el archivo ZUL se da así:

```
<window id="outerWin">
```

[20] http://books.zkoss.org/wiki/File:ZKEssentials_Intro_HelloGoodbye.png

```
<button id="outerBtn" />
 <window id="innerWin">
    <button id="innerBtn"/>
 </window>
</window>
```

Ejemplo de código 2 - 51 Ejemplo de acceso a un componente en JAVA (código ZUML)

Ahora supongamos que tenemos una clase de controlador en la que desea acceder y modificar mediante programación los componentes hijos (outerBtn, innerWin, innerBtn).

¿cómo debemos hacerlo si sólo tenemos acceso a los componentes de la ventana?

Uno de los métodos para lograr esto es el método Component.getFellow(). Por ejemplo, si queremos acceder a la ventana interior, se podría hacer lo siguiente:

```
Window innerWin = (Window)outerWin.getFellow("innerWin");
```

Ejemplo de código 2 - 52 Ejemplo de acceso a un componente en JAVA (código controlador)

En la siguiente figura se ilustra el árbol de componentes que hace referencias al ejemplo de código “2-51”.

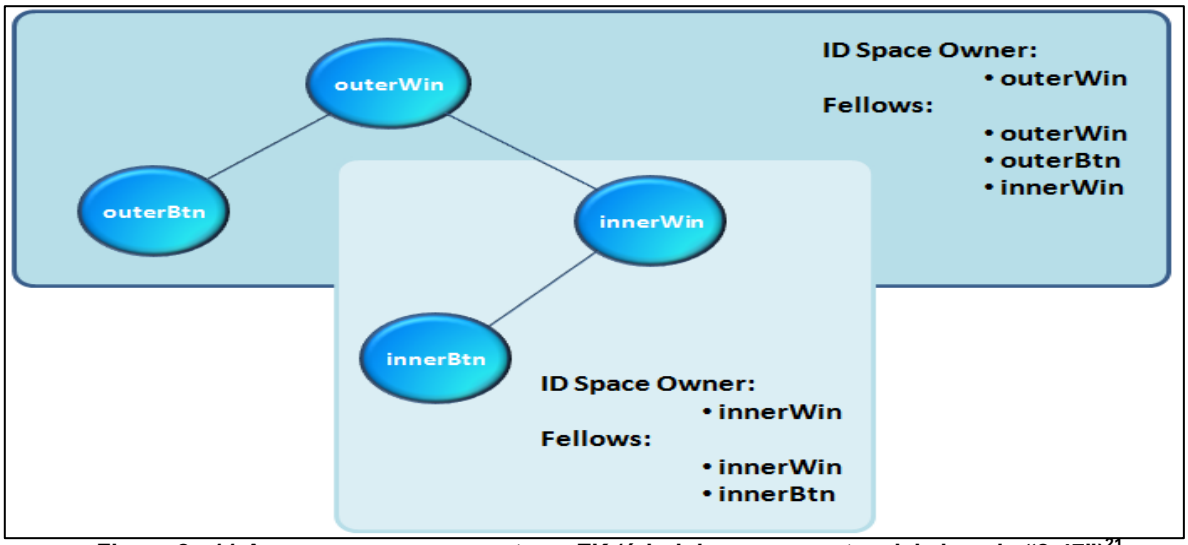

Figura 2 - 11 Acceso a un componente en ZK (árbol de componentes del ejemplo “2-47”)[21]

Hay otros métodos para lograr acceder a los componentes de la figura anterior, la siguiente tabla nos muestra dichos métodos:

[21] http://books.zkoss.org/wiki/File:ZKEssentials_Intro_IDSpace.png

Componente	método
outerBtn	= (Button) outerWin.getFellow ("outerBtn");
innerWin	= (Window) outerWin.getFellow ("innerWin");
innerBtn	= (Button) outerWin.getFellow ("innerWin") getFellow ("innerBtn");
outerBtn	= (Button) Path.getComponent ("/outerBtn");
innerWin	= (Window) Path.getComponent ("/innerWin");
innerBtn	= (Button) Path.getComponent ("/innerWin innerBtn");
outerBtn	= (Button) outerWin.getFirstChild ();
innerWin	= (Window) outerWin.getFirstChild () getNextSibling ().;
innerBtn	= (Button) outerWin.getFirstChild () getNextSibling () getFirstChild ();

Tabla 2 - 2 Métodos para obtener un componente en ZK

2.3.3 ZK Modelo Vista Controlador

En términos de ZK MVC se compone de los siguientes elementos. El "modelo" es su objeto de negocio y servicios de oficina. La "Vista" es el conjunto de componentes en el navegador se define en ZUML (zul, zhtml), son los archivos que no contienen la lógica de

procesamiento de eventos. El "Controlador" es una clase pura en Java que está registrado en un EventListener uno o más componentes en el navegador.
Vamos a ejemplificar el uso de ZK MVC, supongamos que tenemos dos cuadros de texto de entrada para introducir el nombre y apellido, y además, un texto solo de lectura que desplegara automáticamente cuando el texto de entrada ha cambiado.

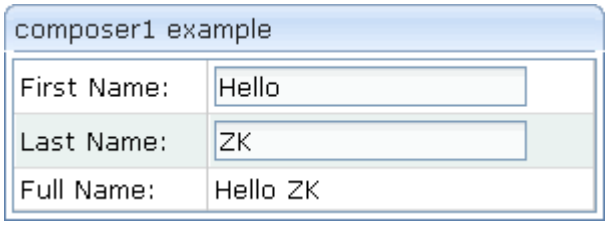

Figura 2 - 12 Ventana de ZK con textos de entrada

- Mediante la clase “Composer”

El código zuml de la figura 2-12 sería el siguiente:

```
<window title="composer1 example" border="normal" width="300px"
apply="MyComposer1">
   <grid>
      <rows>
         <row>First Name: <textbox id="firstName"
forward="onChange=onFirstName"/></row>
         <row>Last Name: <textbox id="lastName"
forward="onChange=onLastName"/></row>
         <row>Full Name: <label id="fullName"/></row>
      </rows>
   </grid>
</window>
```

Ejemplo de código 2 - 53 Ejemplo de código ZUML (composer1.zul)

Como podemos observar el componente “window” contiene el atributo “apply” el cual indica que clase java se va a utilizar en este caso se usa la clase “MyCompeser1” la cual quedaría de la siguiente manera:

```
...
public class MyComposer1 implements Composer {
```

```
    private Textbox firstName;
    private Textbox lastName;
    private Label fullName;

    public void doAfterCompose(Component win) throws Exception {
       firstName = (Textbox) win.getFellow("firstName");
       lastName = (Textbox) win.getFellow("lastName");
       fullName = (Label) win.getFellow("fullName");

       //define and register event listeners
       win.addEventListener("onFirstName",
          new EventListener() {
             public void onEvent(Event event) throws Exception {
                fullName.setValue(firstName.getValue()+" "+lastName.getValue());
             }
          });
       win.addEventListener("onLastName",
          new EventListener() {
             public void onEvent(Event event) throws Exception {
                fullName.setValue(firstName.getValue()+" "+lastName.getValue());
             }
          });
    }
}
```

Ejemplo de código 2 - 54 Código Java del ejemplo 2-12 (MyComposer1.java) mediante la clase "Composer"

La clase MyComposer1.java interviene en el ciclo de vida de la creación de la ventana y es responsable de la definición y el registro del detector de eventos.

- Mediante la clase "GenericComposer"

Por medio de la clase de utilidad "GenericComposer" podemos eliminar las líneas de código que contienen "addEventListener" que se encuentran en el ejemplo del código anterior, para eliminar dichas líneas, la clase "MyComposer1.java" quedaría de la siguiente forma:

```
...
public class MyComposer1 extends GenericComposer {
private Textbox firstName;
private Textbox lastName;
private Label fullName;
public void doAfterCompose(Component win) throws Exception {
      super.doAfterCompose(win);
      //locate ZK components
      firstName = (Textbox) win.getFellow("firstName");
      lastName = (Textbox) win.getFellow("lastName");
      fullName = (Label) win.getFellow("fullName");
      //all addEventListener and new EventListener() codes are removed
   }
    public void onFirstName(Event event) {
      fullName.setValue(firstName.getValue()+" "+lastName.getValue());
   }
   public void onLastName(Event event) {
      fullName.setValue(firstName.getValue()+" "+lastName.getValue());
   }
}
```

Ejemplo de código 2 - 55 Código Java del ejemplo 2-12 (MyComposer1.java) mediante la clase "GenericComposer"

- Mediante la clase "GenericForwardComposer"

Si se desea evitar las líneas de código zuml "forward=onChange=onFirstName" se puede hacer uso de la clase GenericForwardComposer, la clase "MyComposer1.java" quedaría de la siguiente forma:

```
...
public class MyComposer1 extends GenericForwardComposer {
   private Textbox firstName;
   private Textbox lastName;
   private Label fullName;
   //onChange event from firstName component
```

```
public void onChange$firstName(Event event) {
   fullName.setValue(firstName.getValue()+" "+lastName.getValue());
}
//onChange event from lastName component
public void onChange$lastName(Event event) {
   fullName.setValue(firstName.getValue()+" "+lastName.getValue());
}
```

Ejemplo de código 2 - 56 Código Java del ejemplo 2-12 (MyComposer1.java) mediante la clase "GenericForwardComposer"

Cuando se escribe el compositor que se extiende GenericForwardComposer, usted tiene que seguir algunas reglas de nomenclatura en sus métodos de detector de eventos. Como se puede observar en el ejemplo anterior la clase MyComposer1, contiene el oyente de eventos con la siguiente nomenclatura "nombre del evento" seguido del símbolo $ y luego el "id" del Componente.

Así que ahora tenemos una clara separación entre la vista en el archivo. Zul y el controlador.

2.3.4 Integración de ZK Framework con Hibernate

Bueno, Hibernate no tiene nada que ver con la capa de la vista, además posee una gestión de sesiones diferentes. A diferencia del ambiente de ZK, es por eso que los desarrolladores de ZK han implementado un "listener" propio para Hibernate.

Este "listener" tiene que ser configurado en el archivo "zk.xml" de la siguiente forma:

```
<listener>
    <listener-lass>
        modelo.entidades.HibernateListeners
    </listener-class>
</listener>
```

Ejemplo de código 2 - 57 Código xml para configurar la inicialización de Hiberntae en ZK(zk.xml)

La clase "HibernateListeners" que se hace referencia en ejemplo anterior quedaría de la siguiente manera:

```
...
```

```
public class HibernateListeners implements WebAppInit, WebAppCleanup {
   public void init(WebApp webapp) throws Exception {
      //initialize Hibernate
      HibernateUtil.getSessionFactory();
   }
   public void cleanup(WebApp webapp) throws Exception {
      //Close Hibernate
      HibernateUtil.getSessionFactory().close();
   }
}
```

Ejemplo de código 2 - 58 Código Java que contiene la clase HibernetListeners para obtener el sessionFactory de Hibernate

En el ejemplo anterior, se realiza el mapeo correspondiente a Hibernate, al iniciar la aplicación en el ambiente de ZK, además, se hace referencia a la clase HibernateUtil la cuál es usada para obtener el “SessionFactory” de Hibernate, de esta forma se integraría en los archivos de configuración de ZK el mapeador objeto relacional “Hibenate”.

CAPÍTULO III PROCESO DE DESARROLLO

3.1 FASE DE INICIO

3.1.1 Artefacto de Visión del Negocio

3.1.1.1 Introducción

3.1.1.1.1 Propósito

El propósito de este artefacto es definir a alto nivel los requerimientos de la aplicación **Sistema De Atención De Reclamos** (SAR).

El sistema SAR se encargará de gestionar los procesos relacionados con la atención de reclamos de la Empresa Eléctrica Regional Norte (EMELNORTE), además está basado en la regulación del CONELEC No. CONELEC-012/08.

El detalle de cómo el sistema SAR cubrirá las necesidades de los usuarios se especifica en los casos de uso, que son información adicional que se trataran más adelante.

3.1.1.1.2 Alcance

Este artefacto de visión se aplica al **Sistema De Atención De Reclamos** que será implementado por el personal del Centro de Cómputo de EMELNORTE.

3.1.1.2 Posicionamiento

3.1.1.2.1 Oportunidad del negocio

A partir de los procedimientos ya establecidos en el área Comercial y de Distribución, y como parte del plan de automatización de todos los procesos de atención de reclamos establecidos por el Centro de Cómputo, se determina la optimización de la gestión de las actividades relacionadas al manejo de reclamos.

3.1.1.2.2 Definición del problema

El problema de	No Poseer un sistema web que recolecte los reclamos de los clientes de EMELNORTE. El proceso de manejo de reclamos que existe actualmente no cumple con la regulación del CONELEC No. CONELEC-012/08.

Que afecta a	EMELNORTE y clientes de la empresa.
El impacto de ello es	Carencia de información de los reclamos ingresados por los clientes de EMELNORTE, lo cual no permite un registro y atención eficiente de los mismos.
Una solución exitosa debería	Implementar una solución informática de calidad soportada por una metodología eficiente de desarrollo de software, la cual permitirá el registro de reclamos y atención optima de los mismos.

Tabla 3 - 1 Definición del problema (Artefacto de Visión)

3.1.1.3 Descripción de los interesados y usuarios

3.1.1.3.1 Resumen de los interesados

Interesados son todas aquellas personas directamente involucradas en la definición y alcance del proyecto. A continuación se presenta la lista de los interesados:

Nombre	Descripción	Responsabilidad
Coordinador del proyecto	Responsable a nivel directivo de EMELNORTE del proyecto.	Establecer los lineamientos generales para el desarrollo del proyecto. Coordinar a nivel directivo los diferentes requerimientos que surjan en el desarrollo del sistema.
Responsable del proyecto	Responsable del proyecto por parte del Centro De Computo de EMELNORTE.	Responsable del análisis y diseño del proyecto. Gestiona el correcto

		desarrollo de cada uno de los sub-módulos del proyecto en lo referente a la construcción e implantación.
Responsable funcional	Responsable del proyecto por parte del Departamento Comercial.	Responsable de coordinar con los diferentes usuarios la correcta determinación de los requerimientos y la correcta concepción del sistema.

Tabla 3 - 2 Lista de interesados (Artefacto de Visión)

3.1.1.3.2 Resumen de los usuarios

Los usuarios son todas aquellas personas involucradas directamente en el uso del Sistema de Atención de Reclamos a continuación se presenta una lista de los usuarios:

Nombre	Descripción	Responsabilidad
Administrador del sistema	Persona asignada por la dirección comercial que administrara el sistema SAR	Administrar eficazmente el sistema (gestionar acceso a usuarios, facilitar mantenimiento al sistema frente a nuevos requerimientos).
Responsable funcional del sistema	Personal de EMELNORTE para que administre el sistema SAR.	Administrar funcionalmente el sistema.
Usuario del sistema	Clientes de EMELNORTE.	Ingresar la información concerniente a reclamos que permitirá realizar las acciones pertinentes a los responsables de

		EMELNORTE.

Tabla 3 - 3 Resumen de usuarios (Artefacto de Visión)

3.1.1.3.3 Entorno de usuario

3.1.1.3.3.1 Coordinador del proyecto

Representante	Ing. Rene Brown
Descripción	Responsable a nivel directivo del proyecto.
Tipo	Coordinador del proyecto
Responsabilidades	Establecer los lineamientos generales para el desarrollo del proyecto. Coordinar a nivel directivo los diferentes requerimientos que surjan en el desarrollo del sistema.
Criterio de éxito	Mantener una funcionalidad integral en los módulos. Mantener activa la aplicación luego de ser implantada.
Implicación	Revisor de la administración.
Entregable	N/A
Comentario	Mantener una relación constante con el desarrollo del proyecto. Brindar apoyo a nivel gerencial cuando sea necesario.

Tabla 3 - 4 Coordinador del Proyecto (Artefacto de Visión)

3.1.1.3.3.2 Responsable del proyecto

Representante	Sr. David Bolaños
Descripción	Responsable del proyecto por parte del Centro de Computo de EMELNORTE
Tipo	Analista de sistemas
Responsabilidades	Responsable del análisis y diseño del proyecto. Gestiona el correcto desarrollo del proyecto en lo referente a la construcción e implantación.
Criterios de éxito	Cumplir con el cronograma determinado. Obtener un módulo de calidad que cumpla con los requerimientos funcionales establecidos.
Implicación	Jefe de proyecto (Project Manager)
Entregables	Documento de visión Resumen del modelo de casos de uso Especificaciones del modelo de casos de uso Especificaciones complementarias
Comentarios	

Tabla 3 - 5 Responsable del Proyecto (Artefacto de Visión)

3.1.1.3.3.3 Responsable funcional

Representante	Econ. Patricio Talabera

Descripción	Responsable del proyecto por parte del Área Comercial.
Tipo	Experto en el tema
Responsabilidades	Responsable de coordinar con los diferentes usuarios la correcta determinación de los requerimientos y la correcta concepción del módulo. Coordinar las pruebas de validación del nuevo módulo. Coordinar y asegurar la capacitación de los usuarios.
Criterios de éxito	Obtener un módulo de calidad que cumpla con los requerimientos funcionales establecidos.
Implicación	Aprueba las especificaciones funcionales y las pruebas realizadas.
Entregables	Documento de revisión de las especificaciones funcionales. Documento de revisión de las pruebas funcionales
Comentarios	

Tabla 3 - 6 Responsable funcional (Artefacto de Visión)

3.1.1.3.4 Necesidades de los interesados y usuarios

Necesidades	**Prioridad**	**Inquietudes**	**Solución Actual**	**Solución propuesta**
Diseñar un sistema que facilite la consolidación e integración de información concerniente a la atención de reclamos.	Alta	El sistema debe consolidar la información para facilitar el registro y atención de reclamos.	Tiene varios inconvenientes: carencia de facilidades, controles, integración y solidez.	Mejorar el proceso existente para que solucione estas necesidades.

Mantener un registro ordenado de reclamos	Alta	N/A	Actualmente no se lleva un registro de atención de reclamos.	Desarrollar interfaces que faciliten la integración .
Elaborar el sistema utilizando herramientas que facilite y agilice su desarrollo.	Alta	Se debe utilizar las herramientas existentes o adquirir nuevo software de desarrollo.	N/A	Desarrollar el sistema utilizando la herramienta Eclipse.
La interfaz del sistema debe ser fácil de manejar, cumpliendo con todos los requerimientos establecidos.	Alta	Cumplir con todos los requerimientos de los usuarios.	Desarrollo con la ayuda de los expertos en el Área Comercial y de Distribución	Desarrollo con la ayuda de los expertos en el tema.

Tabla 3 - 7 Necesidades de los interesados y usuarios (Artefacto de Visión)

3.1.1.3.5 Alternativas y competencia

3.1.1.3.5.1 Adquirir un sistema desarrollado externamente

Debido a que la regulación del CONELEC fue publicada recientemente no existe en el mercado una solución que se adapte a nuestra necesidad.

3.1.1.4 Vista General del Producto

Esta sección provee información a alto nivel de las funciones del sistema a implantar y de las interfaces con los sub-módulos que se están realizando.

3.1.1.4.1 Perspectiva del producto

Figura 3 - 1 Perspectiva del Producto (Artefacto de Visión)

3.1.1.4.2 Resumen de capacidades

Beneficios para el usuario	Características que lo soportan
Fácil acceso al sistema para ingreso de reclamos.	El sistema tendrá acceso web para registro de reclamos tanto de clientes como sus responsables.
Procesos más automáticos y mayor rapidez en los tramites	Se asignara automáticamente un responsable una vez que se ha ingresado el reclamo.
Los usuarios contarán con una herramienta unificada.	Se evitará la pérdida de tiempo, y se podrá contar una base de datos sólida para evitar pérdida de información.
Facilidades para el análisis de la información.	Brindará diferentes reportes y funciones de consulta.
Se tendrá al cliente informado sobre el estado de su reclamo vía web	Cada responsable cambiara de estado al reclamo ingresado y el cliente se mantendrá informado vía web sobre el estado del reclamo.

Tabla 3 - 8 Resumen de capacidades (Artefacto de Visión)

3.1.1.4.3 Costos y precios

	COSTO ACTUAL
RECURSOS HUMANOS	1500
RECURSOS MATERIALES	
Hojas	15
Material de escritorio	20
Copias	30
Empastados y anillados	100
Tóner para impresión	80
RECURSOS TECNOLÓGICOS	
Servicio de internet	200
Computador Servidor de base de datos	12000
Computador Servidor de Aplicaciones	12000
Computador de desarrollo	1300
Software de base de datos	0
Software sistema operativo para BDD y APP	0
Software para desarrollo	0
SUBTOTAL	27245
5% imprevistos	1362,25
TOTAL	28607,25

Tabla 3 - 9 Costos y precios (Artefacto de Visión)

3.1.1.4.4 Licenciamiento e instalación

Los usuarios del SAR, no necesitaran de licencia alguna, para disfrutar de las bondades que este ofrece. Para la instalación no se requiere del personal del Centro de Cómputo de la empresa, ya que se accede a este vía web.

3.1.1.5 Características del producto

3.1.1.5.1 Facilidad de acceso y uso

El sistema SAR será desarrollado vía web, lo que permitirá a los responsables y clientes un fácil acceso y uso.

3.1.1.5.2 Unificación de la información

Unos de los principales objetivos del SAR es registrar todos los reclamos que están definidos en la regulación No. “012/08” emitida por el CONELEC.

3.1.1.5.3 Mejor control y validación de la información

Los responsables del manejo de reclamos contarán con facilidades para la verificación de la información de cada reclamo ingresado.

3.1.1.6 Rangos de calidad

El desarrollo del sistema SAR se elaborará siguiendo la Metodología de Desarrollo de Software RUP, contemplando los parámetros de calidad que la metodología define.

3.1.1.7 Otros requerimientos del producto

Herramientas a utilizar para la elaboración del SAR:

- La interfaz de usuario estará diseñada en la herramienta “Eclipse Galileo Versión 3.5”, con el plug-in de ZK Studio.
- Frameworks a ser usados en la aplicación: ZK e Hibernate.
- El Servidor de Base de Datos será Oracle versión 11g.
- El Servidor de Aplicaciones a ser usado es Apache Tomcat 6.x.
- Para la elaboración de los Diagramas de los Artefactos de la metodología RUP se usará la herramienta Oracle JDeveloper versión 10135

- Para la elaboración de los formularios establecidos por el CONELEC se usara iReport 3.x
- Máquina Virtual de Java – 1.6.x
- Sistemas Operativos Windows/Linux.

3.1.2 Lista de Riesgos

La calificación de los riesgos presentados a continuación es de 1 al 10.

Ranking	Descripción del Riesgo e Impacto	Estrategia de reducción del riesgo
9	El área comercial y de distribución, no poseen procesos bien definidos para la atención de reclamos.	Coordinar con dichas áreas para establecer correctamente los procesos de acuerdo a la regulación No. “012/08” emitida por el CONELEC.
8	La mayoría del personal de la empresa que se encuentra a cargo del manejo de reclamos tiene poco conocimiento de herramientas basadas en internet.	Diseñar talleres de capacitación sobre el uso de las herramientas básicas basadas en la web.
7	El Centro de Cómputo de la empresa no cuenta con el hardware suficiente, para poner en producción el SAR.	Coordinar con el Director del Centro de Computo, para que dicho director solicite que se realice la compra del hardware necesario para poner en marcha el SAR.
5	El CONELEC aún no tiene definido el formato de envío de reportes del SAR desde las eléctricas hacia su entidad.	Plantear un diseño de base de datos eficiente, para satisfacer los requerimientos solicitados por el ente de regulación “CONELEC” en cuanto al SAR se refiere.
2	Para la capacitación a los	Establecer procesos alternos de

	empleados de la empresa sobre el manejo del SAR de manera personal por parte del Centro de Cómputo, existen inconvenientes por la lejanía de las agencias que se encuentran en el área de concesión de la empresa.	capacitación a los usuarios del sistema que pueden ser: mediante software de acceso remoto, talleres en el auditorio de la empresa.

Tabla 3 - 10 Lista de Riesgos

3.1.3 Plan de Desarrollo de Software

3.1.3.1 Introducción

3.1.3.1.1 Propósito

El propósito de este documento es proporcionar la información necesaria para controlar el proyecto y proveer una visión global del enfoque de desarrollo propuesto para el Sistema de Atención de Reclamos SAR. Este proyecto que ha sido basado en una configuración de la metodología Rational Unified Process (RUP), de acuerdo a las características y necesidades encontradas. En este artefacto de RUP se muestra los roles de los participantes, las actividades a realizar y los artefactos que serán generados.

3.1.3.1.2 Alcance

Este documento describe el plan general a ser usado para el desarrollo del proyecto SAR en Emelnorte S.A. El detalle de las iteraciones individuales se describe en los planes de cada iteración, documentos que se aportan posteriormente en forma separada. Estos planes se elaboran en base a los requerimientos definidos en el artefacto de “Visión”, requerimientos capturados en base a las necesidades expresadas por el stakeholder.

3.1.3.1.3 Resumen

Después de esta introducción, el resto del documento está organizado en las siguientes secciones:

Vista General del Proyecto; proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los artefactos que serán producidos y utilizados durante el proyecto.

Organización del Proyecto; describe la estructura organizacional del equipo de desarrollo.
Gestión del Proceso; explica los costos y planificación estimada, define las fases e hitos del proyecto y describe cómo se realizará su seguimiento.
Planes y Guías de aplicación; proporciona una vista global del proceso de desarrollo de software, incluyendo métodos, herramientas y técnicas que serán utilizadas.

3.1.3.2 Vista General del Proyecto

3.1.3.2.1 Propósito, Alcance y Objetivos

El proyecto SAR contempla el análisis, desarrollo e implantación del sistema de atención de reclamos, de forma tal que se cumplan con los requerimientos definidos por los stakeholders, los mismos que se detallan en los artefactos de casos de uso, información que no forma parte de este artefacto proporcionando de esta manera información necesaria para la toma de decisiones.
El detalle de cómo el sistema SAR cubrirá las necesidades de los usuarios se especifica en los casos de uso, que son información adicional no especificada en este artefacto.

3.1.3.2.2 Suposiciones y Restricciones

Como resultado de las entrevistas con los stakeholders podemos citar las siguientes restricciones:

- Debe diseñarse una arquitectura abierta que permita la adaptación de módulos y prestaciones adicionales a futuro.
- El diseño del sistema del SAR debe realizarse en el menor tiempo posible, de manera que pueda ser utilizado en este año.
- Debe ser flexible de tal manera que permita realizar cambios en el menor tiempo posible.

Como es natural, la lista de suposiciones y restricciones se incrementará durante el desarrollo del proyecto, particularmente una vez establecido el artefacto “Visión”.

3.1.3.2.3 Entregables del Proyecto

Como se explicó anteriormente el proyecto SAR será desarrollado usando la metodología RUP, la misma que genera como resultados los “artefactos” que constituyen los entregables de cada fase.

Estos artefactos son modificados a lo largo del proceso de desarrollo, de manera que al finalizar el proyecto se obtendrá una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un nivel aceptable de estabilidad de los artefactos. Esto será indicado más adelante cuando se presenten los objetivos de cada iteración.

A continuación se presenta la lista de artefactos propuesta para el proyecto SAR:

1) Plan de Desarrollo del Software

Es el presente documento.

2) Visión

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

3) Lista de Riesgos

Este documento presenta una lista de los riesgos conocidos que implica el desarrollo del proyecto, escritos en orden de importancia y asociados con sus respectivas acciones de mitigación o contingencia.

4) Glosario

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología común.

5) Modelo de Casos de Uso

El modelo de Casos de Uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso.

6) Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

7) Modelo de Datos

Previendo que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se

utiliza un Diagrama de Clases (donde se utiliza un perfil UML para Modelado de Datos, para conseguir la representación de tablas, claves, etc.).

8) Modelo de Implementación

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema. (Este modelo es sólo una versión preliminar al final de la fase de Elaboración, posteriormente tiene bastante refinamiento).

9) Manual de Instalación

Este documento incluye las instrucciones para realizar la instalación del producto.

10) Material de Apoyo al Usuario Final

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: Guías del Usuario, Guías de Operación, Guías de Mantenimiento

11) Producto

Los archivos del producto SAR estarán empaquetados y almacenados en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la primera iteración de la fase de Construcción es desarrollado incremental e iterativamente, obteniéndose una nuevo release al final de cada iteración.

3.1.3.2.4 Evolución del Plan de Desarrollo de Software

El Plan de Desarrollo del Software se revisará semanalmente y se refinará antes del comienzo de cada iteración.

3.1.3.3 Organización del Proyecto

3.1.3.3.1 Participantes en el Proyecto

De momento no se incluye el personal que designará Responsable del Proyecto, Comité de Control y Seguimiento, otros participantes que se estimen convenientes para proporcionar los requisitos y validar el sistema.

El resto del personal del proyecto considerando las fases de Inicio, Elaboración y dos iteraciones de la fase de Construcción, estará formado por los siguientes puestos de trabajo y personal asociado:

Jefe de Proyecto. Con una experiencia en metodologías de desarrollo, herramientas CASE y notaciones, en particular la notación UML y el proceso de desarrollo RUP.

Analista de Sistemas. Informático con conocimientos de UML, uno de ellos al menos con experiencia en sistemas afines a la línea del proyecto.
Programadores. Con experiencia en el entorno de desarrollo del proyecto, con el fin de que los prototipos puedan ser lo más cercanos posibles al producto final.
Ingeniero de Software. Persona que participará realizando labores de gestión de requisitos, gestión de configuración, documentación y diseño de datos. Encargada de las pruebas funcionales del sistema, realizará la labor de Tester.

3.1.3.3.2 Interfaces Externas

Se define los participantes del proyecto que proporcionarán los requisitos del sistema, y entre ellos quiénes serán los encargados de evaluar los artefactos de acuerdo a cada módulo y según el plan establecido.
El equipo de desarrollo interactuará activamente con los participantes para especificación y validación de los artefactos generados.

3.1.3.3.3 Roles y Responsabilidades

A continuación se describen las principales responsabilidades de cada uno de los puestos en el equipo de desarrollo durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP.

Puesto	Responsabilidad
Jefe de Proyecto	El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto.
Analista de Sistemas	Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.

Programador	Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario
Ingeniero de Software	Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.

Tabla 3 - 11 Roles y Responsabilidades (Artefacto Plan de Desarrollo de Software)

3.1.3.4 Gestión del Proceso

3.1.3.4.1 Plan de Proyecto

En esta sección se presenta la organización en fases e iteraciones y el calendario del proyecto.

3.1.3.4.1.1 Plan de las Fases

El desarrollo se efectuará en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra una la distribución de tiempos y el número de iteraciones de cada fase (para las fases de Construcción y Transición es sólo una aproximación muy preliminar)

Fase	**Nro. Iteraciones**	**Duración**
Fase de Inicio	1	9 semanas
Fase de Elaboración	1	8 semanas
Fase de Construcción	1	9 semanas
Fase de Transición	1	4 semanas

Tabla 3 - 12 Plan de las Fases (Artefacto Plan de Desarrollo de Software)

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Descripción	Hito
Fase de Inicio	En esta fase desarrollará los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente / usuario del artefacto Visión y el Plan de Desarrollo marcan el final de esta fase.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y / o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera release de la fase de Construcción deben estar analizados y diseñados (en el Modelo de Análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En nuestro caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de una semana.
Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis / Diseño. El producto se construye en base a 2 iteraciones, cada una produciendo una release a la cual se le aplican las pruebas y se valida con el cliente /

	usuario. Se comienza la elaboración de material de apoyo al usuario. El hito que marca el fin de esta fase es la versión de la release 3.0, con la capacidad operacional parcial del producto que se haya considerado como crítica, lista para ser entregada a los usuarios para pruebas beta.
Fase de Transición	En esta fase se prepararán dos releases para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

Tabla 3 - 13 Plan de las Fases- Descripción (Artefacto Plan de Desarrollo de Software)

3.1.3.4.1.1.1 Objetivo de las Iteraciones

Fase	Iteración	Descripción	Hitos Asociados	Riesgos dirigidos
Incepción	Primera Iteración	Definir el modelo de Negocio, los requerimientos del producto, el plan del proyecto.	Revisión de Casos de Negocio	Aclarar las necesidades de los usuarios con anticipación. Elaborar planes de proyectos realistas y alcanzables.
Elaboración	Primera Iteración Desarrollo del Prototipo de	Completar el análisis y desarrollo para todos los casos de uso. Desarrollar el	Prototipo de la Arquitectura	Aclarar los hitos de la arquitectura. Disminuir los riesgos técnicos.

	la Arquitectura	prototipo de la arquitectura		Primeros Prototipos de revisión para el usuario.
Construcción	Iteración C1 – Desarrollo Beta	Implementar y probar los casos de uso para proveer la versión Beta.	Beta	Todos los requerimientos claves de los usuarios y la arquitectura propuesta. Implementado en la versión Beta. Retroalimentación del usuario antes del release del software.
	Iteración C2 Desarrollo del Release inicial	Implementar y probar los casos de uso restantes Corregir los defectos de la versión Beta e incorporar la retroalimentación con esta versión. Desarrollar el sistema inicial.	Software	Revisión completa del software por los usuarios. La calidad del producto debe ser alta. Minimizar defectos. Costo de calidad reducida.
	Iteración C3 Desarrolla el Release completo.	Incorpora las mejoras y defectos del release inicial. Desarrolla el	Software	El sistema provee todas las funcionalidades claves requeridas por el usuario.

		sistema completo.		
Transición	Release del Software	Empaquetar, distribuir e instalar el Release.	Software Released	

Tabla 3 - 14 Objetivo de las Iteraciones (Artefacto Plan de Desarrollo de Software)

3.1.3.4.1.2 Calendario del Proyecto

A continuación se presenta un calendario de las principales tareas del proyecto incluyendo sólo las fases de Inicio y Elaboración. Como se ha comentado, el proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto. La siguiente figura ilustra este enfoque, en ella lo ensombrecido marca el énfasis de cada disciplina (workflow) en un momento determinado del desarrollo.

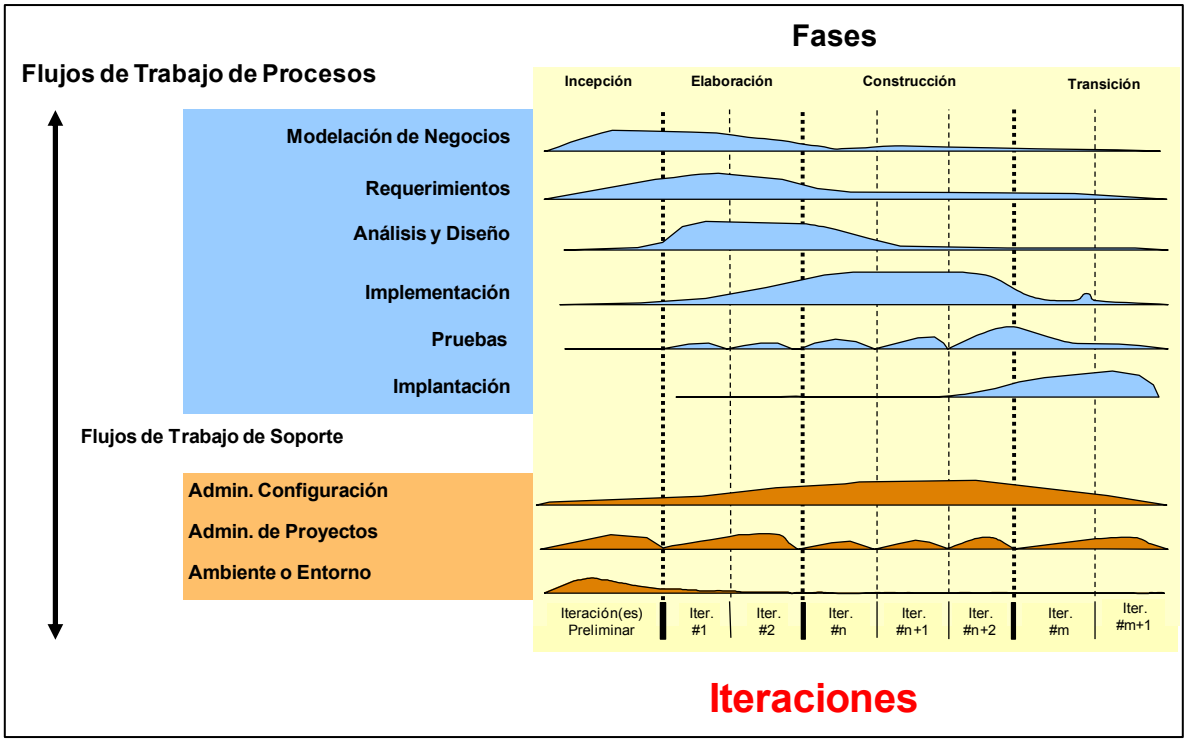

Figura 3 - 2 Fases e Iteraciones (Artefacto Plan de Desarrollo de Software)

Para este proyecto se ha establecido el siguiente calendario:

ACTIVIDADES	INICIO	FINALIZACION
Análisis del sistema.		
Investigación de los reclamos establecidos por CONELEC. Recolección de requerimientos.	Semana 1	Semana 1
Requisitos		
Visión	Semana 2	Semana 2
Modelo de Casos de Uso	Semana 2	Semana 3
Especificación de Casos de Uso	Semana 3	Semana 5
Análisis / Diseño		
Modelo de Análisis / Diseño	Semana 5	Semana 7

Modelo de Datos	Semana 5	Semana 7
Prototipos de Interfaces de Usuario	Semana 7	Semana 9
Implementación		
Prototipos de Interfaces de Usuario	Semana 9	Semana 1
Modelo de Implementación	Semana 11	Semana 13
Pruebas		
Casos de Pruebas Funcionales	Semana 13	Semana 15
Casos de Pruebas Funcionales(Documentadas)	Semana 15	Semana 15
Despliegue		
Modelo de Despliegue	Semana 15	Semana 16
Gestión de Cambios y Configuración	Durante todo el proyecto	

Tabla 3 - 15 Calendario del Proyecto (Artefacto Plan de Desarrollo de Software)

3.1.3.4.2 Seguimiento y Control del Proyecto

- **Gestión de Requisitos**

Los requisitos del sistema son especificados en el artefacto Visión. Cada requisito tendrá una serie de atributos tales como importancia, estado, iteración donde se implementa, etc. Estos atributos permitirán realizar un efectivo seguimiento de cada requisito. Los cambios en los requisitos serán gestionados mediante una Solicitud de Cambio, las cuales serán evaluadas y distribuidas para asegurar la integridad del sistema y el correcto proceso de gestión de configuración y cambios.

- **Control de Plazos**

El calendario del proyecto tendrá un seguimiento y evaluación semanal por el jefe de proyecto.

- **Control de Calidad**

Los defectos detectados en las revisiones y formalizados también en una Solicitud de Cambio tendrán un seguimiento para asegurar la conformidad respecto de la solución de dichas deficiencias. Para la revisión de cada artefacto y su correspondiente garantía de calidad se utilizarán las guías de revisión y checklist (listas de verificación) incluidas en RUP.

- **Gestión de Riesgos**

A partir de la fase de Inicio se mantendrá una lista de riesgos asociados al proyecto y de las acciones establecidas como estrategia para mitigarlos o acciones de contingencia. Esta lista será evaluada al menos una vez en cada iteración.

- **Gestión de Configuración**

Se realizará una gestión de configuración para llevar un registro de los artefactos generados y sus versiones. También se incluirá la gestión de las Solicitudes de Cambio y de las modificaciones que éstas produzcan, informando y publicando dichos cambios para que sean accesibles a todo los participantes en el proyecto. Al final de cada iteración se establecerá un registro del estado de cada artefacto, estableciendo una versión, la cual podrá ser modificada sólo por una Solicitud de Cambio aprobada.

3.2 FASE DE ELABORACIÓN

3.2.1 Diagrama de Casos de Uso

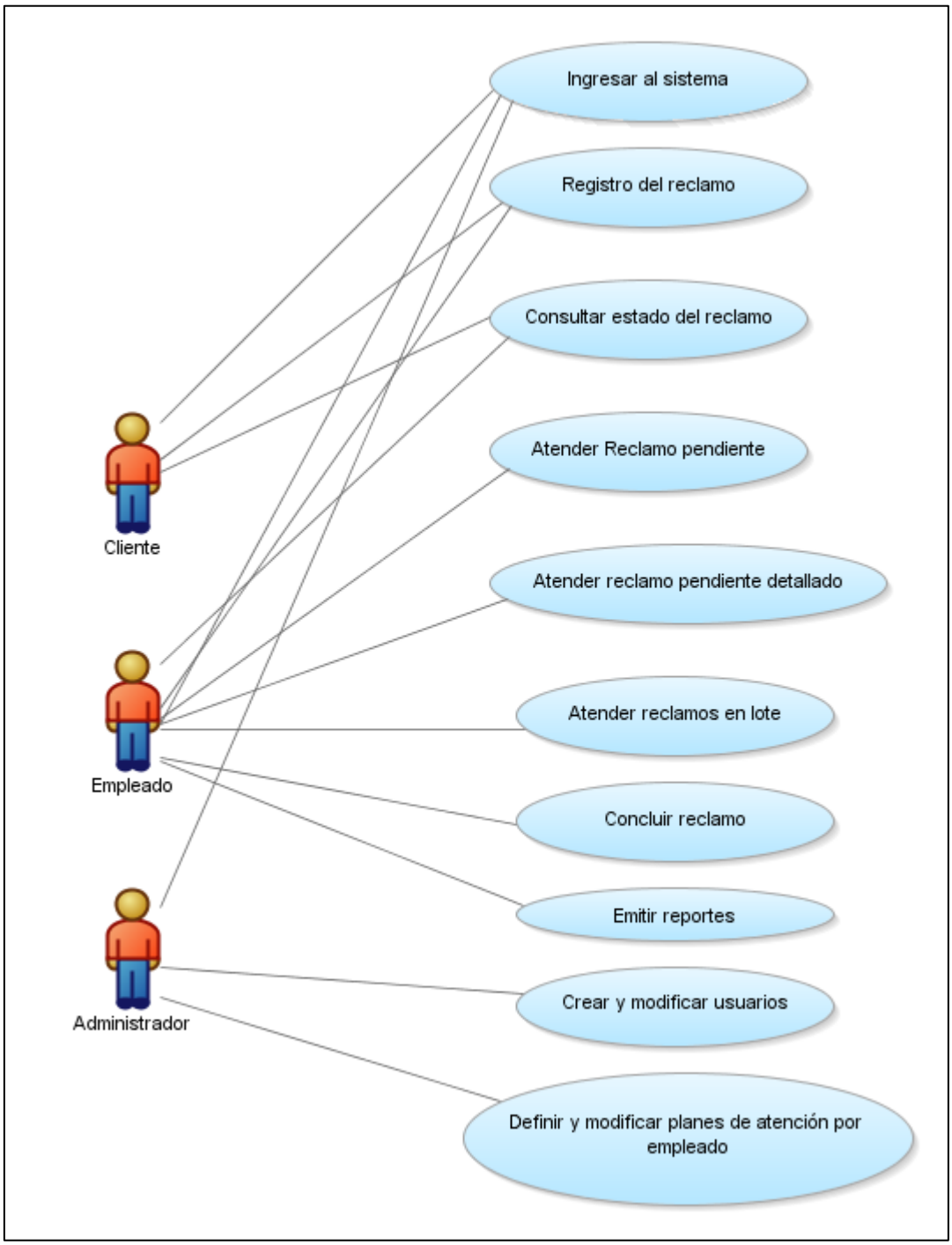

Figura 3 - 3 Diagrama de casos de uso

3.2.2 Especificación de Casos de Uso

3.2.2.1 Ingresar al sistema por parte del cliente.

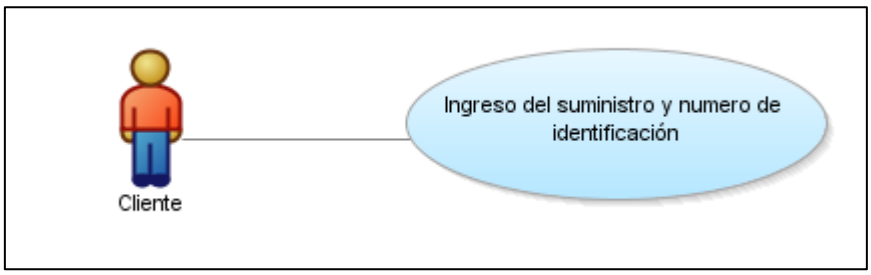

Figura 3 - 4 Ingresar al sistema por parte del cliente (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso de ingresar al sistema por parte del cliente de la empresa vía web.

- **Flujo Básico de Eventos**
  - El cliente de la empresa ingresara al siguiente link http://www.emelnorte.com , luego selecciona el enlace “Atención de Reclamos”
  - Se le presentará la pantalla de ingreso al sistema.
  - Colocará el número de suministro que se encuentra en la factura y el número de identificación (Cédula, Ruc, Pasaporte).
  - Una vez llenos estos campos presiona en “Ingresar”.
  - Se procede a verificar los datos en la tabla SUSCRIPTOR, si está correcta la información procede a ingresar el sistema.
- **Flujos Alternativos**
  - Se puede buscar el número de suministro por apellidos y nombres.
- **Precondiciones**
  - Debe ser cliente de la empresa.
  - No debe tener más de dos planillas impagas.

3.2.2.2 Registro del reclamo por parte del cliente.

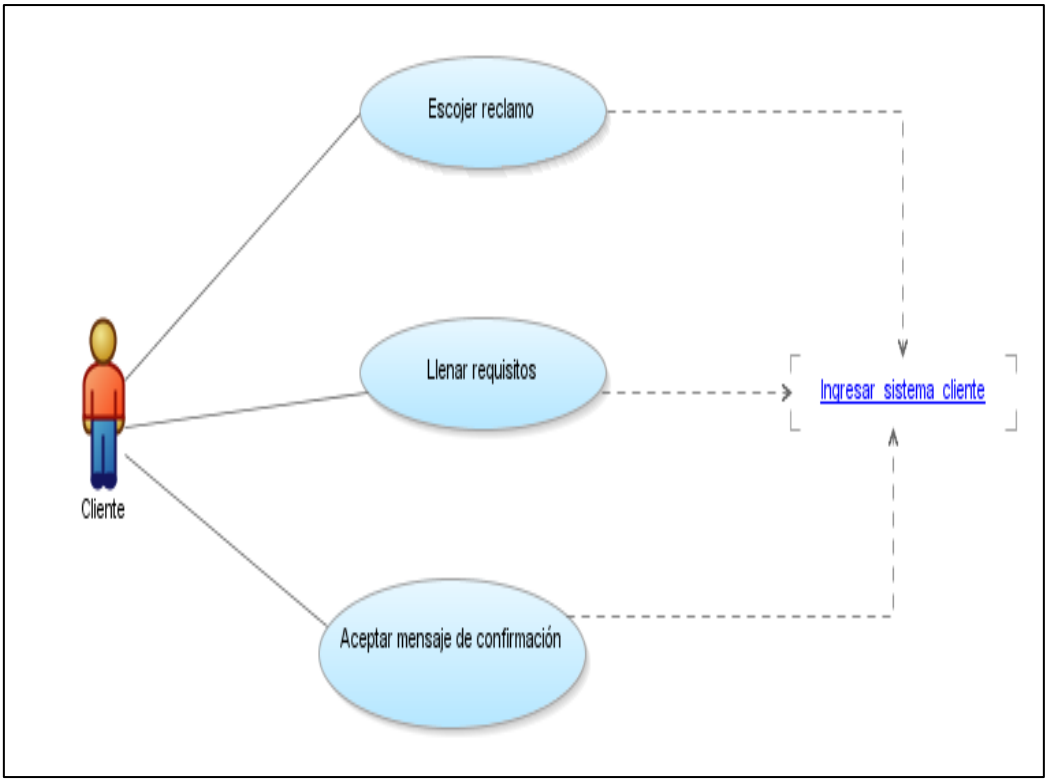

Figura 3 - 5 Registro del reclamo por parte del cliente (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para registrar reclamos por parte del cliente de la empresa vía web de acuerdo a lo que establece la regulación del CONELEC No. “012/08”.

- **Flujo Básico de Eventos**
  - El cliente de la empresa una vez dentro del sistema, seleccionará la opción de “Ingreso del Reclamo”.
  - Se le presentará los reclamos definidos por la regulación del CONELEC No. “012/08”.
  - Una vez que escoja el reclamo, deberá llenar los campos correspondientes a cada reclamo, presionará en el botón “Ingresar”, se desplegará un mensaje de confirmación.
  - Si está de acuerdo se procede a almacenar los datos en las tablas: RECLAMOS_SUSCRIPTOR, HISTORICO_RECLAMOS_SUSCRIPTOR.
- **Flujos Alternativos**
  - Se puede buscar el número de suministro por apellidos y nombres.

- **Precondiciones**
  - Debe ser cliente de la empresa.
  - No debe tener más de dos planillas impagas.
  - Se necesita tener un responsable asignado al plan que tiene el cliente.
- **Post condiciones**
  - Si el cliente ha ingresado un correo electrónico le llegaran notificaciones del estado de su reclamo.

3.2.2.3 Consultar estado del reclamo por parte del cliente.

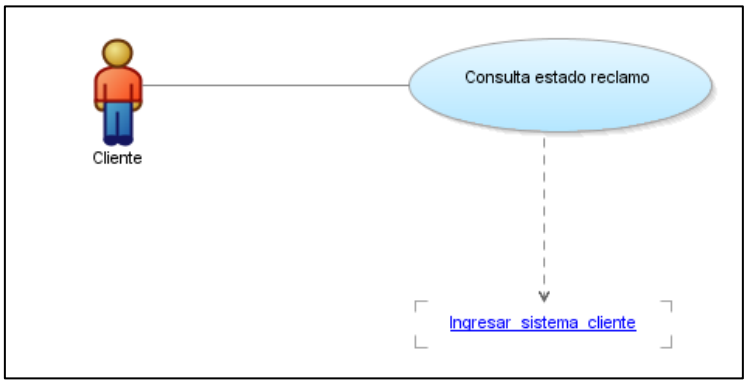

Figura 3 - 6 Consultar estado del reclamo por parte del cliente (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para consultar el estado de un reclamo por parte del cliente de la empresa vía web.

- **Flujo Básico de Eventos**
  - El cliente de la empresa una vez dentro del sistema, seleccionará la opción de "Estado de sus Reclamos".
  - Se le presentara todos los reclamos registrados, indicado el Responsable Actual y el estado en que se encuentran.
- **Flujos Alternativos**
  - Se puede visualizar a mayor detalle la información ingresada del reclamo.
- **Precondiciones**
  - Debe ser cliente de la empresa.
  - No debe tener más de dos planillas impagas.
- **Post condiciones**

- Si el cliente ha ingresado un correo electrónico le llegaran notificaciones del estado de su reclamo.

3.2.2.4 Ingresar al sistema por parte del empleado.

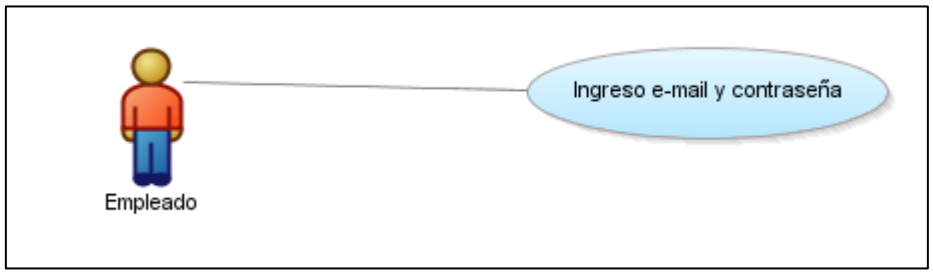

Figura 3 - 7 Ingresar al sistema por parte del empleado (Especificación de Casos De Uso)

- **Descripción**

  Este caso de uso describe el proceso de ingresar al sistema por parte del empleado de la empresa vía web.
- **Flujo Básico de Eventos**
  - El empleado de la empresa ingresara al siguiente link http://www2.emelnorte.com:8889/reclamos_responsables/
  - Se le presentará la pantalla de ingreso al sistema.
  - Colocará su e-mail respectivo y la contraseña asignada.
  - Una vez llenos estos campos presionara en “Ingresar”.
  - Se procede a verificar los datos en la tabla RESPONSABLES, si está correcta la información procede a ingresar al sistema.
- **Flujos Alternativos**
  - Si es el primer ingreso al sistema se despliega una pantalla donde se actualiza la contraseña en la tabla RESPONSABLES.
  - Si olvido su contraseña puede seleccionar el enlace “Ha olvidado su contraseña?”.
- **Precondiciones**
  - Debe ser empleado de la empresa.
  - Debe estar habilitado en el Sistema.
  - Debe encontrarse dentro de la intranet de la empresa.
- **Post condiciones**

- Se cargan las respectivas opciones de acuerdo al rol asignado.

3.2.2.5 Registro del reclamo por parte del empleado.

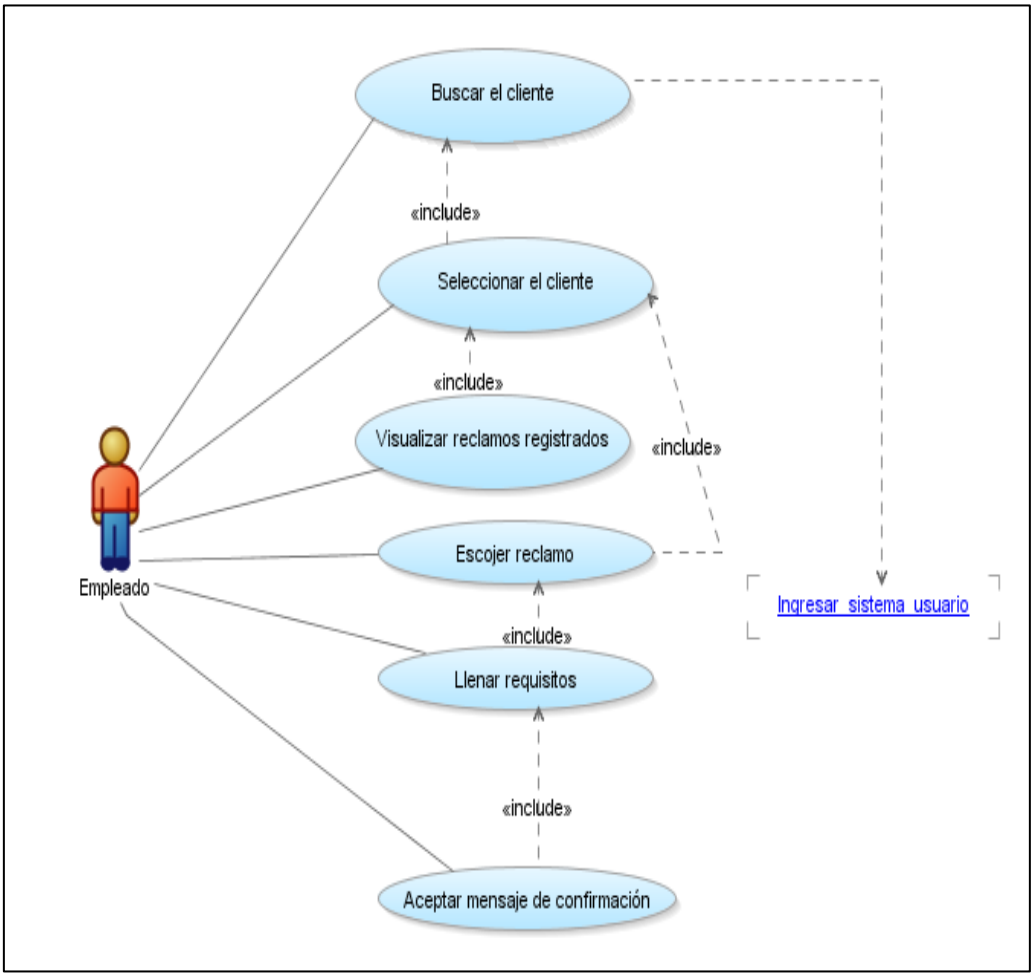

Figura 3 - 8 Registro del reclamo por parte del empleado (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para registrar un reclamo por parte del empleado de la empresa vía web de acuerdo a lo que establece la regulación del CONELEC No. “012/08”.

- **Flujo Básico de Eventos**
  - El empleado de la empresa una vez dentro del sistema, seleccionará la opción “Reclamos/Ingreso” que se encuentra en el menú de “Opciones”.

- Se le presentará la respectiva pantalla de “Ingreso”, para realizar la Búsqueda del cliente por una de las siguientes alteralivas: número de suministro, número de identificación, apellidos y nombres. Después presiona buscar, lo cual verifica la información en la tabla SUSCRIPTOR, posteriormente se desplegaran los posibles clientes que coinciden con el parámetro de búsqueda.
- Seleccionara el cliente deseado, posteriormente se desplegará los reclamos registrados si existieren.
- Se le presentará los reclamos definidos por la regulación del CONELEC No. “012/08”.
- Una vez que escoja el reclamo, deberá llenar los campos correspondientes a cada reclamo, presionará en el botón “Ingresar”, se desplegará un mensaje de confirmación.
- Si está de acuerdo se procede a almacenar los datos en las tablas: RECLAMOS_SUSCRIPTOR, HISTORICO_RECLAMOS_SUSCRIPTOR.
- El sistema asignará automáticamente un responsable para dicho reclamo.

- **Flujos Alternativos**
  - Se puede visualizar el detalle de los reclamos registrados.
- **Precondiciones**
  - Debe ser trabajador de la empresa y estar activo en el sistema.
  - Se necesita tener un responsable asignado al plan que tiene el cliente.
- **Post condiciones**
  - Si se ha ingresado el e-mail del cliente le llegaran notificaciones del estado de su reclamo al mencionado cliente.
  - Se le notificará vía mail al responsable de la empresa indicándole que tiene que atender dicho reclamo.

3.2.2.6 Atender reclamo pendiente

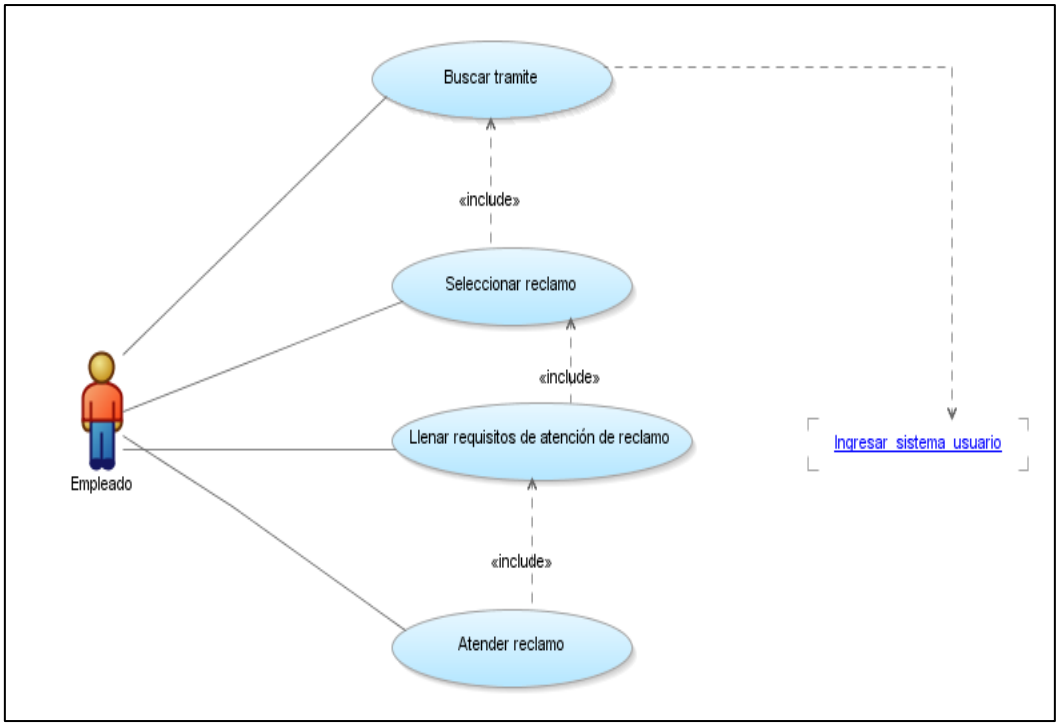

Figura 3 - 9 Atender reclamo pendiente (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para atender un reclamo pendiente por parte del empleado de la empresa vía web.

- **Flujo Básico de Eventos**
    - El empleado de la empresa una vez dentro del sistema, seleccionará la opción "Reclamos/Pendientes" que se encuentra en el menú de "Opciones".
    - Se le presentará la respectiva pantalla de todos los reclamos pendientes paginados de 10 en 10.
    - Seleccionará el tramite deseado, posteriormente se desplegará la información detallada del reclamo.
    - Llenara los requisitos solicitados, de acuerdo a cada tipo de reclamo.
    - Una vez que desee atender el reclamo, se desplegará un mensaje de confirmación indicando si desea atender el reclamo.
    - Si está de acuerdo se procede a almacenar los datos en la tabla: HISTORICO_RECLAMOS_SUSCRIPTOR.
    - El reclamo atendido ya no se encuentra en los pendientes del empleado.
- **Flujos Alternativos**

- Se puede seguir atendiendo reclamos pendientes en el caso de existir.

- **Precondiciones**
  - Debe ser trabajador de la empresa y estar activo en el sistema.
- **Post condiciones**
  - Si al momento de registrar el reclamo del cliente, se guardó el e-mail de este, se procede a notificar vía mail sobre el nuevo estado del reclamo.
  - Si el reclamo posee un estado diferente a “FINALIZADO”, se le notificara al nuevo responsable vía mail, para que proceda a atender el reclamo.

3.2.2.7 Atender reclamo pendiente detallado

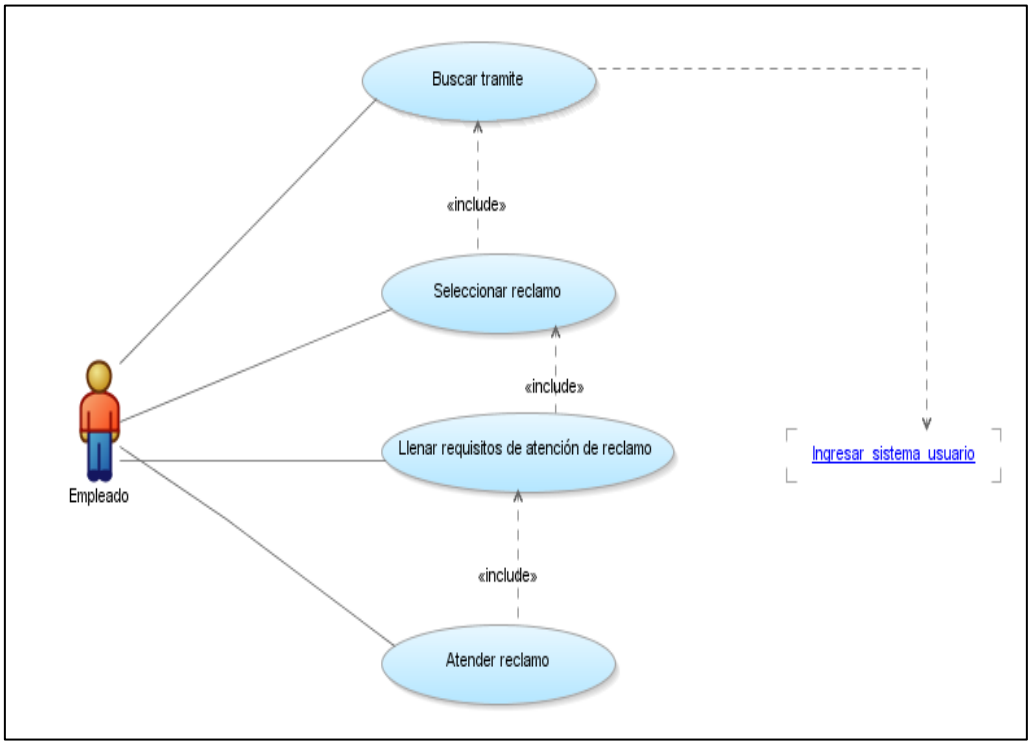

Figura 3 - 10 Atender reclamo pendiente detallado (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para atender un reclamo pendiente detallado por parte del empleado de la empresa vía web.

- **Flujo Básico de Eventos**

- El empleado de la empresa una vez dentro del sistema, seleccionará la opción “Reclamos/Pendientes Detallado” que se encuentra en el menú de “Opciones”.
- Se le presentará la respectiva pantalla para buscar un reclamo por número de trámite.
- Colocará el número de trámite y presionara “Buscar”, se desplegaran los reclamos que coincidan con el parámetro de búsqueda.
- Seleccionará el tramite deseado, posteriormente se desplegará la información detallada del reclamo.
- Llenara los requisitos solicitados, de acuerdo a cada tipo de reclamo.
- Una vez que desee atender el reclamo, se desplegará un mensaje de confirmación indicando si desea atender el reclamo.
- Si está de acuerdo se procede a almacenar los datos en la tabla: HISTORICO_RECLAMOS_SUSCRIPTOR.
- El reclamo atendido ya no se encuentra en los pendientes del empleado.

- **Flujos Alternativos**

  N/A
- **Precondiciones**
  - Debe ser trabajador de la empresa y estar activo en el sistema.
- **Post condiciones**
  - Si al momento de registrar el reclamo del cliente, se guardó el e-mail de este, se procede a notificar vía mail sobre el nuevo estado del reclamo.
  - Si el reclamo posee un estado diferente a “FINALIZADO”, se le notificara al nuevo responsable vía mail, para que proceda a atender el reclamo.

3.2.2.8 Atender varios reclamos pendientes en lote

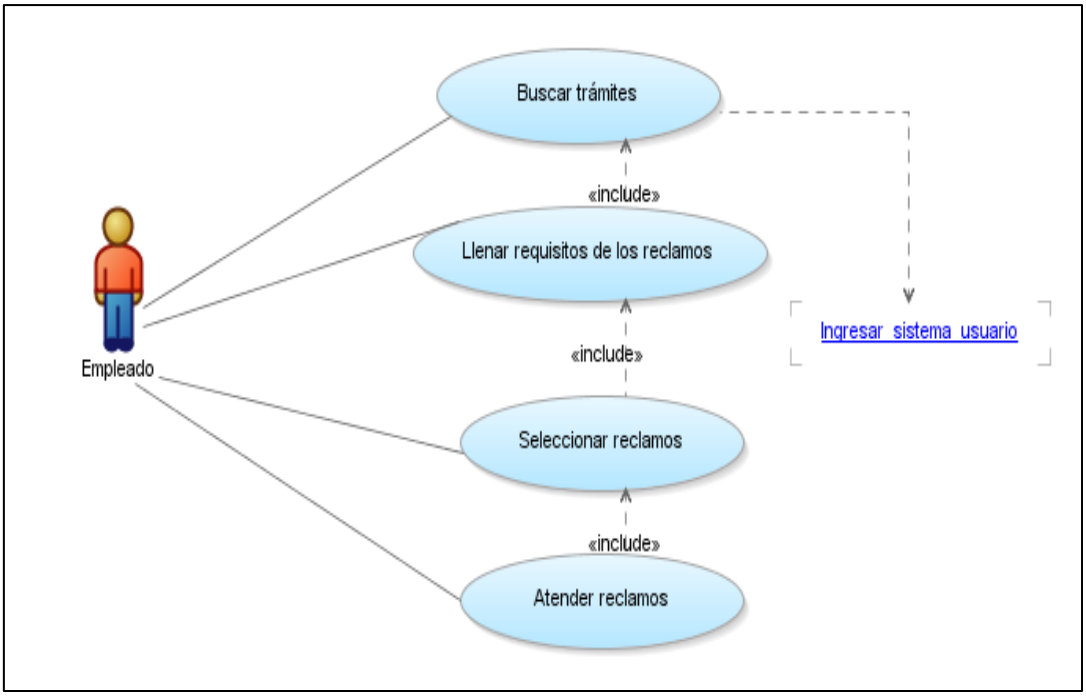

Figura 3 - 11 Atender varios reclamos pendientes en lote (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para atender varios reclamos pendientes en lote por parte del empleado de la empresa vía web.

- **Flujo Básico de Eventos**
    - El empleado de la empresa una vez dentro del sistema, seleccionará la opción "Reclamos/Pendientes en lote" que se encuentra en el menú de "Opciones".
    - Se le presentará la respectiva pantalla de todos los reclamos pendientes paginados de 5 en 5.
    - Llenara los requisitos solicitados de cada reclamo encontrado, de acuerdo a cada tipo de reclamo.
    - Seleccionará los reclamos que desee atender, y se desplegaran los reclamos que se ha seleccionado.
    - Una vez que desee atender los reclamos seleccionados, se desplegará un mensaje de confirmación indicando si desea atender el reclamo.
    - Si está de acuerdo se procede a almacenar los datos en la tabla: HISTORICO_RECLAMOS_SUSCRIPTOR.
    - Los reclamos atendidos ya no se encuentra en los pendientes del empleado.
- **Flujos Alternativos**

N/A

- **Precondiciones**
  - Debe ser trabajador de la empresa y estar activo en el sistema.
- **Post condiciones**
  - Si al momento de registrar el reclamo del cliente, se guardó el e-mail de este, se procede a notificar vía mail sobre el nuevo estado del reclamo.
  - Si el reclamo posee un estado diferente a “FINALIZADO”, se le notificara al nuevo responsable vía mail, para que proceda a atender el reclamo.

3.2.2.9 Concluir reclamo

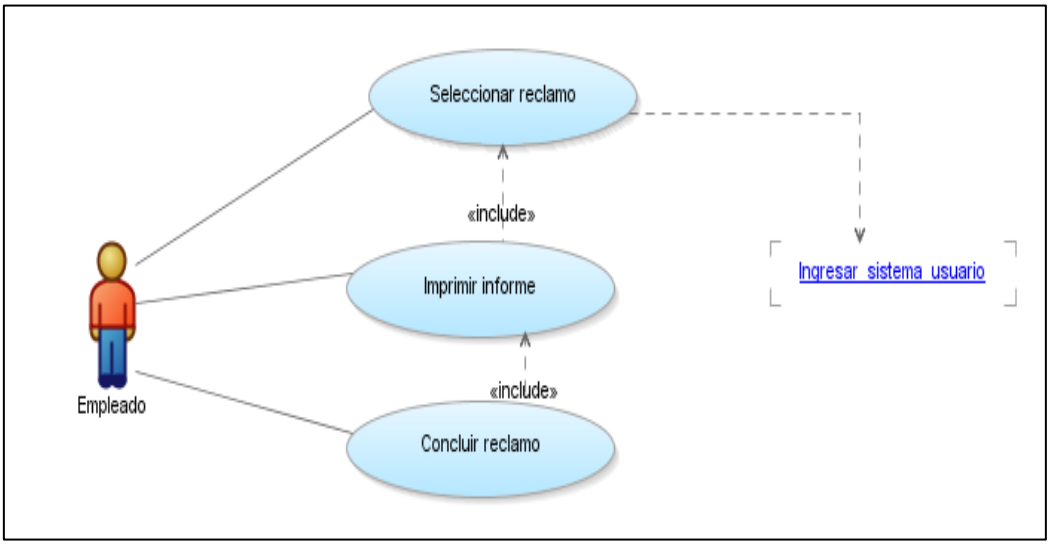

Figura 3 - 12 Concluir reclamo (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso para concluir un reclamo por parte del empleado de la empresa vía web.

- **Flujo Básico de Eventos**
  - El empleado de la empresa una vez dentro del sistema, seleccionará la opción “Reclamos/Finalizados y Concluidos” que se encuentra en el menú de “Opciones”.
  - Se le presentará la respectiva pantalla de todos los reclamos finalizados y concluidos.

- Seleccionara el reclamo que se desea concluir, se desplegara el informe para que el empleado imprima dicho informe.
- Posteriormente se archiva el informe, debidamente firmado por el personal responsable y por el cliente de la empresa, dado este paso se procede a presionar en “Concluir” y se despliega un mensaje de confirmación.
- Si está de acuerdo se procede a almacenar los datos en la tabla: HISTORICO_RECLAMOS_SUSCRIPTOR.
- El reclamo pasa del estado “FINALIZADO” a “CONCLUIDO”.

- **Flujos Alternativos**

  N/A
- **Precondiciones**
  - Debe ser trabajador de la empresa y estar activo en el sistema.
- **Post condiciones**
  - Si al momento de registrar el reclamo del cliente, se guardó el e-mail de este, se procede a notificar vía mail sobre el nuevo estado del reclamo.

3.2.2.10 Crear/Modificar Usuario en el Sistema

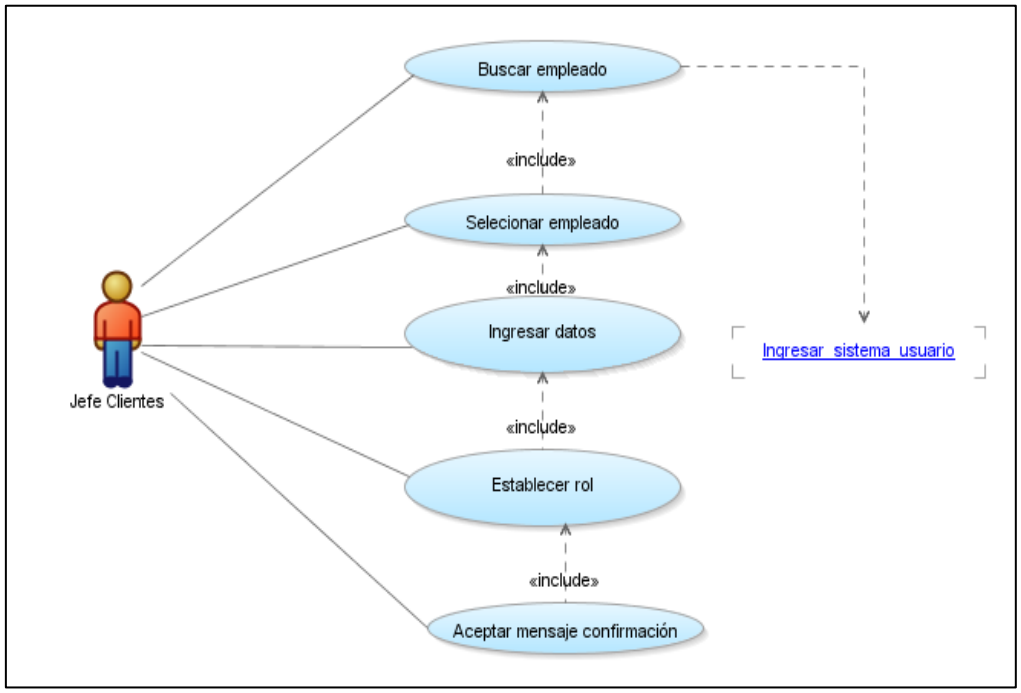

Figura 3 - 13 Crear/Modificar usuario en el sistema (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso de crear usuarios en el sistema por parte del administrador del sistema.

- **Flujo Básico de Eventos**
    - En la pantalla de Creación de Usuarios, el usuario con rol de Administrador, selecciona el código del empleado.
    - Se le presentará los empleados existentes, luego tendrá que escoger el empleado y el rol que va a tener el nuevo usuario del sistema.
    - Se ingresara el email del empleado, luego el sistema enviara al email la contraseña asignada para que este pueda acceder al sistema.
    - Seleccionar la opción guardar, que almacena los datos en las tablas: RESPONSABLES, ROLES_RESPONSABLES.
- **Flujos Alternativos**
    - N/A
- **Precondiciones**
    - El empleado debe existir en nómina para acceder al sistema.
    - Debe existir el requerimiento aprobado por el Jefe respectivo de la empresa.

3.2.2.11 Definir/Modificar planes de atención por empleado

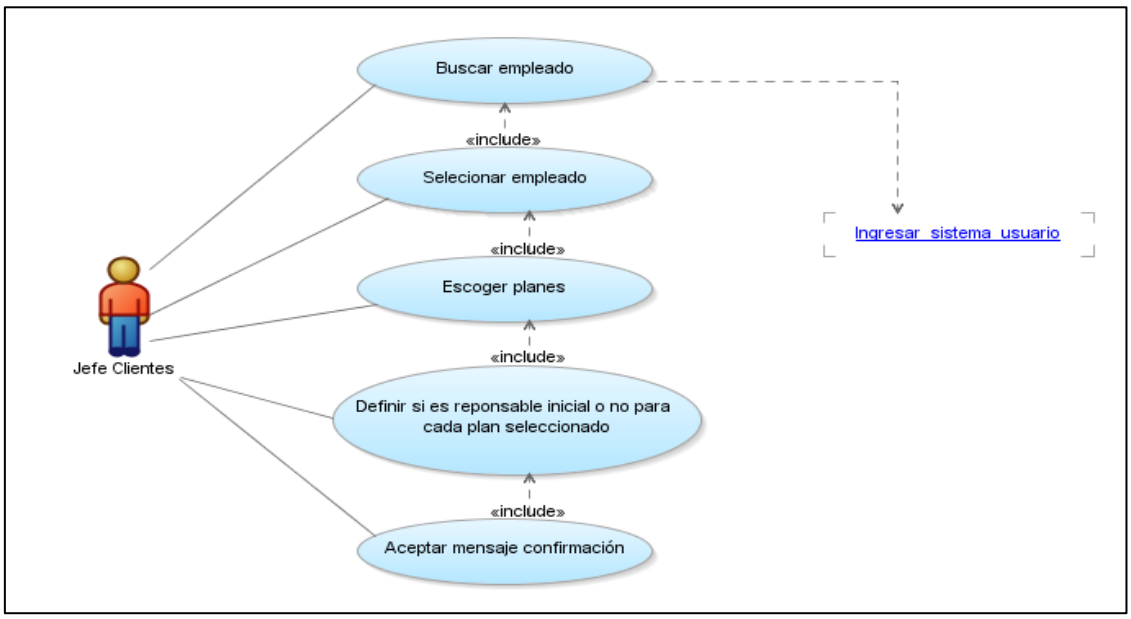

Figura 3 - 14 Definir/Modificar planes de atención por empleado (Especificación de Casos De Uso)

- **Descripción**

Este caso de uso describe el proceso de definir o modificar planes de atención por empleado, para cada plan se establece si el responsable asignado para dicho plan es responsable inicial o no; este proceso lo ejecuta solamente el administrador del sistema.

- **Flujo Básico de Eventos**
    - El empleado de la empresa una vez dentro del sistema, seleccionará la opción “Administrar/Planes por empleado” que se encuentra en el menú de “Opciones”.
    - Se le presentará una pantalla para realizar la búsqueda del empleado por código o por sus apellidos y nombres, luego se procederá a escoger al empleado.
    - Una vez seleccionado el empleado se procede a establecer los respectivos planes que atenderá el empleado.
    - Una vez que desee guardar los planes definidos para el empleado seleccionado, se desplegará un mensaje de confirmación indicando si desea guardar los datos.
    - Si está de acuerdo, se almacena la información en la tabla RESPONSABLES_PLAN_RECLAMOS.
- **Flujos Alternativos**
    - N/A.
- **Precondiciones**
    - El empleado debe existir en nómina para acceder al sistema.
    - Debe existir el requerimiento aprobado por el Jefe respectivo de la empresa.

3.3 FASE DE CONSTRUCCIÓN

3.3.1 Vista Lógica

3.3.1.1 Modelo de Base de Datos

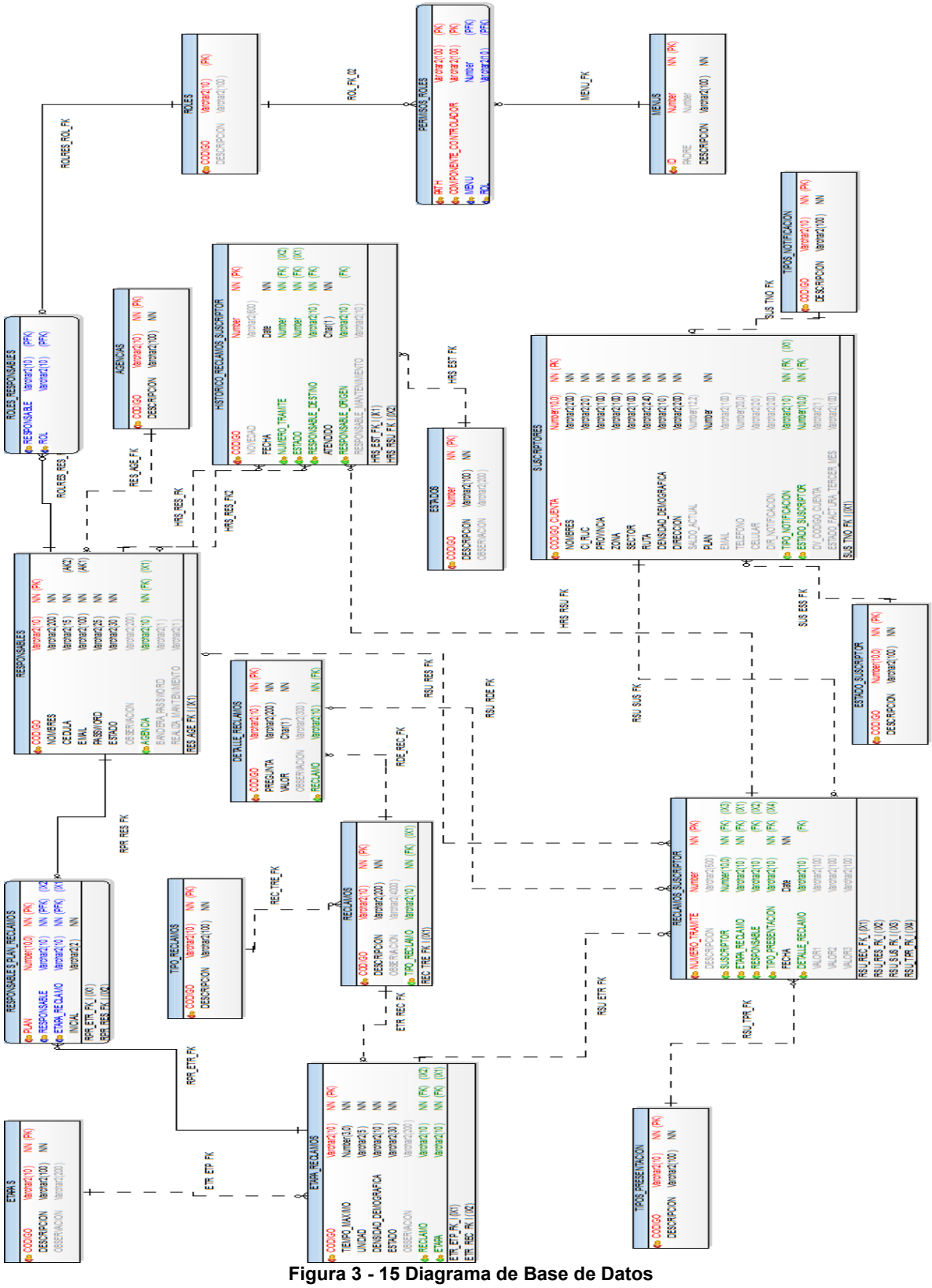

Figura 3 - 15 Diagrama de Base de Datos

3.3.1.2 Diagrama Global de Paquetes

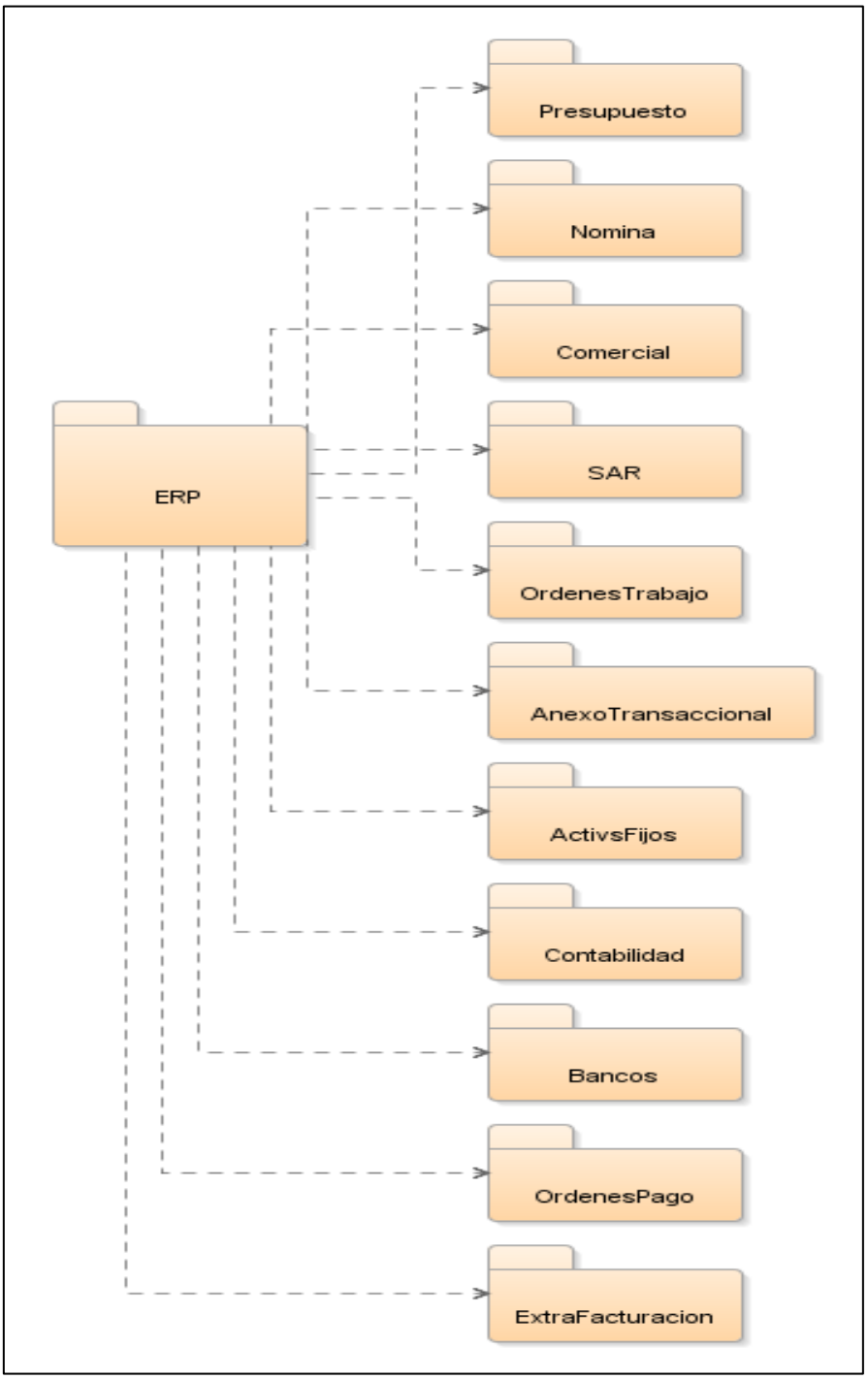

Figura 3 - 16 Diagrama Global de Paquetes

3.3.2 Vista de Implementación

3.3.2.1 Diagramas de Actividades

3.3.2.1.1 Ingresar al sistema por parte del cliente

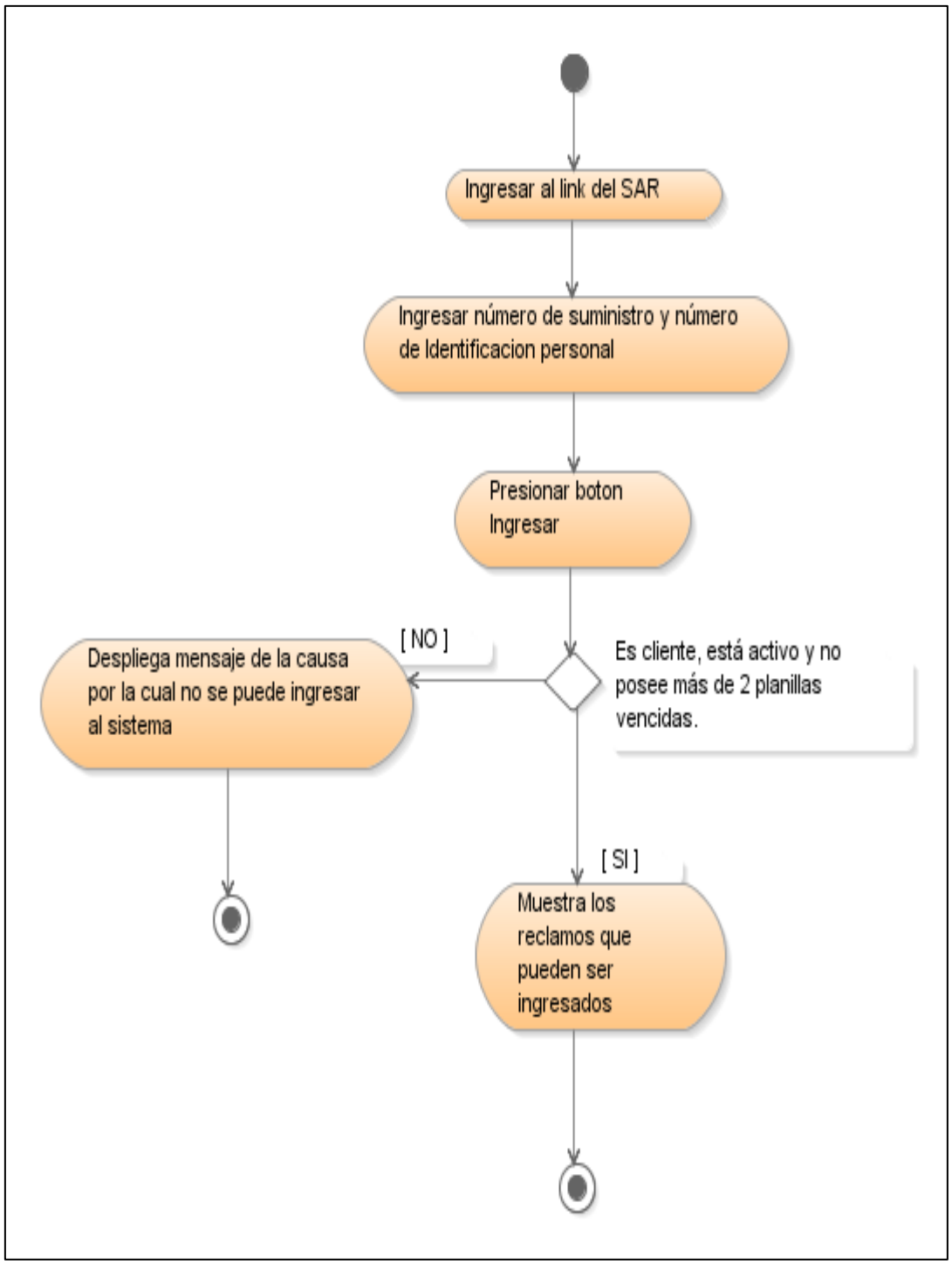

Figura 3 - 17 Ingresar al sistema por Parte del Cliente (Diagrama de Actividades)

3.3.2.1.2 Ingresar al sistema por parte del responsable

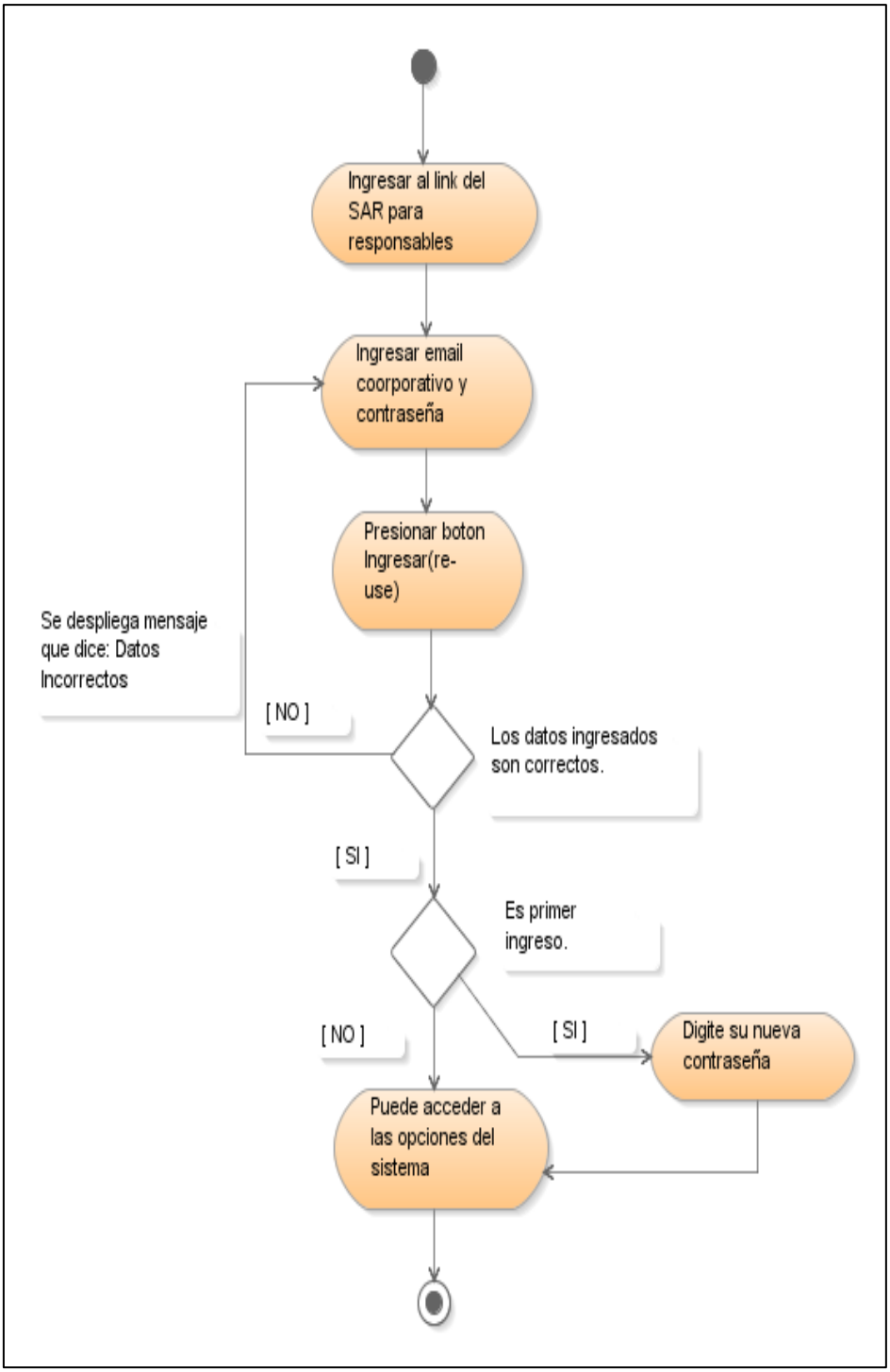

Figura 3 - 18 Ingresar al sistema por Parte del Responsable (Diagrama de Actividades)

3.3.2.1.3 Registrar reclamo

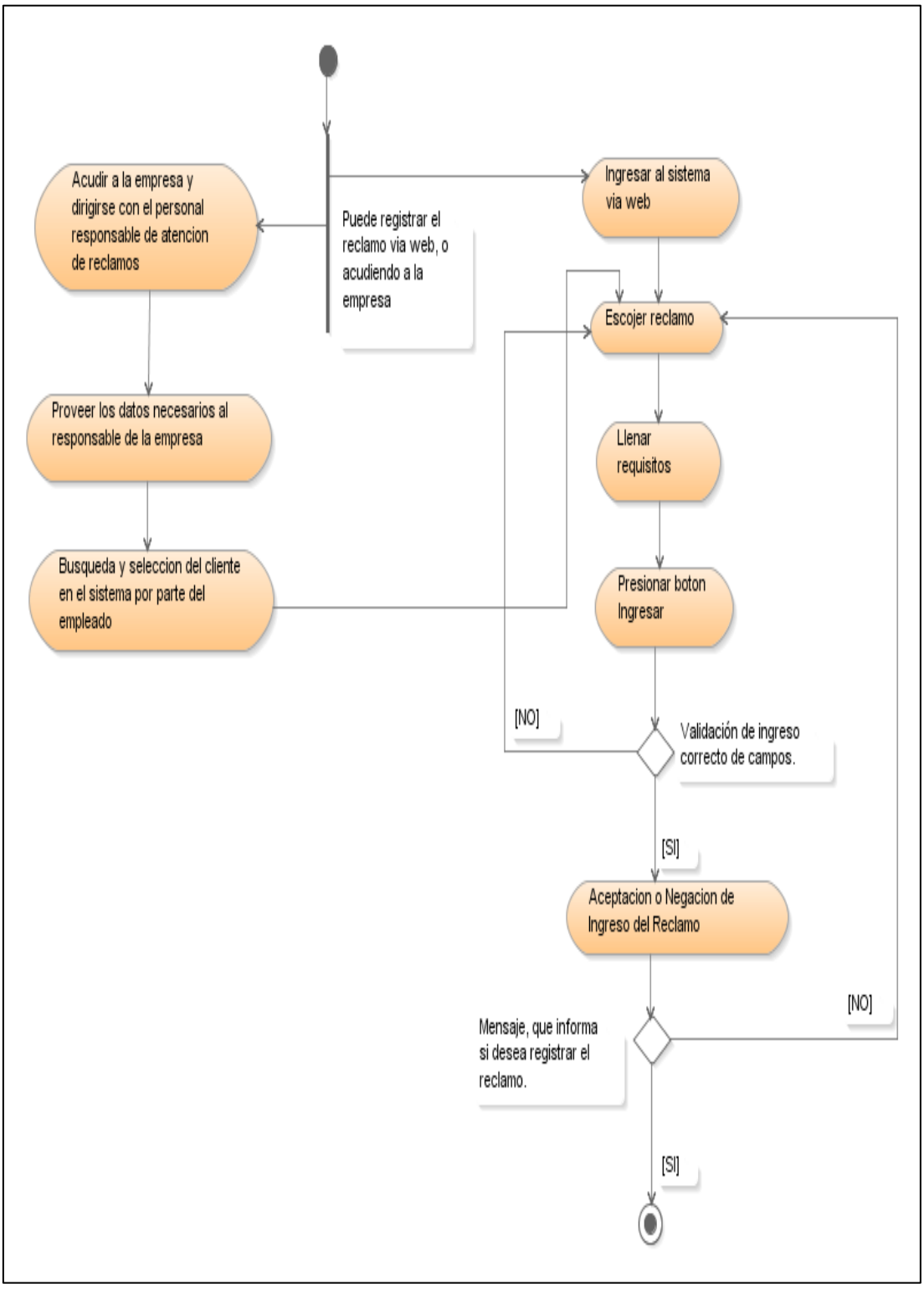

Figura 3 - 19 Registrar reclamo (Diagrama de Actividades)

3.3.2.1.4 Atender reclamo

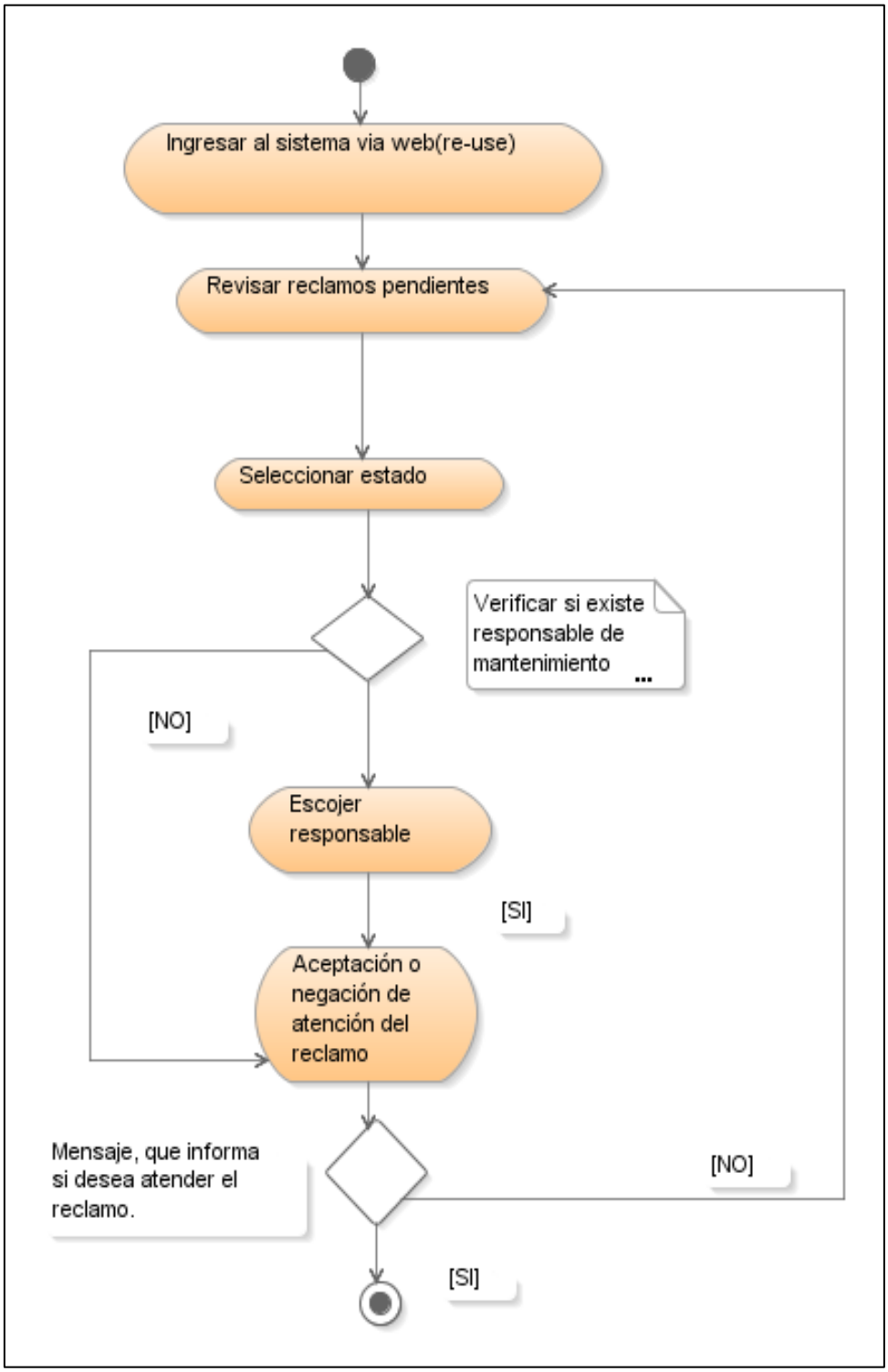

Figura 3 - 20 Atender reclamo (Diagrama de Actividades)

3.3.2.1.5 Pendientes por fecha

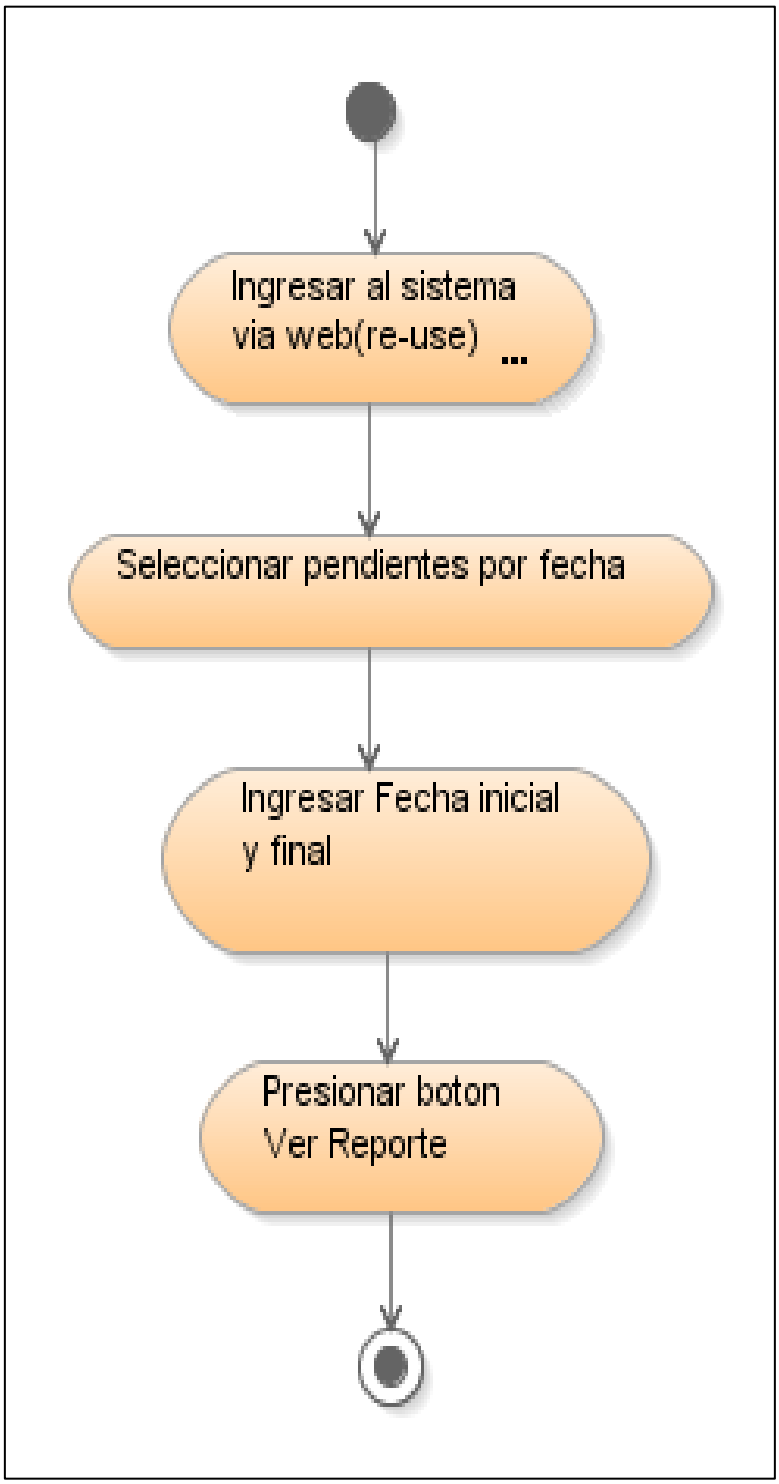

Figura 3 - 21 Pendientes por fecha (Diagrama de Actividades)

3.3.2.1.6 Atender reclamos en lote

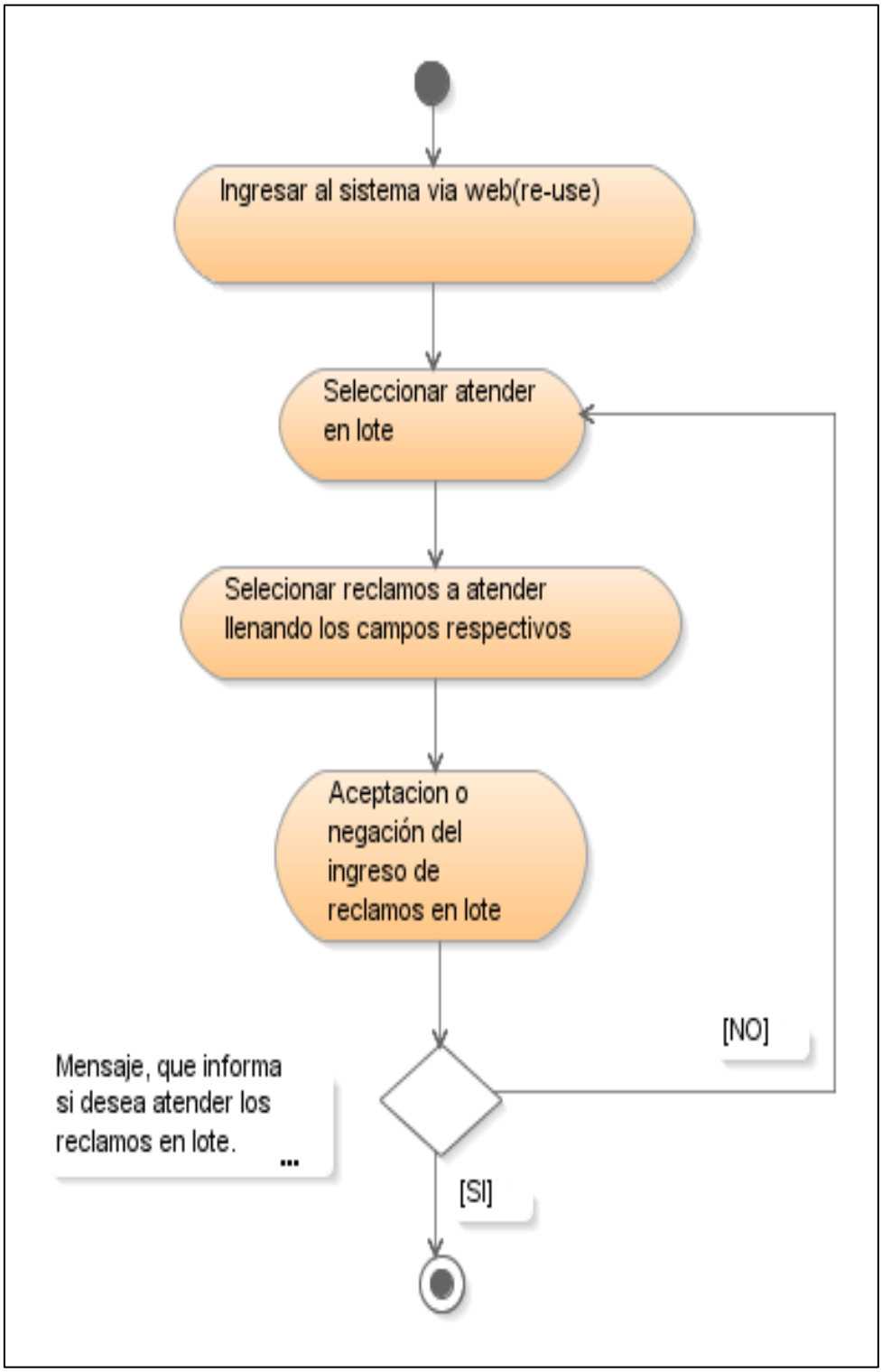

Figura 3 - 22 Atender reclamos en lote (Diagrama de Actividades)

3.3.2.1.7 Consulta de estado de trámite por cliente

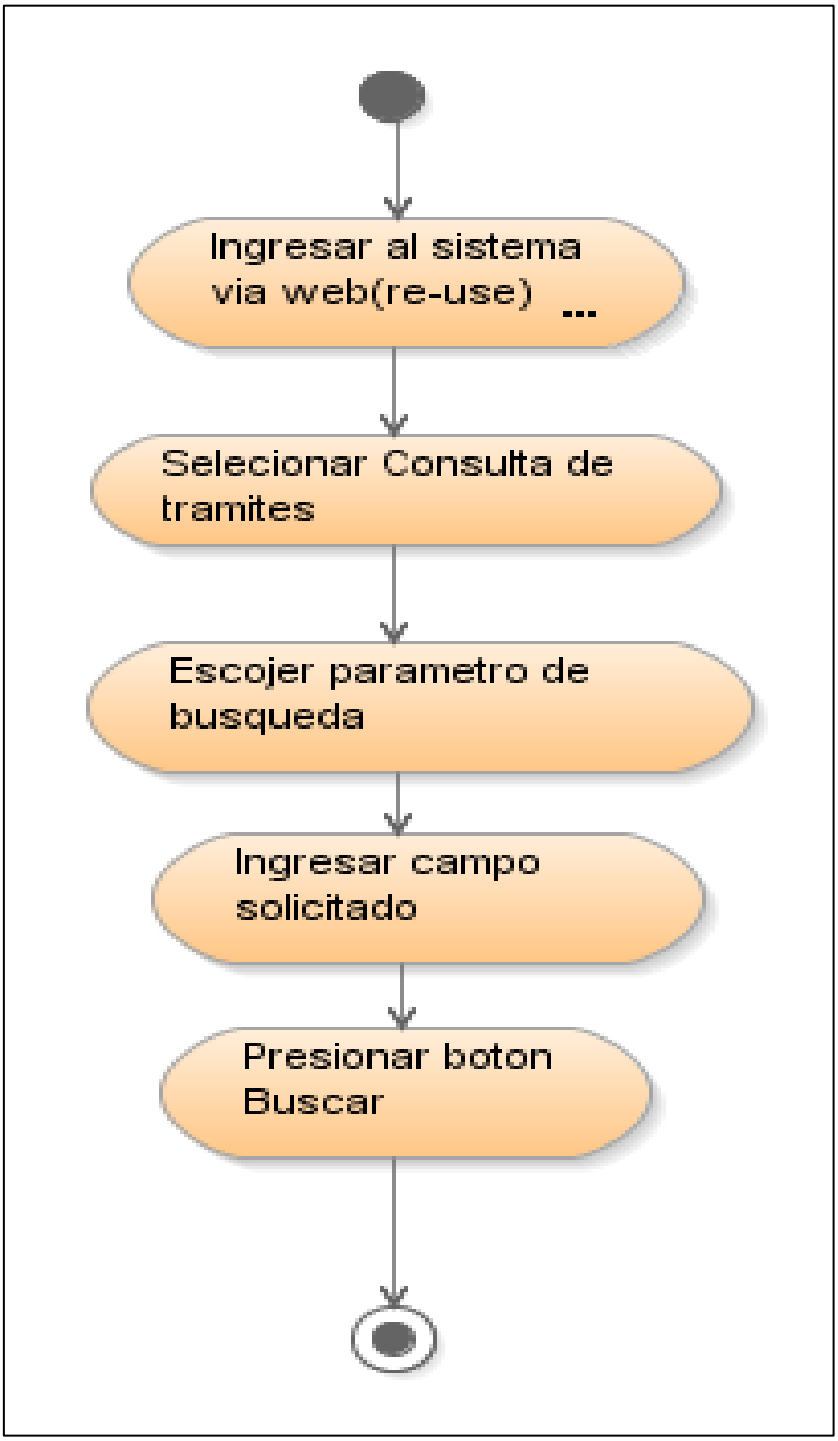

Figura 3 - 23 Consulta de estado de trámite por cliente (Diagrama de Actividades)

3.3.2.1.8 Constancia de ingreso del reclamo

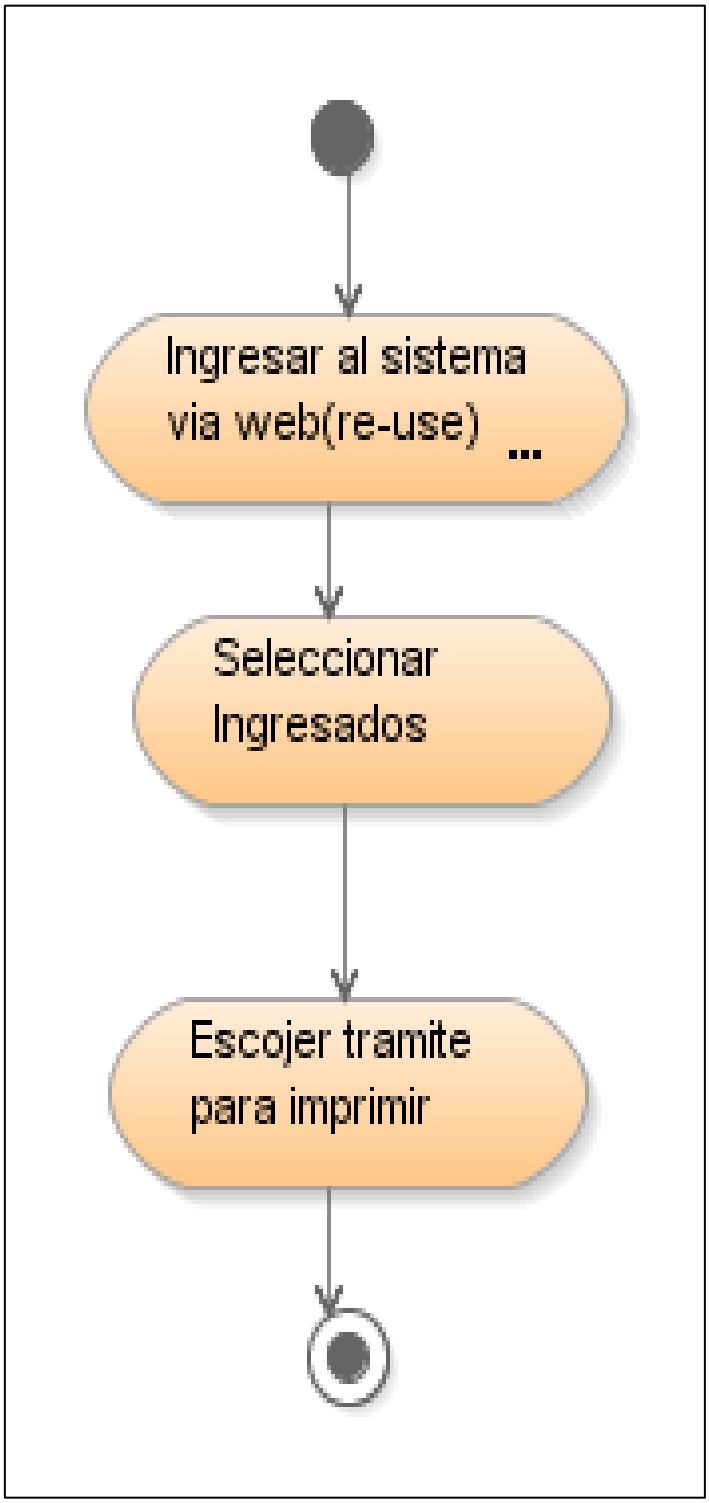

Figura 3 - 24 Constancia de ingreso del reclamo (Diagrama de Actividades)

3.3.2.1.9 Conclusión de un reclamo

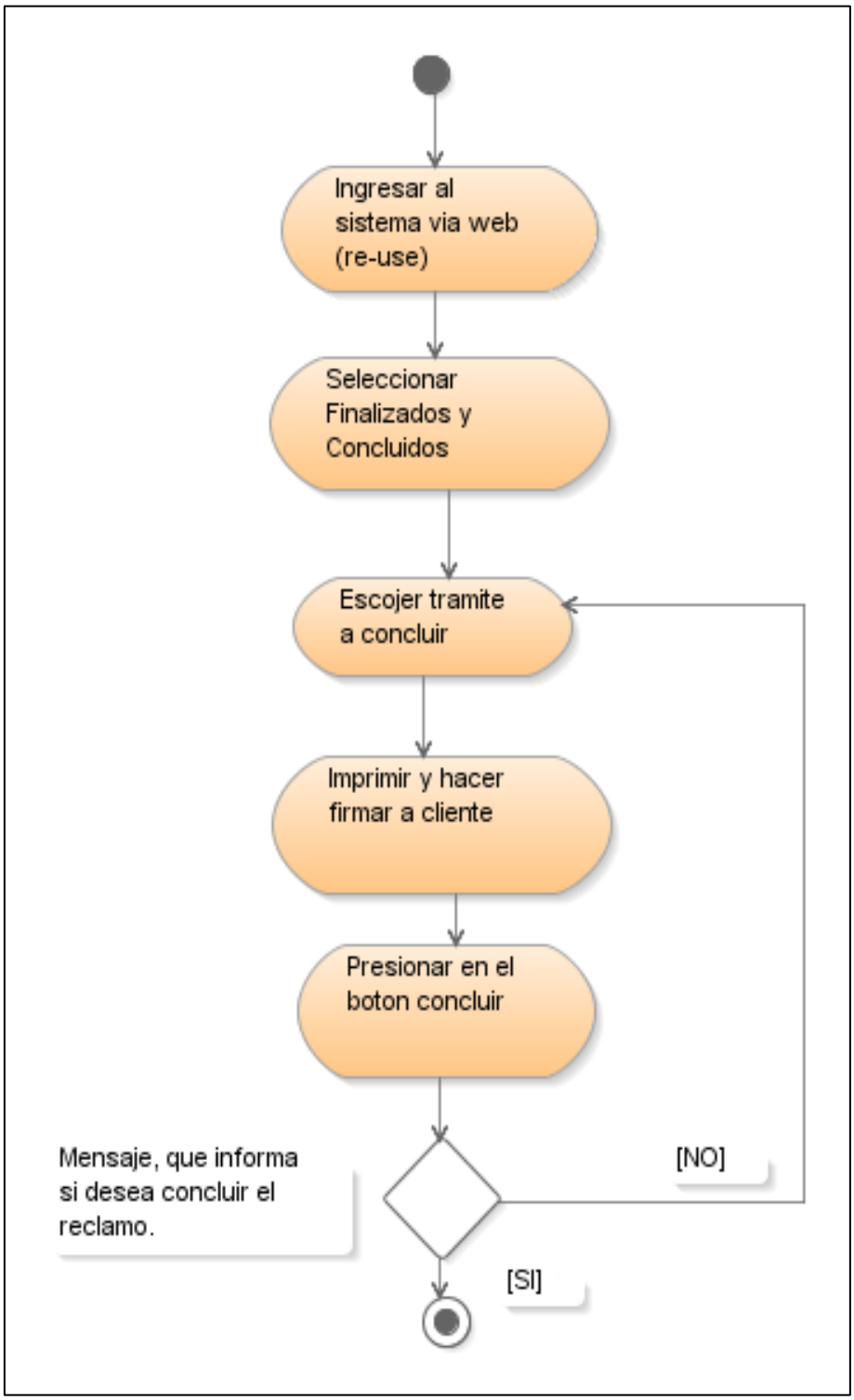

Figura 3 - 25 Conclusión de un reclamo (Diagrama de Actividades)

3.3.2.1.10 Agregar/Modificar usuarios en el sistema

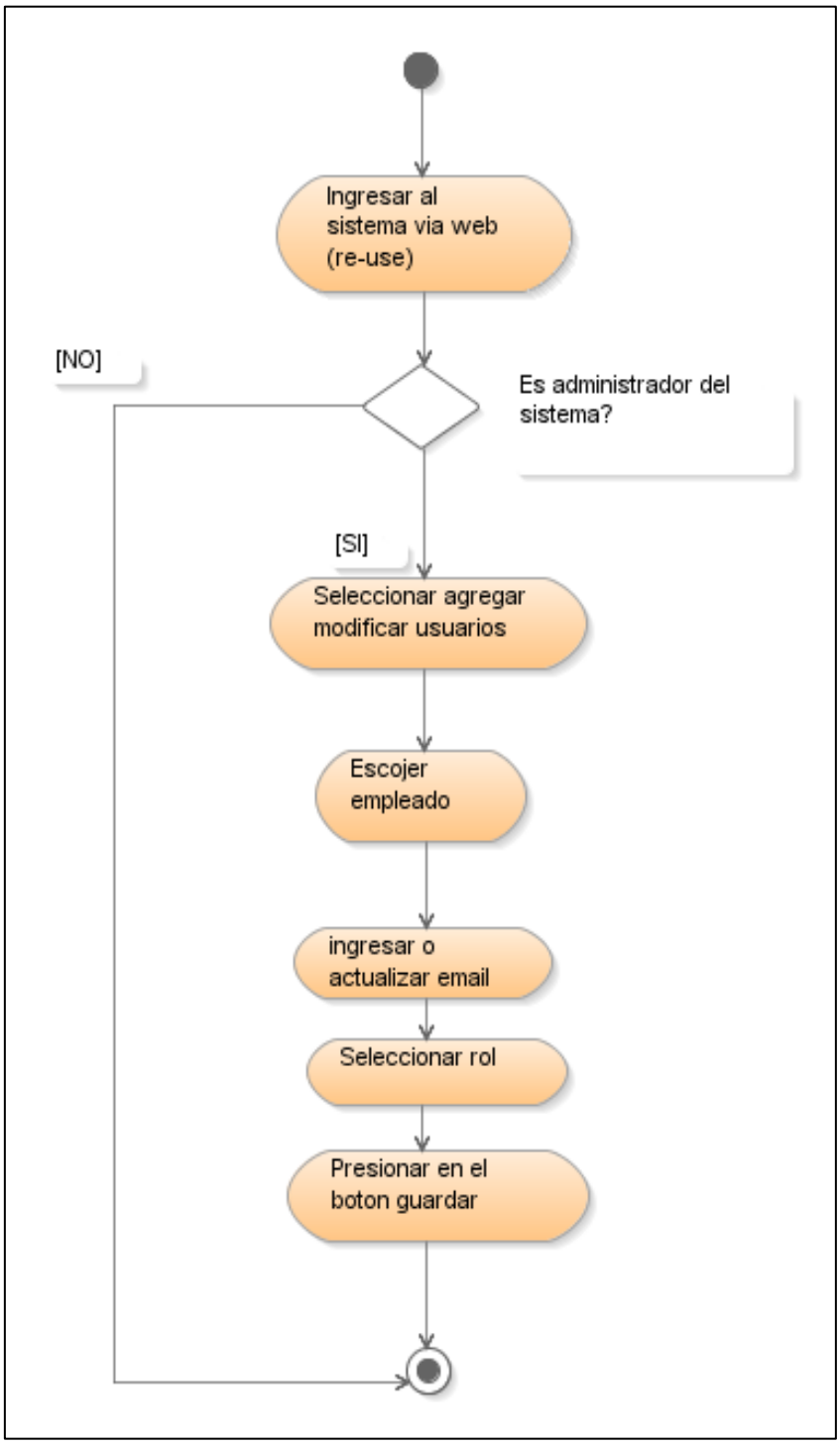

Figura 3 - 26 Agregar/Modificar usuarios en el sistema (Diagrama de Actividades)

3.3.2.1.11 Definir planes de atención por empleado

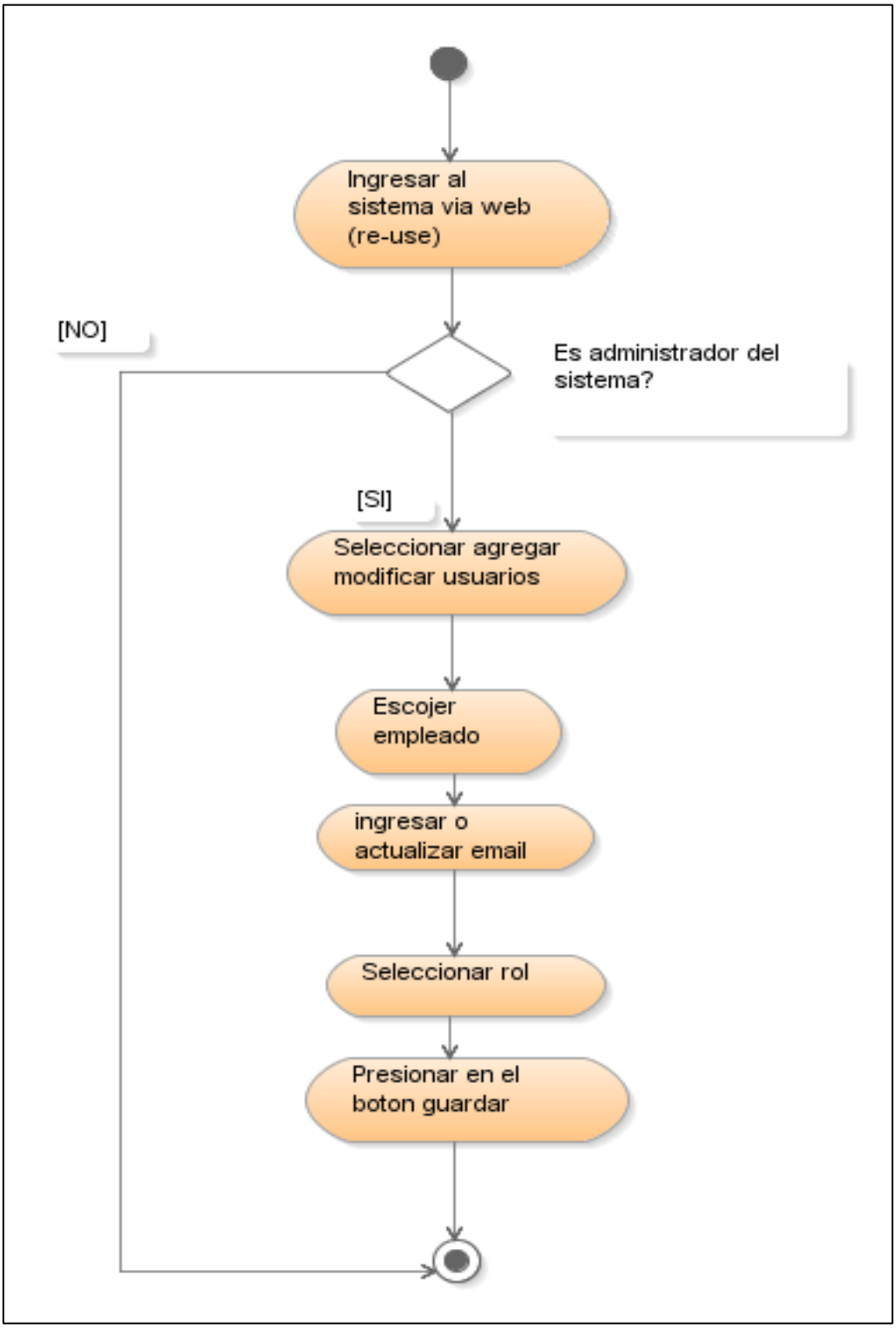

Figura 3 - 27 Definir planes de atención por empleado (Diagrama de Actividades)

3.3.2.2 Diagramas de Arquitectura

3.3.2.2.1 Diagrama de Arquitectura de Software

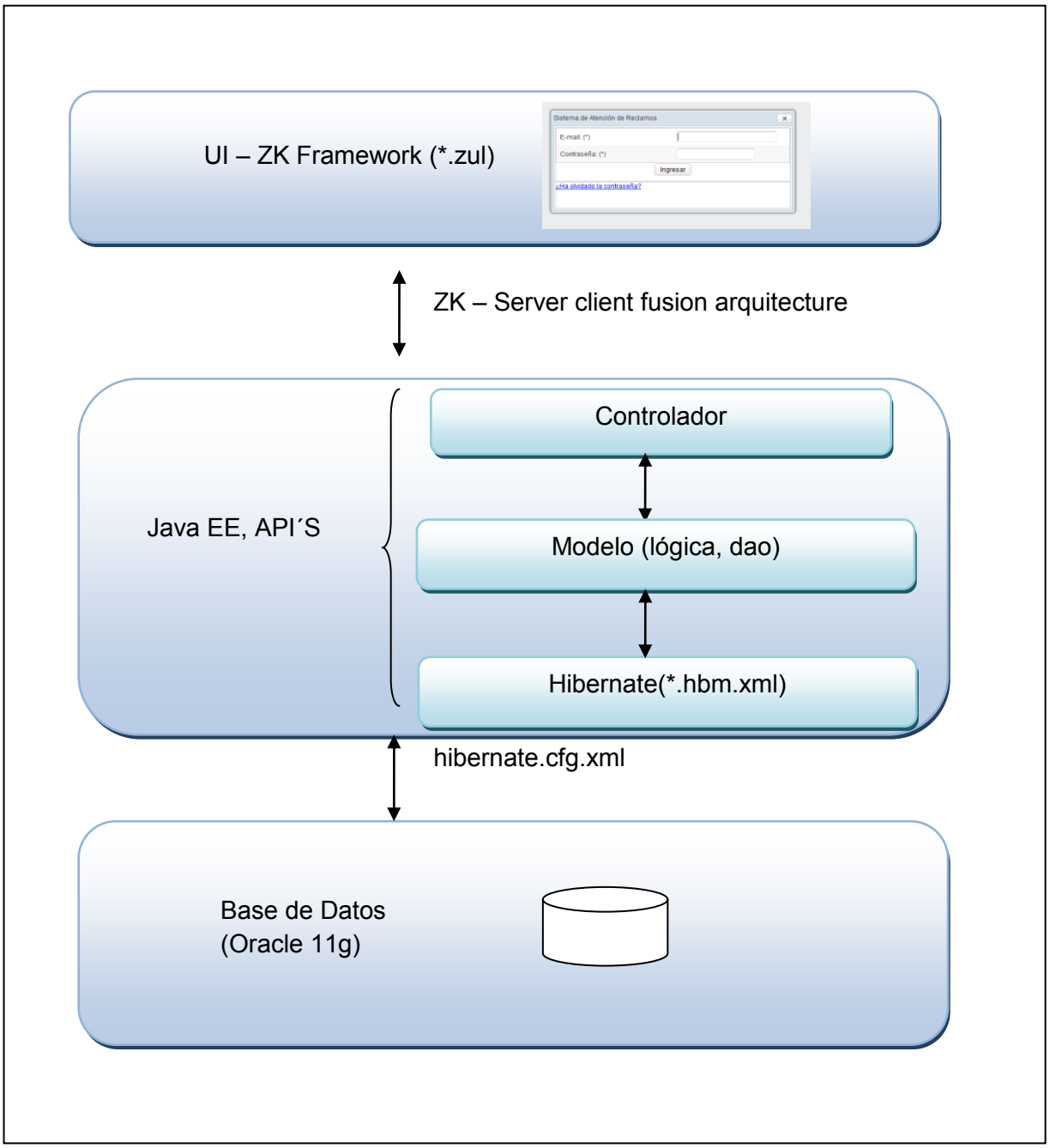

Figura 3 - 28 Arquitectura de software

3.3.2.2.2 Diagrama de Arquitectura de Red y Base de Datos

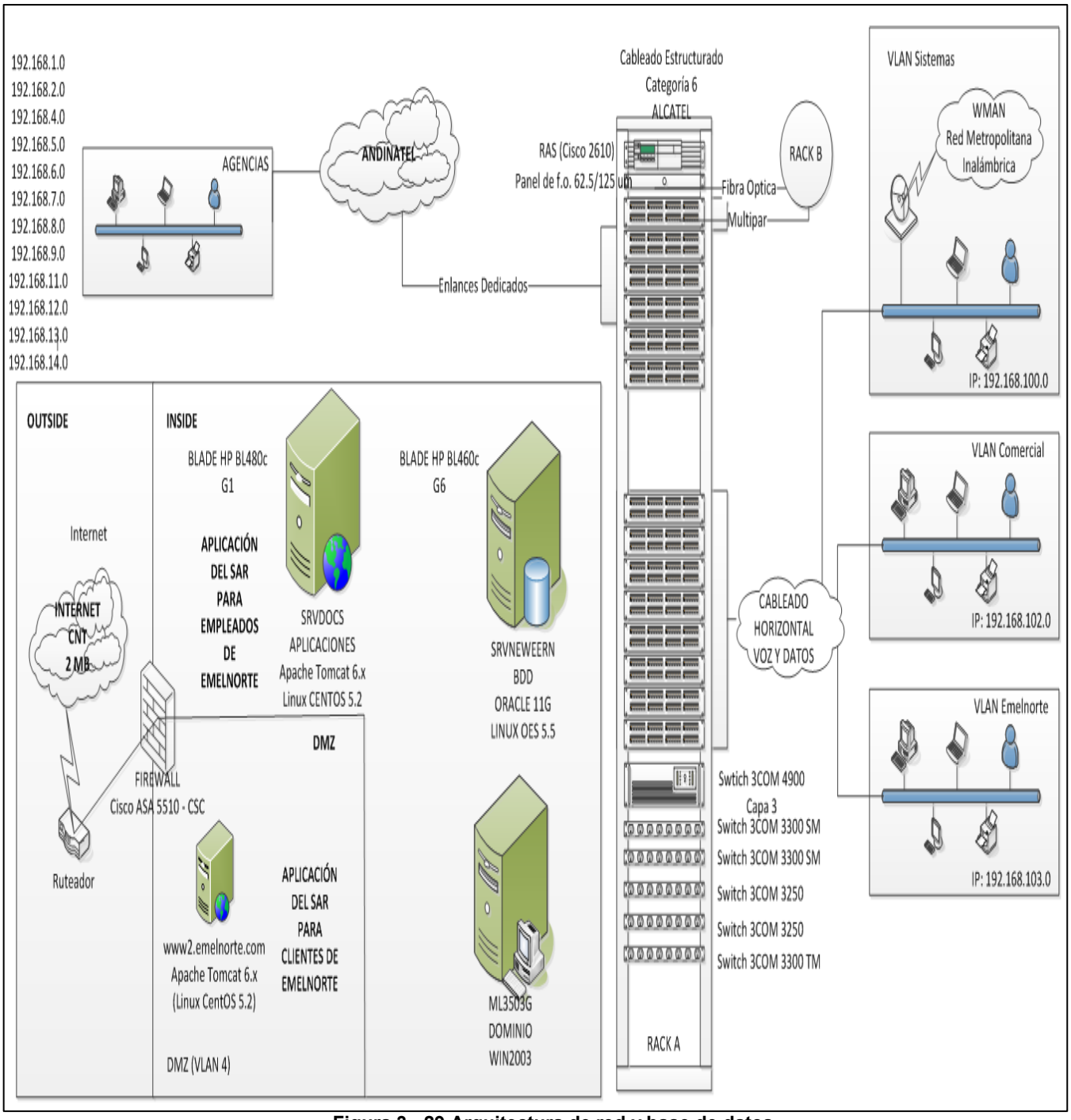

Figura 3 - 29 Arquitectura de red y base de datos

3.4 FASE DE TRANSICIÓN

3.4.2 Especificación de los Casos de Prueba

Nombre:	Ingresar al sistema por parte del cliente
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de validar al usuario para entrar al sistema con su respectiva permiso logia.	
Actores: Cliente de la empresa.	
Precondiciones: El usuario debe ser cliente activo de la empresa, y además no debe tener más de dos facturas vencidas en su planilla eléctrica.	
Flujo Normal: 1. El usuario accede al link de la empresa, en la sección de reclamos se le presenta la pantalla de login. 2. El cliente ingresa el número de suministro asignado a su medidor. 3. El cliente ingresa su número de cédula de identidad, ruc o pasaporte. 4. El sistema confirma los datos. 5. El sistema lo envía a la pantalla principal.	
Flujo Alternativo: En caso de no tener actualizado el número de identificación, el sistema desplegara un mensaje indicándole que se acerque a las oficinas a actualizar sus datos personales.	
Pos condiciones: Aparece la pantalla principal de registro de reclamos establecidos en la regulación Nro. 012/08 emitida por el CONELEC.	
Prueba superado con éxito: SI(X) NO()	

Tabla 3 - 16 Casos de prueba de ingresar al sistema por parte del cliente

Nombre:	Registro del reclamo por parte del cliente
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de registrar los reclamos por parte del cliente de la empresa vía	

web de acuerdo a lo que establece la regulación del CONELEC No. “012/08”.
Actores: Cliente de la empresa.
Precondiciones: El usuario debe estar logeado, y además debe tener asignado un responsable al plan en el que se encuentra el cliente.
Flujo Normal: 1. El cliente de la empresa una vez dentro del sistema, seleccionará la opción de “Ingreso del Reclamo”. 2. Se le presentará los reclamos definidos por la regulación del CONELEC No. “012/08”. 3. Una vez que escoja el reclamo, deberá llenar los campos correspondientes a cada reclamo, presionará en el botón “Ingresar”, se desplegará un mensaje de confirmación. 4. Si está de acuerdo se procede a almacenar los datos en el sistema.
Flujo Alternativo: N/A
Pos condiciones: Aparece un mensaje indicándole el registro correcto del reclamo, en el sistema.
Prueba superado con éxito: SI(X) NO()

Tabla 3 - 17 Caso de prueba registro del reclamo por parte del cliente

Nombre:	Consultar estado del reclamo por parte del cliente.
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de consultar el estado de un reclamo por parte del cliente de la empresa vía web.	
Actores: Cliente de la empresa.	
Precondiciones: El usuario debe estar logeado.	
Flujo Normal: 1. El cliente de la empresa una vez dentro del sistema, seleccionará la opción de “Estado de sus Reclamos”.	

2. Se le presentara todos los reclamos registrados, indicado el Responsable Actual y el estado en que se encuentran.
Flujo Alternativo: N/A
Pos condiciones: Aparece todos los reclamos registrados por el cliente con su respectivo detalle.
Prueba superado con éxito: SI(X) NO()

Tabla 3 - 18 Caso de prueba consultar estado del reclamo por parte del cliente.

Nombre:	Ingresar al sistema por parte del empleado.
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de consultar el estado de un reclamo por parte del cliente de la empresa vía web.	
Actores: Empleado de la empresa.	
Precondiciones: El usuario debe ser empleado de la empresa y estar habilitado en el sistema.	
Flujo Normal: 1. El empleado de la empresa ingresara al siguiente link http://www2.emelnorte.com:8889/reclamos_responsables/ 2. Se le presentará la pantalla de ingreso al sistema. 3. Colocará su e-mail respectivo y la contraseña asignada. 4. Una vez llenos estos campos presionara en “Ingresar”. 5. Se procede a verificar los datos en el sistema, si está correcta la información procede a ingresar al sistema.	
Flujo Alternativo: • Si es el primer ingreso al sistema se despliega una pantalla donde se actualiza la contraseña respectiva. • Si olvido su contraseña puede seleccionar el enlace “Ha olvidado su contraseña?”.	

Pos condiciones: Se cargan las respectivas opciones de acuerdo al rol asignado.
Prueba superado con éxito: SI(X) NO()

Tabla 3 - 19 Casos de prueba ingresar al sistema por parte del empleado

Nombre:	Registro del reclamo por parte del empleado.
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de registrar un reclamo por parte del empleado de la empresa vía web de acuerdo a lo que establece la regulación del CONELEC No. “012/08”.	
Actores: Empleado de la empresa.	
Precondiciones: El usuario debe estar logeado y además se necesita tener un responsable asignado al plan que tiene el cliente.	
Flujo Normal: 1. El empleado de la empresa una vez dentro del sistema, seleccionará la opción “Reclamos/Ingreso” que se encuentra en el menú de “Opciones”. 2. Se le presentará la respectiva pantalla de “Ingreso”, para realizar la Búsqueda del cliente por una de las siguientes alterativas: número de suministro, número de identificación, apellidos y nombres. Después presiona buscar, lo cual verifica la información en el sistema, posteriormente se desplegaran los posibles clientes que coinciden con el parámetro de búsqueda. 3. Seleccionara el cliente deseado, posteriormente se desplegará los reclamos registrados si existieren. 4. Se le presentará los reclamos definidos por la regulación del CONELEC No. “012/08”. 5. Una vez que escoja el reclamo, deberá llenar los campos correspondientes a cada reclamo, presionará en el botón “Ingresar”, se desplegará un mensaje de confirmación. 6. Si está de acuerdo se procede a almacenar los datos en el sistema. **7.** El sistema asignará automáticamente un responsable para dicho reclamo.	
Flujo Alternativo:	

Se puede visualizar el detalle de los reclamos registrados
Pos condiciones: • Si se ha ingresado el e-mail del cliente le llegaran notificaciones del estado de su reclamo al mencionado cliente. • Se le notificará vía mail al responsable de la empresa indicándole que tiene que atender dicho reclamo.
Prueba superado con éxito: SI(X) NO()

Tabla 3 - 20 Casos de prueba registro del reclamo por parte del empleado

Nombre:	Atender reclamo pendiente.
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de atender un reclamo pendiente por parte del empleado de la empresa vía web.	
Actores: Empleado de la empresa.	
Precondiciones: El usuario debe estar logeado y tener reclamos pendientes	
Flujo Normal: 1. El empleado de la empresa una vez dentro del sistema, seleccionará la opción "Reclamos/Pendientes" que se encuentra en el menú de "Opciones". 2. Se le presentará la respectiva pantalla de todos los reclamos pendientes paginados de 10 en 10. 3. Seleccionará el tramite deseado, posteriormente se desplegará la información detallada del reclamo. 4. Llenara los requisitos solicitados, de acuerdo a cada tipo de reclamo. 5. Una vez que desee atender el reclamo, se desplegará un mensaje de confirmación indicando si desea atender el reclamo. 6. Si está de acuerdo se procede a almacenar los datos en el sistema. 7. El reclamo atendido ya no se encuentra en los pendientes del empleado.	
Flujo Alternativo:	

Se puede seguir atendiendo reclamos pendientes en el caso de existir, puede usarse la opción de reclamos en lote o pendiente detallado.
Pos condiciones: • Si al momento de registrar el reclamo del cliente, se guardó el e-mail de este, se procede a notificar vía mail sobre el nuevo estado del reclamo. • Si el reclamo posee un estado diferente a "FINALIZADO", se le notificara al nuevo responsable vía mail, para que proceda a atender el reclamo.
Prueba superado con éxito: SI(X) NO()

Tabla 3 - 21 Caso de prueba atender reclamo pendiente

Nombre:	Concluir reclamo.
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de concluir un reclamo por parte del empleado de la empresa vía web.	
Actores: Empleado de la empresa.	
Precondiciones: El usuario debe estar logeado y tener reclamos pendientes por concluir.	
Flujo Normal: 1. El empleado de la empresa una vez dentro del sistema, seleccionará la opción "Reclamos/Finalizados y Concluidos" que se encuentra en el menú de "Opciones". 2. Se le presentará la respectiva pantalla de todos los reclamos finalizados y concluidos. 3. Seleccionara el reclamo que se desea concluir, se desplegara el informe para que el empleado imprima dicho informe. 4. Posteriormente se archiva el informe, debidamente firmado por el personal responsable y por el cliente de la empresa, dado este paso se procede a presionar en "Concluir" y se despliega un mensaje de confirmación. 5. Si está de acuerdo se procede a almacenar los datos en el sistema. 6. El reclamo pasa del estado "FINALIZADO" a "CONCLUIDO".	

Flujo Alternativo: Puede seguir concluyendo reclamos si existieren.
Pos condiciones: Si al momento de registrar el reclamo del cliente, se guardó el e-mail de este, se procede a notificar vía mail sobre el nuevo estado del reclamo.
Prueba superado con éxito: SI(X) NO()

Tabla 3 - 22 Caso de prueba concluir reclamo

Nombre:	Crear/modificar usuario en el sistema
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de crear o modificar usuarios en el sistema por parte del administrador del sistema.	
Actores: Administrador del sistema.	
Precondiciones: • El usuario debe estar logeado y tener rol de administrador. • El empleado debe existir en nómina para acceder al sistema. • Debe existir el requerimiento aprobado por el Jefe respectivo de la empresa.	
Flujo Normal: 1. En la pantalla de Creación de Usuarios, el usuario con rol de Administrador, selecciona el código del empleado. 2. Se le presentará los empleados existentes, luego tendrá que escoger el empleado y el rol que va a tener el nuevo usuario del sistema. 3. Se ingresara el email del empleado, luego el sistema enviara al email la contraseña asignada para que este pueda acceder al sistema. 4. Seleccionar la opción guardar, que almacena los datos en el sistema.	
Flujo Alternativo: N/A	
Pos condiciones: El sistema despliega un mensaje de actualización de datos	

Prueba superado con éxito:
SI(X) NO()

Tabla 3 - 23 Caso de prueba crear/modificar usuario en el sistema

Nombre:	Definir/modificar planes de atención por empleado.
Autor:	David Bolaños
Fecha:	12-05-2012
Descripción: Se encarga de definir o modificar planes de atención por empleado, para cada plan se establece si el responsable asignado para dicho plan es responsable inicial o no lo es.	
Actores: Administrador del sistema.	
Precondiciones: • El usuario debe estar logeado y tener rol de administrador. • El empleado debe existir en nómina para acceder al sistema. • Debe existir el requerimiento aprobado por el Jefe respectivo de la empresa.	
Flujo Normal: 1. El empleado de la empresa una vez dentro del sistema, seleccionará la opción "Administrar/Planes por empleado" que se encuentra en el menú de "Opciones". 2. Se le presentará una pantalla para realizar la búsqueda del empleado por código o por sus apellidos y nombres, luego se procederá a escoger al empleado. 3. Una vez seleccionado el empleado se procede a establecer los respectivos planes que atenderá el empleado. 4. Una vez que desee guardar los planes definidos para el empleado seleccionado, se desplegará un mensaje de confirmación indicando si desea guardar los datos. 5. Si está de acuerdo, se almacena la información.	
Flujo Alternativo: N/A	
Pos condiciones: El sistema despliega un mensaje de actualización de datos	
Prueba superado con éxito: SI(X) NO()	

Tabla 3 - 24 Caso de prueba definir/modificar planes de atención por empleado

CAPÍTULO IV CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El Sistema de Atención de Reclamos ha sido un éxito en la Empresa Eléctrica Regional Norte, desde su implementación en el mes de enero del año 2010, Emelnorte recibió el premio CIER (Calidad del Servicio Eléctrico Regional) al haber mejorado los índices de satisfacción al cliente.

- Gracias a las bondades que ofrece el SAR, los clientes de la empresa pueden realizar un registro confiable y seguro de los reclamos, como establece el CONELEC, es por ello que en los años: 2010, 2011 y parte del 2012 se almacenó en el sistema alrededor de 10418 trámites versus los pocos reclamos que se archivaban en el pasado.

- Anteriormente, era difícil de coordinar los trabajos de mantenimiento en el alumbrado público que reportaban los clientes, debido al área extensa de concesión que posee Emelnorte, gracias a la plataforma tecnológica que cuenta el SAR, se ha logrado optimizar enormemente los tiempos de atención de reclamos.

- Uno de los factores primordiales en el éxito del SAR, se debe a la integración de la cartera de clientes del Sistema Comercial, debido a que se ha realizado una correcta segmentación de los mismos para poder realizar un óptimo seguimiento a los reclamos que estos realizan.

4.2 Recomendaciones

- Al encontrarse con proyectos de esta magnitud, es de gran importancia el uso correcto de los lineamientos que establece una metodología de desarrollo de software como lo es RUP, enfatizando en el levantamiento de los casos de uso, debido a que este artefacto es la base para elaborar el corazón del sistema.

- En sistemas que no requieran de gran transaccional como los es el SAR, es aconsejable utilizar un mapeador objeto relacional como Hibernate, pero si nuestra aplicación requiere de una mayor transaccional los mapeadores quedan de lado, ya que las sentencias SQL que se solicitan al motor de base de datos no son pre compiladas previamente, si no que dichas sentencias son verificadas en caliente.

- Si lo que deseamos es obtener cierta independencia del motor de base de datos, el uso del framework Hibernate, cumple todas las necesidades de portabilidad con cero dependencia de nuestra base de datos.

- Las tendencias en el desarrollo web requieren de un framework, que ofrezca componentes amigables al usuario, sin que este necesite estar recargando el navegador en cada petición que ejecute; y por otro lado los programadores Java necesitan cambiar las típicas paginas JSP, por herramientas basadas en clientes ricos. ZK cumple con la integridad de componentes Ajax y la madurez de la plataforma JEE, con un coste de aprendizaje mínimo es altamente aplicable para proyectos como el SAR.

Impacto del Sistema de Atención de Reclamos

Gracias a la correcta implementación del SAR desde Enero del 2010, la empresa eléctrica ha logrado registrar 10418 reclamos como se muestra en la siguiente figura:

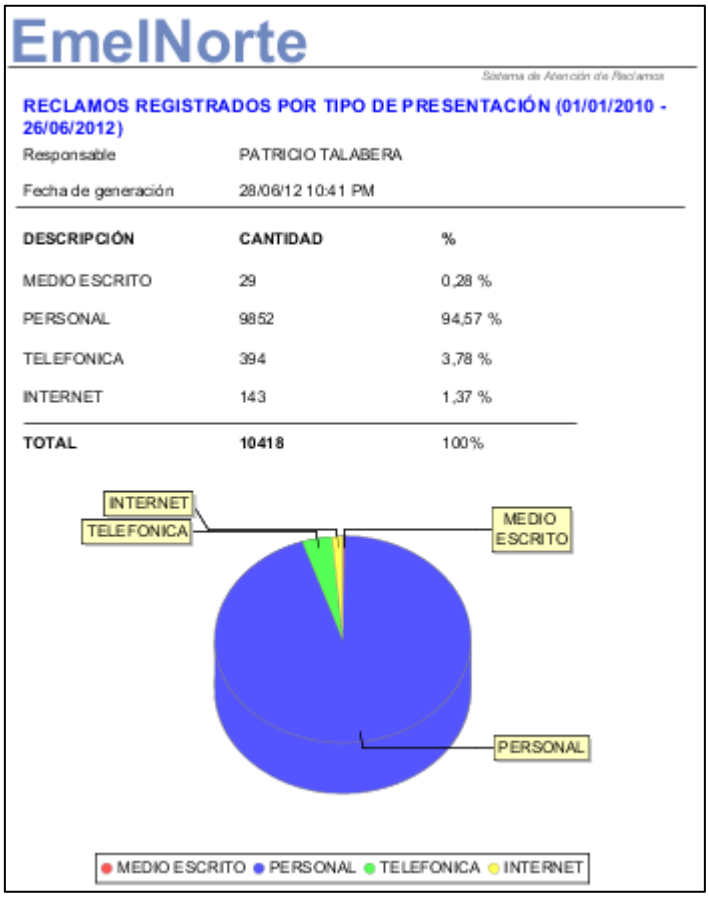

Figura 4 - 1 Reclamos registrados por tipo de presentación

Como se puede observar, la mayor cantidad de reclamos registrados se presentan por la vía personal.

Además se puede comprobar que la mayoría de reclamos finalizados, son del tipo técnicos, como se indica la siguiente figura.

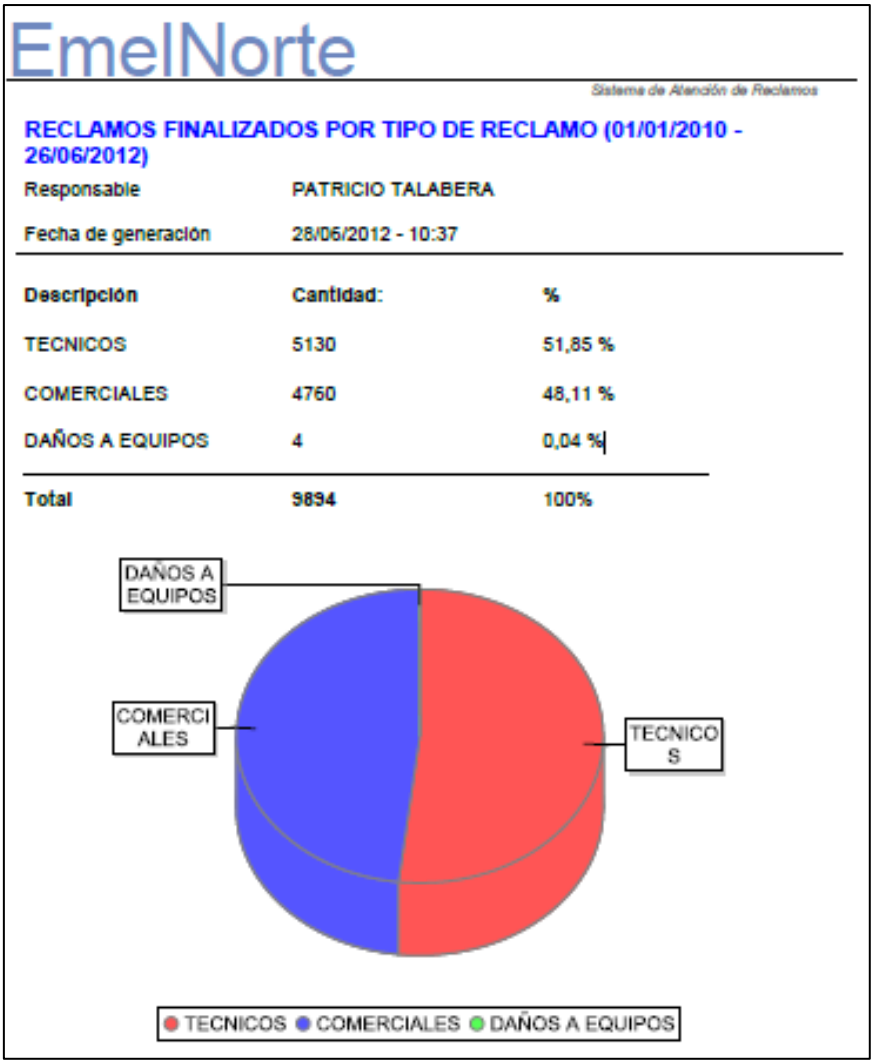

Figura 4 - 2 Reclamos finalizados por su tipo

En la siguiente figura se muestran los reclamos pendientes, demostrándose la minoría que existe en relación a los reclamos finalizados.

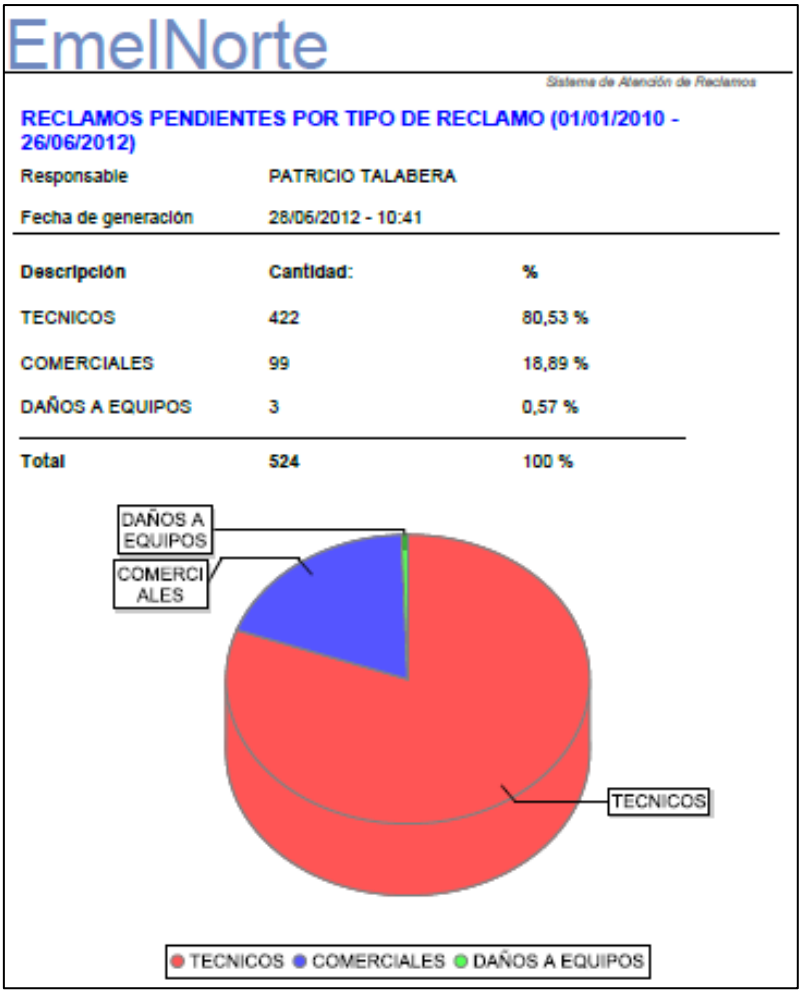

Figura 4 - 3 Reclamos pendientes por su tipo

Gracias, a la información que se puede obtener del SAR, la alta administración de EMELNORTE S.A.; ha logrado ubicar a la empresa entre las mejores eléctricas del país.

Glosario de Términos

Applets generalmente están embebidos en páginas web y pueden ser ejecutados directamente desde un navegador con soporte para Java.

Back-End es la parte que procesa la entrada desde el front-end. La separación del sistema en "front ends" y "back ends" es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. La idea general es que el front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el back-end pueda usar. La conexión del front-end y el back-end es un tipo de interfaz.

Clustering termino común para identificar el mecanismo de distribuir un servicio sobre un número de servidores para incrementar la tolerancia a fallas y soportar mayores cargas que las que podría soportar un servidor simple

CSS hojas de estilo en cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

DHTML El HTML Dinámico o DHTML (del inglés *Dynamic HTML*) designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML estático, un lenguaje interpretado en el lado del cliente (como JavaScript), el lenguaje de hojas de estilo en cascada (CSS) y la jerarquía de objetos de un DOM.

Direct Push el tráfico de Direct Push funciona como un conjunto de solicitudes pequeñas de HTTP a un sitio web de Internet que tarda mucho tiempo en publicar una respuesta.

DOM (Document Object Model - Modelo de Objetos de Documento), especificación que determina cómo los objetos (texto, imágenes, enlaces, etc.) en una página web son representados.

Escalabilidad en telecomunicaciones y en ingeniería informática, la escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Fiabilidad característica de los sistemas informáticos por la que se mide el tiempo de funcionamiento sin fallos. En el caso del hardware, se han conseguido altísimos grados de fiabilidad, mientras que en el software siguen existiendo bugs que dificultan el buen funcionamiento de los programas.

Framework (o marco de trabajo, por su traducción al español), es un componente de software que reúne varias tecnologías para ofrecer un conjunto de herramientas para el desarrollo de aplicaciones.

Front-End es la parte del software que interactúa con el o los usuarios.

HTML, siglas de ***HyperText Markup Language*** (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.

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura.

Java EE o Java Platform, Enterprise Edition (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación parte de la Plataforma Java para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipiado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

JSON, acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

JSF Java Server Faces es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. **JSF** usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.

JSP JavaServer Pages es una tecnología Java que permite generar contenido dinámico para la web, en forma de documentos HTML, XML o de otro tipo.

Persistencia en informática de modo genérico, se refiere a la propiedad de los datos para que estos sobrevivan de alguna manera.

Plug-and-play o ***PnP*** (en español "enchufar y usar") es la tecnología que permite a un dispositivo informático ser conectado a una computadora sin tener que configurar, mediante jumpers o software específico (no controladores) proporcionado por el fabricante, ni proporcionar parámetros a sus controladores. Para que sea posible, el sistema operativo con el que funciona el ordenador debe tener soporte para dicho dispositivo.

Plug-in es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

POJO (acrónimo de *Plain Old Java Object*) es una sigla creada por Martin Fowler, Rebecca Parsons y Josh MacKenzie en septiembre de 2000 y utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.

Release comprende un producto final, preparado para publicarse como versión definitiva a menos que aparezcan errores que lo impidan.

RUP (o proceso unificado de rational, por su traducción al español), es un proceso de desarrollo de software que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Serializable consiste en un proceso de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML o JSON, entre otros.

Stakeholder es un término inglés utilizado por primera vez por R. E. Freeman en su obra: “Strategic Management: A Stakeholder Approach” (Pitman, 1984), para referirse a quienes pueden afectar o son afectados por las actividades de una empresa.

UML (lenguaje unificado de modelado, por su traducción al español), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Widget es un pequeño software que se emplea en la web y que puede ser instalado y ejecutado dentro de un navegador web por un usuario. Usualmente son desarrollados en lenguajes DHTML, JavaScript o Flash. Otros sinónimos utilizados para referirse a los widgets para la web son: portlet, gadget, badge, webjit, snippet, flake, mini, etc.

Worflow (flujo de trabajo, por su traducción al español) es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

XML, siglas en inglés de ***eXtensible Markup Language*** (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

XUL se basa en múltiples estándares y tecnologías web, incluyendo CSS, JavaScript y DOM. Tal dependencia hace XUL relativamente fácil de aprender para las personas con experiencia en programación y diseño web.

ZUML en un lenguaje de marcación para la definición de una potente interfaz de usuario, que es creado por los desarrolladores del framework ZK.

Bibliografía y Web Grafía

Albiol, F. R. (Diciembre de 2004). *Introduccrión a Hibernate*. Recuperado el 12 de Marzo de 2010, de http://www.javahispano.org/contenidos/es/introduccrion_a_hibernate/

Anónimo. (28 de Mayo de 2007). *Frameworks MVC*. Recuperado el 14 de Octubre de 2011, de http://estebansaiz.com/blog/2007/05/28/frameworks-mvc/

Anónimo. (27 de Enero de 2007). *Hibernate*. Recuperado el 12 de Marzo de 2010, de http://mundogeek.net/archivos/2007/01/27/hibernate/

Anónimo. (8 de Febrero de 2009). *Creación de Reportes con JasperRepots y iReports*. Recuperado el 2010 de Febrero de 2010, de http://javatutoriales.blogspot.com/2009_02_01_archive.html

Anónimo. (Noviembre de 2009). *How to persist the java objects using Hibernate ORM framework*. Recuperado el 17 de Marzo de 2010, de http://www.interviewjava.com/2009/11/how-to-persist-java-objects-using.html

Anónimo. (1 de Septiembre de 2011). *Introducción a Hibernate*. Recuperado el 26 de Septiembre de 2011, de http://www.dosideas.com/cursos/mod/resource/view.php?id=20

Anónimo. (s.f.). *AJAX*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/AJAX

Anónimo. (s.f.). *Applet*. Recuperado el 2 de Mayo de 2012, de http://www.alegsa.com.ar/Dic/java%20applet.php

Anónimo. (s.f.). *Clustering*. Recuperado el 2 de Mayo de 2012, de http://es.scribd.com/doc/3101744/CLUSTERING

Anónimo. (s.f.). *CSS*. Recuperado el 2 de Mayo de 2012, de http://www.pergaminovirtual.com.ar/definicion/CSS.html

Anónimo. (s.f.). *Direct Push*. Recuperado el 2 de Mayo de 2012, de http://technet.microsoft.com/es-es/library/dd568843.aspx

Anónimo. (s.f.). *DOM*. Recuperado el 2 de Mayo de 2012, de http://www.alegsa.com.ar/Dic/dom.php

Anónimo. (s.f.). *Escalabilidad*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Escalabilidad

Anónimo. (s.f.). *Fases del desarrollo de software* . Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Fases_del_desarrollo_de_software

Anónimo. (s.f.). *Fiabilidad*. Recuperado el 2 de Mayo de 2012, de http://www.mastermagazine.info/termino/4990.php

Anónimo. (s.f.). *Flujo de trabajo*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Flujo_de_trabajo

Anónimo. (s.f.). *Front-end y back-end*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Front-end_y_back-end

Anónimo. (s.f.). *HTML*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/HTML

Anónimo. (s.f.). *HTML dinámico*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/HTML_din%C3%A1mico

Anónimo. (s.f.). *JASPER REPORTS*. Recuperado el 13 de Enero de 2010, de http://150.185.75.30/atiwiki/index.php/JASPER_REPORTS#Ejemplo_de_uso_de_JasperReports

Anónimo. (s.f.). *Java (lenguaje de programación)*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

Anónimo. (s.f.). *Java EE*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Java_EE

Anónimo. (s.f.). *JavaScript*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/JavaScript

Anónimo. (s.f.). *JavaServer Pages*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/JavaServer_Pages

Anónimo. (s.f.). *JSON*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/JSON

Anónimo. (s.f.). *Lenguaje Unificado de Modelado*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

Anónimo. (s.f.). *Paginacion con hibernate*. Recuperado el 2009 de Diciembre de 2009, de https://ame.endesa.es/confluence/display/AMEBASE/Hibernate+Paginacion

Anónimo. (s.f.). *Persistencia (informática)*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Persistencia_%28inform%C3%A1tica%29

Anónimo. (s.f.). *Plain Old Java Object*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/POJO

Anónimo. (s.f.). *Plug and play*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Plug_and_play

Anónimo. (s.f.). *Plug-in*. Recuperado el 2 de MAyo de 2012, de http://es.wikipedia.org/wiki/Complemento_%28inform%C3%A1tica%29

Anónimo. (s.f.). *RUP*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

Anónimo. (s.f.). *Serialización*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Serializaci%C3%B3n

Anónimo. (s.f.). *Stakeholder*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/Stakeholder

Anónimo. (s.f.). *Widget*. Recuperado el 2 de Mayo de 2012, de http://www.alegsa.com.ar/Dic/widget%20para%20web.php

Anónimo. (s.f.). *XML*. Recuperado el 2 de Mayo de 2012, de Extensible Markup Language: http://es.wikipedia.org/wiki/Extensible_Markup_Language

Anónimo. (s.f.). *XUL*. Recuperado el 2 de Mayo de 2012, de http://en.wikipedia.org/wiki/XUL

Anónimo. (s.f.). *ZK Framework*. Recuperado el 2 de Mayo de 2012, de http://es.wikipedia.org/wiki/ZUML

Chen, H., & Cheng, R. (2007). *ZK Ajax without JavaScript Framework* (1era ed.). New York, EEUU: Potix Corporation.

CONELEC. (s.f.). *Regulaciones CONLEC*. Recuperado el 1 de Septiembre de 2009, de http://www.conelec.gob.ec/normativa.php?categ=1&subcateg=3&cd_centro=4007

CORP, I. (2003). *Rational Unified Process.*

Corporation, P. (s.f.). *ZUML ZK Attributes*. Recuperado el 2009 de Diciembre de 2009, de http://docs.zkoss.org/wiki/ZUML_ZK_Attributes

Elliot, J., Fowler, R., & O'Brien, T. (2008). *Harnessing Hibernate* (1era ed.). EEUU: O'Reilly Media.

Henri Chen, P. E. (28 de Agosto de 2008). *ZK MVC Made Easy*. Recuperado el 2011 de Ocutbre de 2011, de http://docs.zkoss.org/wiki/ZK_MVC_Made_Easy

Jball, E. (2006). *The Java EEE 5 Tutorial* (5ta ed.). Sun Microsystems.

JBoss. (s.f.). *Historia de hibernate*. Recuperado el 26 de Septiembre de 2011, de http://www.hibernate.org/about/history

JBoss. (s.f.). *Por que usar hibernate*. Recuperado el 26 de Septiembre de 2011, de http://www.hibernate.org/about/why-hibernate

JBoss. (s.f.). *What is Object/Relational Mapping*. Recuperado el 26 de Septiembre de 2011, de http://www.hibernate.org/about/orm

Oracle. (s.f.). *Java EE 6 Technologies*. Obtenido de http://www.oracle.com/technetwork/java/javaee/tech/index.html

Páez, F. J. (7 de Julio de 2009). *Instalación y configuración de Eclipse Galileo*. Recuperado el 17 de Marzo de 2010, de http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=galileo

Patricio, A. (18 de Agosto de 2009). *Using Hibernate with Tomcat*. Recuperado el 2010 de Febrero de 2010, de https://community.jboss.org/wiki/UsingHibernateWithTomcat

Potix Corporation. (s.f.). *ZK Developer's Guide/Advanced ZK/MVC in ZK/Introduction*. Recuperado el 13 de Octubre de 2011, de http://books.zkoss.org/wiki/ZK%20Developer%27s%20Guide/Advanced%20ZK/MVC%20in%20ZK/Introduction

Potix Corporation. (s.f.). *ZK Developer's Reference/Internationalization*. Recuperado el 2011 de Julio de 2011, de http://books.zkoss.org/wiki/ZK%20Developer%27s%20Reference/Internationalization

Potix Corporation. (s.f.). *ZK Developer's Reference/MVC*. Recuperado el 13 de Octubre de 2011, de http://books.zkoss.org/wiki/ZK_Developer%27s_Reference/MVC

Potix Corporation. (s.f.). *ZK Essentials/Introduction to ZK/Component Based UI*. Recuperado el 13 de Octubre de 2011, de http://books.zkoss.org/wiki/ZK%20Essentials/Introduction%20to%20ZK/Component%20Based%20UI

Robbie Cheng, E. P. (12 de Junio de 2007). *How to Secure User's Password with Encryption*. Recuperado el 10 de Diciembre de 2009, de http://www.zkoss.org/smalltalks/encryption/encryption.dsp

Robbie Cheng, E. P. (22 de Enero de 2008). *New Features of ZK 3.0.2*. Recuperado el 9 de diciembre de 2009, de http://www.zkoss.org/smalltalks/zk3.0.2/#forward

Stäuble, M., & Schumacher, H.-J. (2008). *ZK Developer's Guide* (1era ed.). Birmingham, UK: Packt Publishing Ltd.

Timothy Clare, T. E. (6 de Julio de 2009). *Handling huge data using ZK*. Recuperado el 10 de Diciembre de 2009, de http://docs.zkoss.org/wiki/Handling_huge_data_using_ZK

ANEXOS

Anexo 1: Instalación de ZK Studio en Eclipse

ZK Studio es un plug-in para Eclipse que nos ayuda en el desarrollo rápido de aplicaciones RIA utilizando el framework ZK Ajax. Con la ayuda de ZK Studio, somos capaces de desarrollar aplicaciones ZK Web rápida y fácilmente.

ZK Studio incluye las herramientas que se muestra y se enumeran a continuación:

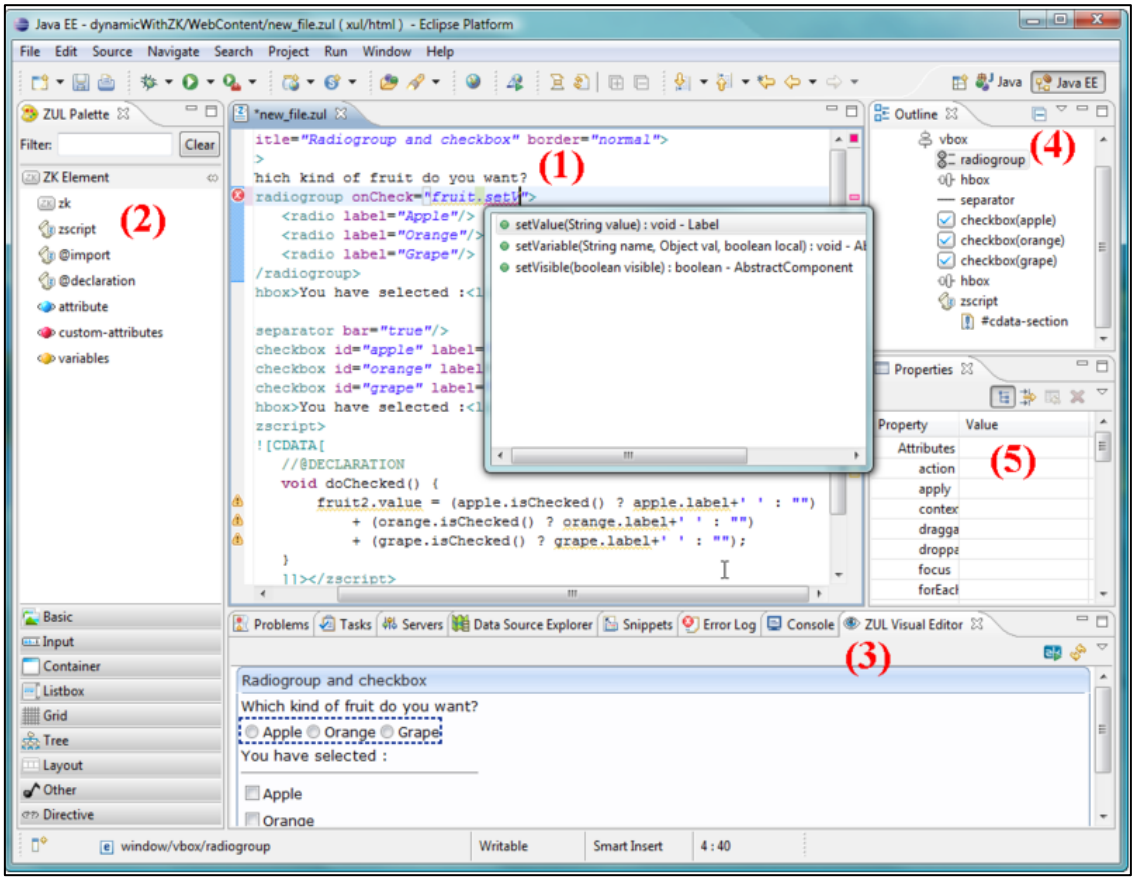

Figura 4 - 4 Elementos de ZK-Studio

1. **Editor ZUL:** Editor ZUL es un editor de documentos para modificar archivos zul y zhtml.
2. **ZUL paleta:** La paleta proporciona una caja de herramientas de arrastrar y soltar los componentes para su uso con archivos ZUL .
3. **ZUL Editor Visual** El editor visual muestra una vista renderizada del archivo zul
4. **Ver esquema de la página Web:** El punto de vista presenta un resumen del contenido en el editor seleccionado ZUL.

5. **Ver propiedades de la interfaz de usuario tag:** La ventana de propiedades muestra una lista de pares de atributos y valores que se corresponden con el nodo seleccionado en el Editor de ZUL o vista jerárquica de la Página Web.

En adición a lo anterior las herramientas GUI, ZK Studio también proporcionan las siguientes características:

1. **ZK nuevo Asistente:** Hay dos asistentes principales, el primero es para crear un nuevo proyecto ZK. El segundo es el asistente que crea un nuevo archivo *. zul.
2. **ZK estilo de diseñador:** El diseñador le ayuda a personalizar el componente ZUL a través de una amigable interfaz gráfica de usuario.
3. **ZK DB Form Builder:** Es un asistente que ayudará a los desarrolladores generar un formulario DB CRUD sin necesidad de escribir ningún código.
4. **ZK Código de ejemplo:** Los códigos de ejemplos que vienen incluidos en el plug-in está dirigido a ayudar a los principiantes a entender rápidamente ZK.
5. **Zscript a MVC Extractor:** Una herramienta para ayudar a refactorizar zscript de forma MVC.
6. **Editor zk.xml:** El editor le ayuda a editar el archivo de configuración de ZK (zk.xml).

- **Compatibilidad con versiones**

ZK-Studio es compatible con: Eclipse 3.6, Eclipse 3.5, Eclipse 3.4, RAD, MyEclipse, Spring Tool Suite 2.5.1.

- **Requisitos previos**

Hay tres piezas de software que usted necesita para descargar e instalar con el fin de utilizar ZK-Studio:

4. **Java SE Development Kit (JDK).** Java versión 5 o 6 se requiere para ejecutar Eclipse.
5. **Eclipse IDE para Java EE Desarrolladores.** ZK-Studio es un plug-in para Eclipse, por lo tanto, debe instalar Eclipse. Se recomienda utilizar el paquete “Eclipse IDE for Java EE Developers”, actualmente se recomiendan las versiones v3.6 (Helios) y v3.5 (Galileo)
6. **Servidores de aplicaciones.** Es necesario instalar un servidor de aplicaciones. Apache Tomcat es uno de los contenedores Web más populares. Tomcat versiones 5.5 y 6 son compatibles.

- **Guía de instalación**

ZK Studio se puede instalar mediante el uso de una URL de actualizaciones que contenga dicho plug-in, o descargar el archivo Zip de instalación.

- **Instalación en línea para Eclipse 3.5**

1. Abierto Eclipse, haga clic en el menú [Ayuda] y seleccione [Instalar nuevo software..]. Una ventana de instalación de ventanas emergentes aparecerá luego haga clic en [Agregar].

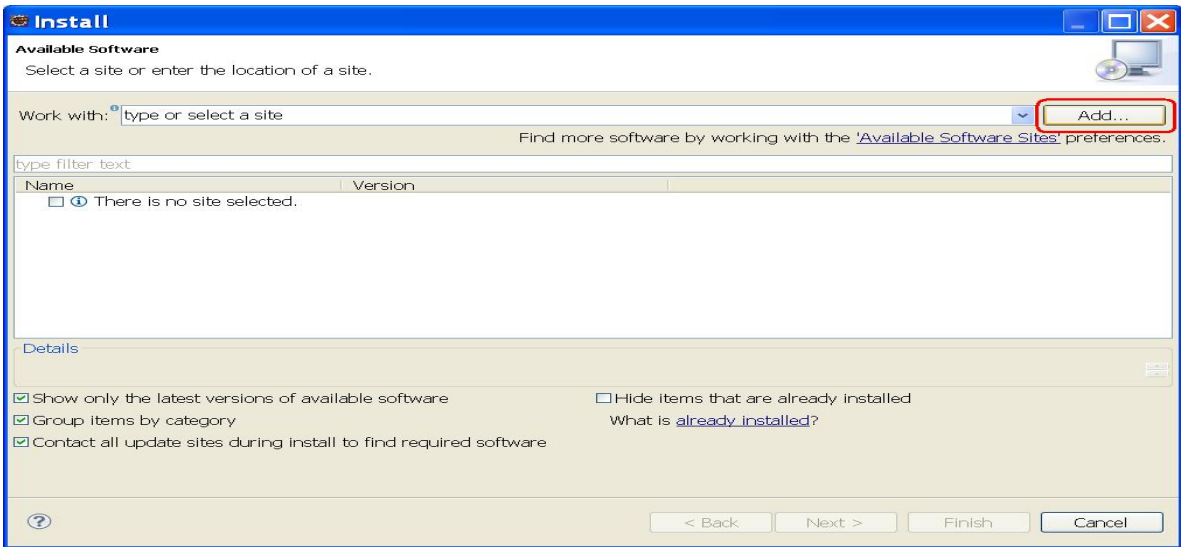

Figura 4 - 5 Instalación de ZK-Studio en línea (paso 1)

2. Coloque el nombre del sitio como “zkstudio” y luego agregue el respectivo repositorio como indica la figura:

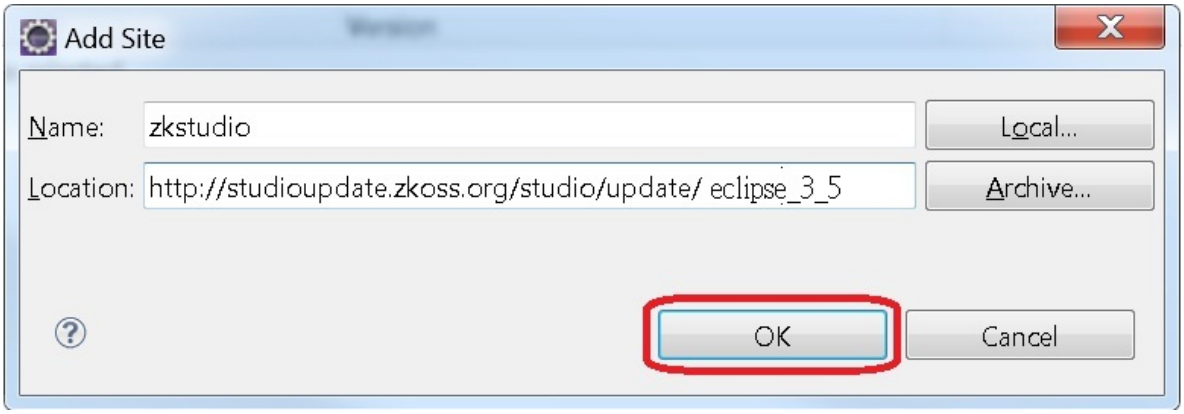

Figura 4 - 6 Instalación de ZK-Studio en línea (paso 2)

3. Desplegar el ZK Studio en la entrada de actualización, marque ZK Studio y haga clic en [Siguiente]

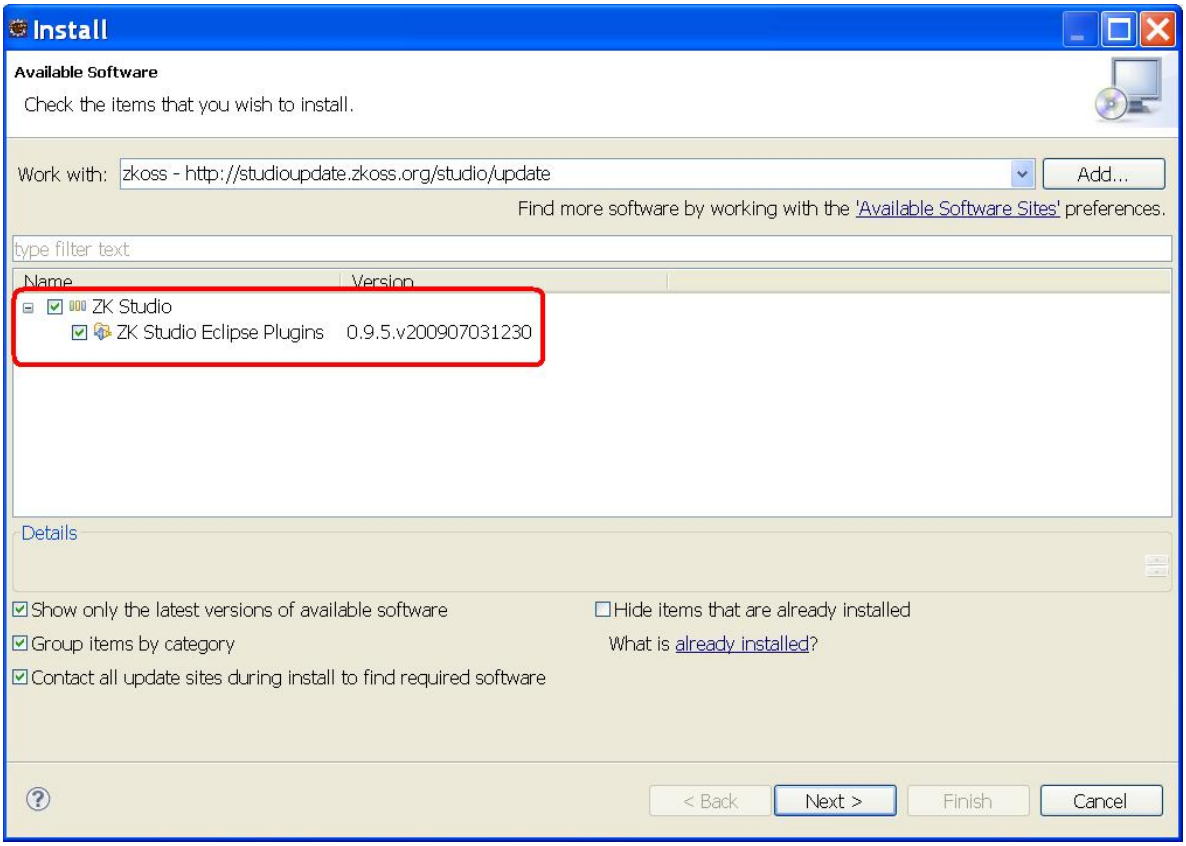

Figura 4 - 7 Instalación de ZK-Studio en línea (paso 3)

El sistema de instalación de Eclipse ahora intentará recuperar la información del sitio de actualizaciones. Esto tomará algún tiempo, dependiendo de su entorno de eclipse, por favor sea paciente.

4. Haga clic en [Finalizar] en la ventana emergente de interfaz de usuario del asistente de instalación.

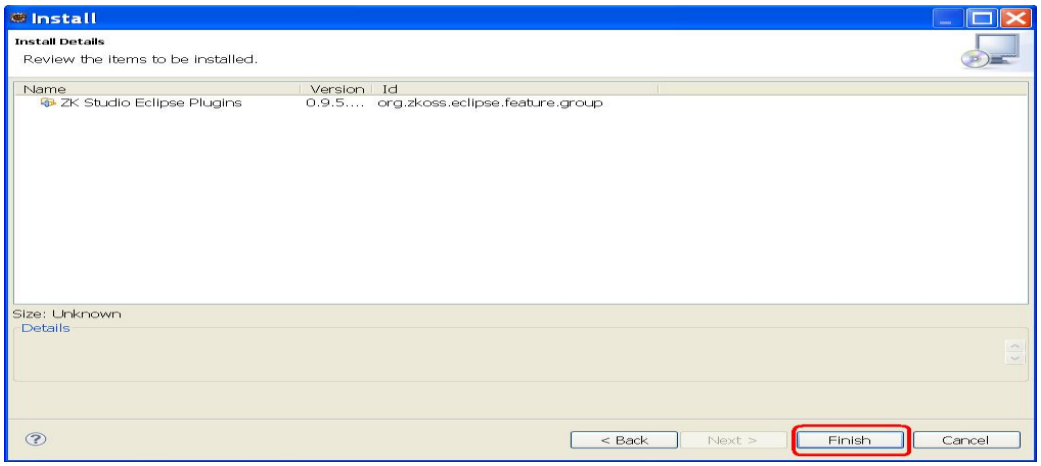

Figura 4 - 8 Instalación de ZK-Studio en línea (paso 4)

5. Una vez completada la instalación, se requiere reiniciar Eclipse.

Figura 4 - 9 Instalación de ZK-Studio en línea (paso 5)

- **Instalación fuera de línea en Eclipse 3.5**
  1. Descargue el archivo zip de instalación.
  2. Los siguientes pasos se parece mucho a la instalación en línea. Sin embargo, haga clic en **Archivo** en el paso 2. A continuación, utilice el Explorador de archivos para seleccionar el archivo descargado en el paso anterior.

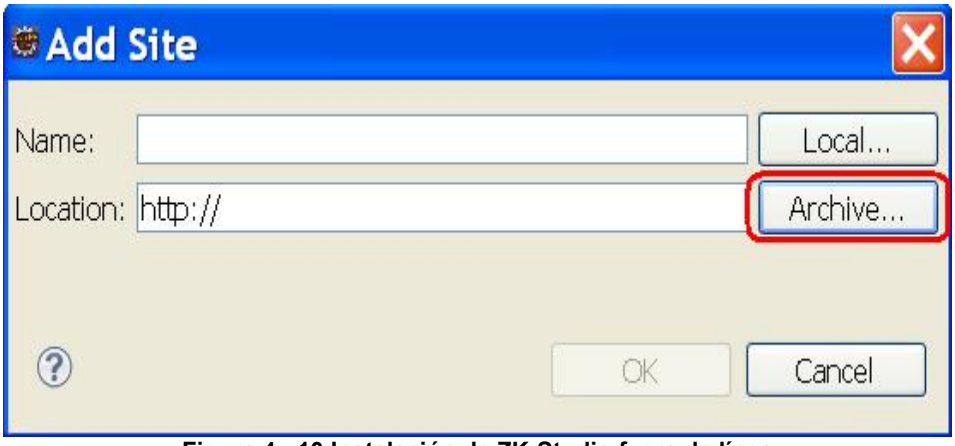

Figura 4 - 10 Instalación de ZK-Studio fuera de línea

- **Activar ZK Studio**

En menú de Eclipse, por favor seleccione **Ayuda** / **Activar Studio ZK.** Un cuadro de diálogo aparecerá:

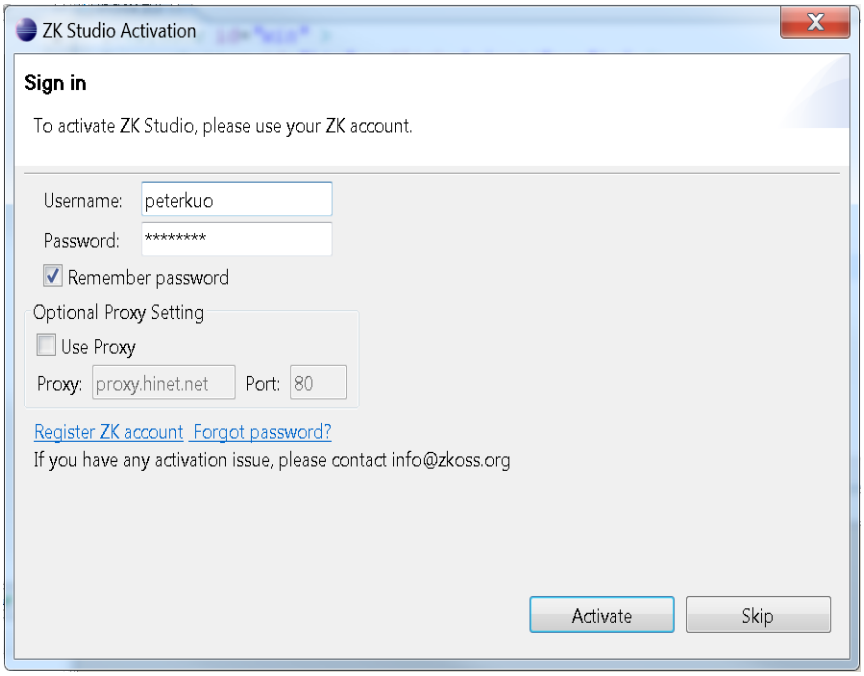

Figura 4 - 11 Activación de ZK-Studio

Por favor ingrese su usuario y contraseña para activar ZK-Studio; si no posee una cuenta de usuario, por favor regístrese en la página oficial de ZK.

Anexo 2: Manual de Usuario

El personal de las áreas comercial y de distribución de la empresa, una vez registrados en el sistema deberán acceder al siguiente link:

http://srvdocs.red.emelnorte.com/reclamos_responsables

El cual deberán ingresar los siguientes datos: correo electrónico y contraseña.

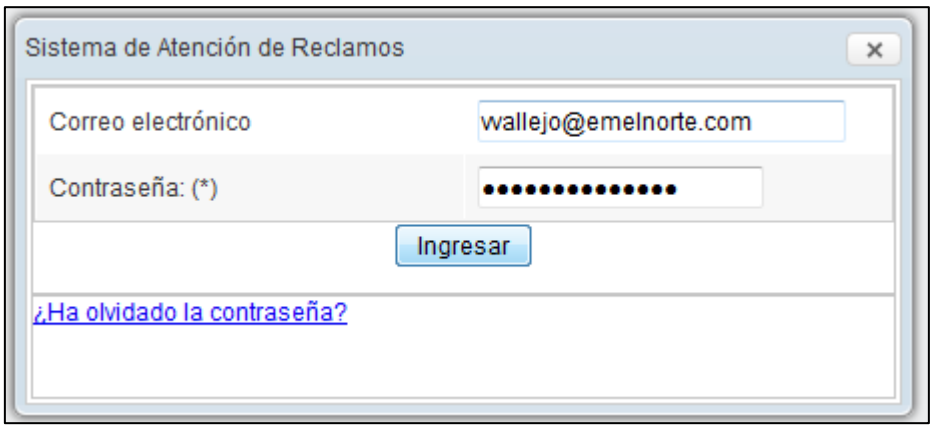

Figura 4 - 12 Pantalla de ingreso al SAR (empleados de la empresa)

En caso de ser el primer el ingreso al SAR o previamente a pulsado “Ha olvidado la contraseña”, el sistema pedirá cambiar la contraseña en la siguiente pantalla.

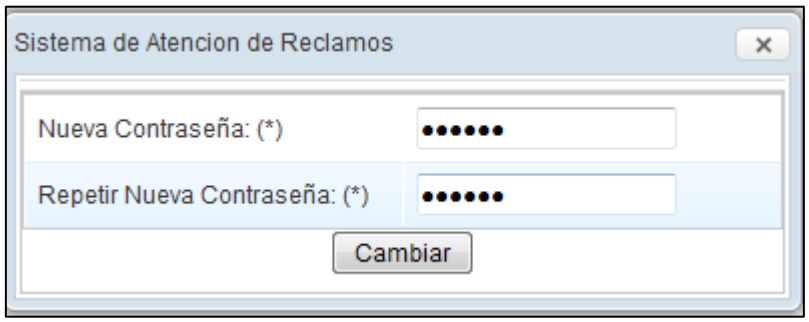

Figura 4 - 13 Pantalla cambio de contraseña (empleados de la empresa)

Cumplidos los pasos anteriores el usuario podrá acceder al SAR, y se le presentará la siguiente pantalla.

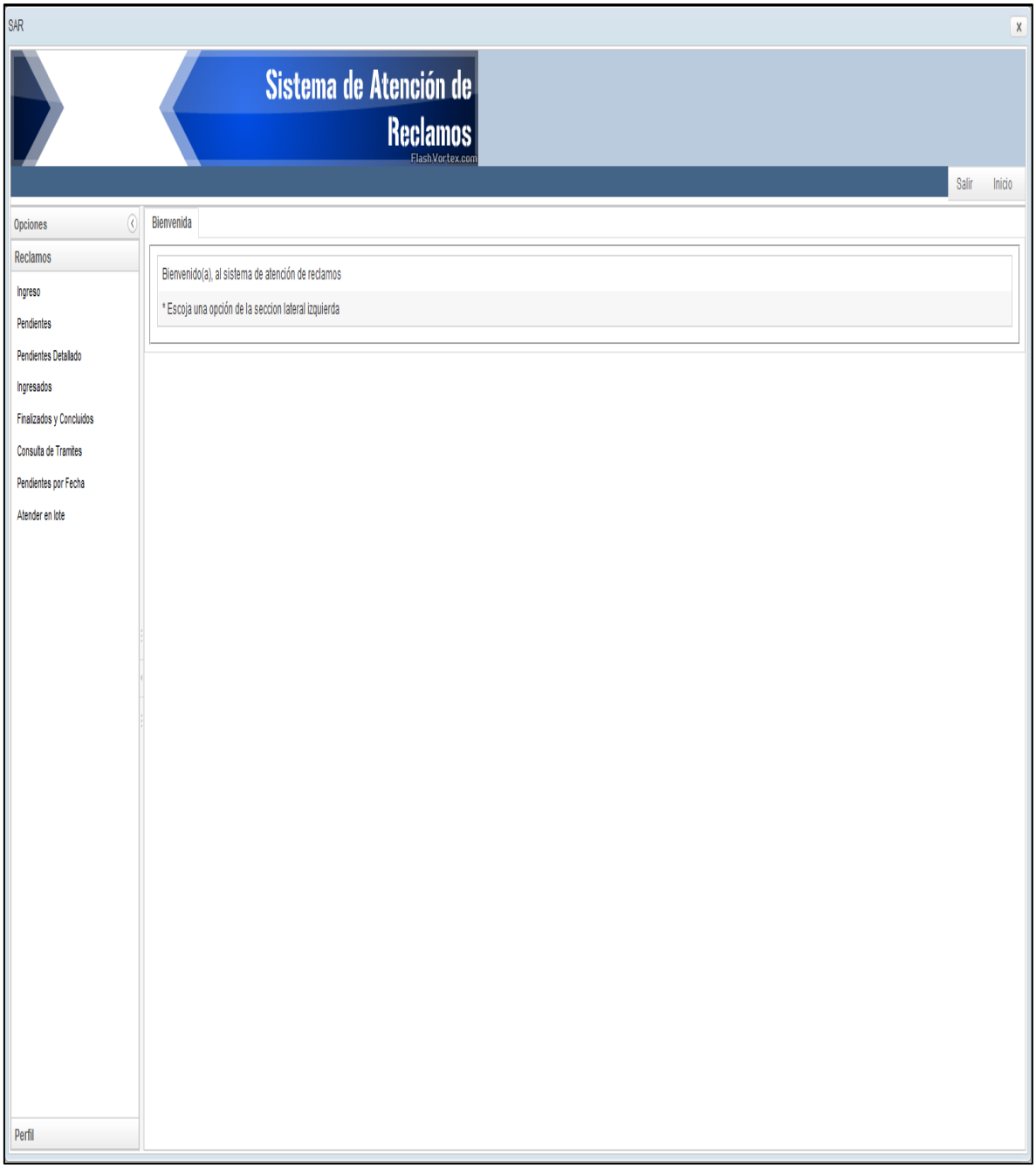

Figura 4 - 14 Pantalla principal del SAR

Las distintas funcionalidades de cada opción se encuentran especificadas en los respectivos casos de uso, que es información proporcionada por este documento.

Para acceder al sistema por parte de los clientes de la empresa, se debe acceder al siguiente link:

http://www2.emelnorte.com:8080/reclamos_clientes/

El cual deberán ingresar los siguientes datos: número de suministro y número de identificación.

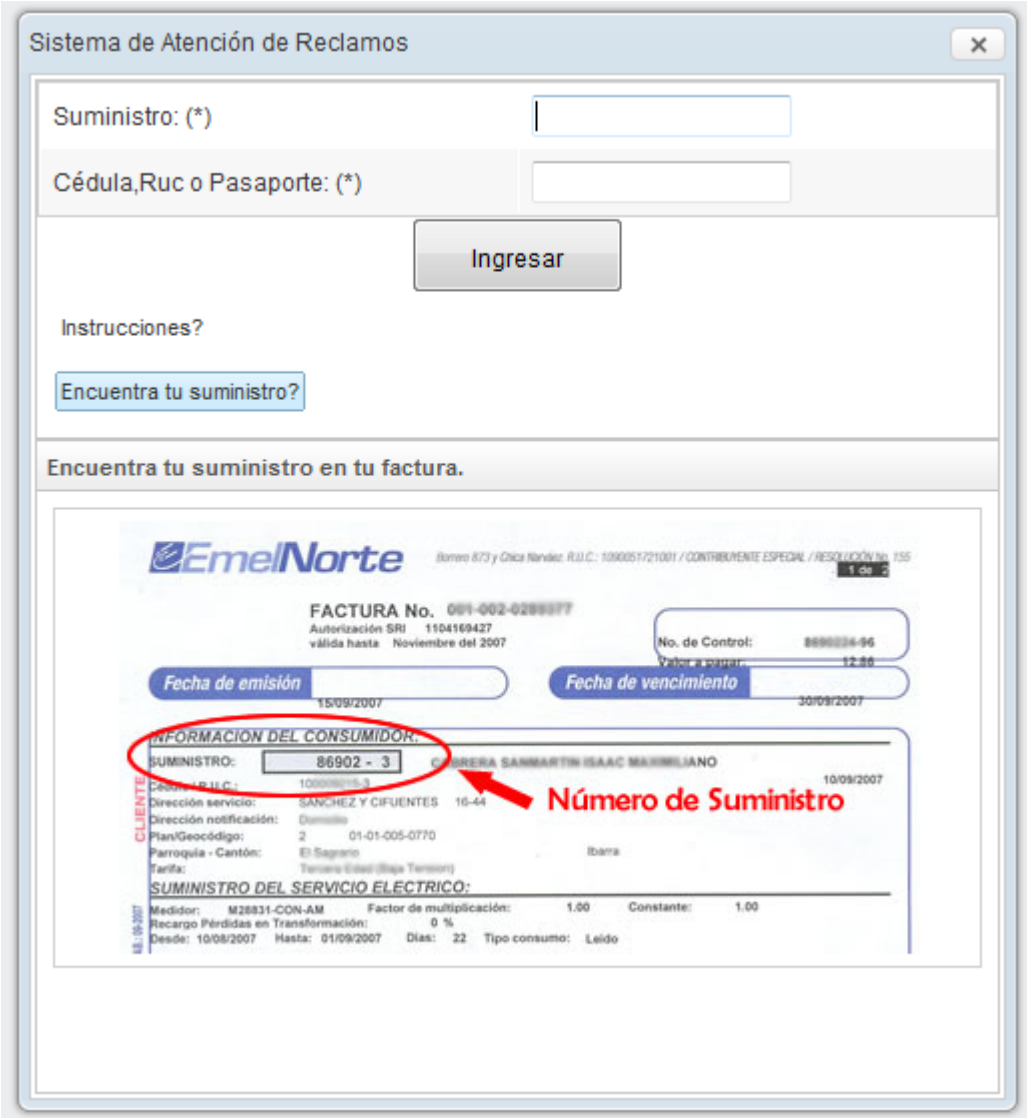

Figura 4 - 15 Pantalla de ingreso para cliente de la empresa

Si se encuentra el cliente en la base de datos del SAR, procede a ingresar a la siguiente pantalla.

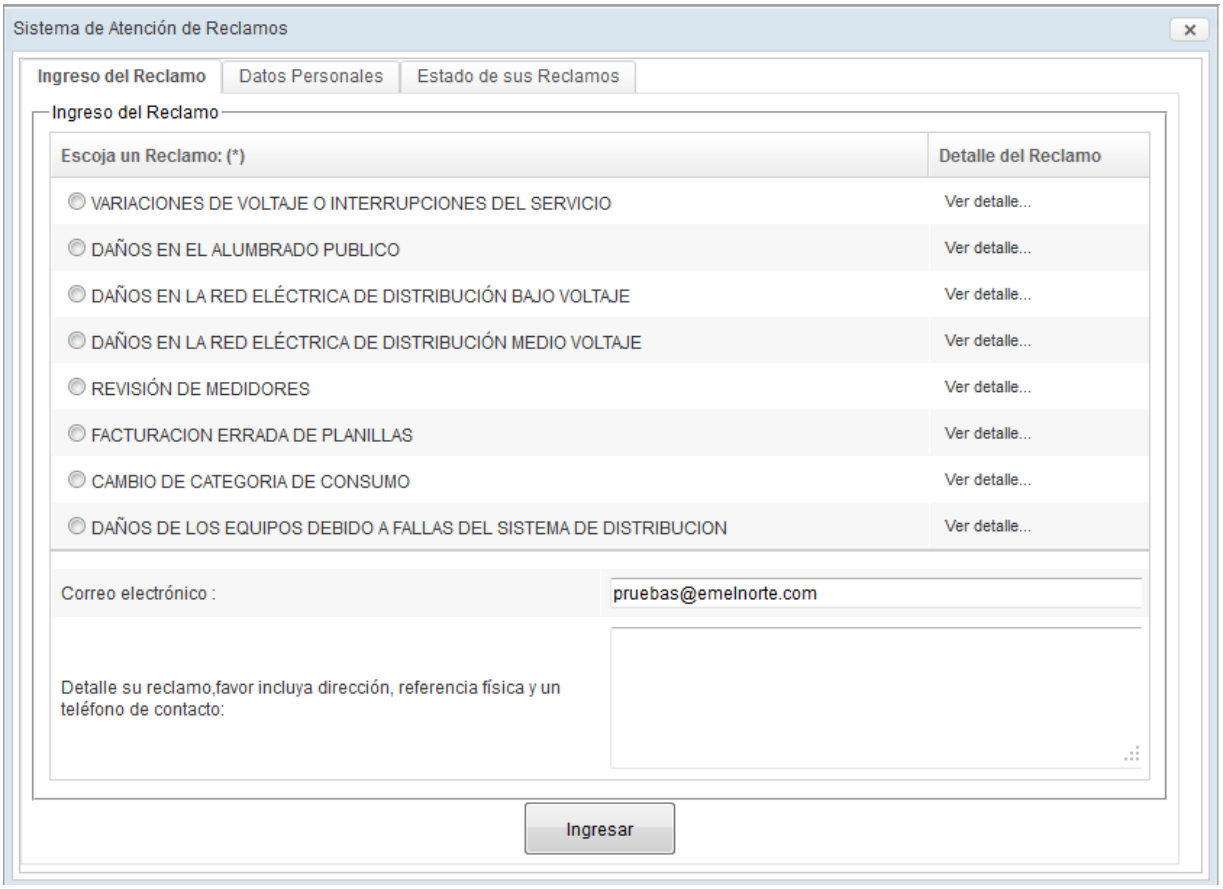

Figura 4 - 16 Pantalla principal del SAR para los clientes de la empresa.

Las distintas funcionalidades de cada opción se encuentran especificadas en los respectivos casos de uso, que es información proporcionada por este documento.