UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".

Presentado por:
Nexolec Del Valle Flores Sánchez

Trabajo de Grado como requisito parcial para optar al título de Ingeniero en Computación.

Puerto La Cruz, Agosto de 2010

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".

Asesor:

_____

Ing. Pedro Dorta
(Asesor Académico)

Puerto La Cruz, Agosto de 2010

UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".

Jurado Calificador:

_____	_____
Ing. José Bastardo	Ing. Manuel Carrasquero
(Jurado)	(Jurado)

_____

Ing. Pedro Dorta
(Asesor Académico)

Puerto La Cruz, Agosto de 2010

RESOLUCIÓN

De acuerdo al Artículo 41 del Reglamento de Trabajo de Grado:

"Los Trabajos de Grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados para fines con el consentimiento del Concejo de Núcleo respectivo, el cual lo participará al Concejo Universitario".

RESUMEN

La Coordinación IS de PDVSA PETROCEDEÑO, se encarga de solucionar las necesidades y/o fallas que se generan en las Aplicaciones existentes en el Mejorador. Para resolver las fallas los Analistas aplican sus conocimientos y conservan un soporte de la implementación que realizaron, y así emplearlo cuando vuelva a ocurrir la misma falla. Los soportes se guardan en físico o en el correo electrónico, lo que resulta incómodo a la hora de buscarlos. Con la finalidad de contar con una herramienta de trabajo en la cual se pueda documentar las fallas generadas en las aplicaciones, la Coordinación IS, plantea la necesidad de una aplicación que cubra sus requerimientos, por lo cual nace la siguiente propuesta: **"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".** Éste Software brindará comodidad, efectividad y seguridad al buscar información, además estadísticas sobre las fallas. Para diseñar el Software se utilizó el Proceso Unificado, UML y Webml. Se implementó SQL SERVER 2000 como sistema manejador de Base de Datos, para la codificación se usó Visual Basic .Net 2003, XML, XSLT.

AGRADECIMIENTOS

Ante todo le doy gracias a Dios por ser tan bondadoso y maravilloso conmigo, además permitirme lograr la meta de culminar mi tesis de grado.

A mi madre Berenice Sánchez, el regalo más hermoso que me ha dado Dios. Gracias mami por ser tan especial conmigo, por apoyarme en todo momento, brindarme educación e impulsarme para luchar y alcanzar mis objetivos.

A mi padre Hernán Flores, otro regalo que Dios me ha dado. Gracias papi por impulsarme siempre a seguir adelante y por el apoyo que me has dado.

A mis hermanos Hernis, Hernesto, Hernelis. Gracias por su preocupación de que culminara mi tesis de grado y por animarme a esforzarme para lograrlo. Y al travieso de mi hermanito Hernán José, gracias corazón por alegrarme la vida.

A todos mis tíos Luz Estela, Carlos, Jesús, Silvina, Yrelis, Anarelis, Ysaura, y a mis abuelas Yrene y Silvina. Gracias a todos por su preocupación, consejos y por animarme a conseguir la culminación de mi tesis de grado.

Al señor Mario Ordosgoitti, gracias por estar pendiente de que realizara mi tesis de grado y por su apoyo.

Al señor Michel Treier, excelente persona de gran calidad humana. Gracias por darme la oportunidad de realizar mi pasantía y por el apoyo que me ha brindado.

A Grace López, gracias por las valiosas horas que me brindaste en PDVSA PETROCEDEÑO, para guiarme con tus conocimientos durante el desarrollo del sistema REFAP

A Luis Carlos Guevara, quien considero un amigo. Gracias por los conocimientos que me aportaste y por estar pendiente de que terminara mi tesis de grado.

A mi asesor académico Pedro Dorta, gracias por guiarme y por los conocimientos que me aportaste en la realización de mi tesis de grado.

A mis primos, que no nombraré aquí ya que es una lista muy larga; pero mis agradecimientos a todos por su preocupación y por estar al pendiente de que yo realizara mi tesis de grado.

Son muchas las personas que de una u otra forma me aportaron un granito de arena para que mi sueño de terminar mi tesis de grado y así obtener mi título a nivel universitario, se cumpliera. Estas personas son familiares, amigas, amigos, compañeros de estudio, compañeros de pasantías, compañeros de trabajo, vecinos. En fin mucha gente que no nombraré aquí para no cometer el error de omitir alguna y además la lista es demasiada larga; pero gracias a todos.

A todos los profesores que me impartieron sus conocimientos en la Universidad de Oriente. Gracias por que sin duda alguna ustedes me aportaron las herramientas necesarias para lograr convertirme en un profesional.

Que Dios los proteja y bendiga siempre a todos.

DEDICATORIA

El presente Trabajo de Grado se lo dedico a:

Dios, mi creador.

Mis padres Berenice Sánchez y Hernán Flores, que a través de ellos Dios me permitió venir a este mundo.

Mis hermanos Hernis, Hernesto, Hernelis, Hernán, quienes forman una parte muy hermosa de mi vida.

Mis tíos, Luz Estela, Carlos, Jesús, Silvina, Yrelis, Anarelis, Ysaura, y a mis abuelas Yrene y Silvina, quienes siempre están pendientes de mi.

Señor Michel Treier, a quien admiro porque es un gran ser humano.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

Nunca dudes cuando Dios demore en darte lo que le has pedido. Él sólo pone a prueba tu paciencia; porque la Espina de hoy, será la flor de

CAPÍTULO 1

INTRODUCCIÓN

Con el propósito de llevar a cabo el desarrollo de las reservas de la Faja del Orinoco, las cuales se estiman en 235 mil millones de barriles de crudo extrapesado, el gobierno venezolano decidió que PDVSA se asociara con empresas extranjeras, estableciéndose en la década de los 90 Sincrudos de Oriente Sincor, C.A, conformada por la estatal venezolana PDVSA (38%), TOTAL de Francia (47%) y Statoil de Noruega (15%). En el año 2007 Sincrudos de Oriente Sincor, C.A, pasa a llamarse PDVSA PETROCEDEÑO, mediante un acuerdo entre sus socios para conformar una empresa mixta, cambiando su composición accionaría: PDVSA (60%), TOTAL (30.3%) y Statoil (9.7%).

PDVSA PETROCEDEÑO, se encarga de la exploración, extracción, producción, mejoramiento y comercialización de crudo. Su producción es de 200 MBD de crudo extrapesado de 8° API proveniente de la Faja del Orinoco, y los mejora en 180 MBD de Zuata Sweet, un crudo liviano y dulce de 30-32° API. Mediante el proceso de mejoramiento, se obtienen 900 toneladas de azufre y 6 mil toneladas de coque diariamente, que son colocadas en los mercados internacionales.

PDVSA PETROCEDEÑO, se encuentra localizada en el Complejo Petrolero, Petroquímico e Industrial José Antonio Anzoátegui; sector Jose entre Barcelona y Píritu, tal como se muestra en la figura 1.1.

Figura 1.1: Ubicación geográfica de PDVSA PETROCEDEÑO
Fuente: PDVSA PETROCEDEÑO, 2007

La misión de PDVSA PETROCEDEÑO, es maximizar la manufactura de crudo sintético "Zuata Sweet", agregando valor para los accionistas, a través de la optimización del procesamiento de crudo extrapesado, el cumplimiento de la legislación vigente en materia de Seguridad, Salud y Ambiente, el uso confiable de sus instalaciones y el mejoramiento continuo de sus procesos, apoyando estrategias comerciales corporativas, incrementando el desarrollo del personal, promoviendo alianzas con los proveedores y desarrollando una conciencia social en las comunidades vecinas.

Su visión es ser reconocido como líder en la manufactura de Crudo Sintético marcador a nivel mundial, maximizando el valor agregado a los accionistas, desarrollando sus operaciones en forma segura y armónica con el ambiente, propiciando el desarrollo sustentable de la Zona Norte de Anzoátegui, con base en una organización flexible, con índices de desempeño que sean referencia internacional y con personal altamente calificado, motivado, e identificado con los Valores de PDVSA PETROCEDEÑO.

PDVSA PETROCEDEÑO, se traza los siguientes Objetivos:

- Maximizar la manufactura de Crudo Sintético, “Zuata Sweet”, con el óptimo procesamiento de crudo extrapesado.
- Optimizar el uso y controlar los recursos de sus instalaciones, manteniendo la continuidad operativa, a través del mejoramiento continuo de los procesos.
- Apoyar estrategias comerciales corporativas.
- Cumplir con la legislación vigente y normativa interna, en materia de Seguridad, Salud y Ambiente.
- Propiciar la confianza del entorno (Zona Norte del estado Anzoátegui), por medio del apoyo de proyectos concretos a entes públicos y privados.
- Contribuir al desarrollo del personal, a través de Programas de Entrenamiento, Desarrollo y Reconocimiento.
- Adaptar la organización en función de los cambios del entorno.
- Promover alianzas estratégicas con proveedores de productos y servicios.
- Implantar sistemas de gestión que apoyen el desempeño del negocio.
- Cerrar administrativamente el proyecto.

1.1 PLANTEAMIENTO DEL PROBLEMA

El Mejorador PDVSA PETROCEDEÑO se encuentra estructurado por 11 gerencias que son: Gerencia de Mejoramiento, Proyecto SION y Parada, Operaciones, Servicios Técnicos, Mantenimiento, Recursos Humanos, Logística y Contratos, Seguridad/Higiene y Ambiente, proyectos, Ejecución, Ejecución de Mantenimiento.

La Gerencia de Servicios Técnicos tiene como función dirigir, planificar y proveer asistencia técnica a los departamentos de Operaciones, Mantenimiento y Planificación y Optimización mediante diagnósticos oportunos de desviaciones, implantación efectiva de soluciones a problemas operacionales a través de herramientas como la tecnología de procesos, economía, calidad de productos y tecnología de información. Ésta tiene como objetivo principal cumplir satisfactoriamente los requerimientos de los clientes funcionales del Mejorador para lograr el funcionamiento óptimo del Mejorador. La Gerencia también provee soporte a la División Técnica Corporativa cuando acometen proyectos para el Mejorador.

La Gerencia de Servicios Técnicos cuenta con 4 superintendencias que son: Superintendencia de Ingeniería de Procesos, Laboratorio, Inspección, e (IS, IT & TELECOM), en esta última se llevará a cabo este proyecto, específicamente en la Coordinación de Sistemas de Información (Coordinación IS).

La Superintendencia de IS/IT & Telecom apoya a todas las organizaciones del Mejorador en las áreas de:

- Red de computadoras (e-mail, Internet, hardware).
- Sistemas de computación como por ejemplo: Sistemas de Información, Transmisión de data, Reportes y Balances Diarios.
- Telecomunicaciones (Telefonía, Radios, Celulares).

La misión de la Superintendencia de IS/IT & Telecom es proveer soluciones tecnológicas de vanguardia, así como un eficiente y eficaz soporte en Tecnologías de Información (IT), Sistemas de Información (IS) y Telecomunicaciones, conformando un excelente equipo de especialistas en cada una de las áreas.

Su visión es mantenerse como el principal soporte tecnológico y constituirse como un pilar fundamental para la ejecución de las actividades medulares de la empresa.

Una de las funciones que desempeña el equipo de trabajo de la Coordinación IS, es darle soluciones a los usuarios que reportan nuevas necesidades y/o fallas que se generan en las diferentes Aplicaciones existentes en el Mejorador. Las Aplicaciones que se manejan en la Coordinación IS son distribuidas entre los integrantes del grupo IS, siendo cada uno de ellos el responsable de solucionar las fallas que se produzcan en las aplicaciones que se le han asignado.

La estrategia utilizada por los integrantes del equipo IS hasta ahora para resolver las fallas, ha resultado satisfactoria y consiste en aplicar sus conocimientos para solucionar las fallas producidas en alguna aplicación y conservar un soporte de la implementación que realizó, dicho soporte es utilizado al efectuarse la misma falla en la aplicación y de esa manera agilizar el proceso de solución de la falla debido a que ya tienen los pasos a seguir para resolver dicha falla. Los soportes que conservan los integrantes del equipo IS se encuentran algunos en físico y otros en el correo electrónico, a la hora de recurrir a dichos soportes, no resulta cómodo de parte de los integrantes del equipo IS, la búsqueda de la información requerida de una determinada falla.

Cuando se produce alguna falla en una aplicación y el encargado de dicha aplicación se encuentra ausente, existe la necesidad de recurrir al mismo para que le indique a otro integrante del equipo IS los pasos a emplear para solucionar la falla, ya

que los soportes que cada integrante del equipo IS guarda de la implementación que realiza cuando resuelve una falla no están a la disposición de todos los integrantes.

Con la finalidad de tener una herramienta de trabajo en el cual se pueda documentar las fallas que se generan en las aplicaciones, la Coordinación IS, plantea la necesidad de una aplicación que solvente sus requerimientos, por lo cual nace la siguiente propuesta: **"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".**

La aplicación a desarrollarse para solventar la problemática existente facilitará a los integrantes de la coordinación IS el ingreso de nuevas fallas, actualizar los datos, realizar consultas, imprimir reportes, poseer conocimientos estadísticos de las veces que se producen las fallas en las aplicaciones, cuáles otras aplicaciones son afectadas al producirse una falla en una determinada aplicación, indicativo de cuáles aplicaciones deben ser revisadas con prioridad tomando en cuenta la cantidad de veces que se produce una falla y su respectivo nivel de criticidad en dicha aplicación, organizar y almacenar la data en una base de datos confiable.

Con la aplicación a desarrollarse, la Coordinación IS contará con un instrumento al alcance de todos sus integrantes, información automatizada y organizada en una sola herramienta, brindándoles comodidad en la búsqueda de la misma. Si se encuentra ausente uno de los integrantes del equipo IS no será necesario recurrir al mismo para poder solventar una falla.

Para el desarrollo del Software se adopta el Proceso Unificado de Desarrollo de Software mediante el cual se transforman los requisitos de un usuario en un software, además es necesario emplear los conocimientos de Base de Datos, Sistema de Gestión

de Base de Datos, Sistemas de Información, Ingeniería de Software Y Lenguaje de Modelado (U.M.L), para realizar el diseño y construcción del software.

Debido a que es una aplicación Web, se empleará Visual Studio.Net 2003 que es una herramienta definitiva para la rápida generación de aplicaciones Web ASP.NET a escala empresarial y aplicaciones de escritorio de alto rendimiento.

Se podrá interactuar con la aplicación desde cualquier ordenador conectado a Internet dentro del Mejorador PDVSA PETROCEDEÑO. Esto facilita el proceso de solventar las fallas, debido a que el acceso de la información es inmediato, disminuyendo el tiempo de espera por parte del usuario que reporte la falla.

La Base de Datos para almacenar toda la información que se utilizará en la aplicación se creará en SQL SERVER 2000, debido a que es un potente motor de bases de datos de alto rendimiento capaz de soportar millones de registros por tabla.

La generación de los reportes se realizará con tecnología XML y XSLT. XSLT es un lenguaje de programación de hoja de estilos para la transformación de documentos XML en otros documentos XML, HTML, XHTML, WML, en este caso se empleará XSLT para elaborar las plantillas de reportes y convertir el documento XML (que describe los datos) en documentos HTML (que muestran los datos).

La aplicación tendrá una interfaz gráfica amigable, permitiendo la fácil interactividad de los usuarios con la misma.

1.2 OBJETIVOS

1.2.1 Objetivo General

Desarrollar un software, para el registro de las fallas que se generan en las diferentes aplicaciones existentes en el Mejorador de PDVSA PETROCEDEÑO, que son reportadas a la coordinación IS.

1.2.2 Objetivos Específicos

- Determinar los requerimientos funcionales y analizarlos.
- Diseñar la estructura del Software a implementarse.
- Establecer el esquema de la base de datos y elaboración de la misma.
- Diseñar una interfaz amigable, considerando los requerimientos de los usuarios de la Coordinación IS.
- Realizar pruebas integrales al Software que garanticen su correcto funcionamiento, identificando y corrigiendo posibles fallas.
- Elaborar el manual de usuario y administración del sistema.

CAPÍTULO 2
MARCO TEÓRICO

En las tiniebla Dios es tu Luz., en la tormenta es tu Calma, en la soledad es tu Compañero y en el momento más difícil es tu Fortaleza.

CAPÍTULO 2

MARCO TEÓRICO

2.1 ANTECEDENTES

En la búsqueda de información que permita encaminar esta investigación, se encuentra que en la Universidad de Oriente Núcleo de Anzoátegui, se han presentado trabajos de grado que se enfocan en la automatización de la información, en los cuales utilizaron técnicas y metodologías que pueden contribuir al desarrollo de este proyecto.

A continuación citaré algunas referencias:

Calsadilla, A (2005) presentó un trabajo titulado: **"DISEÑO DE UN SISTEMA DE INFORMACIÓN PARA LA AUTOMATIZACIÓN DEL PROCESO DE ARCHIVO DE EXPEDIENTES DE LOS EMPLEADOS ADSCRITOS A EL DEPARTAMENTO DE INGENIERÍA DE MANTENIMIENTO DE LA DIVISIÓN PLANTA MACAGUA, C.V.G ELECTRIFICACIÓN DEL CARONÍ C.A. (EDELCA)".** Se utilizó la metodología orientada a objetos, la cual utiliza el Lenguaje de Modelado Unificado (UML), que les aportó la institución de las bases para la codificación y desarrollo de las aplicaciones utilizadas.

Acuña, S / Rosales, Z (2005) presentaron un trabajo titulado: **"DESARROLLO DE UN SISTEMA DE INFORMACIÓN PARA LAS ACTIVIDADES DE PRODUCCIÓN DE UNA EMPRESA CAFETALERA UBICADA EN BERGANTIN EDO-ANZOATEGUI."** Se empleó el Proceso

Unificado de Desarrollo de Software, Visual Basic 6.0, Access 2000 y Cristal Report, para el desarrollo del Sistema.

González, A González (2006) presentó un trabajo titulado: **"DISEÑO DE UN SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LAS FUNCIONES DE SUPERVISIÓN Y CONTROL DEL PRESUPUESTO DEL CONSEJO LEGISLATIVO DEL ESTADO ANZOÁTEGUI."** Se realizó un estudio siguiendo la metodología para el diseño de Sistemas de Información basado en el estudio de Sistemas Blandos, y se empleó el Lenguaje de Modelado Unificado (UML) y la notación UML para diseñar el Sistema**.**

Marcano, A (2007) presentó un trabajo titulado: **"DESARROLLO DE UN SOFTWARE DE GESTIÓN Y CONTROL DE DEMANDAS PARA LA CONSULTORÍA JURÍDICA DE UNA EMPRESA DE TELECOMUNICACIONES. "** Se emplearon metodologías del proceso unificado de desarrollo de software con el fin de producir software de calidad y diagramas UML que permitieron comunicar las ideas a los participantes del proyecto.

Medina, J (2007) presentó un trabajo titulado: **"DESARROLLO DE UN SISTEMA BASADO EN APLICACIONES WEB PARA LA AUTOMATIZACIÓN DEL CONTROL DE PEDIDOS ASOCIADOS AL PROCESO DE VENTAS DE UNA EMPRESA CAFETALERA."** Se usaron las cuatro fases del Proceso Unificado de Desarrollo de Software, permitiendo optimizar las funciones del Departamento de Ventas, Facturación y Administración de la Empresa.

2.2 FUNDAMENTOS TEÓRICOS

2.2.1 Base de datos

Una base de datos es una colección de datos organizados y estructurados según un determinado modelo de información que refleja no sólo los datos en sí mismos, sino también las relaciones que existen entre ellos. Una base de datos se diseña con un propósito específico y debe ser organizada con una lógica coherente. Los datos podrán ser compartidos por distintos usuarios y aplicaciones, pero deben conservar su integridad y seguridad al margen de las interacciones de ambos. La definición y descripción de los datos han de ser únicas para minimizar la redundancia y maximizar la independencia en su utilización.

En una base de datos, las entidades y atributos del mundo real, se convierten en registros y campos. Estas entidades pueden ser tanto objetos materiales como libros o fotografías, pero también personas e, incluso, conceptos e ideas abstractas. Las entidades poseen atributos y mantienen relaciones entre ellas.

2.2.1.1 Tipos de base de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:

Según la variabilidad de los datos almacenados:

- **Bases de datos estáticas**

  Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

- **Bases de datos dinámicas**

  Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

Según el contenido:

- **Bases de datos bibliográficas**

  Solo contienen un surrogante (representante) de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque sino estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

- **Bases de datos de texto completo**

  Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

- **Directorios**

  Un ejemplo son las guías telefónicas en formato electrónico.

- **Bases de datos o "bibliotecas" de información Biológica**

  Son bases de datos que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

  - Aquellas que almacenan secuencias de nucleótidos o proteínas.
  - Las bases de datos de rutas metabólicas

- Bases de datos de estructura, comprende los registros de datos experimentales sobre estructuras 3D de biomoléculas
- Bases de datos clínicas
- Bases de datos bibliográficas (biológicas)

2.2.1.2 Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

- **Bases de datos jerárquicas**

  Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

  Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

  Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

- **Base de datos de red.**

  Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

  Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

- **Base de datos relacional.**

  Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

  En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

- **Bases de datos orientadas a objetos.**

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulación:** Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia:** Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo:** Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

- **Bases de datos documentales.**

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes. Tesauros es un sistema de índices optimizado para este tipo de bases de datos.

- **Base de datos deductivas.**

  Un sistema de base de datos deductivos, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática.

- **Gestión de bases de datos distribuida.**

  La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades, sucursales de tiendas, entre otros.

2.2.2 Sistema de gestión de base de datos

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Propósito:

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos.

Objetivos:

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información:** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una

base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Redundancia mínima:** Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia:** En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad:** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo y recuperación:** Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en

ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Control de la concurrencia:** En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- **Tiempo de respuesta:** Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Ventajas:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- Integridad referencial el terminar los registros.

Inconvenientes:

- El costo de actualización del hardware y software son muy elevados.
- Costo (salario) del administrador de la base de datos es costoso.
- El mal diseño de esta puede originar problemas a futuro.
- Un mal adiestramiento a los usuarios puede originar problemas a futuro.
- Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.

- Generan campos vacíos en exceso.
- El mal diseño de seguridad genera problemas en esta.

2.2.3 Sistema de información

Un sistema de información se define como un conjunto de funciones o componentes interrelacionados que forman un todo, es decir, obtiene, procesa, almacena y distribuye información (datos manipulados) para apoyar la toma de decisiones y el control en una organización. Igualmente apoya la coordinación, análisis de problemas, visualización de aspectos complejos, entre otros aspectos.

Un sistema de información contiene información de sus procesos y su entorno. Como actividades básicas producen la información que se necesita: entrada, procesamiento y salida. La retroalimentación consiste en entradas devueltas para ser evaluadas y perfeccionadas.

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.

- **Entrada de Información:** Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfases automáticas.

  Las unidades típicas de entrada de datos a las computadoras son las terminales, las cintas magnéticas, las unidades de diskette, los códigos de barras, los escáner, la voz, los monitores sensibles al tacto, el teclado y el Mouse, entre otras.

- **Almacenamiento de información:** El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada

en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos. La unidad típica de almacenamiento son los discos magnéticos o discos duros, los discos flexibles o diskettes y los discos compactos (CD-ROM).

- **Procesamiento de Información:** Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general de un año base.
- **Salida de Información:** La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes, cintas magnéticas, la voz, los graficadores y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo. En este caso, también existe una interfase automática de salida. Por ejemplo, el Sistema de Control de Clientes tiene una interfase automática de salida con el Sistema de Contabilidad, ya que genera las pólizas contables de los movimientos procesales de los clientes.

2.2.4 Ingeniería de software

La Ingeniería de Software es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias

computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos.

La ingeniería de software, como las disciplinas tradicionales de ingeniería, tiene que ver con el costo y la confiabilidad. Algunas aplicaciones de software contienen millones de líneas de código que se espera que se desempeñen bien en condiciones siempre cambiantes.

La ingeniería de software se puede considerar como la ingeniería aplicada al software, esto es en base a herramientas preestablecidas, la aplicación de las mismas de la forma más eficiente y óptima; objetivos que siempre busca la ingeniería. No es solo de la resolución de problemas, sino más bien teniendo en cuenta las diferentes soluciones, elegir la más apropiada.

Software

El software es el conjunto de instrucciones que permite al hardware de la computadora desempeñar trabajo útil. En las últimas décadas del siglo XX, las reducciones de costo en hardware llevaron a que el software fuera un componente ubicuo de los dispositivos usados por las sociedades industrializadas.

Metodología

Un objetivo de décadas ha sido el encontrar procesos o metodologías predecibles y repetibles que mejoren la productividad y la calidad.

Pasos del proceso

La ingeniería de software requiere llevar a cabo muchas tareas, sobre todo las siguientes:

- **Análisis de requisitos.**

  Extraer los requisitos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y experiencia en la ingeniería de software para

reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del análisis de requisitos con el cliente se plasma en el documento SRS, Software Requirements Specification, cuya estructura puede venir definida por varios estándares, tales como el de la IEEE. Así mismo, se define un diagrama de Entidad/Relación, en el que se plasman las principales entidades que participarán en el desarrollo del software

- **Especificación.**

  Es la tarea de describir detalladamente el software a ser escrito, en una forma matemáticamente rigurosa. En la realidad, la mayoría de las buenas especificaciones han sido escritas para entender y afinar aplicaciones que ya estaban desarrolladas. Las especificaciones son más importantes para las interfaces externas, que deben permanecer estables.

- **Diseño y arquitectura.**

  Se refiere a determinar como funcionará de forma general sin entrar en detalles. Se definen los casos de uso para cubrir las funciones que realizará el sistema, y se transforman las entidades definidas en el análisis de requisitos en clases de diseño, obteniendo un modelo cercano a la programación orientada a objetos.

- **Programación.**

  Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no es necesariamente la porción más larga.

- **Prueba.**

  Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral.

- **Documentación.**

  Realización del manual de usuario, y posiblemente un manual técnico con el propósito de mantenimiento futuro y ampliaciones al sistema.

- **Mantenimiento.**

  Mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar mantenimiento. Una pequeña parte de este trabajo consiste en arreglar errores, o bugs. La mayor parte consiste en extender el sistema para hacer nuevas cosas.

2.2.5 Desarrollo de software

Es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo"

La ingeniería de software tiene varios modelos o paradigmas de desarrollo en los cuales se puede apoyar para la realización de software, de los cuales podemos destacar a éstos por ser los más utilizados y los más completos:

- Modelo en cascada (ciclo de vida clásico)
- Modelo en espiral
- Modelo de prototipos
- Método en V

2.2.6 Principios de modelado

En cualquier proyecto de ingeniería como la construcción de un gran edificio, un avión, una represa hidroeléctrica, la construcción de un procesador de textos o un software de comunicaciones para Internet, requieren de etapas de modelamiento que permitan experimentar y visualizar el sistema que se construirá. De la experiencia en ingeniería se extractan los siguientes principios de modelado:

a) **La forma como vemos el problema tiene una profunda influencia en forma como acometemos el problema y le damos solución al mismo**.

Si pensamos que el mundo esta compuesto de clases (Abstracciones de la realidad y de la solución del problema) y objetos (instancias de éstas abstracciones) que interactúan entre si para realizar una funcionalidad, así veremos el mundo. Este es precisamente al paradigma a que le apuesta UML: el modelo orientado a objetos. Si vemos la realidad como compuesta de procesos donde cada uno a su vez se puede descomponer en subprocesos entonces estamos concibiendo la realidad según el modelo estructurado y la arquitectura del sistema en desarrollo estará conformada de programas y subprogramas.

b) **Para modelar un sistema complejo no es suficiente un único modelo se requieren múltiples modelos donde cada uno representa una vista (aspecto) del sistema; estos modelos se complementan entre si.**

Esta es la razón de la existencia de varios diagramas en UML que modelan diferentes aspectos del sistema, desde las vistas lógicas y físicas del sistema hasta los aspectos dinámicos, estáticos y funcionales del mismo.

c) **Cualquier modelo puede ser representado con diferentes grados de precisión**.

La precisión se puede ver desde dos ópticas: La primera es el grado de detalle con que se representa un modelo; por ejemplo, si lo que se desea es razonar acerca de los requerimientos del sistema con un cliente o usuario final, se puede elaborar un diagrama de clases que muestra las clases, sus atributos y operaciones así como varios adornos(multiplicidad) en las relaciones; por otro lado, si lo que se desea es transmitir el diagrama de clases para que sea implementado en un DBMS (Data Base Management System, Sistema Administrador de Bases de Datos) por un programador, el diagrama con toda seguridad contendrá la visibilidad de las características (atributos y

operaciones) de las clases, los tipos de datos de los atributos y las signaturas de las métodos de las clases.

La segunda forma de ver la precisión de un modelo se refiere al nivel de abstracción, ese decir, a los detalles y la vista (porción del sistema o realidad) que presenta un modelo al lector; por ejemplo, en un sistema Bancario que maneja los retiros que hacen los clientes ya sea en un cajero automático o humano, el diagrama de clases contiene decenas de éstas; sin embargo las personas encargadas de desarrollar la interfaz de un cajero electrónico estarían interesadas en las clases necesarias para realizar el comportamiento del cajero y omiten el resto de clases del sistema.

d) **Los mejores Modelos están ligados a la realidad.**

El símbolo de un actor en un diagrama de casos de uso representa, de hecho, un actor en el sistema real; así como un componente en un diagrama de componentes representa un componente físico del software. Cada elemento de UML como una clase, objeto, estado, componente o nodo tiene su correspondencia con algún elemento conceptual o físico del mundo real.

2.2.7 Proceso unificado

Es un proceso de desarrollo de Software. Un proceso de desarrollo de Software es un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial del Proceso Unificado.

No obstante, los verdaderos aspectos definitivos del Proceso Unificado se resumen en tres fases claves: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

2.2.7.1 El proceso unificado está dirigido por casos de uso

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso, el cual describe la funcionalidad total del sistema. Puede decirse que una especificación funcional contesta a la pregunta: ¿Qué debe hacer el sistema? La estrategia de los casos de uso puede describirse añadiendo tres palabras al final de esta pregunta: ¿para cada usuario? Estas tres palabras albergan una implicación importante. Nos fuerzan a pensar en términos de importancia para el usuario y no sólo en términos de funciones que serían bueno tener. Sin embargo, los casos de uso no son sólo una herramienta para especificar los requisitos de un sistema. También guían, su diseño, implementación y prueba; estos guían el proceso de desarrollo. Basándose en le modelo de casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso. Los desarrolladores revisan cada uno de los sucesivos modelos para que sean conformes al modelo de casos de uso. Los ingenieros de prueba prueban la implementación para garantizar que los componentes del modelo de implementación implementan correctamente los casos de uso. De este modo, los casos de uso no sólo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Dirigidos por casos de uso quiere decir que el proceso de desarrollo sigue un hilo, avanza a través de una serie de flujos de trabajos que parten de los casos de uso. Los casos de uso se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba.

2.2.7.2 El proceso unificado está centrado en la arquitectura

El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como la perciben los usuarios y los inversores, y se refleja en los casos de uso. Sin embargo también se ve influenciada por muchos otros factores, como la plataforma en la que tiene que funcionar el software, los bloques de construcción reutilizables de que se dispone, consideraciones de implantación, sistemas heredados, y requisitos no funcionales. La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. Debido a que lo que es significativo depende en parte de una valoración, que a su vez, se adquiere con la experiencia, el valor de una arquitectura depende de las personas que se hayan responsabilizado de su creación. No obstante, el proceso ayuda al arquitecto a centrarse en los objetivos adecuados, como la comprensibilidad, la capacidad de adaptación al cambio y la reutilización.

Cada producto tiene tanto una función como una forma. Ninguna es suficiente por sí misma. Estas dos fuerzas deben equilibrarse para obtener un producto con éxito. En esta situación, la función corresponde a los casos de uso y la forma a la arquitectura. Debe haber interacción entre los casos de uso y la arquitectura. En realidad, tanto la arquitectura como los casos de uso deben evolucionaren paralelo.

Por tanto, los arquitectos moldean el sistema para darle una forma. Es esta forma, la arquitectura, la que debe diseñarse para permitir que el sistema evolucione, no sólo en su desarrollo inicial, sino también a lo largo de las futuras generaciones. Para encontrar esa forma, los arquitectos deben trabajar sobre la comprensión general de las funciones clave, es decir, sobre los casos de uso claves del sistema. Estos casos de uso clave pueden suponer solamente entre el 5 y el 10 por ciento de todos los casos de uso, pero son los significativos, los que constituyen las funciones fundamentales del sistema.

2.2.7.3 El proceso unificado es iterativo e incremental

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son miniproyectos.

Los desarrolladores basan la selección de lo que se implementará en una iteración en dos factores. En primer lugar, la iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora. En segundo lugar, la iteración trata los riesgos más importantes. Las iteraciones sucesivas se construyen sobre los artefactos de desarrollo tal como quedaron al final de la última iteración. Al ser miniproyectos, comienzan con los casos de uso y continúan a través del trabajo de desarrollo subsiguiente, análisis, diseño, implementación y prueba, que termina convirtiendo en código ejecutable los casos de uso que se desarrollaban en la iteración. Por supuesto, un incremento no necesariamente es aditivo. Especialmente en las primeras fases del ciclo de vida, los desarrolladores pueden tener que remplazar un diseño superficial por uno más tallado o sofisticado. En fases posteriores, los incrementos son típicamente aditivos.

Cada iteración tiene todo lo que tiene un proyecto de desarrollo de software, planificación, desarrollo de una serie de flujos de trabajo (Requisito, análisis, diseño, implementación y pruebas), y una preparación para la entrega. Las iteraciones se organizan dentro de cuatro fases (Inicio, Elaboración, Construcción y Transición). Aunque cada iteración discurre a lo largo de los flujos de trabajo, tienen énfasis diferentes en las distintas fases, como se muestra en la figura 2.1. Durante la fase de inicio y elaboración, la mayoría de los esfuerzos se dedica a la captura de requisitos y a un análisis y diseño preliminar. Durante la construcción el énfasis pasa al diseño

detallado, la implementación y la prueba. El número de iteraciones planteado para cada fase depende, básicamente de la complejidad del sistema propuesto. Un proyecto muy simple podría ser realizado con una sola iteración por fase, mientras que un proyecto más complicado podría requerir más iteraciones.

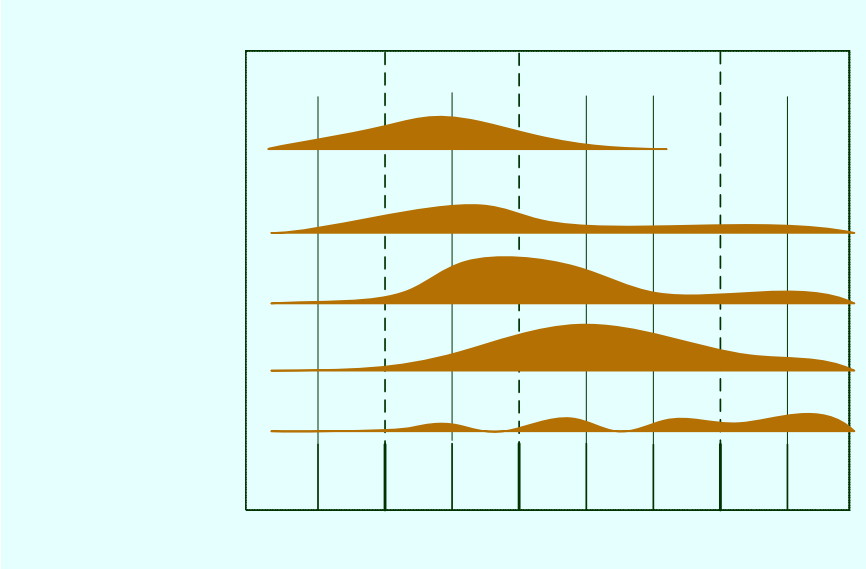


Figura 2.1: El énfasis se desplaza en las iteraciones. Desde la captura de los requisitos, el análisis hacía el diseño, la implementación y prueba.
Fuente: El Proceso Unificado de Desarrollo de Software. Primera Edición. Pearson Educación. Madrid (2000).

2.2.8 Fases del proceso unificado de desarrollo de software

- **Fase de inicio:** Se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto.

  Esencialmente, esta fase responde a las siguientes preguntas:

  - ¿Cuáles son las principales funciones del sistema para sus usuarios más importantes?
  - ¿Cómo podría ser la arquitectura del sistema?
  - ¿Cuál es el plan de proyecto y cuánto costará desarrollar el producto?

La respuesta a la primera pregunta se encuentra en un modelo simplificado que contenga los casos de usos más críticos. Cuando lo tengamos, la arquitectura es provisional y consiste típicamente en un simple esbozo que muestra los subsistemas más importantes. En esta fase, se identifican y priorizan los riesgos más importantes, se planifica en detalle de elaboración, y se estima el proyecto de manera aproximada.

- **Fase de elaboración:** Se identifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La relación entre la máquina del sistema y el propio sistema es primordial.

  Por tanto la arquitectura se expresa en forma de vistas de todos los modelos del sistema, los cuales juntos representan el sistema entero. Esto implica que hay vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y modelo de despliegue. La vista del modelo de implementación incluye componentes para probar que la arquitectura es ejecutable. Durante esta fase del desarrollo, se realizan los casos de uso más críticos que se identifican en la fase de comienzo. El resultado de esta fase es una línea base de la arquitectura. Al final de la fase de elaboración, el director de proyecto está en disposición de planificar las actividades y estimar los recursos necesarios para terminar el proyecto.

- **Fase de construcción:** Se crea el producto. Aquí la línea base de arquitectura crece hasta convertirse en el sistema completo. La descripción evoluciona hasta convertirse en un producto preparado para ser entregado a la comunidad de usuarios. El grueso de los recursos requeridos se emplea durante esta fase del desarrollo. Sin embargo la arquitectura del sistema es estable, aunque los desarrolladores pueden descubrir formas mejores de estructurar el sistema, ya que los arquitectos recibirán sugerencias de cambios arquitectónicos de menor importancia. Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión.

Sin embargo, puede que no esté completamente libre de defectos. Muchos de estos defectos se descubrirán y solucionarán durante la fase de transición. La pregunta decisiva es: ¿cubre el producto las necesidades de algunos usuarios de manera suficiente como para hacer una primera entrega?

- **Fase de transición**: Cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras sugeridas en una versión general dirigida a la totalidad de la comunidad de usuarios. La fase de transición conlleva actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren tras la entrega. El equipo de mantenimiento suele dividir esos defectos en dos categorías: los que tienen suficiente impacto en la operación para justificar una versión incrementada y los que pueden corregirse en la siguiente versión normal.

2.2.9 El lenguaje de modelado unificado UML

UML (Unified Modeling Languaje, Lenguaje Unificado de Construcción de modelo) se define como un lenguaje que permite especificar, visualizar y construcción los artefactos de los sistemas de software. Es un sistema rotacional (que entre otras cosas incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. UML nació como una notación estándar para la construcción de modelos orientados a objetos, que ayuda a crear un modelo con buen diseño.

Cuando se modela algo, se crea una simplificación de la realidad para comprender mejor el sistema que se está desarrollando. Con UML se construyen modelos a partir de bloque de construcción básica, tales como clases, interfaces, colaboraciones, componentes, nodos, dependencias, generalizaciones y asociaciones.

Los diagramas son los medios para ver estos bloques de construcción. Un diagrama es una presentación gráfica de un conjunto de elementos, que la mayoría de las veces se dibuja como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se utilizan para visualizar un sistema desde diferentes perspectivas. Como ningún sistema puede ser comprendido completamente desde una única perspectiva, UML define varios diagramas que permiten centrarse en diferentes aspectos del sistema independiente.

Cuando modelan sistemas reales, sea cual sea el dominio del problema, muchas veces se dibujan los mismos tipos de diagramas, porque representan vistas comunes.

Normalmente, las partes estáticas de un sistema se representarán mediante uno de los cuatros diagramas siguientes: Diagramas de clases, Diagramas de despliegue, Diagramas de componentes, Diagramas de objetos.

A menudo se emplean cinco diagramas adicionales para ver las partes dinámicas de un sistema: Diagramas de casos de uso, Diagramas de secuencia, Diagramas de colaboración, Diagramas de actividades y Diagramas de componentes.

- **Diagramas de clases:** Representa un conjunto de clases, interfaces y colaboraciones, y las relaciones entre ellas. Los diagramas de clases son los diagramas más comunes en el modelado de sistemas orientados a objetos. Los diagramas de clases que incluyen clases para cubrir la vista de procesos estática de un sistema.
- **Diagramas de despliegue:** Muestra un conjunto de nodos y sus relaciones. Los diagramas de despliegue se utilizan para describir la vista de despliegue estática de una arquitectura. Los diagramas de despliegue se racionan con los diagramas de componentes en que un nodo normalmente incluye uno o más componentes.
- **Diagramas de componentes:** Muestra un conjunto de componentes y sus relaciones. Los diagramas de componentes se utilizan para describir la vista de implementación estática de un sistema. Los diagramas de componentes se

relacionan con los diagramas de clases en que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones.

- **Diagramas de casos de uso:** Representa un conjunto de casos de uso y actores (un tipo especial de clases) y sus relaciones. Los diagramas de casos de uso se utilizan para describir la vista de casos de uso estática de un sistema. Los diagramas de casos de uso son especialmente importantes para organizar y modelar al comportamiento de un sistema.
- **Diagramas de colaboración:** Es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes. Un diagrama de colaboración muestra un conjunto de objetos, enlaces entre esos objetos y mensajes enviados y recibidos por esos objetos. Los objetos normalmente son instancias con nombre o anónimas de clases, pero también pueden representar instancias de otros elementos, como colaboraciones, componentes y nodos. Los diagramas de colaboración se utilizan para describir la vista dinámica de un sistema.
- **Diagramas de secuencia:** Es un diagrama de interacción que resalta la ordenación temporal de los mensajes. Un diagrama de secuencia presenta un conjunto de objetos y los mensajes enviados y recibidos por ellos. Los objetos suelen ser instancias con nombres o anónimas de clases, pero también pueden representar instancias de otros elementos, como colaboraciones, componentes y nodos. Los diagramas de secuencia se utilizan para describir la vista dinámica de un sistema.

2.2.10 Visual Studio .net 2003

Visual Studio .NET es la herramienta definitiva para la rápida generación de aplicaciones Web ASP.NET a escala empresarial y aplicaciones de escritorio de alto rendimiento. Visual Studio incluye herramientas de desarrollo basadas en componentes, como Visual C#, Visual J#, Visual Basic y Visual C++, así como

diversas tecnologías suplementarias para simplificar el diseño, desarrollo e implementación en equipo de las soluciones.

Visual Studio admite el entorno de Microsoft .NET Framework, que ofrece Common Language Runtime y las clases de programación unificadas; ASP.NET utiliza estos componentes para crear aplicaciones Web ASP.NET y servicios Web XML. También incluye MSDN Library, que contiene toda la documentación de estas herramientas de programación.

2.2.11 Xml

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

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

2.2.12 Xslt

XSLT es un lenguaje de programación de hoja de estilos para la transformación de documentos XML en otros documentos XML, HTML, XHTML, WML o incluso PDF. XSLT permite:

- Formatear los elementos fuente basados en relaciones de ancestro/descendiente, posición y unicidad.
- La creación de construcciones de formato sofisticadas incluyendo texto generado e imágenes.
- La definición de macros de formateo reutilizables.

- Estilos independices de la dirección en que se escriba el lenguaje.
- Conjunto de objetos de formato extensible.
- Una manera de describir el proceso de transformación es decir que las XSLT transforma un árbol XML de entrada e otro árbol XML de salida.

2.2.13 SQL Server 2000

SQL Server 2000 es un potente motor de bases de datos de alto rendimiento capaz de soportar millones de registros por tabla con un interfaces intuitivo y con herramientas de desarrollo integradas como Visual Studio 6.0 o .NET, además incorpora un modelo de objetos totalmente programable (SQL-DMO) con el que podemos desarrollar cualquier aplicación que manipule componentes de SQL Server, es decir, hacer aplicación para crear bases de datos, tablas, DTS, backups, etc., todo lo que se puede hacer desde el administrador del SQL Server y podemos hacerlo no solo en Visual C++ sino también en Visual Basic, ASP y por supuesto en .NET.

Pero cuidado, que sea muy intuitivo en su administración o instalación no significa que sea fácil, una mala instalación, una base de datos mal creada o diseñada o una mala administración nos puede hacer la vida imposible y nuestras aplicaciones pueden tener un rendimiento malo, debemos tener cuidado y aprender a usarlo correctamente, como también es importante el hardware, lejos de los 64 MB mínimos que requiere el sistema es recomendable que tenga 256 o 512 para su buen funcionamiento y una cantidad suficiente de espacio en disco para que pueda trabajar con las bases de datos.

CAPÍTULO 3
FASE DE INICIO

Mientras más difícil se haga tu camino, Dios multiplicará tus fuerzas y mientras más fuerte se haga tus pruebas, Dios hará más grandes tus

CAPÍTULO 3

FASE DE INICIO

La fase de inicio es la primera en aplicarse al emplear el proceso unificado para desarrollar un software. Mediante la fase de inicio se capturan los requisitos del sistema, empleando métodos que permitan adquirir de los usuarios la información necesaria para enfocar una idea general del funcionamiento que se requiere tenga el sistema.

Cada fase del proceso Unificado pasa por una serie de flujos de trabajo: requisitos, análisis, diseño, implementación, prueba.

En la fase de inicio se capturan los requisitos en forma de casos de usos, luego se analiza y diseña el sistema para cumplir los casos de usos. En la figura 3.1 se puede apreciar que el rectángulo engloba los flujos de trabajos realizados en esta fase.

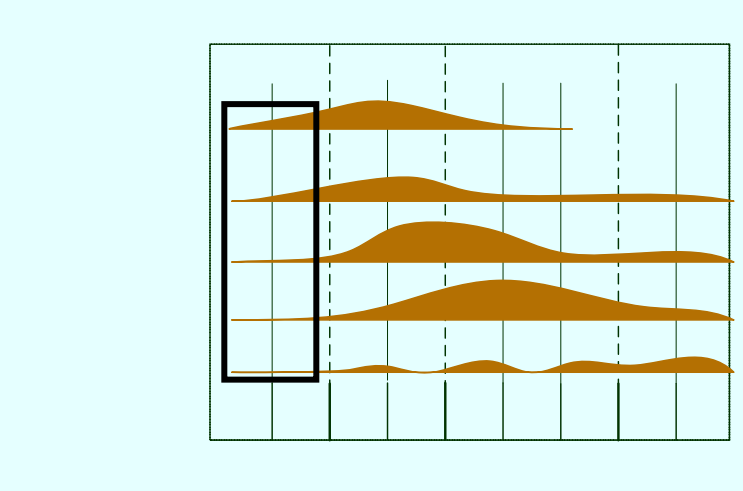


Figura 3.1: Flujos de trabajos de la fase de inicio
Fuente: El Proceso Unificado de Desarrollo de Software".
Primera Edición. Pearson Educación. Madrid (2000)

Básicamente en la fase de inicio se invierte la mayor parte del tiempo en capturar los requisitos del sistema para cumplir los objetivos de esta fase, es decir, delimitar el alcance y el ámbito del sistema propuesto; con el fin de comprender qué debe cubrir la arquitectura del sistema, definir los límites dentro de los cuales se deban buscar los riesgos críticos, delimitar las estimaciones de coste, agenda y recuperación de la inversión; para desarrollar el análisis de negocio.

3.1 REQUISITOS DEL SISTEMA

Los Requisitos del Sistema son las funciones que los usuarios necesitan que desempeñe el Sistema y las propiedades del mismo.

Por lo general no es fácil para el desarrollador del software capturar los requisitos, debido a que la mayoría de las veces la descripción que los usuarios hacen del producto deseado tiende a ser confusa. En esta situación los desarrolladores de software ponen en práctica los fundamentos del flujo de trabajo de los requisitos, para determinar lo que se debe construir y así satisfacer las necesidades del cliente de manera efectiva.

El flujo de trabajo de los requisitos comprende los siguientes pasos:

- Enumerar los requisitos candidatos.
- Comprender el contexto del Sistema.
- Identificar y describir los riesgos.
- Capturar los requisitos funcionales.
- Capturar los requisitos adicionales.

3.1.1 Enumerar los requisitos candidatos

Se dice que son requisitos candidatos aquellas ideas que surgen durante la vida de un sistema, por parte de los usuarios, clientes y desarrolladores; en torno al funcionamiento que se desea tenga el sistema.

Los requisitos candidatos del sistema a desarrollar son los siguientes:

- Elaboración del registro de fallas de forma automatizada.
- Indicación de estadísticas de las fallas producidas.
- Control de las fallas que atiende cada analista.
- Documentación de las fallas que se generan en las distintas aplicaciones.
- Indicación de los sistemas que son afectados por las fallas producidas.
- Manejo de solicitudes de consultas, y actualizaciones.
- Elaboración de reportes.

3.1.2 Comprender el contexto del Sistema

Es indispensable que el desarrollador del software conozca el entorno en que se desarrollará el sistema; por lo tanto se realizó un estudio del contexto en la Coordinación IS de PDVSA PETROCEDEÑO, con el propósito de conocer los procesos que se realizan en dicha Coordinación.

3.1.2.1 Modelo de Dominio

Describe los aspectos más importantes del contexto del sistema y lo representa como objetos o clases, los cuales aparecen como: objetos del negocio, objetos del mundo real y sucesos o eventos que ocurren en el entorno del trabajo del sistema.

El modelo de dominio se describe mediante diagramas de UML, especialmente mediante los diagramas de clases.

El diagrama de clase del dominio se elabora para establecer las relaciones entre las clases del dominio, bien sea de dependencia, herencia, asociación, agregación y composiciones que existen entre ellas. Este diagrama permite visualizar el comportamiento general del sistema en función de los objetos del dominio.

El modelo de dominio es un tipo de diagrama estático, lo más representativo posible al sistema a proponer, que describe la estructura de un sistema mostrando sus clases, las relaciones y la cardinalidad entre ellas

Al realizar el estudio del contexto del sistema en la Coordinación IS de PDVSA PETROCEDEÑO, se obtuvo una lista de clases de dominio. A continuación se realiza una explicación de las clases de dominio que forman parte del sistema:

- **Fallas_Aplicacion:** Registra toda la información concerniente a las fallas que se generan en las diferentes aplicaciones existentes en el Mejorador PDVSA PETROCEDEÑO.
- **Usuarios:** Esta clase es la superclase de las clases que representan los actores que normalmente harán uso del sistema en cuestión (Analista, Coordinador y Administrador).
- **Analista:** Corresponde a los integrantes de la Coordinación IS encargados de atender las fallas que se generan en las distintas aplicaciones que existen en PDVSA PETROCEDEÑO.
- **Coordinador:** Representa a un integrante de la Coordinación IS, que coordina a los Analistas.
- **Administrador:** Representa al usuario encargado de llevar el control de los datos estáticos del sistema.
- **Privilegio**: Almacena todos los módulos del sistema al cual puede acceder cada usuario (Analista, Coordinador, Administrador).
- **Modulos:** Almacena todos los datos concernientes a los módulos del sistema.
- **Aplicacion**: Esta clase encierra los datos de todas las aplicaciones existentes en el mejorador PDVSA PETROCEDEÑO.
- **Aplicfa**: Registra aquellas aplicaciones que son afectadas por una determinada falla.
- **Criticidad**: Se refiere al nivel de impacto que posee una determinada falla.

Una clase determinada puede encontrarse enlazada a otra distinta mediante relaciones, como por ejemplo entre las clases Usuarios y Analista, donde existe una relación denominada “de Herencia”, que se representa por una flecha (◁————). Esta relación se produce debido a que la clase principal o superclase, la cual es la señalada por la punta de la flecha (Usuarios), le transfiere sus propiedades a la subclase, que en este caso viene dada por la clase Analista, indicada por el otro extremo de la flecha. Esta relación es llamada también “es un tipo de”, por lo que se lee de manera que “un Analista es un tipo de Usuario”.

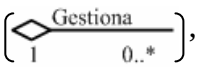
Otro tipo de relación es la que ocurre por ejemplo entre las clases Modulos y Privilegio, conocida como “de Dependencia”, la cual se denotada mediante una línea o flecha abierta y punteada (<- - - - - - -). Dicha relación es sinónimo de que una clase específica “utiliza” a otra clase determinada para que se realice un proceso en particular, por lo que un cambio en el elemento que se está utilizando, en este caso Modulos (clase que es apuntada por la flecha), afecta al elemento que lo utiliza, es decir, Privilegio (clase ubicada en el otro extremo de la flecha). Lo inverso no necesariamente es cierto. Partiendo de esto, un cambio en Modulos (por ejemplo, cambio del nombre de los módulos) afectará a la otra clase, ya que para que se puedan registrar los Privilegio de un usuario se requiere hacer uso de este dato en cuestión.

Existe la relación “de Asociación”, que conecta dos clases mediante una línea sólida entre ellas, acompañada de un nombre que indica la relación y de las cantidades de objetos que pueden ser conectados entre ellos, hecho conocido como cardinalidad o multiplicidad $\left(\overset{1}{\underset{}{\text{Gestiona}}}\overset{0..*}{\underset{}{}}\right)$.

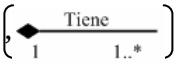
Esta relación “de Asociación” se encuentra, por ejemplo, entre las clases Usuarios y Fallas_Aplicacion, y entre las clases Usuarios y Privilegio, las cuales se leerían “un Usuario gestiona ninguna o muchas Fallas_Aplicacion” y “ un Usuario tiene uno o muchos Privilegio”.

La relación “de Agregación”, es una extensión de la relación de asociación, es decir, que se encuentra presente exactamente donde se encuentre cualquier relación de asociación, trabajando en conjunto con esta misma. Se conoce como una relación “todo-parte”. La clase en un extremo de la relación (el todo) contendrá las clases en el otro extremo de la relación (las partes). La relación “todo-parte” significa la relación “es parte de” o “tiene una”. En las relaciones de asociaciones explicadas en el párrafo anterior, se pueden leer, en función de la agregación, de la siguiente forma: “ninguna o muchas Fallas_Aplicacion son parte de un Usuario” o en su defecto “un Usuario tiene ninguna o muchas Fallas_Aplicacion”, y “un Usuario tiene uno o muchos Privilegio” o “uno o muchos Privilegio son parte de un Usuario”.

Existen dos formas de agregación, la simple y la de composición. En la agregación simple, el tiempo de vida del todo y las partes no están enlazados (si el todo deja de existir las partes siguen existiendo). En la de composición, si están enlazados (si el todo deja de existir, sus partes desaparecen automáticamente).

Entre las clases Usuarios y Fallas_Aplicacion, se presenta una agregación simple, ya que al eliminar del sistema un usuario, la Fallas_Aplicacion registrada por dicho usuario puede seguir existiendo en el sistema. Esta relación de agregación se muestra usando un diamante sin rellenar (

), ubicado cerca de la clase que forma el todo en la relación (Usuarios).

Entre las clases Usuarios y Privilegio, se puede observar una agregación de composición, ya que al eliminar del sistema un usuario, automáticamente desaparecen sus Privilegios. Esta relación se muestra como un diamante negro o relleno (

), que se coloca cerca de la clase que representa el todo en la relación (Usuarios).

Las relaciones entre las clases expuestas en los párrafos anteriores se pueden observar claramente en la figura 3.2, que muestra un diagrama de clases o modelo de dominio.

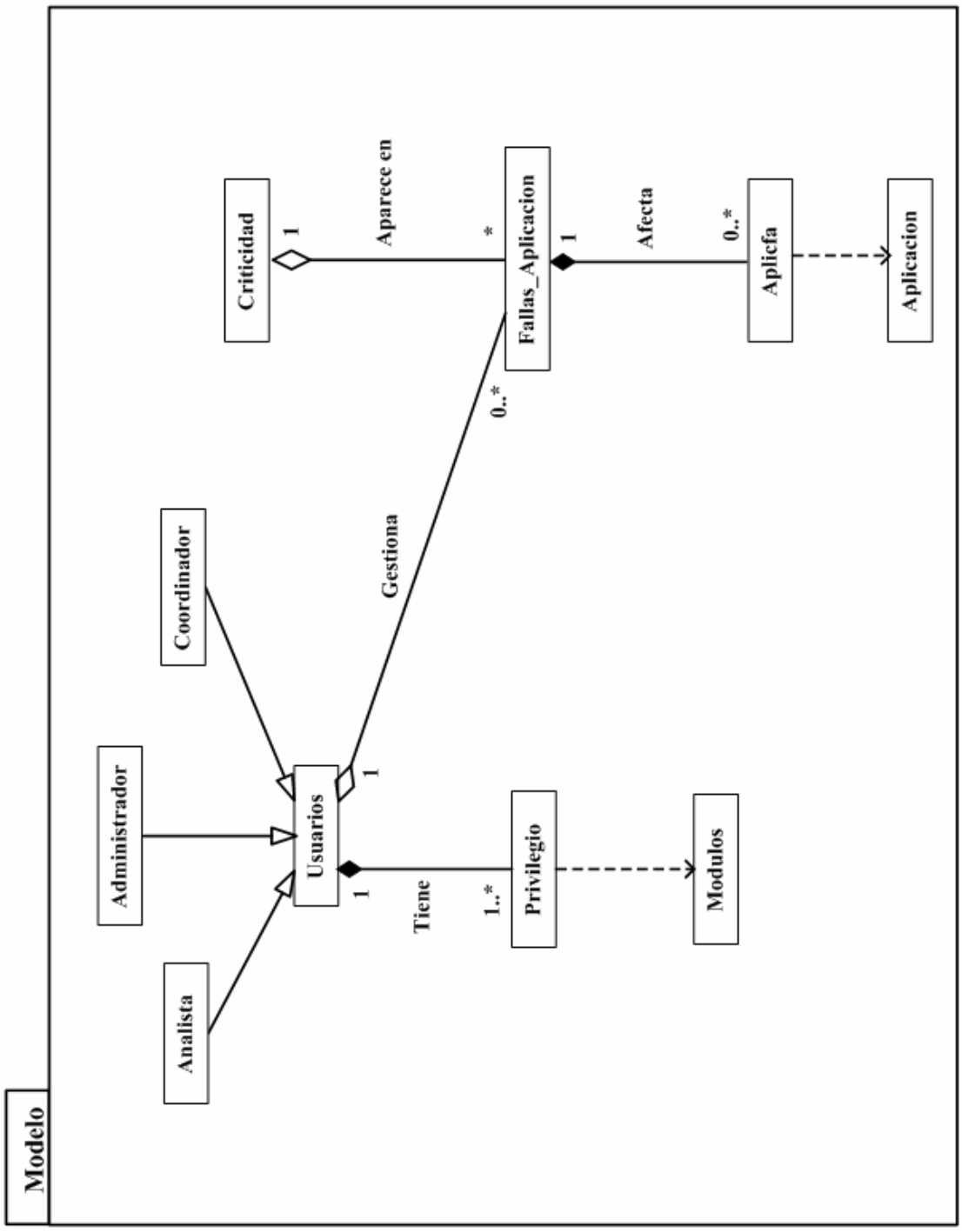


Figura 3.2: Modelo de Dominio o Diagrama de Clases del sistema
Fuente: Propia, 2008

3.1.3 Riesgos del Sistema

Al realizar un Sistema se debe tomar en cuenta los posibles riesgos tanto críticos como secundarios que afectarían al proyecto en su desarrollo o finalmente en su funcionamiento; por lo tanto uno de los pasos de la fase de inicio es identificar los posibles riesgos y solventarlos. A continuación, se muestra la tabla 3.1, que contiene los riesgos críticos del proyecto REFAP con su respectiva acción para mitigarlos:

Tabla 3.1 / 1: Riesgos críticos del proyecto REFAP

Riesgos	Ubicación	Responsable	Contingencia / Solución
Interfaz confusa o de difícil interactividad para los usuarios.	Interfaz	Desarrollador del Software	✓ Desarrollar una interfaz amigable y de fácil interactividad para los usuarios. ✓ Capacitar al personal con respecto al buen funcionamiento del sistema. ✓ Elaborar un manual de ayuda que abarque todo el manejo del sistema para los diferentes usuarios.
Los algoritmos diseñados no gestionan de forma correcta los distintos procesos presentes en el sistema.	Software	Desarrollador del Software	✓ Revisar todos los algoritmos paso a paso y observar si la salida es la adecuada para el comportamiento de cada uno de los procesos del sistema. ✓ Realizar las correcciones necesarias a los algoritmos hasta lograr el funcionamiento eficiente y eficaz de los distintos procesos del sistema.
Configuración ineficiente del sistema	Software	Desarrollador del Software	✓ Diseñar un módulo de configuración robusto que conlleve a un sistema apto según los requisitos exigidos por el servicio.

Tabla 3.1 / 2: Riesgos críticos del proyecto REFAP

Riesgos	Ubicación	Responsable	Contingencia / Solución
Falta de experiencia en el manejo de las herramientas empleadas para el desarrollo del proyecto.	Software	Desarrollador del Software	✓ Adquirir conocimientos sobre las herramientas a emplear en el desarrollo del proyecto mediante cursos y clases de entrenamiento. ✓ Dedicar el tiempo necesario para leer libros y manuales.
Acceso fallido a la Base de Datos.	Base de Datos	Desarrollador del Software	✓ Confirmar que la conexión a la base de datos sea correcta.
Sentencias SQL no adecuadas.	Base de Datos	Desarrollador del Software	✓ Revisar las Sentencias SQL paso a paso y observar si la salida es la esperada para el comportamiento correcto de cada uno de los procesos del sistema. ✓ Realizar las correcciones pertinentes a las Sentencias SQL.
Pérdida de información.	Base de Datos	Desarrollador del Software	✓ Hacer un respaldo constantemente de los datos para evitar inconsistencia en los mismos.
Insatisfacción de los usuarios al presentarles el sistema final.	Usuarios finales	Usuarios finales	✓ Mostrar constantemente prototipos del sistema a los usuarios finales, de forma que al surgir nuevos requisitos puedan ser solventados a tiempo.

Existen otros riesgos secundarios, que afectan levemente el desarrollo del proyecto REFAP, los cuales se pueden observar en la tabla 3.2 que se muestra a continuación:

Tabla 3.2: Riesgos secundarios del proyecto REFAP

Riesgos	Ubicación	Responsable	Contingencia / Solución
Carencia del hardware necesario para dicha aplicación.	Hardware	PDVSA PETROCEDEÑO	✓ PDVSA PETROCEDEÑO, debe adquirir el hardware necesario para el buen funcionamiento del sistema.
Incompatibilidad en la integración de la aplicación con la plataforma.	Plataforma	PDVSA PETROCEDEÑO	✓ Verificar si existe compatibilidad entre el sistema y la plataforma de hardware y de software.
Falta de conexión y de servicios de Internet.	Servidor	PDVSA PETROCEDEÑO	✓ Verificar si existe servicio de Internet o instalarlo y reponerlo si es necesario.

3.1.4 Requisitos Funcionales

Los requisitos funcionales especifican una acción que deberá ser capaz de desempeñar el sistema deseado, las cuales son recopiladas al identificar las necesidades de los usuarios y clientes. Los requisitos funcionales regularmente se expresan en términos de entradas y salidas: dada una entrada específica, el requisito funcional estipula cuál debe ser la salida. Los requisitos funcionales se expresan como casos de usos en un modelo de casos de usos.

3.1.4.1 Modelo de Casos de Uso

El modelo de casos de uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema.

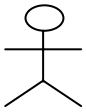
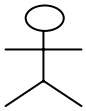
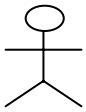
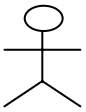
En el modelo de casos de usos se forma por los actores y casos de usos del sistema.

3.1.4.1.1 Actores

Los actores son los usuarios del sistema, que necesitan o usan alguno de los casos de uso. Un usuario puede jugar más de un rol. Un solo actor puede actuar en muchos casos de uso. Los actores pueden ser humanos u sistemas externos que requieren alguna información del sistema actual.

A continuación se puede observar la tabla 3.3, que contiene la descripción de los actores que participan en el sistema REFAP:

Tabla 3.3: Descripción de los actores que participan en el sistema REFAP.

Actores	Descripción
 Administrador	Es el encargado del manejo de los datos estáticos del sistema; puede registrar, modificar y eliminar usuarios, aplicaciones y criticidad. Además es el que autoriza a los usuarios a ingresar a determinadas áreas del sistema.
 Coordinador	Representa a un integrante de la Coordinación IS, que coordina a los Analistas y se encarga de tomar las decisiones respecto a las medidas que deben emplearse en torno a las fallas que se generan en las distintas aplicaciones que existen en PDVSA PETROCEDEÑO, por lo tanto tiene acceso a los reportes.
 Analista	Representa a los integrantes de la Coordinación IS, que se encargan de atender las fallas que se generan en las aplicaciones que existen en PDVSA PETROCEDEÑO, por lo tanto están autorizados para registrar, modificar, consultar y eliminar la información concerniente a las fallas.
 Manejador de Base de Datos	Ejecuta las transacciones relacionadas con el manejo de data (inserción, modificación, eliminación y consulta), en forma transparente, es decir, su participación está implícita en cada una de las tareas solicitadas por los otros actores del sistema, sin que estos se cercioren ni den cuenta de ello.

3.1.4.1.2 Casos de uso

Los casos de uso son una secuencia de acciones que el sistema lleva a cabo interactuando con los usuarios (actores) para dar un resultado de valor.

En el proyecto REFAP se encuentran los siguientes casos de usos:

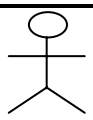
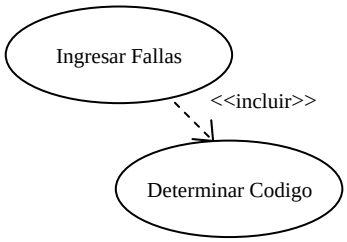
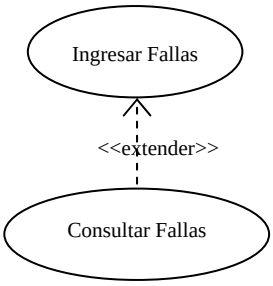
- Gestionar Sesión: Este caso de uso es el proceso mediante el cual los usuarios puede tener acceso al sistema, y además permite que dichos usuarios puedan cambiar su clave de acceso al sistema.
- Gestionar Administración: Es el proceso a través del cual se lleva a cabo el manejo de toda la información estática del sistema.
- Gestionar Fallas: Es el proceso que permite efectuar operaciones concerniente a las fallas que se generan en las distintas aplicaciones existentes en el mejorador PDVSA PETROCEDEÑO.
- Realizar Consultas: Es el proceso que permite consultar información de las fallas que se generan en las distintas aplicaciones existentes en el mejorador PDVSA PETROCEDEÑO.
- Generar Reportes: Este caso de uso concede el acceso a los diferentes reportes del sistema que abarcan información sobre las fallas que se generan en las distintas aplicaciones existentes en el mejorador PDVSA PETROCEDEÑO.

3.1.4.1.3 Diagrama de Caso de Uso

Un diagrama de caso de uso describe un conjunto de caso de uso, usuarios (actores) y sus relaciones, representando la interacción de los usuarios con el sistema.

Un diagrama de caso de uso consiste de un conjunto de casos de uso, actores, relaciones entre casos de usos y asociaciones de comunicación entre actores y casos de usos, los cuales se especifican en la tabla 3.4 que se muestra a continuación:

Tabla 3.4: Componentes de un diagrama de caso de uso

	Detalles	Representación gráfica
Caso de uso	Establece un conjunto de escenarios para realizar algo útil para un actor. En UML, un caso de uso se representa como una elipse con el nombre dentro de la elipse.	
Actores	Usuarios (un humano u oto sistema), que interactúa con el sistema. En UML, un actor se representa con una figura con el nombre debajo de el.	
Asociación de Comunicación entre actores y casos de usos	Comunica un actor con un caso de uso. En UML, una Asociación se representa con una línea que conecta al actor con un caso de uso.	
Relación de Inclusión entre casos de usos	Significa que dentro de un caso de uso se puede utilizar los pasos de otro caso de uso existente. En UML, la inclusión se representa con una línea discontinua con una punta de flecha que conecta a los casos de uso apuntando hacia el caso de uso que se incluye. Sobre la línea se agrega la palabra “incluir” bordeada por dos pares de paréntesis angulares.	
Relación de Extensión entre casos de usos	Significa que se puede crear un caso de uso añadiéndole pasos a uno existente. En UML, la extensión se representa con una línea discontinua con una punta de flecha que conecta a los casos de uso apuntando hacia el caso de uso que es ampliado o extendido. Sobre la línea se agrega la palabra “extender” bordeada por dos pares de paréntesis angulares.	

Para la columna Representación gráfica de la tabla 3.4 mostrada anteriormente, se tomaron ejemplos del proyecto en estudio. Parte de estos ejemplos se pueden

observar en la figura 3.3, que representa al Modelo de caso de uso general del proyecto REFAP. En este diagrama se presenta una relación de “Herencia” entre los actores Administrador, Analista, Coordinador y Usuarios, es decir, éste último les otorga sus atributos a los otros actores mencionados anteriormente. La herencia se representa mediante una flecha vacía que apunta del actor que hereda los atributos al actor que cede dichos atributos.

Además se muestran asociaciones de comunicación entre los actores del sistema y los casos de usos al cual pueden acceder los mismos, como por ejemplo: existe una asociación de comunicación entre el actor Analista y el caso de uso Gestionar Fallas. Este actor también se asocia con los casos de usos Gestionar Sesion y Realizar Consultas, y el actor Administrador se asocia con los casos de usos Gestionar Sesion y Gestionar Administración, mientras el actor Coordinador lo hace con los casos de usos Gestionar Sesion y Generar Reportes.

Como se visualiza en la figura 3.3, el actor Manejador de Base de Datos se asocia con todos los casos de usos, debido a que dicho actor participa implícitamente en todos los procesos que se manejan en el sistema.

El caso de uso Gestionar Sesion es extendido por los casos de usos Iniciar Sesion y Cambiar Clave. El caso de uso Gestionar Administración es extendido por los casos de usos Gestionar Usuario, Gestionar Aplicación y Gestionar Criticidad. El caso de uso Gestionar Fallas es extendido por los casos de usos Ingresar Fallas, Modificar Fallas y Eliminar Fallas. El caso de uso Realizar Consultas es extendido por los casos de usos Consultar fallas por Analista, Consultar fallas por Codigo de Registro y Consultar fallas por Rango de Fechas; mientras que el caso de uso Generar Reportes es extendido por los casos de usos Procesar Reporte aplicaciones afectadas por una falla, Procesar Reporte estadisticas de fallas producidas, Procesar Reporte estadisticas de fallas producidas por aplicacion, Procesar Reporte estadisticas de fallas por criticidad, Procesar Reporte fallas atendidas por analista, Procesar Reporte fallas por aplicacion y Procesar Reporte fallas por rango de fechas.

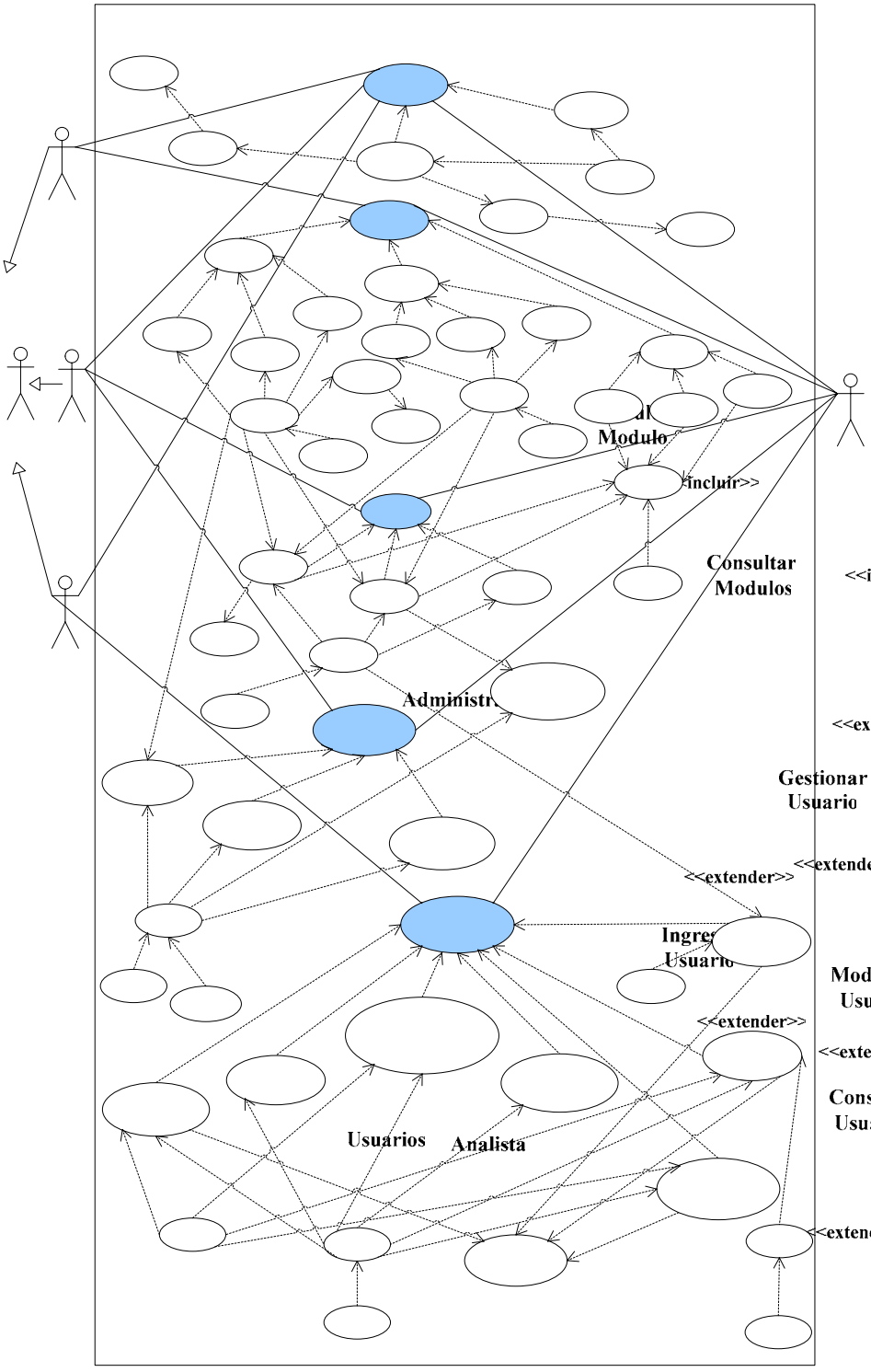


Figura 3.3: Diagrama de caso de uso general del proyecto REFAP
Fuente: Propia, 2008

A continuación se detallaran los diagramas de los casos de uso del proyecto REFAP:

- Diagrama del caso de uso Gestionar Sesion: En la figura 3.4, se puede observar que todos los actores del sistema se asocian a los casos de usos Iniciar Sesion y Cambiar Clave.

  El caso de uso Iniciar Sesion es el proceso utilizado por los usuarios para ingresar al sistema. Este caso de uso incluye a los casos de uso Consultar Modulos y Consultar Privilegios. El caso de uso Consultar Modulos es el proceso mediante el cual se verifican los módulos que contiene el sistema, y el caso de uso Consultar Privilegios permite determinar los módulos del sistema al cual un usuario puede acceder; por lo tanto al iniciarse una sesión, automáticamente se consultan todos los módulos existentes en el sistema y además se verifican los privilegios del usuario que inicia la sesión.

  El caso de uso Cambiar Clave es el proceso que emplean los usuarios para cambiar su contraseña de acceso al sistema.

  Tanto el caso de uso Cambiar Clave como el caso de uso Iniciar Sesion son extendidos por el caso de uso Consultar Usuario. Este último caso de uso es el proceso que permite constatar la existencia de un determinado usuario en el sistema.

  El caso de uso Consultar Modulo es el proceso que permite consultar uno a uno los módulos del sistema. Parecido a este proceso es el de Consultar Privilegio, mediante el cual se verifica uno a uno los módulos a los que un usuario puede acceder.

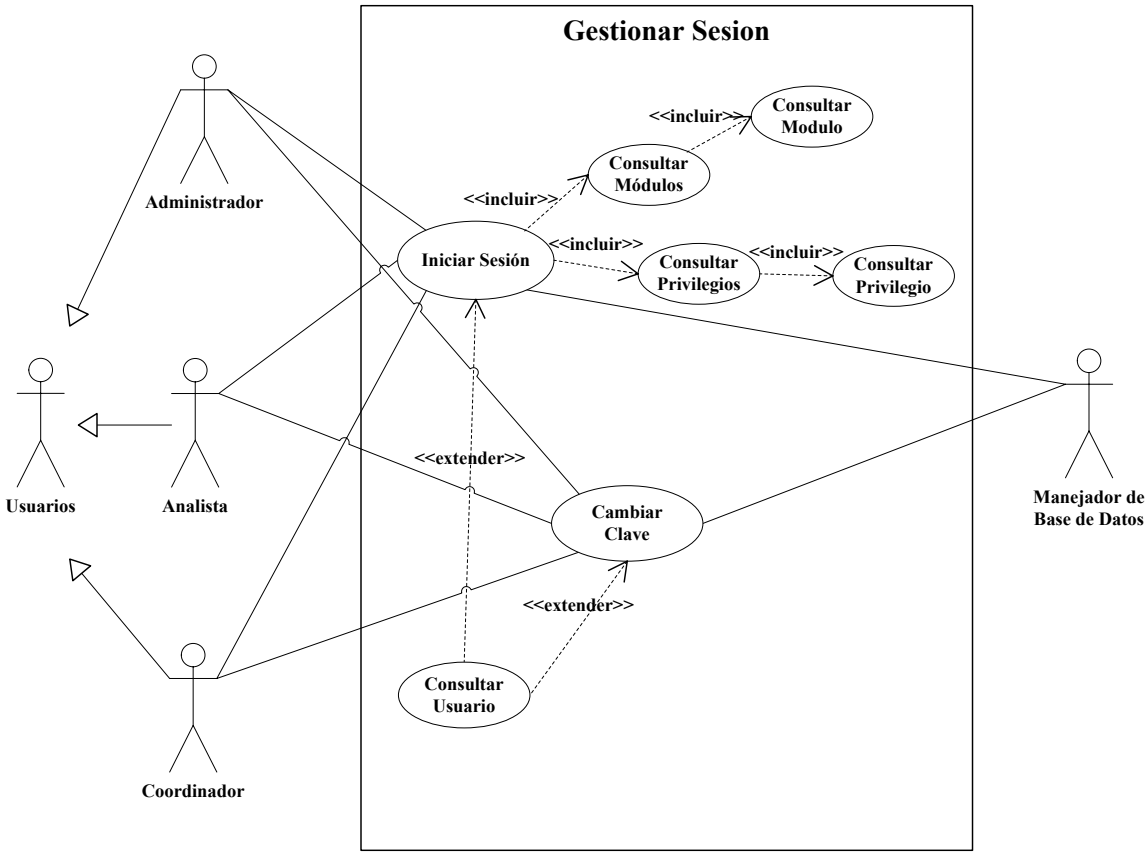


Figura 3.4: Diagrama detallado del caso de uso Gestionar Sesion
Fuente: Propia, 2008

- Diagrama del caso de uso Gestionar Administracion: En la figura 3.5, se puede observar que el actor Administrador tiene una relación de asociación con los casos de usos Gestionar Usuario, Gestionar Aplicacion y Gestionar Criticidad.

  Mediante el caso de uso Gestionar Usuario, el Administrador interactúa con el sistema a través de los casos de usos Ingresar Usuario, Modificar Usuario y Eliminar Usuario, los cuales son extendidos por el caso de uso Consultar Usuarios; este último caso de uso se emplea para verificar que los usuarios que se ingresen, no se encuentren ya registrados en el sistema, además incluye al caso de uso Consultar Privilegios y es extendido por el caso de uso Consultar Usuario.

Por medio del caso de uso Ingresar Usuario, el Administrador puede registrar a los usuarios en el sistema.
El caso de uso Modificar Usuario, le permite al Administrador realizar los cambios que considere necesario a los datos de los usuarios existentes en el sistema.
Empleando el caso de uso Eliminar Usuario, el Administrador puede sacar del sistema a un determinado usuario.

En la figura 3.5, se ilustra que el caso de uso Gestionar Aplicacion es extendido por los casos de usos Ingresar Aplicacion, Modificar Aplicacion, y Eliminar Aplicacion.
Empleando el caso de uso Ingresar Aplicacion, el actor Administrador le puede dar entrada al sistema a todas las aplicaciones que se manejan en el mejorador de PDVSA PETROCEDEÑO y para hacerle cambios a la información de dichas Aplicaciones, utiliza el caso de uso Modificar Aplicación. Si es esencial dar de baja a una determinada aplicación, el Administrador emplea el caso de uso Eliminar Aplicación.
Los casos de usos Ingresar Aplicacion, Modificar Aplicacion y Eliminar Aplicacion, son extendidos por el caso de uso Consultar Aplicaciones. Este último caso de uso, se emplea para verificar que los datos introducidos al registrar una Aplicación nueva o al realizarle cambios a una aplicación existente, no hayan sido almacenados anteriormente en la base de datos.

En la figura 3.5, se puede observar que el caso de uso Gestionar Criticidad es extendido por los casos de usos Ingresar Criticidad, Modificar Criticidad, Eliminar Criticidad.
A través del caso de uso Ingresar Criticidad, el Administrador puede registrar los diferentes tipos de criticidad que se utilizaran en el sistema.

El caso de uso Modificar Criticidad, le permite al Administrador realizar los cambios que considere necesario a las diferentes Criticidades existentes en el sistema.

Empleando el caso de uso Eliminar Criticidad, el Administrador puede sacar del sistema a la Criticidad que no requiera.

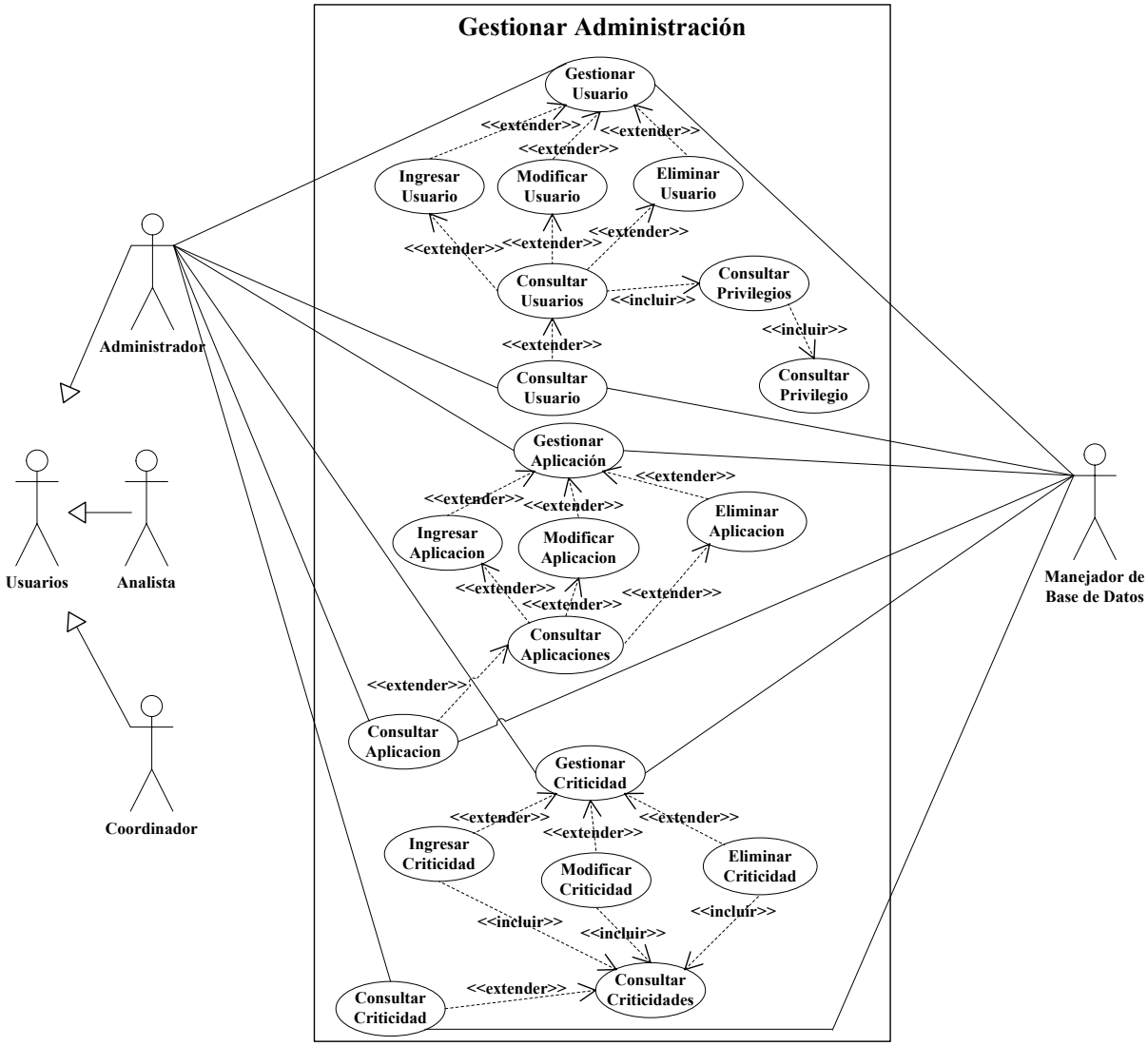


Figura 3.5: Diagrama detallado del caso de uso Gestionar Administracion
Fuente: Propia, 2008

- Diagrama del caso de uso Gestionar Fallas: En la figura 3.6, se ilustra al caso de uso Gestionar Fallas que consiste de 13 casos de usos y 2 actores que interactúan con el sistema.

  El actor Analista interactúa con el sistema a través de los casos de usos Ingresar Fallas, Modificar Fallas, Eliminar Fallas y Consultar Fallas. Mediante los casos de usos mencionados, el Analista puede registrar nuevas fallas, realizarle cambios a los datos de las fallas previamente almacenados, sacar del sistema una determinada falla y verificar que los datos de las fallas que se ingresen o modifiquen no hayan sido registrados con anterioridad en el sistema; respectivamente.

  El caso de uso Ingresar Fallas incluye al caso de uso Determinar Codigo. Este último caso de uso es el proceso mediante el cual se le asigna un código automáticamente a cada falla que se registre en el sistema.

  Tanto el caso de uso Ingresar Fallas como el caso de uso Modificar Fallas, son extendidos por los casos de usos Consultar Usuarios, Consultar Fallas y Consultar Aplicaciones. Además incluyen al caso de uso Consultar Criticidades.

  El caso de uso Consultar fallas es el proceso que permite verificar que los datos introducidos al registrar una nueva falla o al modificar una falla previamente almacenada, no existan en la base de datos.

  El caso de uso Modificar Fallas incluye al caso de uso Consultar Aplicaciones_Afectadas. Este último caso de uso es el proceso mediante el cual se consulta si existen Aplicaciones que han sido afectadas por la falla que se desea modificar.

  El caso de uso Eliminar Fallas es extendido por el caso de uso consultar fallas. En esta ocasión este último caso de uso permite que se realice una consulta de todas las fallas que quedan registradas en el sistema luego de hacer una eliminación de una falla, es decir, se actualiza la lista de las fallas existentes en el sistema.

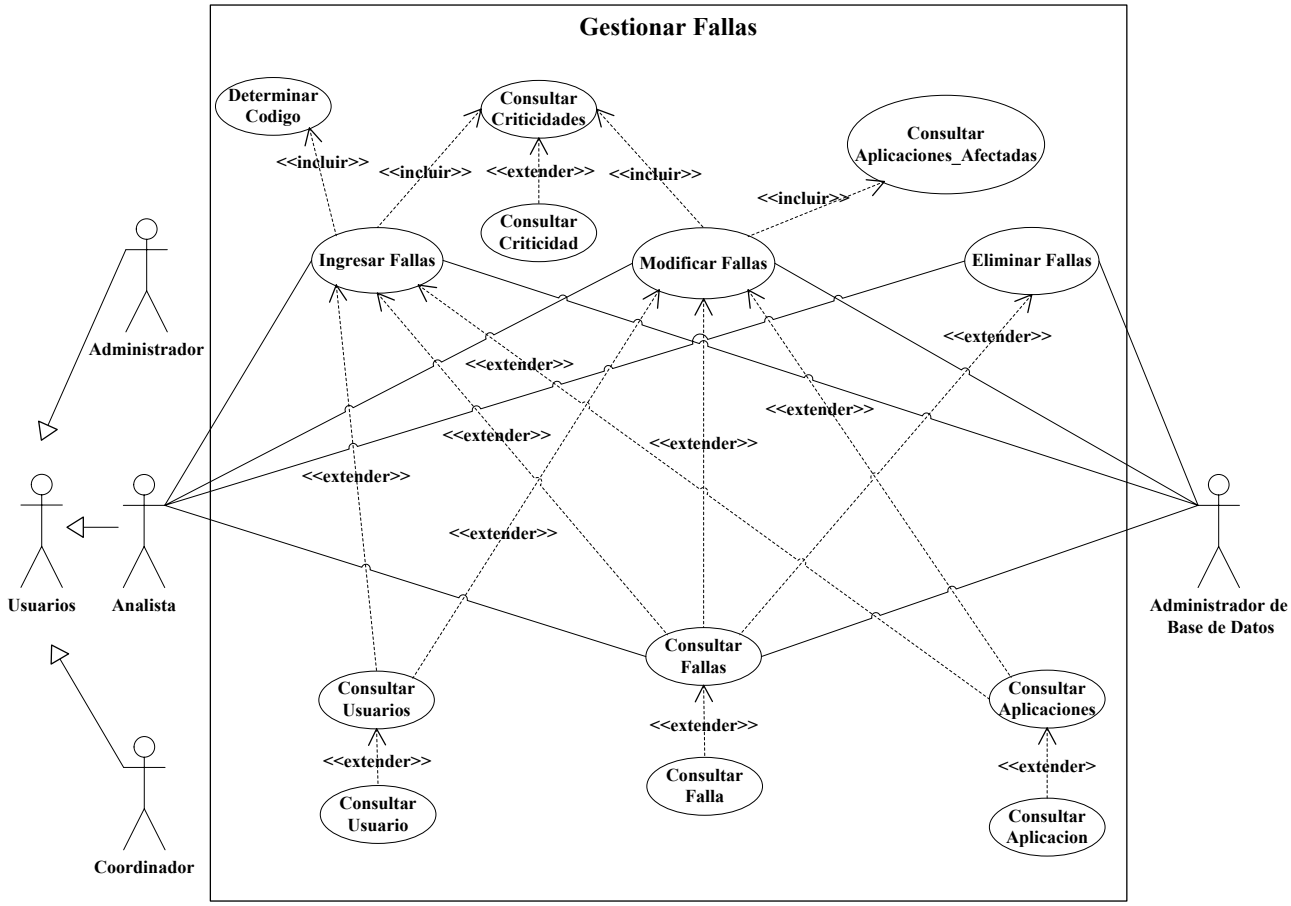


Figura 3.6: Diagrama detallado del caso de uso Gestionar Fallas
Fuente: Propia, 2008

- Diagrama del caso de uso Realizar Consultas: En la figura 3.7, se puede observar que el actor Analista se asocia con los casos de uso Consultar fallas por Analista, Consultar fallas por Codigo de Registro y Consultar fallas por Rango de Fechas. Los casos de usos mencionados son extendidos por el caso de uso Consultar Fallas.

  Mediante el caso de uso Consultar fallas por Analista, el actor Analista puede consultar todas las fallas que él ha registrado en el sistema, y ver toda la información concerniente a dichas fallas.

  El caso de uso Consultar fallas por Codigo de Registro, le permite al actor Analista realizar una consulta de forma exclusiva, es decir, puede consultar

toda la información de sólo una falla, dependiendo del código del registro que el actor desee consultar.

A través del caso de uso Consultar fallas por Rango de Fechas, el actor Analista puede estipular un margen de fechas y realizar una consulta de todas las fallas que se encuentren dentro de dicho margen de fechas.

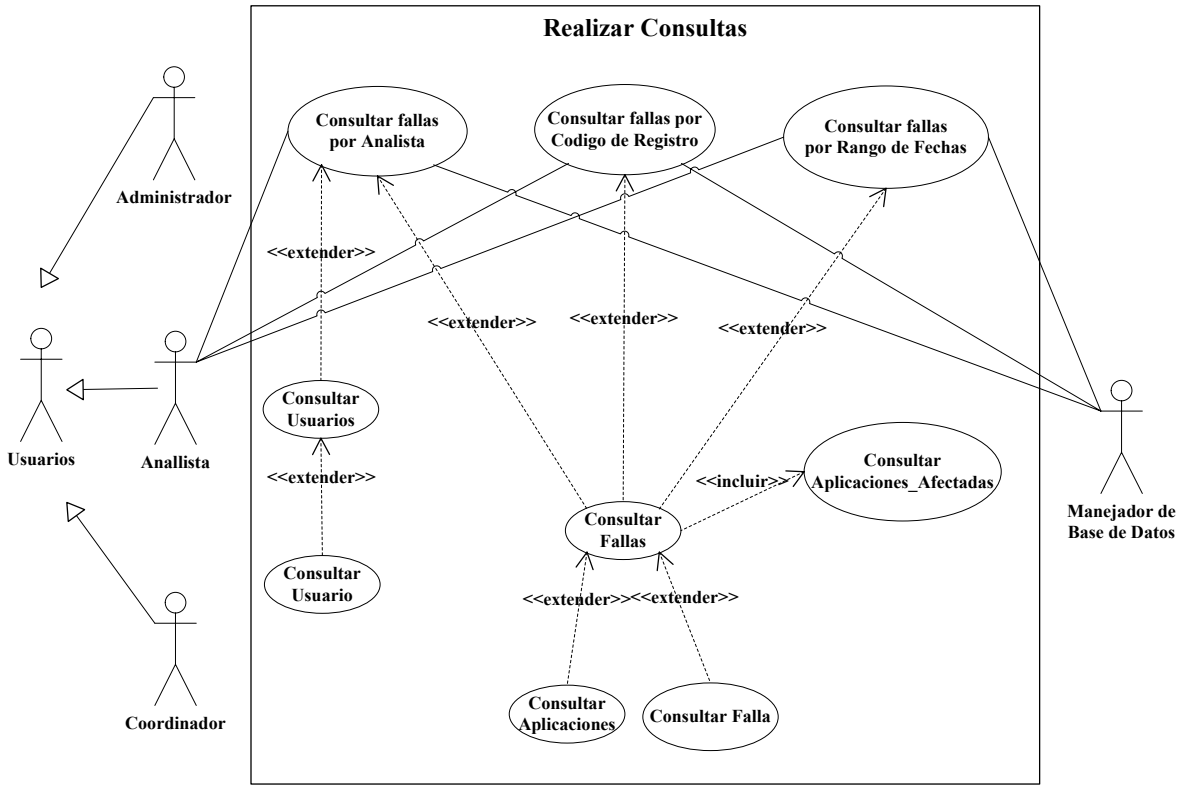


Figura 3.7: Diagrama detallado del caso de uso Realizar Consultas
Fuente: Propia, 2008

- Diagrama del caso de uso Generar Reportes: En la figura 3.8, se ilustra al Diagrama del caso de uso Generar Reportes, que consta de 13 casos de usos y dos actores que interactúan con el sistema a través de 7 casos de usos.

  Por medio del caso de uso Procesar Reporte aplicaciones afectadas por una falla, el actor Coordinador puede acceder a la información de las aplicaciones que son afectadas cuando se produce una determinada falla.

El caso de uso Procesar Reporte estadisticas de fallas producidas, es el proceso que refleja las veces que se ha producido una determinada falla en las aplicaciones existentes en el Mejorador de PDVSA PETROCEDEÑO.

A través del caso de uso Procesar Reporte estadisticas de fallas producidas por aplicacion, el actor Coordinador puede efectuar un reporte que contenga todas las aplicaciones que se manejan en la Coordinación IS de PETROCEDEÑO y las fallas que se han producido en de dichas aplicaciones, al igual que la cantidad de veces de ocurrencia de las fallas.

El caso de uso Procesar Reporte estadisticas de fallas por criticidad, es el proceso mediante el cual el actor Coordinador puede obtener las estadísticas de las fallas producidas en el Mejorador de PDVSA PETROCEDEÑO, dependiendo del tipo de criticidad que posea.

Empleando el caso de uso Procesar Reporte fallas atendidas por analista, el actor Coordinador puede obtener un reporte con todas las fallas que un determinado Analista ha registrado en el sistema, y ver toda la información que dicho Analista ha documentado sobre las fallas que ha gestionado.

El actor Coordinador utilizando el caso de uso Procesar Reporte fallas por aplicacion, puede visualizar todas las fallas que se han producido en una determinada aplicación.

Mediante el caso de uso Procesar Reporte fallas por rango de fechas, el actor Coordinador puede estipular un margen de fechas y realizar un reporte que contenga todas las fallas que se encuentren dentro de dicho margen de fechas.

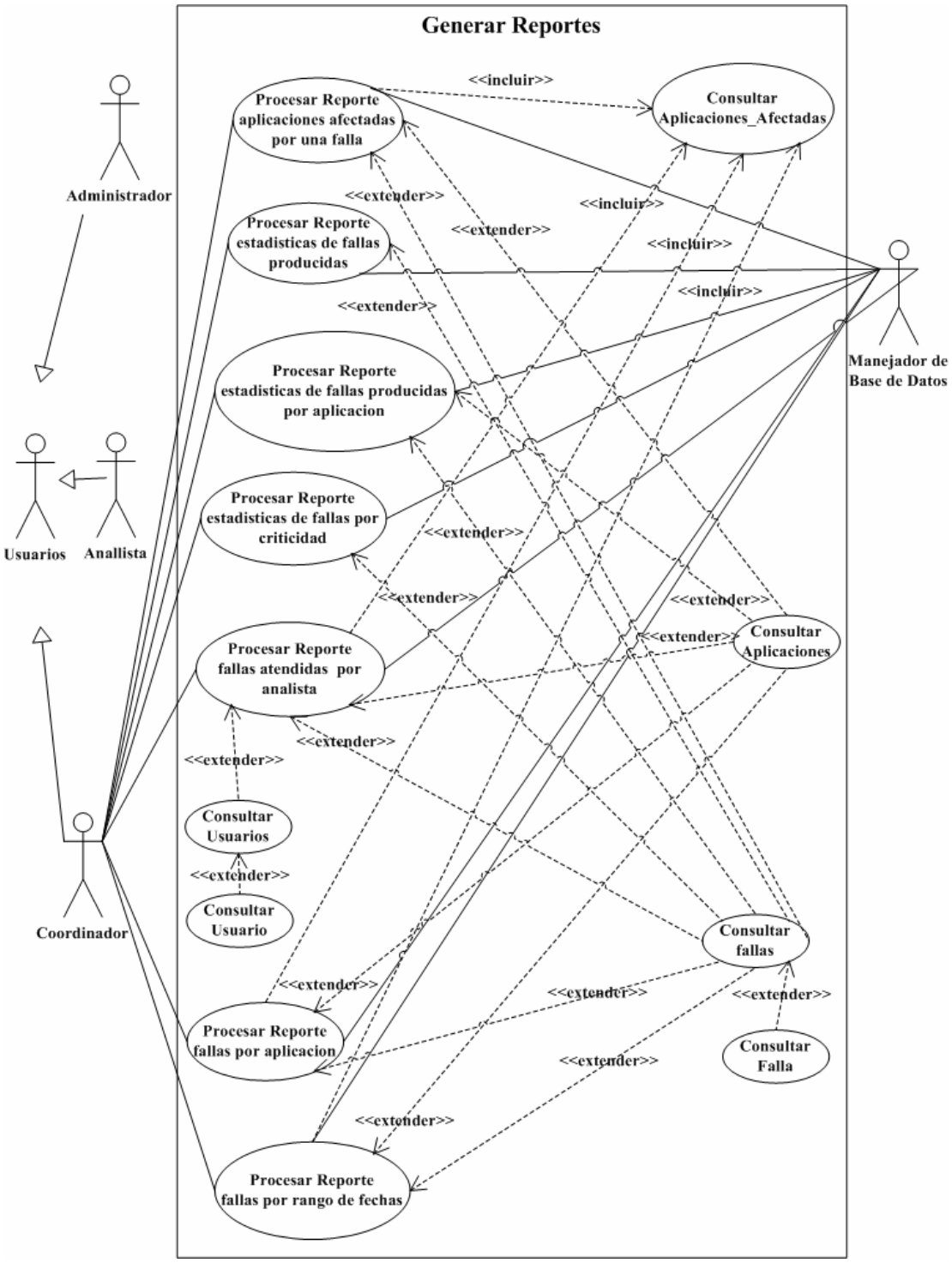


Figura 3.8: Diagrama detallado del caso de uso Generar Reportes
Fuente: Propia, 2008

Una forma de mostrar el comportamiento de un caso de uso, es a través de la descripción de un flujo de eventos. Este flujo de eventos puede ser normal, es decir que se realicen los pasos del caso de uso sin ocurrir excepción o inconveniente alguno, lo cual se denomina “Curso Típico”, o en su defecto que aparezcan dichos inconvenientes o excepciones, lo cual se denomina “Cursos Excepcionales”.

A continuación se mostrarán unas tablas con la realización de ciertos casos de usos del proyecto REFAP:

Tabla 3.5 / 1: Realización del Caso de Uso “Ingresar Fallas”

Nombre Caso de Uso	**INGRESAR FALLAS**	Nº Caso. de Uso.	**01**
Actores	Analista		
Descripción	Permite ingresar al sistema los datos de las fallas que se generan en las distintas aplicaciones existentes en PDVSA PETROCEDEÑO.		
Casos de Uso Relacionados:	➢ Determinar Codigo ➢ Consultar Criticidades ➢ Consultar Usuarios ➢ Consultar Fallas ➢ Consultar Aplicaciones		
Entradas:	Datos de la falla a ingresar (Fecha de reporte, Usuario que reporta la falla, Nombre de la falla, Causa de la falla, Aplicación que falla, Aplicaciones que la falla afecta, Analista que atiende la Falla, Criticidad de la falla, Solución aplicada a la falla).		
Salidas:	Registro de la falla actualizado en la Base de Datos del sistema.		
Curso Típico			
Acción del Actor		Respuesta del Sistema	
1.- El actor ingresa todos los datos sobre la falla, requeridos en el formulario que muestra el sistema (Fecha de reporte, Usuario que reporta la falla, Nombre de la falla, Causa de la falla, Aplicación que falla, Aplicaciones que la falla afecta, Analista que atiende la Falla, Criticidad de la falla, Solución aplicada a la falla).			
2.- El actor presiona el botón adecuado para registrar los datos.			

Tabla 3.5 / 2: Realización del Caso de Uso "Ingresar Fallas"

Curso Típico	
Acción del Actor	Respuesta del Sistema
	3.- El sistema registra la falla, le asigna un código de registro y muestra un mensaje avisando que los datos ya se cargaron en la base de dados.
	4.- El sistema actualiza la Base de Datos.

Tabla 3.6 / 1: Realización del Caso de Uso "Modificar Fallas"

Nombre Caso de Uso	**MODIFICAR FALLAS**	Nº Caso. de Uso.	**02**
Actores	Analista		
Descripción	Permite modificar los datos de las fallas que se generan en las distintas aplicaciones existentes en PDVSA PETROCEDEÑO.		
Casos de Uso Relacionados:	➢ Consultar Aplicaciones ➢ Consultar Criticidades ➢ Consultar Usuarios ➢ Consultar Fallas ➢ Consultar Aplicaciones_Afectadas		
Entradas:	Datos de la falla a modificar (Fecha de reporte, Usuario que reporta la falla, Nombre de la falla, Causa de la falla, Aplicación que falla, Aplicaciones que la falla afecta, Analista que atiende la Falla, Criticidad de la falla, Solución aplicada a la falla).		
Salidas:	Modificación de los datos de la falla en la Base de Datos del sistema.		
Curso Típico			
Acción del Actor	Respuesta del Sistema		
1.- El Analista selecciona el código de registro a verificar y le envía esa información al sistema.			
	2.- El sistema realiza la búsqueda del código de la falla y suministra toda la información almacenada en la base de datos correspondiente a dicho registro.		

Tabla 3.6 /2: Realización del Caso de Uso "Modificar Fallas"

Curso Típico	
Acción del Actor	Respuesta del Sistema
3.- El Analista realiza los cambios a los datos del registro y envía la solicitud para la modificación.	
4.- El actor presiona el botón adecuado para confirmar el registro de los datos.	
	5.- El sistema registra la falla, le asigna un código de registro y muestra un mensaje avisando que los datos ya se cargaron en la base de dados.
	6.- El sistema actualiza la Base de Datos.

Tabla 3.7 / 1 Realización del Caso de Uso "Ingresar Usuarios

Nombre Caso de Uso	**INGRESAR USUARIOS**	Nº Caso. de Uso.	**03**
Actores	Administrador		
Descripción	Permite ingresar los datos de los usuarios del sistema.		
Casos de Uso Relacionados:	➢ Gestionar Usuario ➢ Consultar Usuarios		
Entradas:	Datos del usuario a ingresar (Nombre, Apellido, Usuario, Cargo, Clave, Nombre Modulo).		
Salidas:	Registro de los datos del usuario en la Base de Datos del sistema.		
Curso Típico			
Acción del Actor	Respuesta del Sistema		
1.- El Administrador ingresa los datos del usuario solicitados en el formulario.			
2.- El actor presiona el botón adecuado para registrar los datos			
	3.- El sistema verifica la existencia del usuario ingresado, realizando la búsqueda por el username del usuario, que es el código único para cada usuario.		
4.- El actor presiona el botón adecuado para confirmar el registro de los datos.			

Tabla 3.7 / 2 Realización del Caso de Uso "Ingresar Usuarios

Curso Típico	
Acción del Actor	Respuesta del Sistema
	5.- El sistema registra al usuario, y muestra un mensaje avisando que los datos ya se cargaron en la base de dados.
	6.- El sistema actualiza la Base de Datos.

Tabla 3.8 / 1: Realización del Caso de Uso "Modificar Aplicacion"

Nombre Caso de Uso	**MODIFICAR APLICACION**	Nº Caso. de Uso.	**04**
Actores	Administrador		
Descripción	Permite modificar los datos de las aplicaciones manejadas en el sistema.		
Casos de Uso Relacionados:	➢ Consultar Aplicaciones ➢ Gestionar Aplicacion		
Entradas:	Datos de la aplicación a modificar (Codigo Aplicación, Aplicacion).		
Salidas:	Modificación de los datos de la aplicación en la Base de Datos del sistema.		
Curso Típico			
Acción del Actor	Respuesta del Sistema		
1.- El Analista selecciona la aplicación que desea modificar y le envía esa información al sistema.			
	2.- El sistema realiza la búsqueda de la aplicación y suministra los datos almacenados en la base de datos correspondiente a dicha aplicación y muestra un mensaje indicando que el código de la aplicación no puede ser modificado.		
3.- El Analista realiza el cambio al nombre de la aplicación y envía la solicitud para la modificación.			
4.- El actor presiona el botón adecuado para confirmar el registro de los datos.			
	5.- El sistema verifica la existencia del dato ingresado.		

Tabla 3.8 / 2: Realización del Caso de Uso "Modificar Aplicacion"

Curso Típico	
Acción del Actor	Respuesta del Sistema
	6.- El sistema actualiza la Base de Datos.
Cursos Excepcionales	
Curso Excepcional #1: La falla ya está registrada en la BDD	
Pre-Condición:	En el paso 3 del Curso Típico el Analista ingresa el nombre de la aplicación.
Acción del Actor	Respuesta del Sistema
	1.- En la consulta de la Aplicación se consigue una ocurrencia de registro de la misma.
	2.- El sistema muestra un mensaje, indicando que la Aplicación ya se encuentra registrada.
3.- El caso de uso continúa en el paso 3 del Curso Típico.	

3.2 ARQUITECTURA CANDIDATA DEL SISTEMA

En la fase inicial, el flujo de trabajo de diseño es breve debido a que aún no se poseen todos los detalles del sistema que se construirá. Lo que se obtiene de este flujo de trabajo es un bosquejo del modelo de diseño y una arquitectura inicial estable. Basándose en los requisitos obtenidos, y haciendo uso de los análisis realizados se puede definir una arquitectura candidata del sistema, en lo referente a software y a hardware, como a continuación se explica.

3.2.1 Arquitectura de Software

La computación distribuida puede definirse como una red de computadores que son alojamientos (hosts) para componentes que se comunican y coordinan unos a otros por medio de pase de mensajes. Dichos componentes pueden ser componentes de hardware o software (programas). Se refiere a un ambiente en el cual los programas,

los datos que ellos usan y los cómputos actuales se extienden a través de una red. En la computación distribuida los programas realizan llamadas a otros espacios de direcciones generalmente presentes en otras máquinas, mientras que en la computación local dichos componentes comparten un espacio de direcciones comunes para comunicarse; la velocidad de comunicación local es mucho mayor comparada con la distribuida; y cuando ocurre una falla parcial en una porción de la computación distribuida, las otras partes pueden no enterarse y continuar la ejecución como si nada hubiese ocurrido, a diferencia de la computación local, en la que por causa de dicha falla todos los componentes se pueden colgar, o en su defecto actúa un detector encargado de transmitir la información de la falla a aquellos componentes que no estén afectados por la misma. [8] Para el escenario planteado en este sistema, se puede proponer una arquitectura de computación distribuida que divida el trabajo del sistema en cuatro capas, como se puede observar en la figura 3.9;

- ✓ **Capa de interfaz de usuario**: en esta capa estarán todos los componentes relacionados con el diseño, captura de datos, validación de entradas e inicio y cierre de sesión, es decir, con todo lo que tenga que ver con las acciones llevadas a cabo por los usuarios a emplear el sistema.
- ✓ **Capa de controlador de interfaz**: en esta capa se encuentran todos los componentes que toman decisiones sobre el flujo de eventos, dependiendo de las solicitudes de los usuarios del sistema (llámese solicitud a cualquier tarea que se le pide al sistema ejecutar de manera automática o no, desde consultas hasta actualizaciones). Aquí están los componentes que determinan cuáles otros componentes atenderán una solicitud específica.
- ✓ **Capa de lógica del negocio**: se encuentran inmersas todas las entidades (clases) que fueron definidas para implementar las funcionalidades y comportamientos (casos de uso) del sistema, según los requerimientos de la aplicación, el alcance del sistema y las políticas de la empresa.
- ✓ **Capa de acceso a datos**: aquí se encuentran los componentes dedicados a establecer la conexión con las bases de datos asociadas al sistema. Estos

componentes son responsables de establecer los privilegios de los diferentes tipos de usuarios que intentan acceder a la información presente en las Bases de Datos respectivas.

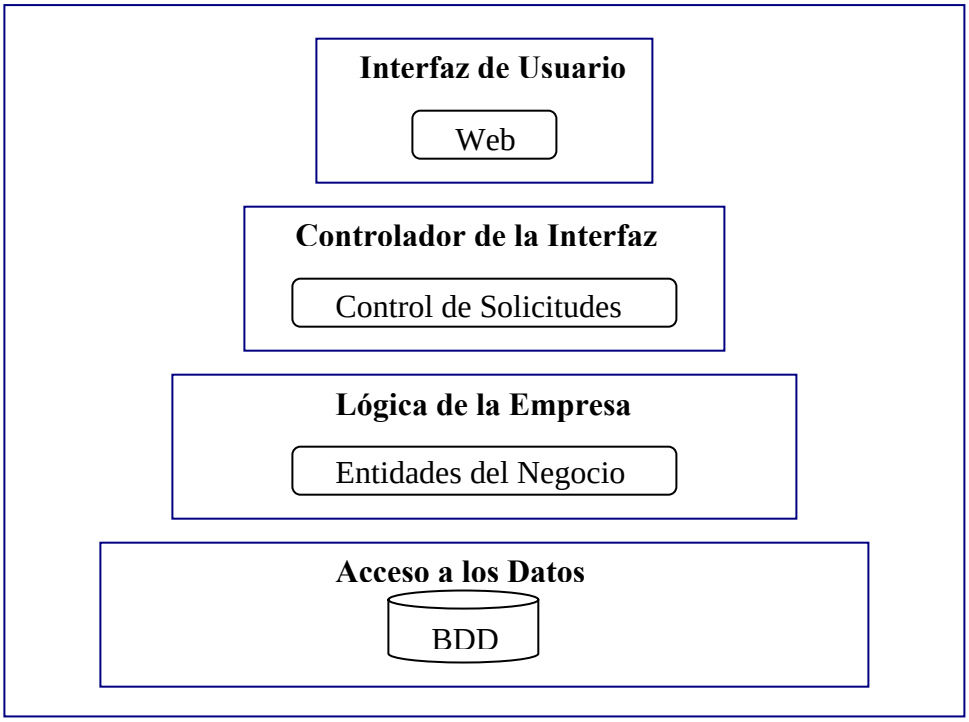


Figura 3.9: Modelo de Arquitectura de software Candidata
Fuente: Guía de Estudiante del Curso de Java Empresarial Libro Nº. 1".
Curso: CY760. Versión 4.0., 2006

3.3 EVALUACIÓN DE LA FASE DE INICIO

En esta fase se logró obtener todos los casos de usos del sistema, ya que los usuarios que requerían el sistema tenían una amplia idea de lo que necesitaban y empleando las técnicas para la captura de requisitos se hizo posible la adquisición de toda la información necesaria para la realización de los diagramas de casos de usos en una sola interacción.

Además se adquirieron los requisitos que permitieron obtener una arquitectura candidata del mismo. Así como también se identificaron la mayoría de los riesgos iniciales, consiguiéndose estrategias de mitigación de riesgos para cada uno.

Otro avance que se consiguió en esta fase es el analizar el contexto del sistema, lo cual permitió obtener un diseño potencial del modelo de dominio, se definieron las clases conceptuales que representan el negocio, y se describieron para comprender este modelo conceptual.

Se definieron los actores principales del sistema, se relacionaron con los respectivos casos de uso y la función que cumple cada uno en el sistema.

Luego se obtuvo un esbozo del sistema usando las clases hasta ahora definidas y tablas de realizaciones de casos de uso, que guiaron al siguiente paso: definir la arquitectura del sistema.

En la culminación de esta fase se presenta la propuesta de una arquitectura candidata robusta y estable a ser implementada en las fases venideras.

CAPÍTULO 4
FASE DE ELABORACIÓN

Que las bendiciones de Dios te motiven a: Soñar, Amar, Reír, Luchar, Compartir, y seguir

CAPÍTULO 4

FASE DE ELABORACIÓN

En esta fase se lleva a cabo la realización de una la serie inicial de iteraciones durante las cuales el equipo de desarrollo de software lleva a cabo un análisis mas profundo de RUP. Se define el núcleo central de la arquitectura, se aclara la mayoría de los requisitos, y se abordan los problemas de alto riesgo. Por tanto, el trabajo inicial podría incluir la implementación de los escenarios que se consideran importantes, pero que no son especialmente arriesgados desde el punto de vista técnico.

En la fase de inicio se obtienen muchos detalles esenciales como son la visión del sistema, la arquitectura candidata, los usuarios del sistema, entre otros. Pero estos sólo son bocetos, los cuales serán refinados y algunos de ellos validados en esta nueva fase. La meta de la fase de elaboración es definir y establecer la base de la arquitectura del sistema, brindando así un soporte estable para la mayor parte del esfuerzo de diseño e implementación en la fase de construcción. Esta meta general se traduce en cuatro objetivos importantes, donde cada uno trata un área imprescindible de riesgo. Se manejan riesgos asociados con los requerimientos y con la arquitectura. También se manejan los riesgos asociados con los costos, cronogramas y finalmente se necesita manejar los problemas relacionados al proceso y al ambiente de desarrollo. Manejar estos riesgos asegura un mínimo de problemas cuando se pase a la fase de construcción. (Jacobson, Booch y Rumbaugh, 2000)

Los diagramas de secuencia UML y el diseño de la base de datos son puntos que se toman en cuenta en esta fase del desarrollo del proyecto. Los primeros explican cómo es la interacción entre un conjunto de objetos específicos para llevar a cabo cierto caso de uso, mientras que el segundo modelo muestra la distribución de los datos de manera que la redundancia de los mismos sea mínima. Considerando que uno de los requisitos para este sistema es que sea una aplicación Web, se

seleccionó WebML, una herramienta novedosa que ofrece modelos para representar sistemas basados en Web, y que integrando dicha herramienta con el proceso unificado RUP, se obtendrá un buen soporte en la modelación e implementación del área Web del proyecto. Estos modelos serán de gran ayuda al momento de describir de manera sencilla el funcionamiento de las páginas Web del sistema, y los mismos colaboran significativamente en el proceso de codificación de las páginas e interfaces que se realizarán de forma definitiva en la fase siguiente a la presente.

Hasta ahora se ha obtenido un esbozo claro de la arquitectura que se desea implementar. Se estima culminar esta fase con una o dos iteraciones.

En la fase de inicio se obtuvo un diagrama completo de casos de uso, el cual se tomará en cuenta en el análisis de esta fase de elaboración.

WebML proporciona los "SiteViews" que permiten representar la organización lógica y física de las páginas de la aplicación. La siguiente actividad del flujo de trabajo de análisis será realizar el análisis de las interfaces. Es importante también realizar el diseño lógico de las páginas mediante el modelo de hipertexto de WebML. En esta fase se realizará el diseño de la base de datos del sistema mediante el modelo relacional. Para el final de esta fase se ha de obtener la arquitectura que sustentará al sistema y un diseño conceptual de las páginas bastante sólido para proceder a la codificación de las mismas. Los objetivos de esta fase son:

- **Obtener un entendimiento más detallado de los requerimientos:** durante la fase de elaboración es necesario tener un buen entendimiento de la gran mayoría de los requerimientos. En este caso en la fase de inicio se obtuvo una amplia idea de los requerimientos del sistema, lo cual facilita la creación de un plan más detallado, así como también ganar un entendimiento amplio de los requerimientos más críticos y relevantes, para validar y garantizar que la arquitectura propuesta y seleccionada cubra con todas las bases, algo que

únicamente puede alcanzarse construyendo una implementación parcial de estos requerimientos en cuestión.

- **Diseñar, implementar y validar la arquitectura base: la** funcionalidad del sistema no estará completa, pero la mayoría de las interfaces entre los bloques de construcción son implementadas durante la fase de elaboración, para poder compilar y probar la arquitectura. A esto se le denomina "arquitectura ejecutable" hasta el punto que se puede conducir alguna carga inicial de datos y ejecutar ciertas pruebas sobre la arquitectura.

- **Mitigar los riesgos significativos, producir un cronograma más exacto y estimar el costo:** durante esta fase se manejan los riesgos significativos y primordiales. La mayoría serán manejados como el resultado detallado de los requerimientos, el diseño, la implementación y prueba de la arquitectura. Al final de esta fase, se tendrá información más exacta que permitirá actualizar el cronograma, el plan de proyecto y la estimación de costos.

- **Refinar y configurar el ambiente de desarrollo:** se refina el proceso definido en la fase de inicio, para reflejar lo que se ha averiguado en las iteraciones previas. Se establece un ambiente de soporte y se define cuáles herramientas de desarrollo serán necesarias y cuáles otras tendrían que ser actualizadas o descartadas. Se instala y configura el ambiente establecido, el conjunto de todas las herramientas seleccionadas, entornos de desarrollo, arquitecturas, servidores, herramientas gráficas, entre otros.

4.1 REQUISITOS

4.1.1 Modelo de Casos De Uso

Entre los materiales o requisitos generales y básicos que se emplean en esta nueva fase de RUP para proseguir con este estudio, se encuentran todos los diagramas de casos de uso detallados de la fase anterior. Por lo tanto no es necesario que se repita dicha información en esta fase, aunque los casos de uso van a ser empleados para desarrollar nuevos artefactos y diagramas referentes a otras áreas del desarrollo del sistema, tales como los Diagramas de Secuencia, o los Modelos de Navegación y de Hipertexto de WebML, además de ciertos prototipos de pantallas o interfaces que permiten la realización de los mismos, e inclusive para etapas y fases posteriores a esta.

4.1.2 Actores, Requisitos Y Arquitectura del Sistema

Con respecto a los actores, se mantienen los mismos encontrados en la fase de inicio, al igual que los mismos requisitos funcionales y adicionales determinados. Y en cuanto a la arquitectura, tanto de software como de hardware, se mantiene la que se escogió previamente, pero se detallarán ciertas especificaciones adicionales de la misma con la finalidad de mejorar la ya expuesta, y explicarla de mejor manera, para así dar a entender definitivamente en qué consistirá dicha arquitectura en cuestión.

4.1.3 Modelo De Dominio O Diagrama De Clases

Otro de los artefactos indispensables para la continuación de este proyecto en la fase actual es el modelo o diagrama de dominio, obtenido en la etapa previa o fase de inicio, que al igual que en los modelos de casos de uso obtenidos, también este diagrama no le surgieron cambios.

4.2 ANÁLISIS

En el flujo de trabajo de análisis se tomará como datos de entrada tanto los diagramas de casos de uso, como la información presente en el diagrama de modelo de dominio obtenidos hasta el momento. Se desea obtener una vista más dinámica del sistema y sus elementos, para así tener una base de conocimiento y análisis lo suficientemente estable y fuerte para realizar un diseño óptimo y efectivo. Haciendo uso de dichos casos de uso se desarrollan otros diagramas vitales para la continuidad de este proyecto, como son los diagramas de secuencia que muestran el comportamiento y tiempo de vida de los objetos que interactúan en estos casos, y una introducción de algunos modelos del lenguaje WebML, para reforzar los diagramas UML de secuencia mencionados anteriormente, además de orientarlos a la navegabilidad de las futuras páginas Web que se han de fabricar.

4.2.1 Diagramas de Secuencia

Un diagrama de secuencia es una descripción de una colección de objetos que interactúan para implementar un cierto comportamiento dentro de un contexto. Describe una sociedad de objetos cooperantes unidos para realizar un propósito determinado. Los objetos trabajan juntos para realizar tareas comunicándose el uno con el otro, por lo tanto los diagramas de secuencia modelan a nivel de objetos más que a nivel de clases. Dicho diagrama utiliza dos elementos fundamentales:

- **Línea de vida del objeto**: la notación de la línea de vida de un objeto es una combinación de un objeto con una línea de tiempo, la cual es una línea discontinua que sale de la base del objeto mismo. El largo de dicha línea representa el tiempo que el objeto interactúa en el escenario.

- **Mensajes**: es la definición para la comunicación entre objetos. Estos mensajes pueden invocar una operación, causar una creación o destrucción de un objeto,

entre otros. Los mensajes se representan con una flecha de línea punteada, pero según el tipo de mensaje el tipo de flecha puede cambiar. Mensajes asíncronos (que no ocurren simultáneamente, sino uno después del otro) son los que imperan en estos diagramas, y los mismos están representados por una flecha con punta completa; a diferencia de los síncronos (ocurren al mismo tiempo), que se denotan con una flecha con media punta. La cola de la flecha representa al remitente del mensaje, mientras que la punta representa el receptor del mismo.

A continuación se muestran algunos diagramas de secuencia referentes a ciertos casos de uso que intervienen en esta aplicación:

- **Diagrama de Secuencia del Caso de Uso "Iniciar Sesión"**: como se puede observar en la figura 4.1, se describe la realización del procedimiento para que un usuario inicie sesión, este caso de uso lo puede llevar a cabo cualquier tipo de actor del sistema (Administrador, Coordinador, Analista) registrado en la aplicación. Para realizar este caso de uso el usuario introduce en los campos respectivos su nombre de usuario y su contraseña. Al presionar el botón de ingreso que desencadenará los procesos de autentificación para este actor, el sistema crea una instancia del objeto USUARIO y le es traspasado dichos datos (Textusu, Textcon) acompañados del mensaje "Iniciar Sesión". Este objeto USUARIO le envía al objeto SGBDD, igualmente creado con anterioridad por el sistema, el mensaje de "Consultar Datos", acompañado como es debido de los datos ingresados por el actor en uso de la aplicación. Esto con la finalidad de determinar si es un usuario registrado en el sistema el que está intentando ingresar al mismo.

En caso de cumplir con todos estos requisitos, el objeto SGBDD le devuelve al objeto USUARIO la respuesta de que efectivamente es un usuario aprobado, junto

con los demás datos presentes en la base de datos propios de dicho usuario, para luego ser destruido por el sistema, después el objeto USUARIO le envía un mensaje de “Iniciar Sesión” al actor y finalmente el sistema destruye al objeto USUARIO.

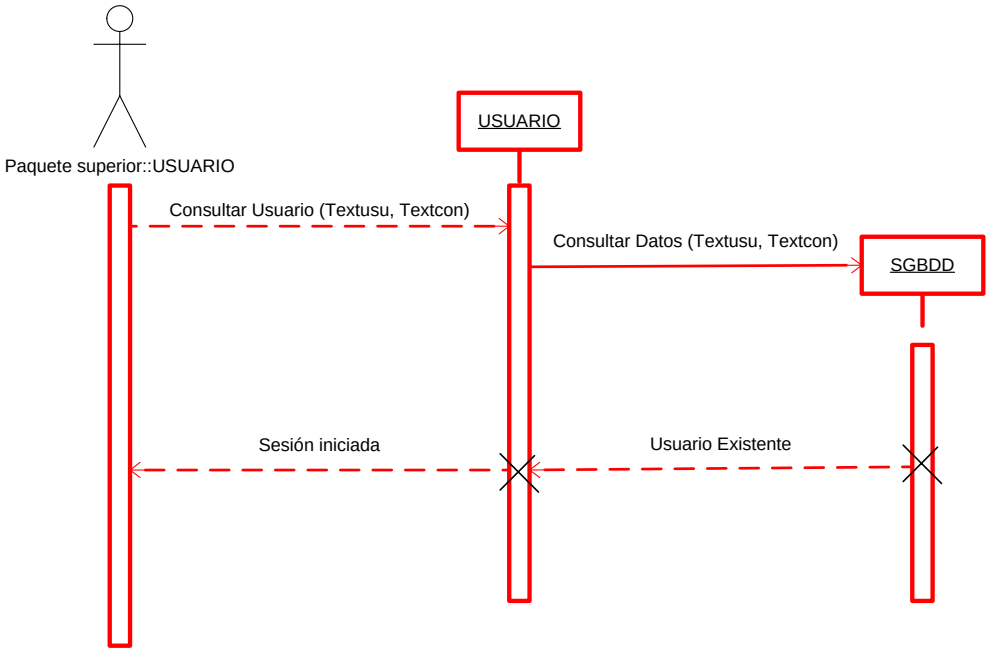


Figura 4.1: Diagrama de Secuencia del Caso de Uso “Iniciar Sesión”
Fuente: Propia, 2008

- **Diagrama de Secuencia del Caso de Uso “Ingresar Aplicación”:** En la figura 4.2se puede observar que el caso de uso Ingresar Aplicación es llevado a cabo por el Administrador del Sistema. Al momento de realizar la solicitud para ingresar una aplicación en particular, donde se incluyen todos los dados propios de la misma (Código, Nombre de la Aplicación), el sistema forma una instancia de la clase APLICACIÓN, la cual recibe dichos datos de la aplicación a registrar. A su vez dicho objeto APLICACIÓN se auto envía un mensaje “ConsultarAplicacion”, acompañado del argumento CodAplicacion. Luego el sistema crea el objeto SGBDD, el cual recibe de parte del objeto

APLICACIÒN, el mensaje “ConsultarDatos” cuyo argumento es el mismo CodAplicacion. Este objeto SGBDD revisa si dicho código de aplicación ya se encuentra registrado en la base de datos, en caso de no ser así, este objeto le devuelve al objeto APLICACIÓN un mensaje indicando que es una aplicación nueva, y el mismo es reenviado nuevamente a dicho objeto APLICACIÒN. Posteriormente este último objeto mencionado le envía al objeto SGBDD el mensaje “IngresarDatos”, acompañado en definitiva por todos los datos correspondientes a la aplicación en cuestión. Este objeto SGBDD realiza los procesos necesarios para guardar esta nueva aplicación en la base de datos y luego de ello le devuelve al objeto APLICACIÒN la señal de que su ingreso fue exitoso, para luego ser destruido por el sistema. Para finalizar, el objeto APLICACIÒN es destruido por el sistema de igual modo que lo fue el objeto SGBDD, no sin antes devolverle al usuario del sistema el mensaje de que se ingreso con éxito la aplicación.

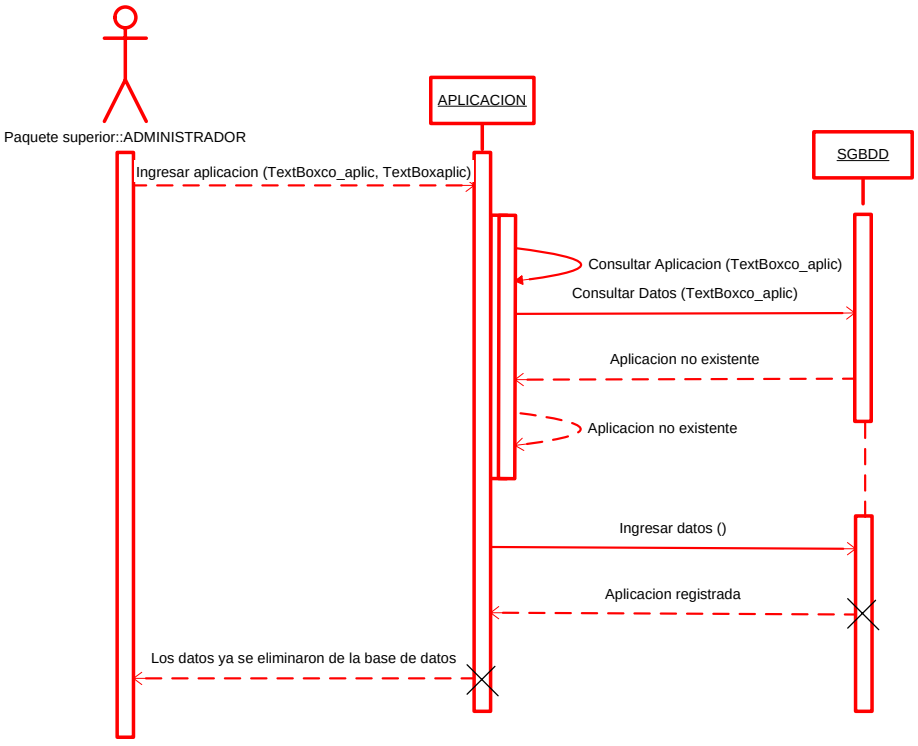


Figura 4.2: Diagrama de Secuencia del Caso de Uso “Ingresar Aplicacion”
Fuente: Propia, 2008

4.2.2 Diagramas Webml

Los diagramas WEBML más resaltantes que encajan en esta fase de elaboración son el diagrama o Modelo Estructural y el diagrama o Modelo de Hipertexto. El primero muestra los objetos publicados en el sitio Web y cómo ellos están relacionados con respecto a los objetos que describe. El segundo, como ya se mencionó anteriormente es una fusión del modelo de Navegación y del modelo de Composición. En el modelo de Navegación se observan la navegabilidad y accesibilidad entre las diversas páginas, mientras que en el de Composición se observan las unidades de contenido, las páginas y los enlaces.

4.2.2.1 Modelo Estructural Webml

El modelo estructural propio de este proyecto se observa en la figura 4.2. El mismo muestra los objetos publicados en el sitio Web y cómo ellos están relacionados. Se observan las características comunes de todo objeto, tales como **Entidad**: el nombre o tipo de objeto (Usuario). **Atributos**: propiedades escalares de una entidad, que por cuestiones de espacio se usarán atributos claves tales como códigos. **Relaciones**: conexiones entre entidades, como las existentes entre Usuarios y Privilegio, donde un (1:1) Usuario tiene uno o muchos Privilegio (1:N), **Jerarquía ES_UN**: clasifica y agrupa a las entidades, tal como se observa en los objetos actores, ya que un Analista, Coordinador o Administrador constituyen tipos de Usuario. El modelo estructural en cierta forma es una simplificación del modelo Entidad – Relación del diseño de Base de Datos, ya que las relaciones son sólo binarias entre entidades y para las entidades se emplean atributos simples mas no compuestos. De igual forma tiene un cierto parecido con el Diagrama de Clases propio de UML, pero para efectos de diseño se tratan ambos diagramas por separado, no como un mismo artefacto, ya que uno es para la construcción de clases y este otro para la construcción de páginas Web.

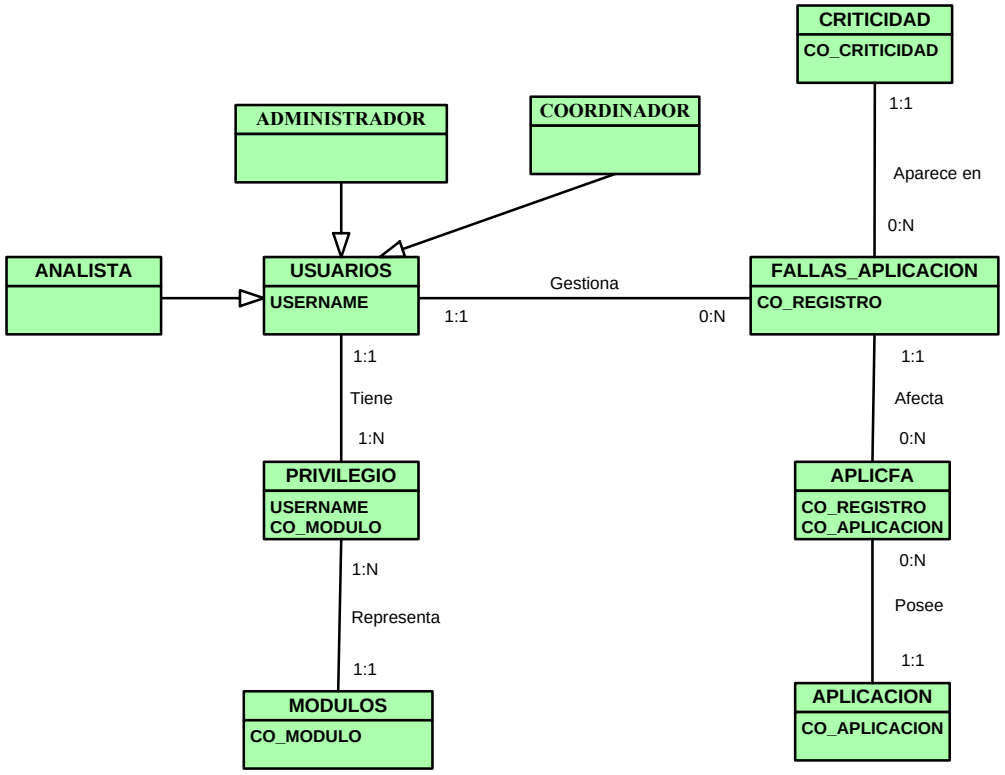


Figura 4.3. Modelo Estructural WebML del sistema
Fuente: Propia, 2008

4.2.2.2 Siteviews Webml

Un SiteView es un sub-conjunto de páginas en la que el usuario puede experimentar como un sitio Web completo. Representa un número de páginas y/o áreas que forman una vista coherente del sitio Web. A continuación, se realiza una descripción de los SiteViews, mediante tablas explicativas, tanto público como privados, propios de cada tipo de usuario, además de sus respectivos mapas de SiteView, y que conforman la estructura lógica y organizacional de la aplicación Web. Una vez definidos todos los SiteViews del sistema, se posee una visión clara de la navegabilidad y direccionamiento de cada uno de los enlaces.

Tabla 4.1. SiteView Privado "Principal para Analista"

Nombre del SiteView	**PRINCIPAL PARA ANALISTAS**
Descripción	Es la primera vista que se le muestra al Analista cuando este ingresa como usuario válido al sistema. En esta vista se muestran las opciones a las que puede acceder y cuáles son permitidas a dicho Analista en cuanto a Gestionar y Administrar Fallas, tales como Ingresar Fallas, modificar las mismas, Consultar las fallas registradas.
Grupo de Usuarios	Empleados de PDVSA PETROCEDEÑO pertenecientes al grupo de la Coordinación IS, que son autorizados para ingresar al sistema REFAP como Analistas.
Casos de Uso Asociados	❖ Gestionar Fallas ❖ Realizar Consultas

Tabla 4.2. SiteView Privado "Principal para Coordinador"

Nombre del SiteView	**PRINCIPAL PARA COORDINADOR**
Descripción	Es la primera vista que se le muestra al Coordinador cuando este ingresa como usuario válido al sistema. En esta vista se muestran las opciones a las que puede acceder y cuáles son permitidas a dicho Coordinador en cuanto a Generar Reportes.
Grupo de Usuarios	Empleado de PDVSA PETROCEDEÑO perteneciente al grupo de la Coordinación IS, que son autorizados para ingresar al sistema REFAP como Coordinador.
Casos de Uso Asociados	❖ Generar Reportes

Tabla 4.3. SiteView Privado "Principal para Administrador"

Nombre del SiteView	**PRINCIPAL PARA ADMINISTRADOR**
Descripción	Es la primera vista que se le muestra al Administrador cuando este ingresa como usuario válido al sistema. En esta vista se muestran las opciones a las que puede acceder y cuáles son permitidas a dicho Administrador en cuanto a Gestionar Administración.
Grupo de Usuarios	Empleado de PDVSA PETROCEDEÑO perteneciente al grupo de la Coordinación IS, que son autorizados para ingresar al sistema REFAP como Administrador.
Casos de Uso Asociados	❖ Gestionar Administracion

4.5 DISEÑO

El flujo de trabajo de diseño toma como entrada el análisis previamente realizado de los requisitos. En el diseño se realizarán actividades dedicadas y cuidadosas que darán como resultado los componentes que se implementarán en la siguiente fase. Entre ellos tenemos el diseño de prototipos de interfaces, diseño de la base de datos, y el diseño de la arquitectura de software y de hardware.

Un prototipo de interfaz representa un ejemplar original o primer molde en que se fabrica la interfaz visual definitiva, por ello se denomina prototipo (una representación limitada del diseño de un producto que permite a las partes responsables de su creación experimentar, probarlo en situaciones reales y explorar su uso). En lo que respecta a la base de datos, se comenzará con el diseño de las tablas que intervienen en el sistema, con la descripción de todos y cada uno de sus campos, para luego obtener sus respectivos Scripts generadores. En cuanto a la arquitectura del software se especifica el sistema manejador de bases de datos que se utilizará para este proyecto, la arquitectura .Net Framework la cual ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables, y los modelos de hipertexto de WebML para observar el comportamiento visual en lo que a la navegabilidad de la aplicación se refiere. Mientras que en lo que corresponde a la

arquitectura del hardware, en esta sección se detallan las especificaciones indicadas en la fase de inicio en cuanto a este punto o tema.

4.5.1 Prototipos de Interfaces

El prototipo de una interfaz es como un boceto de lo que verdaderamente serán las futuras interfaces de usuario definitivas del sistema. Es recomendable realizar estos esquemas una vez se hayan descritos todos los casos de uso del sistema, en otras palabras, que se hayan definido la mayoría de los requerimientos, los usuarios que interactuarán con el sistema y la forma en que interactuarán con el mismo, respectivamente. De allí que se considere incluirlos en esta fase del desarrollo del proyecto, y para ello se han seleccionado dos prototipos de interfaces, la de "Inicio de Sesión" del sistema y la interfaz "Principal".

Por solicitud de los Integrantes de la Coordinación IS de PDVSA PETROCEDEÑO, los prototipos de pantalla del proyecto REFAP se realizaron en PowerPoint.

- **Prototipo de la Interfaz de "Inicio de Sesión"**: En la figura 4.4, se puede observar una interfaz que presenta un logotipo que identifica a la empresa, el nombre del sistema, la fecha, además de una ventana que contiene los campos necesarios para iniciar sesión, tales como, el Nombre de usuario y la contraseña. Esta ventana abarca tres opciones con las cuales, aceptar iniciar la sesión, cancelar el inicio de sesión o cambiar la calve de acceso al sistema.

Figura 4.4. Prototipo de interfaz "Principal Externa" o de "Inicio de Sesión"
Fuente: Propia, 2007

- **Prototipo de la "Interfaz Usuario"**: Esta interfaz al igual que la interfaz externa, presenta un marco superior que con un logotipo que identifica a la empresa, una ventana que contiene todos los campos necesarios para gestionar todo lo relacionado con los usuarios del sistema. Del lado izquierdo muestra un menú desplegable vertical que le permite al usuario (Administrador, Coordinador, Analista) navegar por las diferentes opciones, y del lado derecho se observa la información según la opción seleccionada en el menú desplegable.

Figura 4.5. Prototipo de interfaz "Usuarios"
Fuente: Propia, 2007

4.5.2 Diseño de la Base de Datos

El almacenamiento de la información ingresada a través de la aplicación en desarrollo implica el uso de una base de datos. Para ello se implementó el modelo relacional, mediante asociaciones de entidades e interrelaciones, el cual sintetiza eficazmente la implementación de la base de datos, ya que elimina la redundancia de datos, determinándose las posibles estructuras de las tablas que requiere el sistema, según los requerimientos previos.

- **Identificación de Tablas**: durante el proceso de definición de la estructura de la Base de Datos se consiguieron 7 tablas que conformarán el sistema de datos

del software, las cuales se detallan en las siguientes tablas descriptivas. En dichas tablas se pueden observar los campos clave de las tablas, definidos de acuerdo a las relaciones conseguidas en el diseño mismo, son señalados mediante un fondo de celda amarillo, un formato de letra en **negrita** y un dibujo de una llave que hace suponer que representa una "clave principal". 

Tabla 4.4. Tabla FALLAS_APLICACION y sus respectivos campos

Nombre Tabla	**FALLAS_APLICACION**	
Descripción Tabla	Esta tabla encierra todos los datos necesarios para gestionar todo lo relacionado a las fallas que son registradas en este sistema.	
Datos de los Campos		
Nombre Campo	**Tipo Dato**	**Descripción**
CO_REGISTRO	Numeric(9)	Código que le asigna el sistema a cada falla.
USERNAME	Varchar(50)	Nombre de usuario del Analista que gestiona la falla.
USERNAME_CLILENTE	Varchar(50)	Nombre del usuario que reporta la falla.
NOMBRE_FALLA	Varchar(200)	Nombre que se le asigna a la falla.
FECHA_REPORTE	datetime(8)	Fecha en que se registró la falla en el sistema.
CAUSA_FALLA	Varchar(1000)	Motivo que ocasionó la falla.
SOLUCION_APLICADA	Varchar(5000)	Pasos seguidos para corregir la falla.
APLIC_FALLANTE	Varchar(50)	Aplicación en la que se origina la falla.
CRITICIDAD_FALLA	Varchar(50)	Nivel de impacto que tiene la falla.

Tabla 4.5 Tabla USUARIOS y sus respectivos campos

Nombre Tabla	**USUARIOS**	
Descripción Tabla	Esta tabla encierra todos los datos de los usuarios que manipulan el sistema.	
Datos de los Campos		
Nombre Campo	**Tipo Dato**	**Descripción**
USERNAME	Varchar(50)	Nombre de Usuario.
NOMBRE	Char(20)	Nombre del usuario que manipula el sistema.
APELLIDO	Char(20)	Apellido del usuario que manipula el sistema
CARGO	Char(30)	Cargo del usuario que manipula el sistema
CLAVE	Varchar(20)	Contraseña de acceso al sistema.

Tabla 4.6 Tabla MODULOS y sus respectivos campos

Nombre Tabla	**MODULOS**	
Descripción Tabla	Esta tabla encierra todos los datos de los módulos que abarca el sistema.	
Datos de los Campos		
Nombre Campo	**Tipo Dato**	**Descripción**
CO_MODULO	Varchar(50)	Código del modulo hijo.
NOMBRE_MODULO	Varchar(100)	Nombre de los módulos.
CO_MODULO_PADRE	Varchar(50)	Código del modulo padre.
URL	Varchar(200)	Dirección de la página a la cual puede acceder el módulo.

Tabla 4.7 Tabla PRIVILEGIO y sus respectivos campos

Nombre Tabla	PRIVILEGIO		
Descripción Tabla	Esta tabla encierra todos los datos necesarios para la asignación de los módulos a los cuales puede acceder un determinado usuario del sistema.		
Datos de los Campos			
Nombre Campo		Tipo Dato	Descripción
	USERNAME	Varchar(50)	Nombre de Usuario.
	CO_MODULO	Varchar(50)	Código del modulo al que puede acceder el usuario.

Tabla 4.8 Tabla APLICACION y sus respectivos campos

Nombre Tabla	APLICACION		
Descripción Tabla	Esta tabla encierra los datos de las aplicaciones registradas en el sistema.		
Datos de los Campos			
Nombre Campo		Tipo Dato	Descripción
	CO_APLICACION	Varchar(20)	Código de la aplicación.
	NOMBRE_APLICACION	Varchar(50)	Nombre de la aplicación.

Tabla 4.9 Tabla APLICFA y sus respectivos campos

Nombre Tabla	APLICFA		
Descripción Tabla	Esta tabla encierra los datos de las aplicaciones que fallan cuando se produce una determinada falla.		
Datos de los Campos			
Nombre Campo		Tipo Dato	Descripción
	CO_REGISTRO	Numeric(9)	Código que le asigna el sistema a cada falla.
	CO_APLICACION	Varchar(20)	Código de la aplicación.

Tabla 4.10 Tabla CRITICIDAD y sus respectivos campos

Nombre Tabla		**CRITICIDAD**	
Descripción Tabla		Esta tabla encierra los datos de los niveles de impacto que se le asignan a una determinada falla.	
Datos de los Campos			
Nombre Campo		**Tipo Dato**	**Descripción**
	CO_CRITICIDAD	Varchar(20)	Código de la criticidad.
	**CRITICIDAD**	Varchar(50)	Nombre asignado a la criticidad.

- **Scripts de la Base de Datos**: Para la realización de la base de datos se escogió SQL SERVER, por tal razón los Scripts que generan las tablas de dicha base de datos son sentencias SQL. Se emplea la sentencia "CREATE TABLE" seguido del nombre de la tabla a crear, acompañado por sus respectivos campos en orden de aparición y colocación dentro de la tabla, así como el tipo de dato que impera en ese campo, bien sea cadena de caracteres corta (CHAR), cadena de caracteres larga (VARCHAR), incluyendo la longitud de cada uno de ellos, y punto flotante (FLOAT) entre otros tipos de datos. También se indica cuales campos representan tanto clave primaria (PRIMARY KEY) como claves foráneas (FOREIGN KEY) en la tabla, como se describen a continuación:

```
CREATE TABLE FALLAS_APLICACION (
 CO_REGISTRO  NUMERIC(9) NOT NULL
, USERNAME VARCHAR(50) NULL
, USERNAME_CLIENTE VARCHAR(50) NULL
, NOMBRE_FALLA VARCHAR(200) NULL
, FECHA_REPORTE DATETIME(8) NULL
, CAUSA_FALLA VARCHAR(1000) NULL
, SOLUCION_APLICADA VARCHAR(5000) NULL
, APLIC_FALLANTE VARCHAR(50) NULL
, CRITICIDAD_FALLA VARCHAR(50) NULL
```

```
    , PRIMARY KEY (CO_REGISTRO));

CREATE TABLE  USUARIOS (
      USERNAME VARCHAR(50) NOT NULL
    , NOMBRE CHAR(20) NOT NULL
    , APELLIDO CHAR(20) NOT NULL
    , CARGO CHAR(30) NOT NULL
    , CLAVE VARCHAR(20)
    , PRIMARY KEY (USERNAME));

CREATE TABLE APLICFA (
      CO_REGISTRO NUMERIC(9) NOT NULL
    , CO_APLICACION VARCHAR(20)
    , FOREIGN KEY (CO_REGISTRO)
            REFERENCES FALLAS_APLICACION (CO_REGISTRO)
    , FOREIGN KEY (CO_APLICACION)
            REFERENCES APLICACION (CO_APLICACION));

CREATE TABLE APLLICACION (
      CO_APLICACION VARCHAR(20) NOT NULL
    , NOMBRE_APLICACION VARCHAR(50) NOT NULL
    , PRIMARY KEY (CO_APLICACION));

CREATE TABLE MODULOS (
      CO_MODULO VARCHAR(50) NOT NULL
    , NOMBRE_MODULO VARCHAR(100) NOT NULL
    , CO_MODULO_PADRE VARCHAR(50)
    , URL VARCHAR(200)
    , PRIMARY KEY (CO_MODULO));

CREATE TABLE CRITICIDAD (
      CO_CRITICIDAD VARCHAR(20) NOT NULL
    , CRITICIDAD VARCHAR(50) NOT NULL
    , PRIMARY KEY (CRITICIDAD));

CREATE TABLE PRIVILEGIO (
      USERNAME VARCHAR (50) NOT NULL
    , CO_MODULO VARCHAR(50)
    , FOREIGN KEY (USERNAME)
            REFERENCES USUARIOS (USERNAME)
    , FOREIGN KEY (CO_MODULO)
            REFERENCES MODULOS (CO_MODULO));
```

En la figura 4.6 se muestra el modelo relacional de la Base de Datos propio de esta aplicación donde se observan las diferentes tablas que interactúan en el sistema de base de datos escogido, cada una con sus respectivos nombres de tabla y con todos los campos denotados en las tablas anteriormente explicadas, y de las cuales se derivaron los diferentes Scripts que se describieron anteriormente. Se puede observar que los campos que representan claves principales o foráneas se encuentran indicadas con un tipo de letra un poco más oscura. De igual modo se pueden detallar las diversas relaciones de multiplicidad existente entre determinadas tablas, enlazadas por medio de sus claves, como por ejemplo la que existe entre la tabla FALLAS_APLICACION y la tabla USUARIOS, donde dicha relación expresa que un (1) Usuario atiende muchas (∞) fallas. Todo esto con la finalidad de diseñar una base de datos lo más compacta posible y con la menor redundancia de datos.

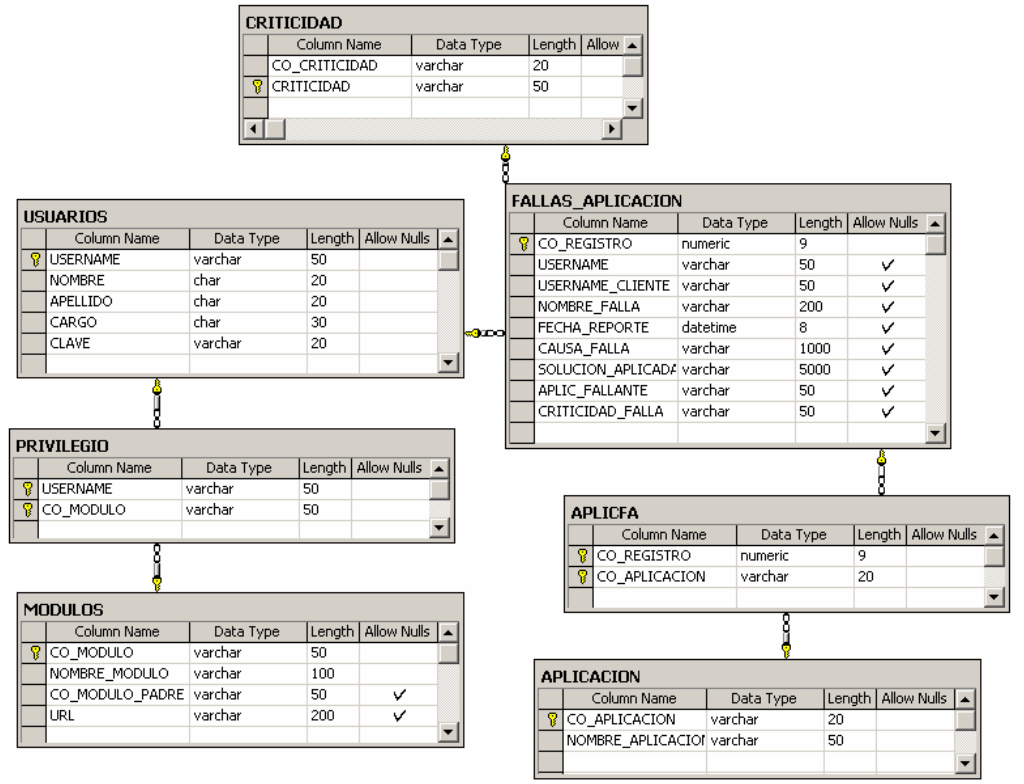


Figura 4.6 Modelo Relacional de la Base de Datos
Fuente: Propia, 2008

4.5.3 Arquitectura de Software

La arquitectura candidata seleccionada en la fase de inicio se conserva en la etapa presente, sin embargo en esta parte del proyecto se ha de detallar y especificar un poco más la misma, obteniéndose una versión mejorada. En esta sección se explica entre otras cosas, las razones por las cuales se seleccionó para este proyecto un determinado sistema manejador de base de datos relacionales, se explica el .Net Framework como arquitectura distribuida para desarrollar aplicaciones empresariales multinivel apta para ser empleada en este sistema en desarrollo, y finalmente se ahonda en el diseño de Modelos de Composición propios de WebML.

4.5.3.1 Sistema De Base De Datos

Con lo referente al diseño y administración de la base de datos a utilizar, en el mercado se encuentran disponibles diferentes tipos de Sistemas Manejadores de Bases de Datos Relacionales (RDBMS), de los cuales los más populares son Oracle, Microsoft SQL Server, MySQL, PostgreSQL y el que se va a emplear para este proyecto es Microsoft SQL Server, el cual tiene como características y propiedades que, además de ser un manejador de base de datos propietario, es un sistema que responde rápidamente a la demanda de transacciones en los periodos de mayor carga de trabajo, se amplía para acoger cantidades crecientes de información, posee las características de manejar pedidos de procesamiento de datos de forma exacta y eficiente, escalabilidad a medida que el negocio crece, y permite soporte para multiprocesamiento simétrico y procesadores masivamente paralelos, es decir, puede configurarse para correr en múltiples servidores y procesadores (maneja lo que se conoce como multihilos).

4.5.3.2 Visión De La Arquitectura .Net Framework

.Net Framework es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación
- Biblioteca de clases de .Net
- CLR (Common Language Runtime)

El .Net Framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes. Uno de esos lenguajes es el Visual Basic .Net, el cual se usa en el desarrollo del proyecto REFAP.

CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR. De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador.

Sin embargo, el código generado en MSIL no es código máquina y por tanto no puede ejecutarse directamente. Se necesita un segundo paso en el que una

herramienta denominada compilador JIT (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora. De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.

La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y, el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente.

4.5.3.3 Modelos de Composición O Gestión de Contenido Webml

Este modelo comprende el uso de operaciones, que sirven para expresar como se manejan los datos en el sistema, ya sea mediante el uso de bases de datos, variables globales, acceso a sistemas, transacciones, salidas de sesiones, transporte de datos, entre otros. El modelo de gestión de contenido o de composición nos otorga una visión más completa del funcionamiento intrínseco de la aplicación Web. El modelo de composición en WebML es el encargado de mostrar al usuario el contenido conceptual de las páginas que formarán parte de la aplicación Web a desarrollar, y lo hacen utilizando los SiteViews que conforman el sistema, las áreas por la que están formadas los SiteViews, las páginas que contiene cada área y por supuesto, el contenido de cada página utilizando los elementos que nos proporciona WebML.

El diseño de hipertextos es una de las actividades que consume más tiempo en esta fase de desarrollo de RUP. El mismo define cómo ha de ser el comportamiento del sistema en función de páginas Web, es decir, cómo será la navegación entre páginas interrelacionadas pertenecientes a un caso de uso específico del sitio Web de la aplicación. Es imperativo poseer ya un conocimiento absoluto de cómo funciona el sistema y de lo que se requiere específicamente. Durante esta actividad se diseñarán el conjunto de páginas que integrarán los SiteViews requeridos. En este tipo de

diagramas se observan ciertas unidades denominadas unidades de contenido, que indican los procesos a los que son sometidos los datos: unidades de entrada de datos, unidades de consulta de datos, unidades de índice, unidades de creación, modificación y eliminación de datos, transacciones, unidades de desplazamiento de datos y enlaces, entre otros.

A continuación se muestran algunos modelos de composición de ciertos casos de uso del software en construcción:

- **Diagrama de Composición o de Gestión de Contenido del Caso de Uso "Iniciar Sesión"**: este diagrama se detalla en la figura 4.7, y en dicho diagrama se puede observar el proceso que realizaría cualquier usuario del sistema (Administrador, Coordinador o Analista), para poder ingresar a la aplicación. El caso de uso a nivel de páginas Web comienza con la interfaz exterior del sistema, donde se observa, entre otras cosas, un formulario con dos campos para que el usuario coloque su nombre de usuario y su contraseña. Al hacer la solicitud de ingreso a la aplicación, el sistema consulta si existe un usuario registrado cuyo nombre de usuario y clave sean los indicados. Esto es llevado a cabo utilizando una unidad de consulta sencilla o simple encargada de buscar dichos datos en la tabla USUARIOS. En caso de que no se cumpla alguna de estas condiciones se muestra un mensaje de rechazo y se vuelve a mostrar la interfaz externa. De lo contrario, si cumple todos los requisitos y es un usuario registrado y autentificado, automáticamente se le permite ingresar al sistema para iniciar una sesión.

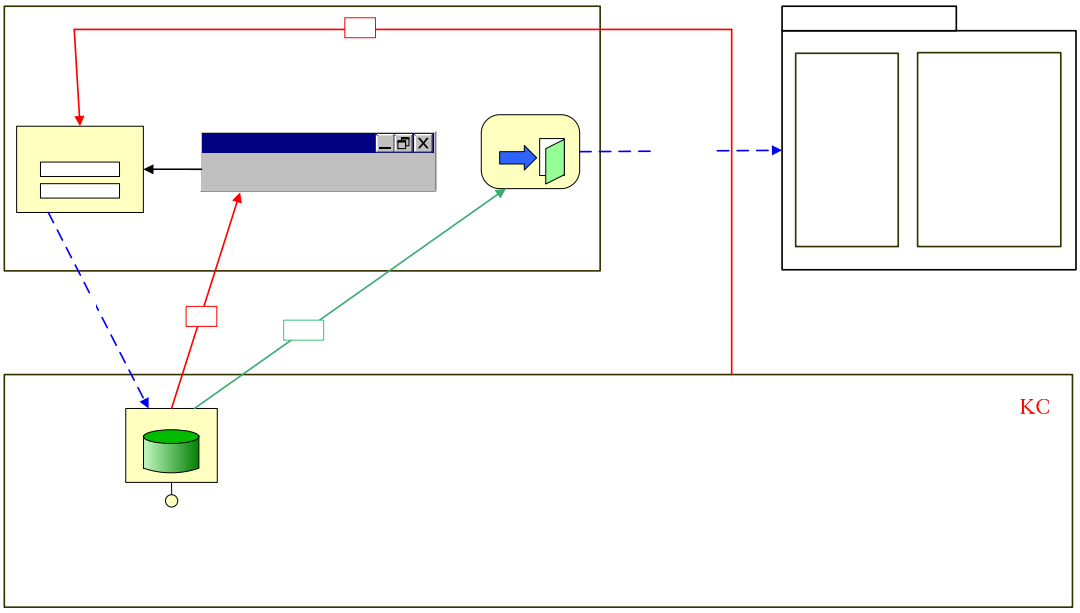


Figura 4.7 Diagrama de Composición del Caso de Uso "Iniciar Sesión"
Fuente: Propia, 2008

- **Diagrama de Composición o de Gestión de Contenido del Caso de Uso "Ingresar Usuario"**: En la figura 4.8 se puede observar el proceso que realiza el Administrador para ingresar un usuario en el sistema. En la interfaz se encuentra un menú principal único para el Administrador y un formulario el cual debe llenarle los campos y posteriormente solicita el registro o ingreso de los mismos, acción que lleva al sistema a consultar si existe un usuario que posea el USERNAME ingresado por el Administrador. Esta búsqueda se realiza empleando una unidad de consulta simple o sencilla la cual revisa en la tabla USUARIOS alguna ocurrencia del USERNAME ingresado en el formulario en cuestión. Si existe algún usuario que concuerde con esos datos se muestra un mensaje indicando la existencia del mismo. De lo contrario, una unidad de registro o ingreso de datos se encarga de registrar los datos del nuevo usuario ingresado por el Administrador a la base de datos.

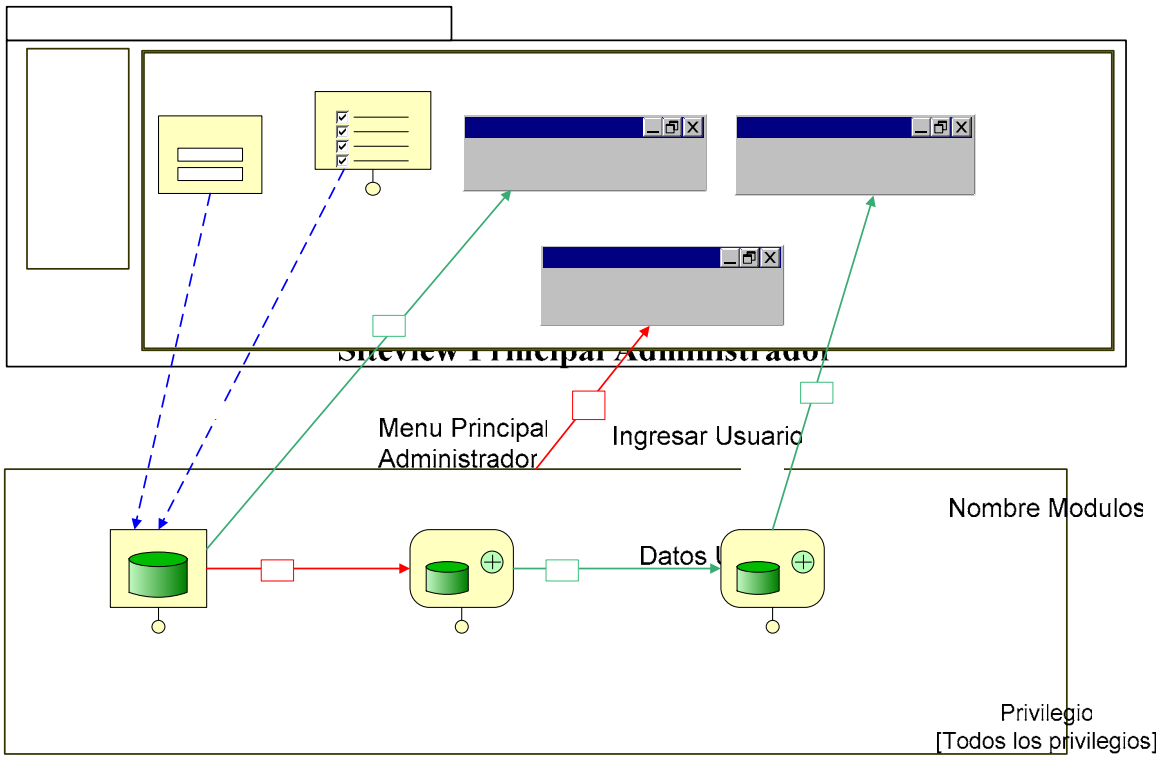


Figura 4.8 Diagrama de Composición del Caso de Uso "Ingresar Usuario"

Fuente: Propia, 2008

- **Diagrama de Composición o de Gestión de Contenido del Caso de Uso "Modificar Fallas"**: este caso de uso es realizado exclusivamente por un Analista, ya que es el tipo de actor autorizado para tal finalidad. Obsérvese detalladamente la figura 4.9, la cual corresponde a la realización completa a nivel de tecnología Web de este modelo de composición. Considérese que la página donde se encuentran todos los campos de datos de la falla a modificar se está observando actualmente en la interfaz del sistema. El Analista hace las modificaciones de rigor correspondientes a dicha falla y procede a solicitar y confirmar la modificación de los datos y una unidad de modificación de datos recibe los datos substitutivos, y dicha unidad se encarga de realizar el cambio de la información en la tablas FALLAS_APLICACION y APLICFA de la base

de datos, para luego devolverlos a la interfaz del sistema y ser colocados nuevamente en sus campos respectivos.

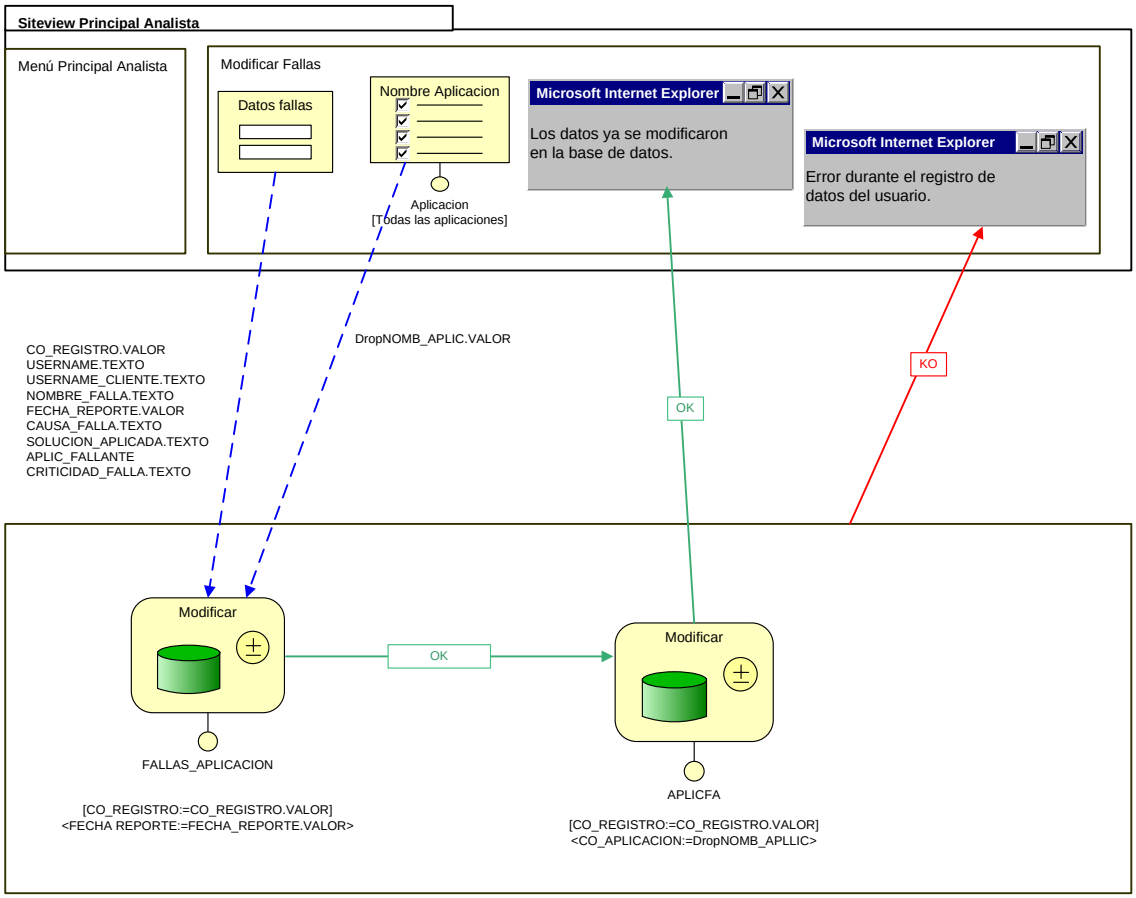


Figura 4.9 Diagrama de Composición del Caso de Uso "Modificar Fallas"
Fuente: Propia, 2008

- **Diagrama de Composición o de Gestión de Contenido del Caso de Uso "Eliminar Aplicacion"**: este modelo se puede observar en la figura 4.10. En la interfaz del sistema se muestra un menú único para el Administrador, además de un formulario en el cual el Administrador quien es el autorizado para llevar a cabo este proceso, selecciona la aplicación que desea eliminar, envía la respectiva solicitud, la confirma y se emplea una unidad de eliminación de datos para sacar del sistema dicha Aplicación.

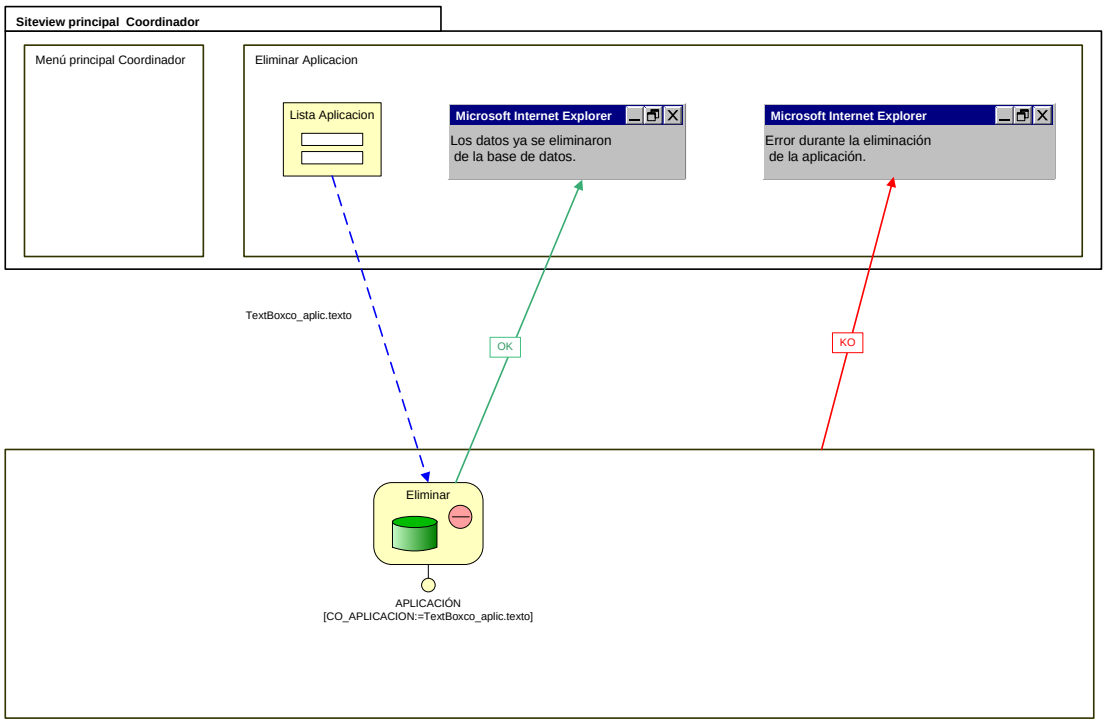


Figura 4.10 Diagrama de Composición del Caso de Uso "Eliminar Aplicacion"

Fuente: Propia, 2008

- **Diagrama de Composición o de Gestión de Contenido del Caso de Uso "Consultar fallas por Analista"**: en la figura 4.11 se puede observar este modelo. En la interfaz del sistema se muestra un menú único para el Analista, quien es el autorizado para llevar a cabo este proceso, además de un formulario en el cual selecciona el nombre del Analista y envía la solicitud de realizar la consulta. El sistema hace la consulta en la base de datos y luego de encontrar las fallas atendidas por el Analista en cuestión, el sistema devuelve todos los datos extraídos de la base de datos y los coloca en los campos respectivos de la nueva interfaz de Consultar fallas por Analista.

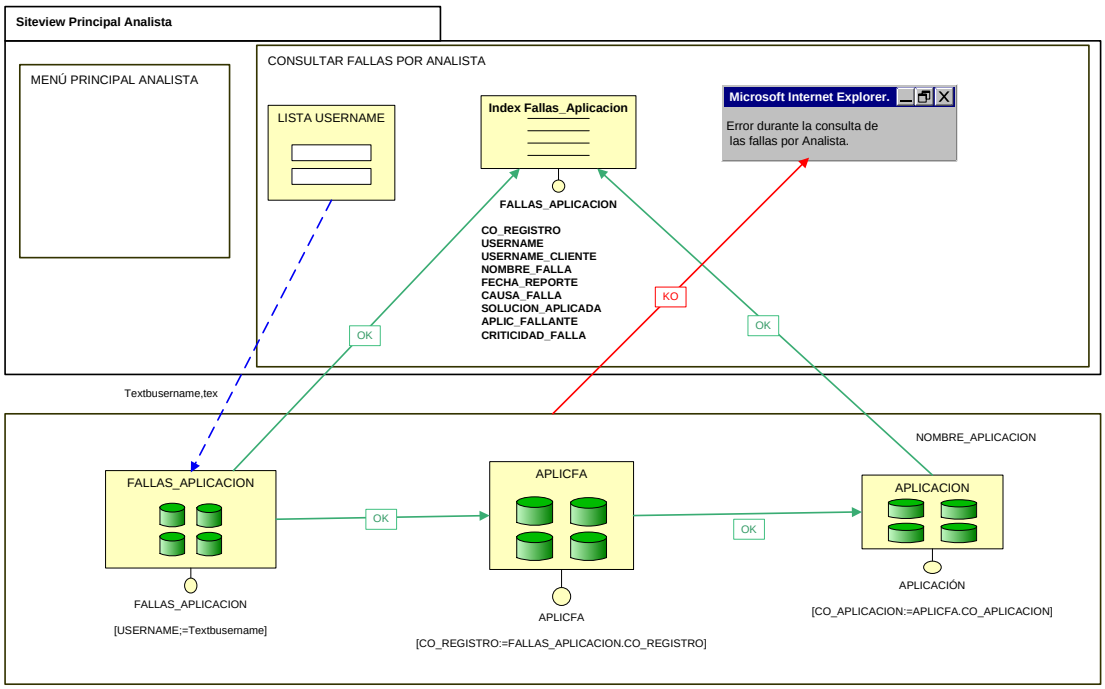


Figura 4.11 Diagrama de Composición del Caso de Uso "Consultar fallas por Analista"
Fuente: Propia, 2008

4.6 EVALUACIÓN DE LA FASE DE ELABORACIÓN

Al final de esta fase se encuentra el "Hito de Arquitectura del Sistema". En este punto se examina en detalle el alcance y objetivos del sistema, la selección de la arquitectura y se reafirman los riesgos obtenidos en la fase de inicio. Se requirió de una sola iteración para completar los objetivos planteados al inicio de la fase. Al revisar este hito aplicado al proyecto que actualmente se desarrolla se constataron ciertos criterios de evaluación, arrojándose los siguientes resultados:

- Se reafirma la viabilidad y estabilidad de la visión y los requerimientos del proyecto, así como de la arquitectura del mismo.
- El diseño de las bases de datos propias para el almacenamiento de la información relacionada con los procesos administrativos a ser manejados por el sistema en cuestión fue llevado a cabo de manera exitosa, al igual que el

diseño de la Arquitectura del Software, en función de las restricciones computacionales existentes.

- Los planes de iteración para la construcción son lo suficientemente detallados y fiables para permitir iniciar el trabajo de desarrollar la aplicación como tal.
- Todas las personas involucradas en el proyecto (Administrador, Coordinador, Analistas, diseñador de software) están de acuerdo con la visión actual del mismo.
- Se logró hacer un diseño efectivo de las páginas Web que conformarán el sistema mediante el uso del modelo Estructural y los modelos de Navegación de WebML.
- En definitiva se considera que ya se han obtenido todos los componentes necesarios para construir el sistema en la siguiente fase.

CAPÍTULO 5
FASE DE CONSTRUCCIÓN

La belleza de las personas no se encuentra en su exterior, se encuentra en los sentimientos que posee en su interior.

CAPÍTULO 5

FASE DE CONSTRUCCIÓN

El propósito principal de la fase de construcción es lograr un sistema completo de alta calidad, funcionalidad y costo efectivo, es decir, obtener una solución de software operativa, satisfaciendo todos los requerimientos planteados en las fases anteriores; por lo tanto se requieren etapas de diseño, implementación y pruebas que permitan alcanzar dicho propósito.

En esta fase se deben resolver los requerimientos restantes y completar el desarrollo del sistema sobre la arquitectura base. A esta altura del desarrollo muchos casos de uso posiblemente no hayan sido implementados del todo, sólo parcialmente o lo suficiente para probar algunas hipótesis y mitigar riesgos. A medida que se implementen más y más casos de uso, se refinarán los requerimientos para asegurar que la funcionalidad correcta será entregada. La fase de construcción del Proceso Unificado desarrolla los componentes del software que permiten que cada caso de uso sea operativo para los usuarios finales. Lograr esto requiere que los modelos de análisis y diseño, realizados durante la fase de elaboración, reflejen la versión final del software a construir.

En la fase de construcción se muestran tanto la codificación de las diferentes interfaces como lo referente a interacciones entre las diversas entidades u objetos del sistema, que igualmente deben de ser desarrollados para funcionar como un todo.

Para esta nueva fase de desarrollo del proyecto REFAP, los diagramas de secuencia descritos en la fase anterior y los diagramas WebML, son piezas fundamentales.

La versión del software que se obtendrá se denomina como “versión beta”, una versión lista para ser entregada a los usuarios finales de la aplicación.

5.1 DISEÑO

En esta fase se inicia la construcción del sistema REFAP, y se plantean las herramientas a emplear para el desarrollo y construcción de dicho sistema.

5.1.1 Herramientas de Desarrollo Empleadas

Las herramientas utilizadas para el diseño del software contribuyen al buen desempeño y correcto funcionamiento del mismo, entre ellas se encuentran:

- **SQL Server:** es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. *[Microsoft SQL Server, 2000]*

  Entre sus características figuran: Soporte de transacciones, gran estabilidad, gran seguridad, Escalabilidad, entre otras.
- **Visual Basic.Net:** Es un lenguaje que ofrece un modo fácil y rápido de crear aplicaciones Web. Además contiene numerosas características nuevas y mejoradas como la herencia, interfaces y sobrecarga entre otras.
- **XML:** XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. La idea que subyace bajo el XML es la de crear un lenguaje muy general que sirva para muchas cosas.
- **XSLT:** Es un leguaje que permite definir una presentación o formato para un documento XML.

5.2 IMPLEMENTACIÓN

En la etapa de implementación se elaboran las interfaces del programa, además se convierte el resultado del diseño en código fuente y en un sistema ejecutable.

5.2.1 Interfaces Gráficas de Usuario

Partiendo de los prototipos de interfaces diseñadas en la fase de elaboración, se desarrolla el resto de las interfaces que, al mejorarlas, constituyen las definitivas a utilizar por sus respectivos usuarios. A continuación se muestran algunas de las interfaces finales que formarán parte definitiva de la aplicación en desarrollo, clasificadas según el tipo de usuario:

- **Interfaz de "Principal Externa" o "Gestionar Sesión":** Es la pantalla inicial del sistema donde los usuarios se identifican para ser autenticados e iniciar su sesión. Es una interfaz que no sólo incluye un logotipo que identifica a la empresa PDVSA PETROCEDEÑO, sino que además incluye los cuadros de texto receptores de información propia de los usuarios registrados, como son el nombre de usuario y su contraseña. También presenta la opción de cambiar la clave de ingreso al sistema, tal como se observa en la figura 5.1.

Figura 5.1. Interfaz "Principal Externa" o de "Gestionar Sesión"
Fuente: REFAP, 2008

- **Interfaz "Principal para el Administrador del sistema REFAP":** Es formada por una página que a sus vez enmarca tres páginas más: la página superior presenta un logo que identifica a la empresa PDVSA PETROCEDEÑO, acompañado del nombre del sistema REFAP, un link que permite que el Administrador pueda salir del sistema en el momento que lo desee, el día y fecha actualizada. La página izquierda inferior posee un menú desplegable por medio del cual el Administrador puede navegar y trasladarse hacia las diferentes secciones de la aplicación en cuestión que le son permitidas, y finalmente la página inferior derecha que inicialmente presenta una imagen de una de las áreas de PDVSA PETROCEDEÑO y que constituye la página de información o de contenido, en donde se muestran las interfaces solicitadas según la opción seleccionada por el Administrador en el menú

desplegable mencionado anteriormente, tal como se puede apreciar a continuación en la en la figura 5.2:

Figura 5.2. Interfaz "Principal para el Administrador"
Fuente: REFAP, 2008

- **Interfaz "Gestionar Usuarios" para el Administrador del sistema REFAP:** haciendo uso de esta página el Administrador de REFAP, quien es el usuario capacitado para llevar a cabo este caso de uso, puede ingresar, modificar y eliminar los datos de un usuario en particular. Para registrar un nuevo usuario es obligatorio que se introduzcan todos los campos requeridos en el formulario, en caso de que se omita algún dato necesario el sistema mostrará un mensaje al presionar el botón adecuado, indicando la falta del mismo. De igual modo, al ser exitoso el registro del usuario, la aplicación emitirá un mensaje anunciando el ingreso definitivo del mismo en el sistema. Al modificar los datos de un

usuario o al intentar eliminar del sistema un usuario, la aplicación envía un mensaje solicitando la confirmación de la acción a realizar. Véase la interfaz en la figura 5.3.

Figura 5.3. Interfaz "Gestionar Usuarios" para el Administrador de REFAP
Fuente: REFAP, 2008

- **Interfaz "Gestionar Fallas" para Analistas del sistema REFAP:** haciendo uso de esta página, cualquier Analista del sistema REFAP, puede ingresar, modificar y eliminar los datos de una falla en particular. Para registrar una nueva falla es obligatorio que se introduzcan todos los campos requeridos en el formulario, en caso de que se omita algún dato necesario el sistema mostrará un mensaje al presionar el botón adecuado, indicando la falta del mismo. De igual modo, al ser exitoso el registro de la falla, la aplicación emitirá un mensaje anunciando el ingreso definitivo de la misma al sistema. Al modificar

los datos de una falla o al intentar eliminar del sistema una falla, la aplicación envía un mensaje solicitando la confirmación de la acción a realizar. Véase la interfaz en la figura 5.4.

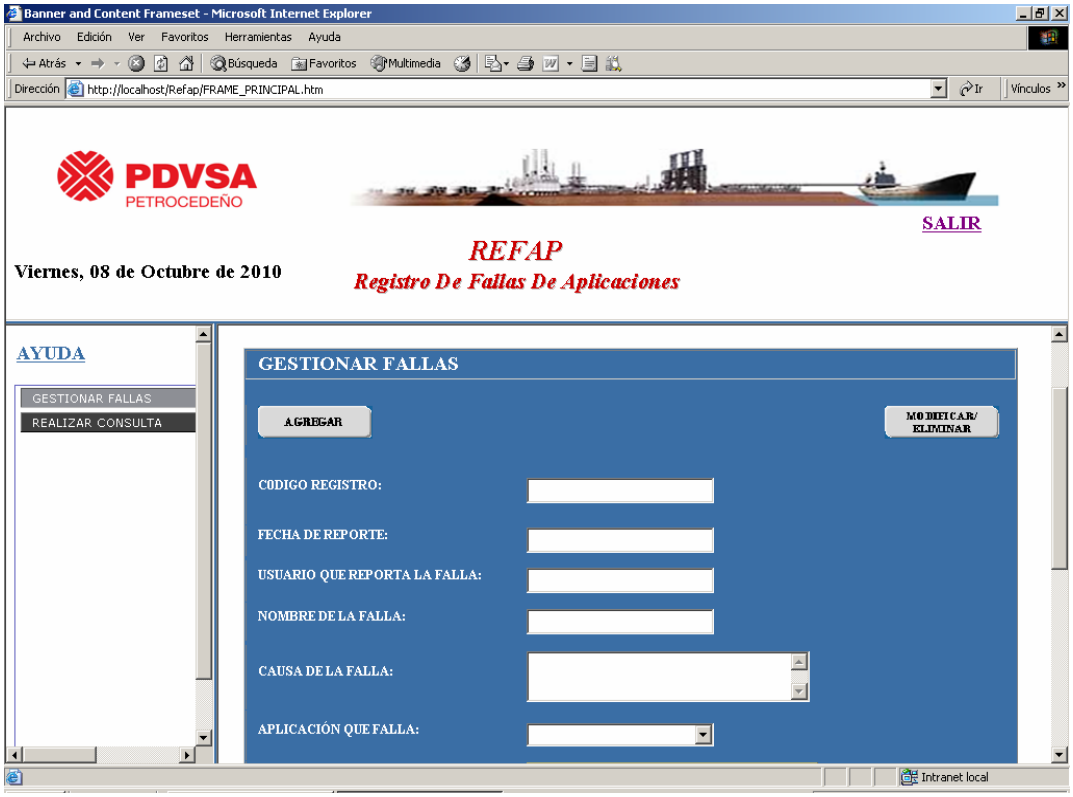


Figura 5.4. Interfaz "Gestionar Fallas" para Analistas del sistema REFAP
Fuente: REFAP, 2008

En la figura anterior no se puede observar completamente el formulario de Gestionar Fallas, debido al espacio que abarca. A continuación se muestra en la figura 5.5 el formulario en cuestión.

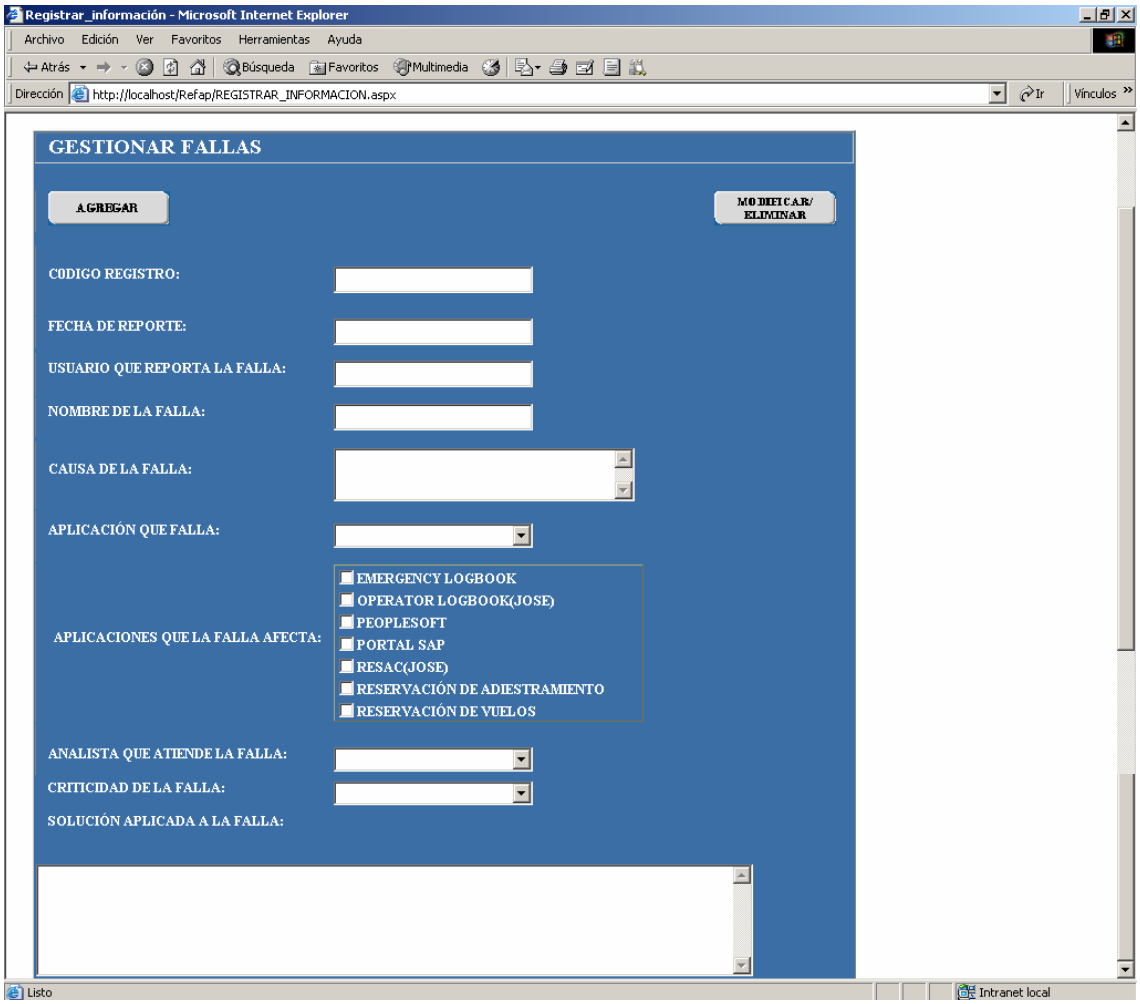


Figura 5.5. Interfaz "Formulario de Gestionar Fallas" para Analistas del sistema REFAP
Fuente: REFAP, 2008

- **Interfaz "Consultar Fallas por Analista" para Analistas del sistema REFAP:** al seleccionar esta opción se muestra en el área de información o de contenido una ventana que contiene una lista en la cual se puede seleccionar el nombre del usuario del Analista a consultar y enviar la solicitud respectiva. Luego se muestra una tabla con todos los datos de las fallas atendidas por dicho Analista. Esta tabla contiene un link en la columna denominada Detalles, mediante el cual se puede observar en otra tabla el contenido de tres campos (causa falla, solución aplicada a la falla y aplicaciones que la falla afecta),

debido a que estos campos generalmente tienen gran contenido de información. Lo expuesto anteriormente se puede observar en la figura 5.6. y figura 5.7.

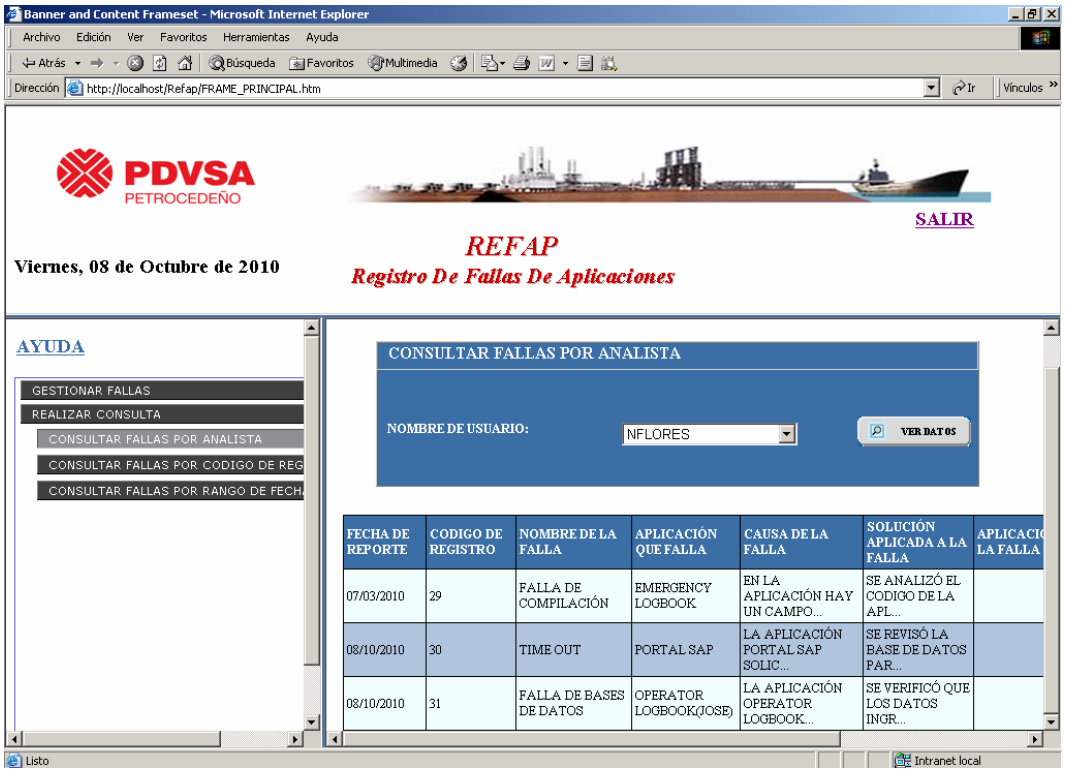


Figura 5.6. Interfaz "Consultar Fallas por Analista" para Analistas del sistema REFAP
Fuente: REFAP, 2008

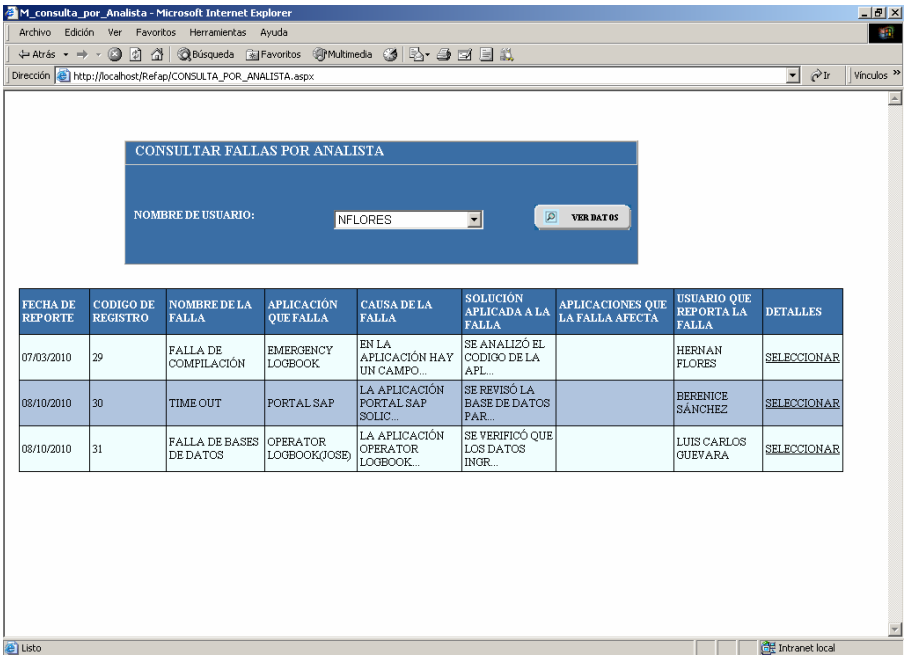


Figura 5.7 Interfaz "Consultar Fallas por Analista" para Analistas del sistema REFAP (Continuación)
Fuente: REFAP, 2008

- **Interfaz "Procesar Reporte Fallas por Rango de Fechas" para el Coordinador del sistema REFAP:** Al seleccionar esta opción se observa en la página de información una ventana que contiene dos campos, uno para la fecha de inicio de la consulta y el otro para la fecha final. Al lado de cada campo mencionado anteriormente se encuentra un calendario para escoger las dichas fechas. Al enviar la solicitud de consulta el sistema verifica que las fechas ingresadas estén en un rango válido, de ser así, se muestra el reporte que contempla todas las fallas registradas en el rango de fechas ingresado, ordenadas por fecha y código de registro. Ver figura 5.8 y 5.9.

Figura 5.8. Interfaz "Procesar Reporte Fallas por Rango de Fechas" para el Coordinador del sistema REFAP.
Fuente: REFAP, 2008

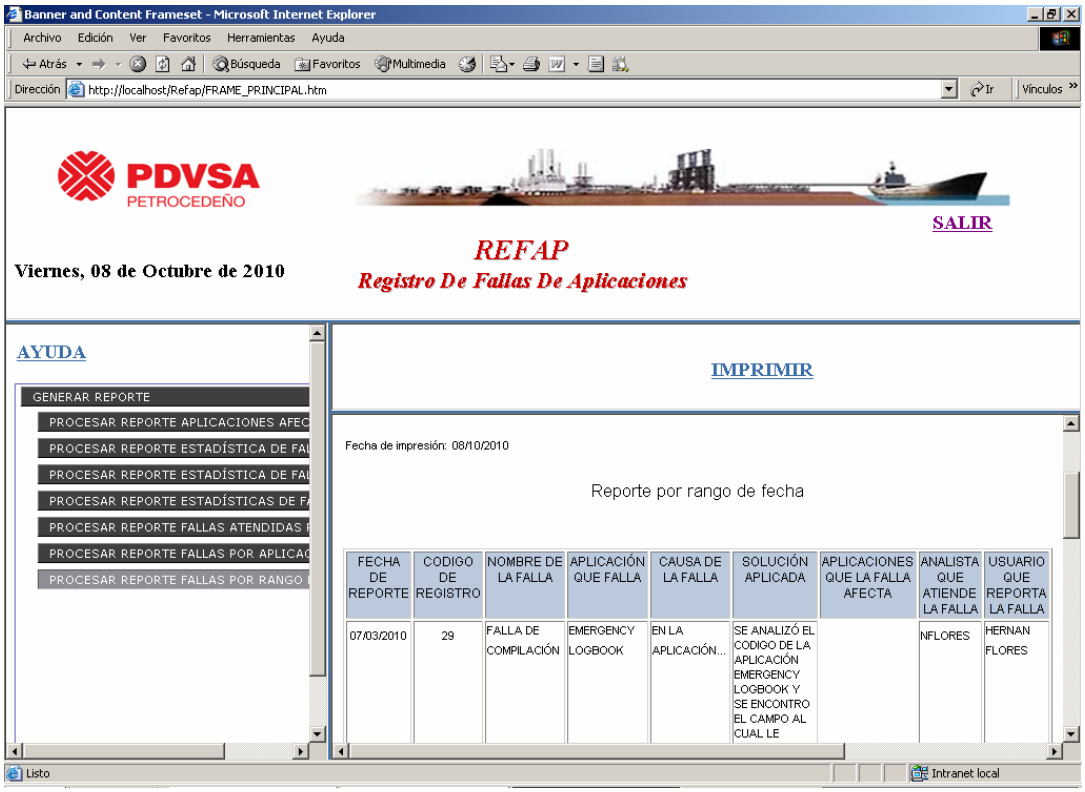


Figura 5.9. Interfaz "Reporte Fallas por Rango de Fechas" para el Coordinador del sistema REFAP.
Fuente: REFAP, 2008

5.2.2 Código Fuente

Las páginas Web que han sido mostradas se deben implementar en código fuente. Para la codificación de las páginas se usará el lenguaje de programación Visual Basic.Net. Las operaciones de base datos serán gestionadas por el Sistema Manejador de Base de Datos Relacionales SQL SERVER. Los reportes serán realizados con XML y XSLT. La integración de todos estos elementos formará la arquitectura de trabajo estable que se diseñó en la fase de Elaboración de RUP.

A continuación se observa una muestra del código fuente de la página Gestionar Fallas, incluyendo los códigos de formulario en HTML, y validaciones de campos usando JavaScript.

Código Visual Basic.Net del caso de uso Gestionar Fallas:

```
'Librerías que se  importan o  incluyen para que  la clase o
'función se lleve a cabo sin inconvenientes

Imports Microsoft.VisualBasic
Imports System
Imports System.Windows
Imports System.Timers
Imports System.Security.Cryptography
Imports System.Text

'Declaración de la clase Gestionar Fallas

Public Class Gestionar_Fallas
   Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

   'Requerida para el diseñador de formularios Web.

   <System.Diagnostics.DebuggerStepThrough()>

Private Sub InitializeComponent()

 End Sub

   Dim arreglo(100) As String
   Protected WithEvents Dropco_registro As System.Web.UI.WebControls.DropDownList
   Protected WithEvents Label13 As System.Web.UI.WebControls.Label
   Protected WithEvents TextBusu_reportant As System.Web.UI.WebControls.TextBox
   Protected WithEvents TextBoxnom_falla As System.Web.UI.WebControls.TextBox
   Protected WithEvents Form1 As System.Web.UI.HtmlControls.HtmlForm
   Protected WithEvents Label16 As System.Web.UI.WebControls.Label
   Protected WithEvents TextBox4 As System.Web.UI.WebControls.TextBox
   Protected WithEvents Label1 As System.Web.UI.WebControls.Label
   Protected WithEvents Ibtmodeli As System.Web.UI.WebControls.ImageButton
   Protected WithEvents Ibtagre As System.Web.UI.WebControls.ImageButton
   Protected WithEvents Ibtsalvar As System.Web.UI.WebControls.ImageButton
   Protected WithEvents DropDCO_REGISTRO As System.Web.UI.WebControls.DropDownList
   Protected WithEvents TextBco_registro As System.Web.UI.WebControls.TextBox
   Protected WithEvents Label2 As System.Web.UI.WebControls.Label
   Protected WithEvents Ibtver_datos As System.Web.UI.WebControls.ImageButton
   Protected WithEvents Label3 As System.Web.UI.WebControls.Label
   Protected WithEvents Ibtmodificar As System.Web.UI.WebControls.ImageButton
   Protected WithEvents TextBusu_reportante As System.Web.UI.WebControls.TextBox
   Protected WithEvents Label5 As System.Web.UI.WebControls.Label
   Protected WithEvents Ibteliminar As System.Web.UI.WebControls.ImageButton
   Protected WithEvents Label6 As System.Web.UI.WebControls.Label
   Protected WithEvents Textcausa_falla As System.Web.UI.WebControls.TextBox
```

```
    Protected WithEvents Label8 As System.Web.UI.WebControls.Label
    Protected WithEvents DropDapli_fallante As System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label9 As System.Web.UI.WebControls.Label
    Protected WithEvents CheckBoxapli_afect As System.Web.UI.WebControls.CheckBoxList
    Protected WithEvents Label10 As System.Web.UI.WebControls.Label
    Protected WithEvents DropDanalista As System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label11 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBoxapli_fallante As System.Web.UI.WebControls.TextBox
    Protected WithEvents TextBanalista As System.Web.UI.WebControls.TextBox
    Protected WithEvents ImageButton1 As System.Web.UI.WebControls.ImageButton
    Protected WithEvents TextBfecha As System.Web.UI.WebControls.TextBox
    Protected WithEvents TextBnom_falla As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label7 As System.Web.UI.WebControls.Label
    Protected WithEvents Dropcriticidad As System.Web.UI.WebControls.DropDownList
    Protected WithEvents Textcriticidad As System.Web.UI.WebControls.TextBox
    Protected WithEvents Textsolucion As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label15 As System.Web.UI.WebControls.Label
    Protected WithEvents Calendar As System.Web.UI.WebControls.Calendar


 'Declaración requerida por el diseñador Web.
    'No se puede borrar ni mover de aqui.

Private designerPlaceholderDeclaration As System.Object

   Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Init

'Método necesario para admitir el Diseñador. No se puede modificar el contenido del método con el
editor de código.
            InitializeComponent()

 'Evento que ocurre al iniciarse la página.

           CargarListaAplicacion()
 End Sub

#End Region

'Declarando la variable strcon que permite la conexión con la base de
datos.
 Protected strcon As String = System.Configuration.ConfigurationSettings.AppSettings("strcon")

'Declaración que permite que se visualize los mensajes en la
aplicación.
Protected WithEvents MsgBox As MsgBox.MsgBox


    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
       If Not Page.IsPostBack Then
```

```
        End If
    End Sub


'Carga la lista de las aplicaciones existentes en la base de datos.

Sub CargarListaAplicacion()
      Dim str1, str2, sqlvar, modulo As String
       Dim cnn1 As OleDb.OleDbConnection
       Dim resultadoSQL As OleDb.OleDbDataAdapter
       Dim dataset As New DataSet
       Dim numero As Integer
       Dim valor As Boolean
       Dim codigo As String

       cnn1 = New OleDb.OleDbConnection(strcon)
       cnn1.Open()
       sqlvar = "SELECT * FROM APLICACION ORDER BY NOMBRE_APLICACION"
       resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
       valor = resultadoSQL.Fill(dataset, "Categ")
       If valor Then
          numero = dataset.Tables("Categ").Rows.Count
          For numero = 0 To dataset.Tables("Categ").Rows.Count - 1
             modulo =dataset.Tables("Categ").Rows(numero).Item("NOMBRE_APLICACION")
             arreglo(numero) = dataset.Tables("Categ").Rows(numero).Item("CO_APLICACION")
             Me.CheckBoxapli_afect.Items.Add(modulo)
          Next numero
       End If
       cnn1.Close()
    End Sub


'Proceso que abarca tres opciones(Ingresar, Modificar, y Eliminar ).

 Private Sub MsgBox_YesChoosed(ByVal sender As Object, ByVal key As String) Handles MsgBox.YesChoosed
Select Case key
          Case "A"
'Declaración de variables.

             Dim numero As Integer
             Dim miOleDbConexion As OleDb.OleDbConnection, cnn1, cnn3 As OleDb.OleDbConnection
             Dim miOleDbCommand1, miOleDbCommand2 As OleDb.OleDbCommand
             Dim resultadoSQL As OleDb.OleDbDataAdapter
             Dim sqlInsert, sqlInsert2, str1, str2, str3, sqlvar, registro, modulo As String
             Dim dataset As New DataSet
             Dim ejecutar As Integer
             Dim total, i As Integer
             Dim valor As Boolean
             Dim valor1, valor2 As Boolean
             Dim cnn2 As OleDb.OleDbConnection
             Dim sqlvar1, User, Userp, tapli, tanali, st As String
```

```
        Dim resultadoSQL1 As OleDb.OleDbDataAdapter
        Dim datase As New DataSet
        Dim apli, var_fecha As String
        Dim codigo, apli_vacia As String
        Dim dataset1 As New DataSet
        Dim dataset2 As New DataSet
        Dim dataset3 As New DataSet
        Dim dataset4 As New DataSet
        Dim iden As Long

'Se le otorga visibilidad al boton.

        Me.ImageButton1.Visible = True


var_fecha = IIf(Day(Date.Today) < 10, "0", "") + Day(Date.Today).ToString + "/" + IIf(Month(Date.Today) < 10, "0", "") +
Month(Date.Today).ToString + "/" + Year(Date.Today).ToString

'Se realiza la conexión con la base de datos.

        cnn1 = New OleDb.OleDbConnection(strcon)
        cnn1.Open()
        User = Request.Form("TextBco_registro")
        tanali = Request.Form("DropDanalista")
        tapli = Request.Form("DropDapli_fallante")
        st = Request.Form("TextBfecha")
        str3 = comparar_registrar(st, var_fecha)

        If str3 = 1 Then
           UserMsgBox("No puede registrar fallas con fecha superior a la actual.")
        Else

'Permite insertar los datos en la base de datos.
            sqlInsert = "INSERT INTO FALLAS_APLICACION
(USERNAME,USERNAME_CLIENTE,NOMBRE_FALLA,FECHA_REPORTE,CAUSA_FALLA,
SOLUCION_APLICADA, APLIC_FALLANTE, CRITICIDAD_FALLA) VALUES ('"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("DropDanalista")), VbStrConv.UpperCase) + "',
'"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("TextBusu_reportante")),
VbStrConv.UpperCase) + "', '"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("TextBnom_falla")), VbStrConv.UpperCase) +
"', '"
            sqlInsert = sqlInsert + convierte_fecha_guardar(Request.Form("TextBfecha")) + "', '"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("Textcausa_falla")), VbStrConv.UpperCase) + "',
'"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("Textsolucion")), VbStrConv.UpperCase) + "', '"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("DropDapli_fallante")), VbStrConv.UpperCase)
+ "', '"
            sqlInsert = sqlInsert + StrConv(Trim(Request.Form("Dropcriticidad")), VbStrConv.UpperCase) + "')"
            miOleDbCommand1 = New OleDb.OleDbCommand(sqlInsert, cnn1)
            ejecutar = miOleDbCommand1.ExecuteNonQuery()
            sqlvar = "SELECT @@IDENTITY as 'identity'"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset3, "Categ")
```

```
            If valor Then
               numero = dataset3.Tables("Categ").Rows.Count
               For numero = 0 To dataset3.Tables("Categ").Rows.Count - 1
                  iden = dataset3.Tables("Categ").Rows(numero).Item("identity")
               Next numero
            End If

             With Me

'Se registran en la table APLICFA, las aplicaciones que han sido afectasdas por una falla.
               With .CheckBoxapli_afect
                  total = .Items.Count
                  For i = 0 To total - 1
                     valor1 = Me.CheckBoxapli_afect.Items(i).Selected()
                     If Me.CheckBoxapli_afect.Items(i).Selected() = True Then
                        sqlInsert = "INSERT INTO APLICFA VALUES('"
                        sqlInsert = sqlInsert + Str(iden) + "', '"
                        sqlInsert = sqlInsert + arreglo(i) + "')"
                        miOleDbCommand1 = New OleDb.OleDbCommand(sqlInsert, cnn1)
                        ejecutar = miOleDbCommand1.ExecuteNonQuery()
                     End If
                  Next i
               End With

'Se actualiza la lista de usuarios.

               With .DropDanalista
                  numero = .Items.Count
                  For i = 0 To numero - 1
                     User = .Items(0).Text
                     .Items.Remove(User)
                     arreglo(i) = ""
                  Next i
                  cnn1 = New OleDb.OleDbConnection(strcon)
                  cnn1.Open()
                  sqlvar = "SELECT * FROM USUARIOS ORDER BY USERNAME"
                  resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
                  valor = resultadoSQL.Fill(dataset, "Categ")
                  If valor Then
                     numero = dataset.Tables("Categ").Rows.Count
                     For numero = 0 To dataset.Tables("Categ").Rows.Count - 1
                        registro = dataset.Tables("Categ").Rows(numero).Item("USERNAME")
                        Me.DropDanalista.Items.Add(registro)
                     Next numero
                  End If
                  cnn1.Close()
               End With

'Se actualiza la lista de los registros de las fallas.

               With .DropDCO_REGISTRO
                  numero = .Items.Count
```

```
                    For i = 0 To numero - 1
                        User = .Items(0).Text
                        .Items.Remove(User)
                        arreglo(i) = ""
                    Next i
                    cnn1 = New OleDb.OleDbConnection(strcon)
                    cnn1.Open()
                    sqlvar = "SELECT * FROM FALLAS_APLICACION ORDER BY
CO_REGISTRO"
                    resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
                    valor = resultadoSQL.Fill(dataset1, "Categ")
                    If valor Then
                        numero = dataset1.Tables("Categ").Rows.Count
                        For numero = 0 To dataset1.Tables("Categ").Rows.Count - 1
                            registro = dataset1.Tables("Categ").Rows(numero).Item("CO_REGISTRO")
                            Me.DropDCO_REGISTRO.Items.Add(registro)
                        Next numero
                    End If
                    cnn1.Close()
                End With

'Se actualiza la lista de criticidades.

                With .Dropcriticidad
                    numero = .Items.Count
                    For i = 0 To numero - 1
                        User = .Items(0).Text
                        .Items.Remove(User)
                        arreglo(i) = ""
                    Next i
                    cnn1 = New OleDb.OleDbConnection(strcon)
                    cnn1.Open()
                    sqlvar = "SELECT * FROM CRITICIDAD ORDER BY CRITICIDAD"
                    resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
                    valor = resultadoSQL.Fill(dataset4, "Categ")
                    If valor Then
                        numero = dataset4.Tables("Categ").Rows.Count
                        For numero = 0 To dataset4.Tables("Categ").Rows.Count - 1
                            registro = dataset4.Tables("Categ").Rows(numero).Item("CRITICIDAD")
                            Me.Dropcriticidad.Items.Add(registro)
                        Next numero
                    End If
                    cnn1.Close()
                End With

'Se actualiza la lista de las aplicaciones.

                With .DropDapli_fallante
                    numero = .Items.Count
                    For i = 0 To numero - 1
                        User = .Items(0).Text
```

```
                    .Items.Remove(User)
                    arreglo(i) = ""
                Next i
                cnn1 = New OleDb.OleDbConnection(strcon)
                cnn1.Open()
                sqlvar = "SELECT * FROM APLICACION ORDER BY
NOMBRE_APLICACION"
                resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
                valor = resultadoSQL.Fill(dataset2, "Categ")
                If valor Then
                    numero = dataset2.Tables("Categ").Rows.Count
                    For numero = 0 To dataset2.Tables("Categ").Rows.Count - 1
                        registro =
dataset2.Tables("Categ").Rows(numero).Item("NOMBRE_APLICACION")
                        Me.DropDapli_fallante.Items.Add(registro)
                    Next numero
                End If

                UserMsgBox("Los datos ya se cargaron en la Base de Datos")
                Me.TextBco_registro.Visible = True
                Me.Label2.Visible = True
                Me.TextBco_registro.Text = iden
                Me.DropDanalista.Visible = False
                Me.DropDapli_fallante.Visible = False
                Me.TextBoxapli_fallante.Visible = True
                Me.TextBanalista.Visible = True
                Me.TextBoxapli_fallante.Text = tapli
                Me.TextBanalista.Text = tanali

                cnn1.Close()
            End With
        End With
        cnn1.Close()
    End If

Case "B"
    Dim numero As Integer
    Dim miOleDbConexion, cnn1, cnn2 As OleDb.OleDbConnection
    Dim miOleDbCommand1, miOleDbCommand2 As OleDb.OleDbCommand
    Dim resultadoSQL As OleDb.OleDbDataAdapter
    Dim sqlInsert, sqlInsert2, str1, str2, sqlvar, sqlvar1, st, str3, USER, registro As String
    Dim dataset As New DataSet
    Dim dataset1 As New DataSet
    Dim dataset2 As New DataSet
    Dim dataset4 As New DataSet
    Dim valor1 As Boolean
    Dim ejecutar, total, i, USE, USE1, USE2 As Integer
    Dim valor As Boolean
    Dim codigo, var_fecha, criti As String

    Me.ImageButton1.Visible = True
```

```
            var_fecha = IIf(Day(Date.Today) < 10, "0", "") + Day(Date.Today).ToString + "/" +
IIf(Month(Date.Today) < 10, "0", "") + Month(Date.Today).ToString + "/" +
Year(Date.Today).ToString

            cnn1 = New OleDb.OleDbConnection(strcon)
            cnn1.Open()
            USE = Me.DropDCO_REGISTRO.SelectedIndex
            str1 = Me.DropDanalista.SelectedIndex
            str2 = Me.DropDapli_fallante.SelectedIndex
            criti = Me.Dropcriticidad.SelectedIndex

            st = Request.Form("TextBfecha")
            str3 = comparar_registrar(st, var_fecha)

            If str3 = 1 Then
               UserMsgBox("No puede registrar fallas con fecha superior a la actual.")
            Else
'Permite modificar los datos en la base de datos.

               sqlInsert = "update FALLAS_APLICACION "
               sqlInsert = sqlInsert + "SET USERNAME = '" +
StrConv(Trim(Request.Form("DropDanalista")), VbStrConv.UpperCase) + "', "
               sqlInsert = sqlInsert + "USERNAME_CLIENTE = '" +
StrConv(Trim(Request.Form("TextBusu_reportante")), VbStrConv.UpperCase) + "', "
               sqlInsert = sqlInsert + "NOMBRE_FALLA = '" +
StrConv(Trim(Request.Form("TextBnom_falla")), VbStrConv.UpperCase) + "', "
               sqlInsert = sqlInsert + "FECHA_REPORTE = '" +
convierte_fecha(Request.Form("TextBfecha")) + "', "
               sqlInsert = sqlInsert + "CAUSA_FALLA = '" +
StrConv(Trim(Request.Form("Textcausa_falla")), VbStrConv.UpperCase) + "', "
               sqlInsert = sqlInsert + "SOLUCION_APLICADA = '" +
StrConv(Trim(Request.Form("Textsolucion")), VbStrConv.UpperCase) + "', "
               sqlInsert = sqlInsert + "APLIC_FALLANTE = '" +
StrConv(Trim(Request.Form("DropDapli_fallante")), VbStrConv.UpperCase) + "', "
               sqlInsert = sqlInsert + "CRITICIDAD_FALLA = '" +
StrConv(Trim(Request.Form("Dropcriticidad")), VbStrConv.UpperCase) + "' "
               sqlInsert = sqlInsert + " where CO_REGISTRO = '" +
Trim(Request.Form("DropDCO_REGISTRO")) + "' "
               miOleDbCommand1 = New OleDb.OleDbCommand(sqlInsert, cnn1)
               ejecutar = miOleDbCommand1.ExecuteNonQuery()
               sqlvar = "delete from APLICFA where APLICFA.CO_REGISTRO = '" &
Request.Form("DropDCO_REGISTRO") & "'"
               miOleDbCommand1 = New OleDb.OleDbCommand(sqlvar, cnn1)
               ejecutar = miOleDbCommand1.ExecuteNonQuery()

'Permite modificar  las aplicaciones que han sido afectadas .

               With Me.CheckBoxapli_afect
                  total = .Items.Count
                  For i = 0 To total - 1
```

```
        valor1 = Me.CheckBoxapli_afect.Items(i).Selected()
        If valor1 Then
            sqlInsert = "INSERT INTO APLICFA VALUES('"
            sqlInsert = sqlInsert + Request.Form("DropDCO_REGISTRO") + "', '"
            sqlInsert = sqlInsert + arreglo(i) + "')"
            miOleDbCommand1 = New OleDb.OleDbCommand(sqlInsert, cnn1)
            ejecutar = miOleDbCommand1.ExecuteNonQuery()
        End If
    Next i
End With

With Me.DropDanalista
    numero = .Items.Count
    For i = 0 To numero - 1
        USER = .Items(0).Text
        .Items.Remove(USER)
        arreglo(i) = ""
    Next i
    cnn1 = New OleDb.OleDbConnection(strcon)
    cnn1.Open()
    sqlvar = "SELECT * FROM USUARIOS ORDER BY USERNAME"
    resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
    valor = resultadoSQL.Fill(dataset, "Categ")
    If valor Then
        numero = dataset.Tables("Categ").Rows.Count
        For numero = 0 To dataset.Tables("Categ").Rows.Count - 1
            registro = dataset.Tables("Categ").Rows(numero).Item("USERNAME")
            Me.DropDanalista.Items.Add(registro)
        Next numero
    End If
    cnn1.Close()
End With

With Me.DropDCO_REGISTRO
    numero = .Items.Count
    For i = 0 To numero - 1
        USER = .Items(0).Text
        .Items.Remove(USER)
        arreglo(i) = ""
    Next i
    cnn1 = New OleDb.OleDbConnection(strcon)
    cnn1.Open()
    sqlvar = "SELECT * FROM FALLAS_APLICACION ORDER BY CO_REGISTRO"
    resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
    valor = resultadoSQL.Fill(dataset1, "Categ")
    If valor Then
        numero = dataset1.Tables("Categ").Rows.Count
        For numero = 0 To dataset1.Tables("Categ").Rows.Count - 1
            registro = dataset1.Tables("Categ").Rows(numero).Item("CO_REGISTRO")
            Me.DropDCO_REGISTRO.Items.Add(registro)
        Next numero
```

```
            End If
            cnn1.Close()
        End With

        With Me.Dropcriticidad
            numero = .Items.Count
            For i = 0 To numero - 1
                USER = .Items(0).Text
                .Items.Remove(USER)
                arreglo(i) = ""
            Next i
            cnn1 = New OleDb.OleDbConnection(strcon)
            cnn1.Open()
            sqlvar = "SELECT CRITICIDAD FROM CRITICIDAD ORDER BY CRITICIDAD"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset4, "Categ")
            If valor Then
                numero = dataset4.Tables("Categ").Rows.Count
                For numero = 0 To dataset4.Tables("Categ").Rows.Count - 1
                    registro = dataset4.Tables("Categ").Rows(numero).Item("CRITICIDAD")
                    Me.Dropcriticidad.Items.Add(registro)
                Next numero
            End If
            cnn1.Close()
        End With

        With Me.DropDapli_fallante
            numero = .Items.Count
            For i = 0 To numero - 1
                USER = .Items(0).Text
                .Items.Remove(USER)
                arreglo(i) = ""
            Next i
            cnn1 = New OleDb.OleDbConnection(strcon)
            cnn1.Open()
            sqlvar = "SELECT * FROM APLICACION ORDER BY NOMBRE_APLICACION"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset2, "Categ")
            If valor Then
                numero = dataset2.Tables("Categ").Rows.Count
                For numero = 0 To dataset2.Tables("Categ").Rows.Count - 1
                    registro =
dataset2.Tables("Categ").Rows(numero).Item("NOMBRE_APLICACION")
                    Me.DropDapli_fallante.Items.Add(registro)
                Next numero
            End If
            cnn1.Close()
        End With
        UserMsgBox("Los datos ya se modificaron en la Base de Datos")

        cnn1.Close()
```

```
            Me.DropDCO_REGISTRO.SelectedIndex() = USE
            Me.DropDanalista.SelectedIndex() = str1
            Me.DropDapli_fallante.SelectedIndex() = str2
            Me.Dropcriticidad.SelectedIndex() = criti
        End If


    Case "C"
        Dim cnn1, cnn4 As OleDb.OleDbConnection
        Dim miOleDbCommand1 As OleDb.OleDbCommand
        Dim sqlvar, user As String
        Dim ejecutar, numero, i As Integer
        Dim str1, str2, modulo, registro As String
        Dim resultadoSQL As OleDb.OleDbDataAdapter
        Dim dataset As New DataSet
        Dim dataset1 As New DataSet
        Dim dataset2 As New DataSet
        Dim valor As Boolean
        Dim codigo, limpi As String


        cnn1 = New OleDb.OleDbConnection(strcon)
        cnn1.Open()
        str1 = Request.Form("DropDCO_REGISTRO")
        sqlvar = "delete from FALLAS_APLICACION where
FALLAS_APLICACION.CO_REGISTRO = '" & str1 & "'"
        miOleDbCommand1 = New OleDb.OleDbCommand(sqlvar, cnn1)
        ejecutar = miOleDbCommand1.ExecuteNonQuery()

'Permite limpiar los campos del formulario.

        Call Me.limpiar1()

        With Me.DropDCO_REGISTRO
            numero = .Items.Count
            For i = 0 To numero - 1
                user = .Items(0).Text
                .Items.Remove(user)
                arreglo(i) = ""
            Next i
            cnn1 = New OleDb.OleDbConnection(strcon)
            cnn1.Open()
            sqlvar = "SELECT * FROM FALLAS_APLICACION ORDER BY CO_REGISTRO"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset1, "Categ")
            If valor Then
                numero = dataset1.Tables("Categ").Rows.Count
                For numero = 0 To dataset1.Tables("Categ").Rows.Count - 1
                    registro = dataset1.Tables("Categ").Rows(numero).Item("CO_REGISTRO")
                    Me.DropDCO_REGISTRO.Items.Add(registro)
                Next numero
```

```
            End If
            cnn1.Close()
        End With

        Me.DropDanalista.Visible = False
        Me.DropDapli_fallante.Visible = False
        Me.Dropcriticidad.Visible = False
        Me.TextBanalista.Visible = True
        Me.TextBoxapli_fallante.Visible = True
        Me.Textcriticidad.Visible = True

        UserMsgBox("Los datos ya se eliminaron de la Base de Datos")

        cnn4 = New OleDb.OleDbConnection(strcon)
        cnn4.Open()

'Permite eliminar los datos en la base de datos.

        sqlvar = "delete from APLICFA where APLICFA.CO_REGISTRO = '" & str1 & "'"
        miOleDbCommand1 = New OleDb.OleDbCommand(sqlvar, cnn4)
        ejecutar = miOleDbCommand1.ExecuteNonQuery()
        cnn4.Close()
        'cnn1.Close()
    End Select
  End Sub

'Evento que ocurre cuando se desea registrar una falla.

  Private Sub Ibtsalvar_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles Ibtsalvar.Click
    If (Request.Form("DropDanalista") = "") Or (Request.Form("TextBusu_reportante") = "") Or
(Request.Form("TextBnom_falla") = "") Or (Request.Form("Textcausa_falla") = "") Or
(Request.Form("Textsolucion") = "") Or (Request.Form("DropDapli_fallante") = "") Or
(Request.Form("TextBfecha") = "") Then
      UserMsgBox("Todos los campos son obligatorios")
    Else
      MsgBox.ShowConfirmation("¿DESEA GUARDAR LOS DATOS?", "A", True, True)
    End If
  End Sub

'Evento que ocurre cuando se desea modificar  una falla.


  Private Sub Ibtmodificar_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles Ibtmodificar.Click
    If (Request.Form("TextBfecha") = "") Or (Request.Form("TextBusu_reportante") = "") Or
(Request.Form("TextBnom_falla") = "") Or (Request.Form("Textcausa_falla") = "") Or
(Request.Form("DropDapli_fallante") = "") Or (Request.Form("DropDanalista") = "") Or
(Request.Form("Textsolucion") = "") Then
      UserMsgBox("Todos los campos son obligatorios")
    Else
```

```
        MsgBox.ShowConfirmation("¿ESTÁ SEGURO QUE DESEA MODIFICAR LOS DATOS?",
"B", True, True)
      End If
   End Sub

'Evento que ocurre cuando se desea eliminar una falla.


   Private Sub Ibteliminar_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles Ibteliminar.Click
      MsgBox.ShowConfirmation("¿ESTÁ SEGURO QUE DESEA ELIMINAR LOS DATOS?", "C",
True, True)
   End Sub

'Evento que ocurre al  seleccionar la opción agregar.

   Private Sub Ibtagre_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles Ibtagre.Click
      Dim str1, str2, sqlvar, sqlvar1, registro, apli, apli1, modulo As String
      Dim cnn1, cnn2 As OleDb.OleDbConnection
      Dim resultadoSQL, resultadoSQL1 As OleDb.OleDbDataAdapter
      Dim dataset As New DataSet
      Dim dataset1 As New DataSet
      Dim dataset2 As New DataSet
      Dim dataset3 As New DataSet
      Dim dataset4 As New DataSet

      Dim numero, numero1 As Integer
      Dim valor As Boolean
      Dim codigo As String
      Dim USER As String
      Dim i As Integer

      Me.Ibtsalvar.Visible = True
      Me.Ibteliminar.Visible = False
      Me.Ibtmodificar.Visible = False
      Me.Ibtver_datos.Visible = False
      Me.DropDCO_REGISTRO.Visible = False
      Me.TextBco_registro.Visible = False
      Me.TextBanalista.Visible = False
      Me.TextBoxapli_fallante.Visible = False
      Me.Textcriticidad.Visible = False
      Me.DropDanalista.Visible = True
      Me.DropDapli_fallante.Visible = True
      Me.Dropcriticidad.Visible = True
      Me.Label2.Visible = False
      Me.ImageButton1.Visible = True
      Me.TextBusu_reportante.ReadOnly = False
      Me.TextBnom_falla.ReadOnly = False
      Me.Textcausa_falla.ReadOnly = False
      Me.Textsolucion.ReadOnly = False
```

```
        Me.Calendar.Visible = False

        Call Me.limpiar1()
        Me.TextBfecha.Text = IIf(Day(Date.Today) < 10, "0", "") + Day(Date.Today).ToString + "/" +
IIf(Month(Date.Today) < 10, "0", "") + Month(Date.Today).ToString + "/" +
Year(Date.Today).ToString


        With Me
            With .DropDanalista
                numero = .Items.Count
                For i = 0 To numero - 1
                    USER = .Items(0).Text
                    .Items.Remove(USER)
                    arreglo(i) = ""
                Next i
                cnn1 = New OleDb.OleDbConnection(strcon)
                cnn1.Open()
                sqlvar = "SELECT * FROM USUARIOS ORDER BY USERNAME"
                resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
                valor = resultadoSQL.Fill(dataset, "Categ")
                If valor Then
                    numero = dataset.Tables("Categ").Rows.Count
                    For numero = 0 To dataset.Tables("Categ").Rows.Count - 1
                        registro = dataset.Tables("Categ").Rows(numero).Item("USERNAME")
                        Me.DropDanalista.Items.Add(registro)
                    Next numero
                End If
                cnn1.Close()
            End With

            With .Dropcriticidad
                numero = .Items.Count
                For i = 0 To numero - 1
                    USER = .Items(0).Text
                    .Items.Remove(USER)
                    arreglo(i) = ""
                Next i
                cnn1 = New OleDb.OleDbConnection(strcon)
                cnn1.Open()
                sqlvar = "SELECT * FROM CRITICIDAD ORDER BY CRITICIDAD"
                resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
                valor = resultadoSQL.Fill(dataset4, "Categ")
                If valor Then
                    numero = dataset4.Tables("Categ").Rows.Count
                    For numero = 0 To dataset4.Tables("Categ").Rows.Count - 1
                        registro = dataset4.Tables("Categ").Rows(numero).Item("CRITICIDAD")
                        Me.Dropcriticidad.Items.Add(registro)
                    Next numero
                End If
                cnn1.Close()
```

```
        End With


        With .DropDCO_REGISTRO
            numero = .Items.Count
            For i = 0 To numero - 1
                USER = .Items(0).Text
                .Items.Remove(USER)
                arreglo(i) = ""
            Next i
            cnn1 = New OleDb.OleDbConnection(strcon)
            cnn1.Open()
            sqlvar = "SELECT * FROM FALLAS_APLICACION ORDER BY CO_REGISTRO"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset1, "Categ")
            If valor Then
                numero = dataset1.Tables("Categ").Rows.Count
                For numero = 0 To dataset1.Tables("Categ").Rows.Count - 1
                    registro = dataset1.Tables("Categ").Rows(numero).Item("CO_REGISTRO")
                    Me.DropDCO_REGISTRO.Items.Add(registro)
                Next numero
            End If
            cnn1.Close()
        End With


        With .DropDapli_fallante
            numero = .Items.Count
            For i = 0 To numero - 1
                USER = .Items(0).Text
                .Items.Remove(USER)
                arreglo(i) = ""
            Next i
            cnn1 = New OleDb.OleDbConnection(strcon)
            cnn1.Open()
            sqlvar = "SELECT * FROM APLICACION ORDER BY NOMBRE_APLICACION"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset2, "Categ")
            If valor Then
                numero = dataset2.Tables("Categ").Rows.Count
                For numero = 0 To dataset2.Tables("Categ").Rows.Count - 1
                    registro = dataset2.Tables("Categ").Rows(numero).Item("NOMBRE_APLICACION")
                    Me.DropDapli_fallante.Items.Add(registro)
                Next numero
            End If
            cnn1.Close()
        End With
    End With

End Sub
```

```
'Evento que ocurre al  seleccionar la opción modificar/eliminar.

  Private Sub Ibtmodeli_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles Ibtmodeli.Click
      Dim str1, str2, sqlvar, registro, apli, modulo, codigo, USER As String
      Dim cnn1 As OleDb.OleDbConnection
      Dim resultadoSQL As OleDb.OleDbDataAdapter
      Dim dataset As New DataSet
      Dim dataset1 As New DataSet
      Dim dataset2 As New DataSet
      Dim dataset3 As New DataSet
      Dim dataset4 As New DataSet
      Dim numero, i As Integer
      Dim valor As Boolean


      Me.Ibtsalvar.Visible = False
      Me.Ibteliminar.Visible = False
      Me.Ibtmodificar.Visible = False
      Me.Ibtver_datos.Visible = True
      Me.DropDCO_REGISTRO.Visible = True
      Me.DropDapli_fallante.Visible = False
      Me.DropDanalista.Visible = False
      Me.Dropcriticidad.Visible = False
      Me.TextBoxapli_fallante.Visible = True
      Me.TextBanalista.Visible = True
      Me.Textcriticidad.Visible = True
      Me.TextBco_registro.Visible = False
      Me.Label2.Visible = True
      Me.Calendar.Visible = False

      Me.TextBusu_reportante.ReadOnly = True
      Me.TextBnom_falla.ReadOnly = True
      Me.Textcausa_falla.ReadOnly = True
      Me.Textsolucion.ReadOnly = True
      Me.Calendar.Visible = False

      Call Me.limpiar1()

      With Me
        With .DropDanalista
          numero = .Items.Count
          For i = 0 To numero - 1
            USER = .Items(0).Text
            .Items.Remove(USER)
            arreglo(i) = ""
          Next i
          cnn1 = New OleDb.OleDbConnection(strcon)
          cnn1.Open()
          sqlvar = "SELECT * FROM USUARIOS ORDER BY USERNAME"
          resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
```

```
    valor = resultadoSQL.Fill(dataset, "Categ")
    If valor Then
        numero = dataset.Tables("Categ").Rows.Count
        For numero = 0 To dataset.Tables("Categ").Rows.Count - 1
            registro = dataset.Tables("Categ").Rows(numero).Item("USERNAME")
            Me.DropDanalista.Items.Add(registro)
        Next numero
    End If
    cnn1.Close()
End With

With .Dropcriticidad
    numero = .Items.Count
    For i = 0 To numero - 1
        USER = .Items(0).Text
        .Items.Remove(USER)
        arreglo(i) = ""
    Next i
    cnn1 = New OleDb.OleDbConnection(strcon)
    cnn1.Open()
    sqlvar = "SELECT * FROM CRITICIDAD ORDER BY CRITICIDAD"
    resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
    valor = resultadoSQL.Fill(dataset4, "Categ")
    If valor Then
        numero = dataset4.Tables("Categ").Rows.Count
        For numero = 0 To dataset4.Tables("Categ").Rows.Count - 1
            registro = dataset4.Tables("Categ").Rows(numero).Item("CRITICIDAD")
            Me.Dropcriticidad.Items.Add(registro)
        Next numero
    End If
    cnn1.Close()
End With


With .DropDCO_REGISTRO
    numero = .Items.Count
    For i = 0 To numero - 1
        USER = .Items(0).Text
        .Items.Remove(USER)
        arreglo(i) = ""
    Next i
    cnn1 = New OleDb.OleDbConnection(strcon)
    cnn1.Open()
    sqlvar = "SELECT * FROM FALLAS_APLICACION ORDER BY CO_REGISTRO"
    resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
    valor = resultadoSQL.Fill(dataset1, "Categ")
    If valor Then
        numero = dataset1.Tables("Categ").Rows.Count
        For numero = 0 To dataset1.Tables("Categ").Rows.Count - 1
            registro = dataset1.Tables("Categ").Rows(numero).Item("CO_REGISTRO")
            Me.DropDCO_REGISTRO.Items.Add(registro)
```

```
            Next numero
          End If
          cnn1.Close()
        End With


        With .DropDapli_fallante
          numero = .Items.Count
          For i = 0 To numero - 1
            USER = .Items(0).Text
            .Items.Remove(USER)
            arreglo(i) = ""
          Next i
          cnn1 = New OleDb.OleDbConnection(strcon)
          cnn1.Open()
          sqlvar = "SELECT * FROM APLICACION ORDER BY NOMBRE_APLICACION"
          resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
          valor = resultadoSQL.Fill(dataset2, "Categ")
          If valor Then
            numero = dataset2.Tables("Categ").Rows.Count
            For numero = 0 To dataset2.Tables("Categ").Rows.Count - 1
              registro = dataset2.Tables("Categ").Rows(numero).Item("NOMBRE_APLICACION")
              Me.DropDapli_fallante.Items.Add(registro)
            Next numero
          End If
          cnn1.Close()
        End With
      End With

    End Sub


'Limpia algunos campos del formulario.

 Sub limpiar()
      Dim i As Integer

      With Me
        .TextBco_registro.Text = ""
        .TextBfecha.Text = ""
        .TextBusu_reportante.Text = ""
        .TextBnom_falla.Text = ""
        .Textcausa_falla.Text = ""
        .Textsolucion.Text = ""

        With .CheckBoxapli_afect
          For i = 0 To .Items.Count - 1
            .Items(i).Selected() = False
          Next i
        End With
      End With
```

```
    End Sub

'Limpia  todos los campos del formulario.
Sub limpiar1()
        Dim i As Integer

        With Me
            .TextBco_registro.Text = ""
            .TextBfecha.Text = ""
            .TextBusu_reportante.Text = ""
            .TextBnom_falla.Text = ""
            .Textcausa_falla.Text = ""
            .Textsolucion.Text = ""
            .TextBanalista.Text = ""
            .TextBoxapli_fallante.Text = ""
            .Textcriticidad.Text = ""

            With .CheckBoxapli_afect
                For i = 0 To .Items.Count - 1
                    .Items(i).Selected() = False
                Next i
            End With
        End With
    End Sub

'Evento que ocurre al seleccionar la opción  ver datos.

    Private Sub Ibtver_datos_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles Ibtver_datos.Click
        Dim cnn1 As New OleDb.OleDbConnection
        Dim resultadoSQL, resultadoSQL1 As OleDb.OleDbDataAdapter
        Dim sqlvar, sqlvar1 As String
        Dim int1 As Integer
        Dim str1, str2, pos, pos1, User, registro As String
        Dim dataset As New DataSet
        Dim dataset1 As New DataSet
        Dim dataset2 As New DataSet
        Dim valor, valor1 As Boolean
        Dim numero, modulo, n, i, total As Integer

        Me.Ibteliminar.Visible = True
        Me.Ibtmodificar.Visible = True
        Me.DropDCO_REGISTRO.Visible = True
        Me.DropDanalista.Visible = True
        Me.DropDapli_fallante.Visible = True
        Me.Dropcriticidad.Visible = True
        Me.TextBanalista.Visible = True
        Me.TextBoxapli_fallante.Visible = False
        Me.Textcriticidad.Visible = True
        Me.ImageButton1.Visible = True
```

```
        Me.TextBusu_reportante.ReadOnly = False
        Me.TextBnom_falla.ReadOnly = False
        Me.Textcausa_falla.ReadOnly = False
        Me.Textsolucion.ReadOnly = False
        Me.Calendar.Visible = False

        Call Me.limpiar1()

        cnn1 = New OleDb.OleDbConnection(strcon)
        cnn1.Open()
        str1 = Request.Form("DropDCO_REGISTRO")
        '
        If (Request.Form("DropDCO_REGISTRO") = "") Then
            UserMsgBox("No hay registros que mostrar")
        Else
            'Permite mostrar  los datos del registro de falla seleccionado.
            sqlvar = "SELECT * FROM FALLAS_APLICACION WHERE
FALLAS_APLICACION.CO_REGISTRO = '" & str1 & "'"
            resultadoSQL = New OleDb.OleDbDataAdapter(sqlvar, cnn1)
            valor = resultadoSQL.Fill(dataset, "Categ")
            Me.TextBanalista.Text = dataset.Tables("Categ").Rows(0).Item("USERNAME")
            Me.TextBusu_reportante.Text =
dataset.Tables("Categ").Rows(0).Item("USERNAME_CLIENTE")
            Me.TextBnom_falla.Text =
Trim(dataset.Tables("Categ").Rows(0).Item("NOMBRE_FALLA"))
            Me.TextBfecha.Text =
convierte_fecha_consulta(dataset.Tables("Categ").Rows(0).Item("FECHA_REPORTE"))
            Me.Textcausa_falla.Text = dataset.Tables("Categ").Rows(0).Item("CAUSA_FALLA")
            Me.Textsolucion.Text = dataset.Tables("Categ").Rows(0).Item("SOLUCION_APLICADA")
            Me.DropDapli_fallante.SelectedValue =
dataset.Tables("Categ").Rows(0).Item("APLIC_FALLANTE")
            Me.Textcriticidad.Text = dataset.Tables("Categ").Rows(0).Item("CRITICIDAD_FALLA")
            Me.TextBco_registro.Text = str1

            sqlvar1 = "SELECT * FROM APLICFA WHERE APLICFA.CO_REGISTRO = '" & str1 & "'"
            resultadoSQL1 = New OleDb.OleDbDataAdapter(sqlvar1, cnn1)
            valor = resultadoSQL1.Fill(dataset1, "Categ1")
            If valor Then
                numero = dataset1.Tables("Categ1").Rows.Count
                For numero = 0 To dataset1.Tables("Categ1").Rows.Count - 1
                    str2 = dataset1.Tables("Categ1").Rows(numero).Item("CO_APLICACION")
                    For n = 0 To Me.CheckBoxapli_afect.Items.Count - 1
                        If arreglo(n) = str2 Then
                            Me.CheckBoxapli_afect.Items(n).Selected() = True
                        End If
                    Next n
                Next numero
            End If

            Me.DropDCO_REGISTRO.SelectedValue = str1
            cnn1.Close()
```

```
        End If
    End Sub

'Permite mostrar las alertas de mensajes en la aplicación.

    Public Sub UserMsgBox(ByVal sMsg As String)

        Dim sb As New StringBuilder
        Dim oFormObject As System.Web.UI.Control

        sMsg = sMsg.Replace("'", "\f '")
        sMsg = sMsg.Replace(Chr(34), "\" & Chr(34))
        sMsg = sMsg.Replace(vbCrLf, "\n")
        sMsg = "<script languaje=javascript>alert('" & sMsg & "')</script>"
        sb = New StringBuilder

        sb.Append(sMsg)
        For Each oFormObject In Me.Controls
            If TypeOf oFormObject Is HtmlForm Then
                Exit For
            End If
        Next
        oFormObject.Controls.AddAt(oFormObject.Controls.Count, New LiteralControl(sb.ToString()))
    End Sub

'Permite seleccionar la fecha en el calendario.

    Private Sub Calendar_SelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Calendar.SelectionChanged
        Me.TextBfecha.Text = IIf(Day(Calendar.SelectedDate.Date) < 10, "0", "") +
Day(Calendar.SelectedDate.Date).ToString + "/" + IIf(Month(Calendar.SelectedDate.Date) < 10, "0",
"") + Month(Calendar.SelectedDate.Date).ToString + "/" +
Year(Calendar.SelectedDate.Date).ToString
        Calendar.Visible = False
    End Sub

'Muestra el calendario al presionar el botón indicado.

    Private Sub ImageButton1_Click(ByVal sender As System.Object, ByVal e As
System.Web.UI.ImageClickEventArgs) Handles ImageButton1.Click
        Me.Calendar.Visible = True
    End Sub
End Class
```

Código HTML y validaciones de campos en JavaScript del caso de uso Gestionar Fallas:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
      <title>Gestionar_Fallas</title>
      <meta content=”Microsoft Visual Studio .NET 7.1” name=”GENERATOR” >
      <meta content=”Visual Basic .NET 7.1” name=”CODE_LANGUAGE” >
      <meta content=”JavaScript” name=”vs_defaultClientScript” >
      <meta content=”http://schemas.microsoft.com/intellisense/ie5”
       name=”vs_targetSchemas” >
  </HEAD>
  <body aLink=”#0066ff” MS_POSITIONING=”GridLayout” >

<!-- Aquí se muestra el código HTML que genera la página Web. Posee
etiquetas propias de este lenguaje para formar tablas, imágenes, estilos, así como
enlaces hacia otros archivos para la buena presentación de la página en cuestión.
Además de código para validar los campos en cuanto a caracteres y longitud -->

      <form id=”Form1” onsubmit=”return validar(this)” action=”formularios.asp”
method=”post” runat=”server”>

          <script>

      function validar(formulario) {
        if (formulario.TextBusu_reportante.value.length > 50) {
          alert("EL CAMPO \"USUARIO QUE REPORTA LA FALLA\" ACEPTA HASTA 50 CARACTERES.");
          formulario.TextBusu_reportante.focus();
          return (false);}
      var checkOK = "abcdefghijklmnñopqrstuvwxyzáéíóú0123456789
ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁÉÍÓÚ \n" + String.fromCharCode(13);
       var checkStr = formulario.TextBusu_reportante.value;
       var allValid = true;
       for (i = 0; i < checkStr.length; i++) {
            ch= checkStr. charAt(i);
            for (j = 0; j < checkOK.length; j++)
              if (ch == checkOK.charAt(j))
                break;
        if (j == checkOK.length) {
          allValid = false;
       break;
```

```
        }
 }
          if (!allValid) {
            alert("EL CAMPO \"USUARIO QUE REPORTA LA FALLA\" SÓLO
ACEPTA CARACTERES ALFANUMÉRICOS.");
            formulario.TextBusu_reportante.focus();
            return (false); }


           if (formulario.TextBnom_falla.value.length > 200) {
              alert("EL CAMPO \"NOMBRE DE  LA FALLA\" ACEPTA HASTA 200 CARACTERES.");
              formulario.TextBnom_falla.focus();
              return (false);}
          var checkOK = "abcdefghijklmnñopqrstuvwxyzáéíóú0123456789
ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁÉÍÓÚ \n" + String.fromCharCode(13);
          var checkStr = formulario.TextBnom_falla.value;
          var allValid = true;
          for (i = 0; i < checkStr.length; i++) {
                ch= checkStr. charAt(i);
                for (j = 0; j < checkOK.length; j++)
                   if (ch == checkOK.charAt(j))
                     break;
           if (j == checkOK.length) {
              allValid = false;
          break;
        }
 }
          if (!allValid) {
            alert("EL CAMPO \"NOMBRE DE LA FALLA\" SÓLO ACEPTA
CARACTERES ALFANUMÉRICOS Y LOS CARACTERES (, . ; : _ - / * [ ] ).");
            formulario.TextBnom_falla.focus();
            return (false); }


if (formulario.Textcausa_falla.value.length > 1000) {
    alert("EL CAMPO \"CAUSA DE LA FALLA\" ACEPTA HASTA 1000
CARACTERES.");
    formulario.Textcausa_falla.focus();
    return (false);  }
var checkOK = "abcdefghijklmnñopqrstuvwxyzáéíóú0123456789
ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁÉÍÓÚ,.;:_-/*[] \n ()" +
String.fromCharCode(13) + String.fromCharCode(9);
var checkStr = formulario.Textcausa_falla.value;
```

```
 var allValid = true;
          for (i = 0; i < checkStr.length; i++) {
                   ch= checkStr. charAt(i);
                   for (j = 0; j < checkOK.length; j++)
                      if (ch == checkOK.charAt(j))
                        break;
           if (j == checkOK.length) {
                allValid = false;
                break;
           }
 }

 if (!allValid) {
     alert("EL CAMPO \"CAUSA DE LA FALLA\" SÓLO ACEPTA CARACTERES
ALFANUMÉRICOS.");
     formulario.Textcausa_falla.focus();
     return (false);
}


if (formulario.Textsolucion.value.length > 5000) {
   alert("EL CAMPO \"SOLUCIÓN APLICADA A LA FALLA\" ACEPTA HASTA
5000 CARACTERES.");
   formulario.Textsolucion.focus();
   return (false);
}
var checkOK = "abcdefghijklmnñopqrstuvwxyzáéíóú0123456789
ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁÉÍÓÚ, . ; : _- / * [] \n ()" +
String.fromCharCode(13) + String.fromCharCode(9);
var checkStr = formulario.Textsolucion.value;
 var allValid = true;
    for (i = 0; i < checkStr.length; i++) {
          ch= checkStr. charAt(i);
             for (j = 0; j < checkOK.length; j++)
                if (ch == checkOK.charAt(j))
                 break;
            if (j == checkOK.length) {
          allValid = false;
                break;
           }
 }

  if (!allValid) {
```

```
   alert("EL CAMPO \"SOLUCIÓN APLICADA A LA FALLA\" SÓLO ACEPTA
CARACTERES ALFANUMÉRICOS.");
   formulario.Textsolucion.focus();
   return (false);
}
}

        </script>


<TABLE id=”Table 2”  style=”Z-INDEX: 102; LEFT: 24px; WIDTH: 744px;
POSITION: absolute; TOP: 120 px; HEIGHT: 791px cellSpacing=”1” align=”center"
bgColor=”#3a6ea5” border = "1">
         <TR>
            <TD style=” HEIGHT: 23px; BORDER-BOTTOM-STYLE:solid”
colSpan=”13”>  
               <asp: label id=”Label1” runat=”server” Font-Bold=”True”
Height=”24px” Width=”248px” ForeColor=”White” Font-Names=”Times New
Roman” Font-Size=”Small”> GESTIONAR FALLAS </asp: label>
            </TD>
         </TR>
      </TABLE>
   </form>
 </BODY>
</HTML>
```

5.3 PRUEBAS

Para entregar un producto de software al cliente libre de defectos y de errores, es necesaria la realización de pruebas que garanticen que dicho software se encuentra en óptimas condiciones.

Las pruebas constituyen una de las etapas más importantes del ciclo de vida de desarrollo de software. Mediante ellas se ejecuta un programa con la intención de encontrar defectos en el mismo.

- **Pruebas de Unidad:** las pruebas de unidad evalúan los componentes implementados como unidades individuales. Ayudan a que el módulo

probado se haga independiente, es decir, sin tomar en cuenta el resto del sistema. Conociendo la función específica para la cual se diseñará el producto, se aplican pruebas que demuestren que cada función es plenamente operacional, mientras se buscan los errores en dichas funciones. Para efectuar las pruebas de unidad se utiliza la técnica de pruebas de Caja Negra. La prueba de Caja Negra, también conocida como prueba Funcional, es la que se aplica a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica del software. La prueba de Caja Negra se concentra en la funcionalidad global del software y considera la selección de los datos de prueba y la interpretación de los resultados de dicha prueba realizada basándose en las características funcionales del software.

- **Partición Equivalente:** la partición equivalente es un método de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El objetivo principal es definir tan solo un caso de prueba en cada clase de equivalencia en lugar de múltiples casos de prueba que pertenecen a la misma clase de equivalencia. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse.

El diseño de casos de prueba para una partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana. Las clases de equivalencia se definen de acuerdo con ciertas

directrices que ayuden a determinar todos los posibles casos de clases de equivalencia que imperan en un sistema a ser examinado.

5.3.1 Determinación de Clases de Equivalencia

Al aplicar estos conceptos para la derivación de clases de equivalencia, se desarrollarán y ejecutarán los casos de prueba para cada objeto de los datos del dominio de entrada. A continuación, en la tabla 5.1 de esta sección se detallan todas las posibles clases de equivalencia que intervienen en los casos de prueba del sistema en general.

Tabla 5.1. Clases de Equivalencia generales presentes en el sistema

N°. Clase	Nombre de la Clase	Descripción	Ejemplo a Utilizar
1.	Fecha	Esta clase de equivalencia recibe primero dos números que representan el día seguido del carácter especial "/", luego dos números que representan el mes, igualmente seguido de otro "/", y finalmente cuatro números que indican el año.	10/07/2009
2.	Alfanumérico	Recibe caracteres alfabéticos en mayúscula o en minúscula, además de numéricos y caracteres especiales. Este tipo de datos son los que recibe nombres de las fallas, causa de la falla, solución aplicada a la falla.	?NE"X_flo'res/3
3.	Valor vacío	Se refiere a que ningún dato se ingresa en el campo, también es conocido como una cadena de caracteres vacía. Existen algunos campos que sí aceptan este valor (dejando sus campos libres) mientras que otros no.	""

5.3.2 Casos de Prueba de Caja Negra

A continuación se muestran ciertos casos de prueba de Caja Negra en que se resumen el uso de las clases de equivalencia anteriormente descritas. La idea de estos casos de prueba es ingresar en los campos de texto requeridos un valor representativo de cada clase de equivalencia y determinar si es un argumento válido o no para el campo en cuestión, tanto en lo que se refiere a tipo de datos que representa como en longitud de dicho conjunto de caracteres. Así, en cada campo se ingresarán cada una de las tres clases de equivalencia encontradas anteriormente, y si dicho valor representativo de dicha clase de equivalencia es aceptable teóricamente, y a la vez es aceptado por el sistema, en función de sus requerimientos tanto de tipo de dato (cadena de caracteres, numérico, fecha, cadena vacía, etc.) como de longitud (por ejemplo, un campo clave sólo acepta 20 caracteres, ni un carácter más, ni un carácter menos) pues se cotejará como "Válido" en dicho campo, de lo contrario será declarado "No Válido". Se comenzará con el caso de uso que lleva a cabo un usuario al tratar de autentificarse e iniciar una sesión. Véase tabla 5.2.

Tabla 5.2. Caso de Prueba de Caja Negra "Iniciar Sesión"

Nombre del Campo	Caso de Prueba	Válido	No Válido	Clase de Equiv. Cubierto
Nombre de Usuario	10/07/2009		✓	1
	?NE"HE_nov's/3		✓	2
	""		✓	3
	Abnaf	✓		2
Contraseña o Password	10/07/2009		✓	1
	?hec"ne_FL'O/3		✓	2
	""		✓	3
	Abnaf	✓		2

Tabla 5.3 pruebas del caso de uso "Ingresar Fallas".

Nombre del Campo	Caso de Prueba	Válido	No Válido	Clase de Equiv. Cubierto
Usuario que reporta la falla	10/07/2008		✓	1
	?ne"IS_HE'Be/3		✓	2
	Ab2naf0	✓		2
	""		✓	3
Nombre de la falla	20/07/2006	✓		1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
Causa de la falla	10/07/2009	✓		1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
Aplicación que falla	10/07/2009		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
Aplicaciones que la falla afecta	10/07/2009		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
Analista que atiende la falla	10/07/2009		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
Criticidad de la falla	10/07/2009		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
Solución aplicada a la falla	10/07/2009	✓		1
	?ne"NX_Be'He/3		✓	2
	""		✓	3

Tabla 5.4 pruebas del caso de uso "Modificar Usuario".

Nombre del Campo	Caso de Prueba	Válido	No Válido	Clase de Equiv. Cubierto
Nombre	10/07/2008		✓	1
	?ne"IS_HE'Be/3		✓	2
	Abnaf	✓		2
	""		✓	3
Apellido	20/07/2006		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
	Abnaf	✓		2
Cargo	10/07/2009		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
	Abnaf	✓		2
Clave	10/07/2009		✓	1
	?ne"NX_Be'He/3		✓	2
	""		✓	3
	Abnaf	✓		2

5.3.3. Casos de Prueba de Integración

Conociendo el funcionamiento interno del producto, se aplican pruebas para asegurarse que "todas las piezas encajan", en otras palabras, están integradas correctamente. Se prueban las rutas lógicas del software y la colaboración entre los componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos. Esta técnica de evaluación de software es denominada pruebas de Caja Blanca. Las pruebas se aplicaron a varios componentes de software desarrollados, comprobando así la exitosa integración de la aplicación. Además, se probaron los enlaces de las páginas, que no hubiesen enlaces rotos o vacíos, ni enlaces que direccionaran hacia alguna página indebida. Los casos de prueba deben realizarse sabiendo cómo funciona el programa, es decir, cómo es su desempeño. Se

comienza con el caso de uso que realiza un usuario al tratar de autentificarse, como se muestra en la tabla 5.5 para lograr ingresar como usuario válido:

Tabla 5.5. Caso de Prueba de Integración "Iniciar Sesión"

N°. Caso de Prueba	**01**	**Caso de Prueba**	**Iniciar Sesión**
Entrada	➢ **Nombre usuario**: nflores ➢ **Contraseña**: fsnex		
Resultado	El usuario es autentificado y de inmediato ingresa al sistema REFAP.		
Condiciones	➢ El usuario debe estar registrado previamente en la BDD.		
Procedimiento	1. Escribir el nombre usuario y la contraseña en los cuadros de texto correspondientes. 2. Presione el botón "ACEPTAR". 3. Se ingresa finalmente al sistema.		
Situación del Caso de Prueba	✓ **EXITOSO**		

Luego de iniciar satisfactoriamente la sesión, se procede a Ingresar una falla en el sistema, como se observa en la tabla 5.6. Dicho procedimiento lo puede realizar un Analista:

Tabla 5.6 / 1. Casos de Prueba de Integración "Ingresar Falla"

N°. Caso de Prueba	**02**	**Casos de Prueba**	**Ingresar Fallas**
Entrada	➢ Fecha de reporte: 09/10/2010 ➢ Usuario que reporta la falla: Berenice Sánchez ➢ Nombre de la falla: Conexión Errada ➢ Causa de la falla: Incompatibilidad de datos ➢ Aplicación que falla: Resac(Jose) ➢ Aplicaciones que la falla afecta: Portal Sap ➢ Analista que atiende la falla: nflores ➢ Criticidad de la falla: Baja ➢ Solución aplicada a la falla: Depuración de la base de datos		

Tabla 5.6 / 2. Casos de Prueba de Integración "Ingresar Falla"

N°. Caso de Prueba	**02**	**Casos de Prueba**	**Ingresar Fallas**
Resultado	Los datos de la falla ingresados se almacenan en la base de datos y se le genera un código de registro.		
Condiciones	➢ El Analista que realiza este caso de uso debe estar registrado en el sistema como usuario. ➢ Dicho usuario debe estar previamente autentificado, es decir, haber cumplido requisitos para entrar al sistema. ➢ El usuario que lleva a cabo este caso de uso posee los privilegios adecuados para realizar el registro de la falla.		
Procedimiento	1. Seleccionar del menú la opción de Gestionar Fallas. 2. Seleccionar la opción Agregar 3. Llenar los campos que se encuentran vacios y seleccionar la información adecuada de las listas que están previamente cargadas. 4. Presionar el botón Salvar. 5. Confirmar el mensaje de solicitud de procedimiento que el sistema presenta. ¿Desea guardar los datos? 6. Confirmar el mensaje de solicitud de procesamiento que el sistema presenta. Los datos ya se cargaron en la base de datos.		
Situación del Caso de Prueba	✓ **EXITOSO**		

Después de registrar una nueva falla, el Analista procede a realizar una modificación de los datos de dicha falla, tal como se aprecia en la tabla 5.7.

Tabla 5.7 / 1. Casos de Prueba de Integración "Modificar Falla"

N°. Caso de Prueba	**03**	**Casos de Prueba**	**Modificar Fallas**
Entrada	➢ **Código de registro como criterio usado de búsqueda**: 30 ➢ Modificar los siguientes datos: o Fecha de reporte anterior: 09/10/2010 o **Nueva Fecha de reporte**: 09/10/2010		

Tabla 5.7 / 2. Casos de Prueba de Integración "Modificar Falla"

N°. Caso de Prueba	**03**	**Casos de Prueba**	**Modificar Fallas**
Entrada	o Usuario que reporta la falla anterior: Berenice Sánchez o **Nuevo Usuario que reporta la falla:** Hernán Flores o Nombre anterior de la falla: Conexión Errada o **Nuevo Nombre de la falla:** Falla de conexión a la base de datos o Causa de la falla anterior: Incompatibilidad de datos o **Nueva Causa de la falla:** Incompatibilidad de datos o Aplicación que falla anterior: Resac(Jose) o **NuevaAplicación que falla**: Resac(Jose) o Aplicaciones que la falla afecta anterior: Portal Sap o **Nuevas Aplicaciones que la falla afecta:** o Analista que atiende la falla anterior: nflores o **Nuevo Analista que atiende la falla**: nflores o Criticidad de la falla anterior: Baja o **Nueva Criticidad de la falla**: Alta o Solución aplicada a la falla anterior: Depuración de la base de datos o **Nueva Solución aplicada a la falla**: Depuración de la base de datos		
Resultado	Los datos de la falla modificados se actualizan en la base de datos.		
Condiciones	➢ El Analista que realiza este caso de uso debe estar registrado en el sistema como usuario. ➢ Dicho usuario debe estar previamente autentificado, es decir, haber cumplido requisitos para entrar al sistema. ➢ Existe un registro de la falla a modificar en la base de datos con el dato considerado como criterio de búsqueda. ➢ El usuario que lleva a cabo este caso de uso posee los privilegios adecuados para realizar las modificaciones.		
Procedimiento	1. Seleccionar del menú la opción de Gestionar Fallas. 2. Seleccionar la opción Modificar/Eliminar 3. Escoger de la lista el código del registro de la falla a modificar. 4. Presionar el botón Ver Datos.		

Tabla 5.7 / 3. Casos de Prueba de Integración "Modificar Falla"

N°. Caso de Prueba	03	Casos de Prueba	Modificar Fallas
Procedimiento	5. Realizar los cambios a los datos de los campos necesarios por los nuevos datos indicados anteriormente. 6. Presionar el botón Modificar para proceder con la modificación de los datos de la falla. 7. Confirmar el mensaje de solicitud de procedimiento que el sistema presenta. ¿Está seguro que desea modificar los datos? 8. Confirmar el mensaje de solicitud de procesamiento que el sistema presenta. Los datos ya se modificaron en la base de datos.		
Situación del Caso de Prueba	✓ **EXITOSO**		

5.4 EVALUACIÓN DE LA FASE DE CONSTRUCCIÓN

El objetivo de obtener un software operativo libre de errores y defecto trazado para esta fase se logró. Se puede decir que el producto está listo para ser publicado en una versión beta, y que la fase de construcción se ha ejecutado con éxito. Se requirió una iteración donde se abarcaron los flujos de trabajo de implementación y pruebas. Durante la implementación se realizó la codificación efectiva de las páginas Web y los distintos componentes que conforman el software. En el transcurso del desarrollo de esta fase, se revisaron ciertos criterios de evaluación, arrojándose los siguientes resultados:

- Se diseñaron las interfaces del programa que permiten la interacción entre el usuario y el sistema de manera cómoda y amigable.
- Tanto la codificación de los módulos que conforman el proyecto, como la integración de cada uno de ellos se realizaron de manera exitosa, en combinación con la respectiva base de datos.

- Las pruebas necesarias y llevadas a cabo para la certificación de la confiabilidad y efectividad de la aplicación permitió la detección y corrección oportuna de las fallas existentes.
- Las herramientas que ofrece el lenguaje de modelado WebML fueron de mucha utilidad, ya que facilitan la codificación mediante estándares de programación Web.
- La ejecución de los procesos de implementación y pruebas han dado como resultado un producto estable y confiable como para ser entregado a la comunidad de usuarios y así ser probado y evaluado por los mismos.
- Con la obtención de la versión funcional del producto,
- Ahora que se obtuvo una versión funcional del producto, se continúa a la fase de transición de dicha versión, donde se determinará si hay necesidad de otro ciclo de desarrollo, es decir, otra iteración, o si efectivamente se puede entregar el producto a los usuarios finales del mismo.

CAPÍTULO 6
FASE DE TRANSICIÓN

"El tiempo no espera por nadie: el pasado es historia, el futuro un misterio y el presente un regalo de Dios". ¡Aprovéchalo!

CAPÍTULO 6

FASE DE TRANSICIÓN

El objetivo fundamental de la fase de transición es asegurar que el software esté listo para ser entregado a los usuarios finales.

En este punto la mayor parte de la estructura debería estar terminada y la retroalimentación de los usuarios debería enfocarse principalmente en la entonación, configuración, instalación y resultados de reutilización. Es posible que la fase de transición se extienda algunas iteraciones, incluyendo las pruebas del producto dentro de la preparación para su publicación y el hacer los ajustes menores basados en la retroalimentación de los usuarios. La fase de transición en el enfoque de RUP difiere del desarrollo tradicional, principalmente porque se entra en esta fase con una versión del sistema estable, integrada y probada.

En la fase anterior se obtuvo un software preliminar que es capaz de cumplir con éxito los requerimientos solicitados por el grupo de integrantes de la Coordinación IS de PDVSA PETROCEDEÑO, minimizando los riesgos y maximizando la productividad y efectividad de los usuarios. La fase de transición cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporarán alguna de las mejores sugeridas en una versión general dirigida a la totalidad de la comunidad de usuarios. La fase de transición conlleva actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos en dos categorías: los que tienen suficiente impacto en la operación para justificar una versión incrementada y los que pueden corregirse en la siguiente versión nominal. Además de proveer el soporte básico necesario a través de un manual de usuario y manual del administrador.

Al final de la fase de transición los objetivos del ciclo de vida deben haber sido alcanzados y el proyecto debería estar en posición de ser cerrado. El fin del actual ciclo de vida puede coincidir con el inicio de otro ciclo de vida, sobre el mismo producto, dirigiéndose a la próxima generación o versión del producto.

Se estima que la duración de esta fase sea más corta que la anterior, ya que la realización de actividades en cualquier flujo de trabajo (requisitos, análisis, diseño, implementación o pruebas) depende del resultado que arroje las pruebas sobre la versión beta del software. La corrección de todos los posibles errores y modificaciones que aparezcan sugeridas por los usuarios es lo que determinará la calidad funcional del software, debido a que son los usuarios finales los que reportarán las fallas o modificaciones aceptables en pro del mejoramiento del producto final que utilizarán.

6.1 IMPLEMENTACIÓN

En la fase de transición del RUP se procede directamente a la implementación del proyecto en cuestión, de allí que no se lleve a cabo ninguna etapa de diseño, y mucho menos de análisis. En esta fase el número de iteraciones puede variar de algo muy directo a algo extremadamente complejo, dependiendo del producto, es decir, que el producto requiera simplemente la corrección de algunos errores menores y se emplee sólo una iteración; en cambio si al producto se le adicionan otras características o se integrará con otros sistemas se necesitarán más iteraciones.

6.1.1 Preparación de la Versión Beta

Se ha pautado instalar la estructura de hardware necesaria para que funcione el software en un área de trabajo determinada, de forma de preparar el lanzamiento de la versión beta del software. Además de seleccionar un grupo de usuarios para que utilicen la aplicación e instruirlos en el uso correcto de la misma, cubriendo el

entrenamiento al personal que incluye requisitos mínimos para la instalación y uso del software, niveles de permisología por usuario, es decir, las restricciones o limitaciones de acceso de cada usuario, y los manuales de usuario y del administración para aprender cómo utilizar la aplicación.

6.1.2. Instalación de la Versión Beta

La instalación del software se realizó en un servidor de la Coordinación IS de PDVSA PETROCEDEÑO. Se instalaron todas las herramientas requeridas para el correcto funcionamiento del software bajo la plataforma Microsoft Windows XP, así como el manejador de base de datos SQL SERVER 2000, y Visual Studio .Net.

El software es en ambiente Web por lo tanto puede ser accedido desde cualquier computador que esté conectado a la red de PDVSA PETROCEDEÑO.

Todos los computadores involucrados cumplen con las características mínimas que se necesitan para un desempeño óptimo del software, tales como:

- Procesador PENTIUM IV 3.0 GHz.
- Memoria RAM de 512 MB o más.
- Disco duro 80 GB o más.
- Teclado, monitor y mouse.

6.2 EVALUACIÓN DE LA FASE DE TRANSICIÓN

Para dar por terminada la fase de transición se debe tomar en cuenta la versión del producto Software obtenido, es decir, si dicho Software cumple con los requerimientos de los usuarios finales, la fase de transición se da por terminada, de lo contrario sería necesario comenzar otro ciclo de desarrollo.

Al finalizar esta fase, se encontraron los siguientes resultados:

- Se ha completado con éxito el desarrollo de este proyecto; el hecho de haber culminado esta fase así lo confirma.
- El sistema implantado cumple con los requerimientos y necesidades de los integrantes de la Coordinación IS de PDVSA PETROCEDEÑO.
- Se han logrado los objetivos planteados al inicio del proyecto REFAP.
- No fue necesario corregir o modificar significativamente algún componente de software.
- La arquitectura implementada es robusta y estable.
- Los manuales de usuario y del administrador del sistema REFAP son de fácil comprensión e ilustrativo al usar.

CONCLUSIONES

Una vez culminado el proyecto REFAP, se puede concluir los siguientes:

- El Proceso Unificado de Desarrollo de Software permitió crear la arquitectura del proyecto REFAP, de una manera adaptable, robusta y confiable.

- El obtener una visión amplia de los procedimientos que se llevan a cabo en la Coordinación IS de PDVSA PETROCEDEÑO, identificar los requisitos, actores y casos de uso del sistema, fue la base fundamental para lograr en la Fase de Inicio enfocar una idea general del funcionamiento que se esperaba cumpliera el sistema.

- Utilizar el modelo relacional, mediante asociaciones de entidades e interrelaciones, sintetiza eficazmente la implementación de la base de datos, ya que elimina la redundancia de datos lo cual permite un almacenamiento y recuperación eficiente de la información.

- Utilizar el lenguaje de modelado UML, acompañado también del WebML para diseñar los módulos facilitó la construcción de la aplicación en lo que a navegabilidad y funcionamiento del sistema se refiere, así como a visualización e integración de los módulos.

- El software REFAP ofrece interfaz flexible, amigable, y de fácil manejo para los usuarios finales del mismo.

- Mediante la realización de diversas pruebas necesarias y llevadas a cabo para la certificación de la confiabilidad y efectividad de la aplicación, se permitió la detección y corrección de las fallas existentes, garantizando un producto de buena calidad.

- El software REFAP cumplió con los objetivos propuestos, ya que se ha logrado cubrir con todos los requisitos solicitados por los usuarios finales del mismo.

RECOMENDACIONES

- Añadir una opción al sistema que permita llevar un registro de los usuarios que inician sesión, tales como la hora de inicio y finalización de la sesión, fecha, módulos al que accedió, tiempo de duración en dichos módulos y si ingresa, modifica o elimina algún registro; con la finalidad de establecer responsabilidades en cuanto a la información manejada en el sistema.

- Es necesario leer el manual de usuario y del Administrador para obtener un mejor desempeño del sistema y evitar un mal uso del mismo.

- Realizar respaldos periódicamente de la Base de Datos.

- Crear una opción en el sistema que permita guardar imágenes, con la finalidad de tener una visualización del error que arroja una determinada falla.

BIBLIOGRAFÍA

[1] Acuña S y Rosales Z., **"DESARROLLO DE UN SISTEMA DE INFORMACIÓN PARA LAS ACTIVIDADES DE PRODUCCIÓN DE UNA EMPRESA CAFETALERA UBICADA EN BERGANTIN EDO-ANZOATEGUI." (2005)**

[2] Calsadilla A., **"DISEÑO DE UN SISTEMA DE INFORMACIÓN PARA LA AUTOMATIZACIÓN DEL PROCESO DE ARCHIVO DE EXPEDIENTES DE LOS EMPLEADOS ADSCRITOS A EL DEPARTAMENTO DE INGENIERÍA DE MANTENIMIENTO DE LA DIVISIÓN PLANTA MACAGUA, C.V.G ELECTRIFICACIÓN DEL CARONÍ C.A. (EDELCA)." (2005)**

[3] Camping, F. "Ingeniería de Software". [On – line].
http://.WWW.monografias.com **(2001).**

[4] Conceptos de Bases de Datos.
http://www.lafacu.com/apuntes/informatica/base_datos/default.htm **(2001).**

[5] González A., **"DISEÑO DE UN SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LAS FUNCIONES DE SUPERVISIÓN Y CONTROL DEL PRESUPUESTO DEL CONSEJO LEGISLATIVO DEL ESTADO ANZOÁTEGUI." (2006)**

[6] IBM IT EDUCATION SERVICES WORLDWIDE CERTIFIED MATERIAL. **"Guía de Estudiante del Curso Análisis y Diseño de Sistemas Orientados a Objetos Libro Nº. 1"**. Código del Curso: CY450. Versión 5.0. (2006).

[7] IBM LEARNING SERVICES WORLDWIDE CERTIFIED MATERIAL. **"Guía de Estudiante del Curso de Ingeniería de Software (Volumen 1: Fundamentos de Análisis y Diseño)"**. Código del Curso: CY440. Versión 4.1. (2006).

[8] IBM LEARNING SERVICES WORLDWIDE CERTIFIED MATERIAL. **"Guía de Estudiante del Curso de Ingeniería de Software (Volumen 2: Métricas, Calidad y Pruebas)"**. Código del Curso: CY440. Versión 4.1. (2006).

[9] IBM IT EDUCATION SERVICES WORLDWIDE CERTIFIED MATERIAL. **"Guía de Estudiante del Curso Desarrollo de Aplicaciones con XML Libro Nº.1"**. Código del Curso: CY730. Versión 3.0. (2006).

[10] IBM IT EDUCATION SERVICES WORLDWIDE CERTIFIED MATERIAL. **"Guía de Estudiante del Curso Desarrollo de Aplicaciones con XML Libro Nº.2"**. Código del Curso: CY730. Versión 3.0. (2006).

[11] Jacobson I., Booch G. y Rumbaugh J. "**El Lenguaje Unificado de Modelado - Manual de Referencia".** Primera Edición. Pearson Educación. Madrid **(2000).**

[12] Jacobson I., Booch G. y Rumbaugh J. "**El Proceso Unificado de Desarrollo de Software".** Primera Edición. Pearson Educación. Madrid **(2000).**

[13] Kendall, J. y Kendall, K. "**Análisis y Diseño de Sistemas".** Tercera edición. Editorial Pearson Educación, Mexico, **(1999).**

[14] Marcano A., **"DESARROLLO DE UN SOFTWARE DE GESTIÓN Y CONTROL DE DEMANDAS PARA LA CONSULTORÍA JURÍDICA DE UNA EMPRESA DE TELECOMUNICACIONES. "(2007)**

[15] Medina J., **"DESARROLLO DE UN SISTEMA BASADO EN APLICACIONES WEB PARA LA AUTOMATIZACIÓN DEL CONTROL DE PEDIDOS ASOCIADOS AL PROCESO DE VENTAS DE UNA EMPRESA CAFETALERA." (2007)**

[16] Murillo, V. "**Bases de Datos",** http://.galeon.com/materiasis/seminariotemas.html **(2001).**

[17] Politecnico di Milano. **"Modelado de Hipertexto"**. Disponible en http://www.webml.org/webml/page18.do?dau70.oid=12&UserCtxParam=0&GrouptParam=0&ctx1=EN.

APÉNDICE A

MANUAL DE USUARIO

REGISTRO DE FALLAS DE APLICACIONES

(REFAP)

CONTENIDO

1 INTRODUCCIÓN

Refap es una aplicación que permite registrar información concerniente a las fallas que se generan en las diferentes aplicaciones existentes en el Mejorador PDVSA PETROCEDEÑO.

El objetivo de Refap es mantener de forma automatizada y organizada un respaldo de las soluciones que los analistas de la coordinación IS aplican para resolver las fallas de aplicaciones, brindándoles comodidad a la hora de buscar información sobre dichas fallas.

REFAP cuenta con una interfaz gráfica amigable y de fácil interactividad para los usuarios.

El propósito que se persigue con el presente manual es dar a conocer a los usuarios finales las características y las formas de funcionamiento del sistema REFAP.

REFAP está constituido por 4 módulos que se identifican a continuación:

- Gestionar Administración. (Ver detalles en el manual del administrador).
- Gestionar Fallas.
- Realizar Consultas.
- Generar Reportes.

2 INICIO DE SESIÓN

Refap es una aplicación Web; por lo tanto; su acceso es a través de la dirección electrónica: http://sup-36/appsnet/Refap/INICIAR_SESION.aspx en la cual se visualizará la siguiente ventana:

Para iniciar la sesión se debe ingresar el Usuario y la contraseña, presionar (clic con el Mouse) en el botón aceptar. Se validan los datos introducidos contra los datos cargados en la base de datos refap y se accederá al sistema siempre y cuando sea un usuario valido registrado.

Si el usuario y la clave ingresada son correctos, se verifican los privilegios que tiene el usuario, es decir, se determinan a que módulos puede acceder de acuerdo al

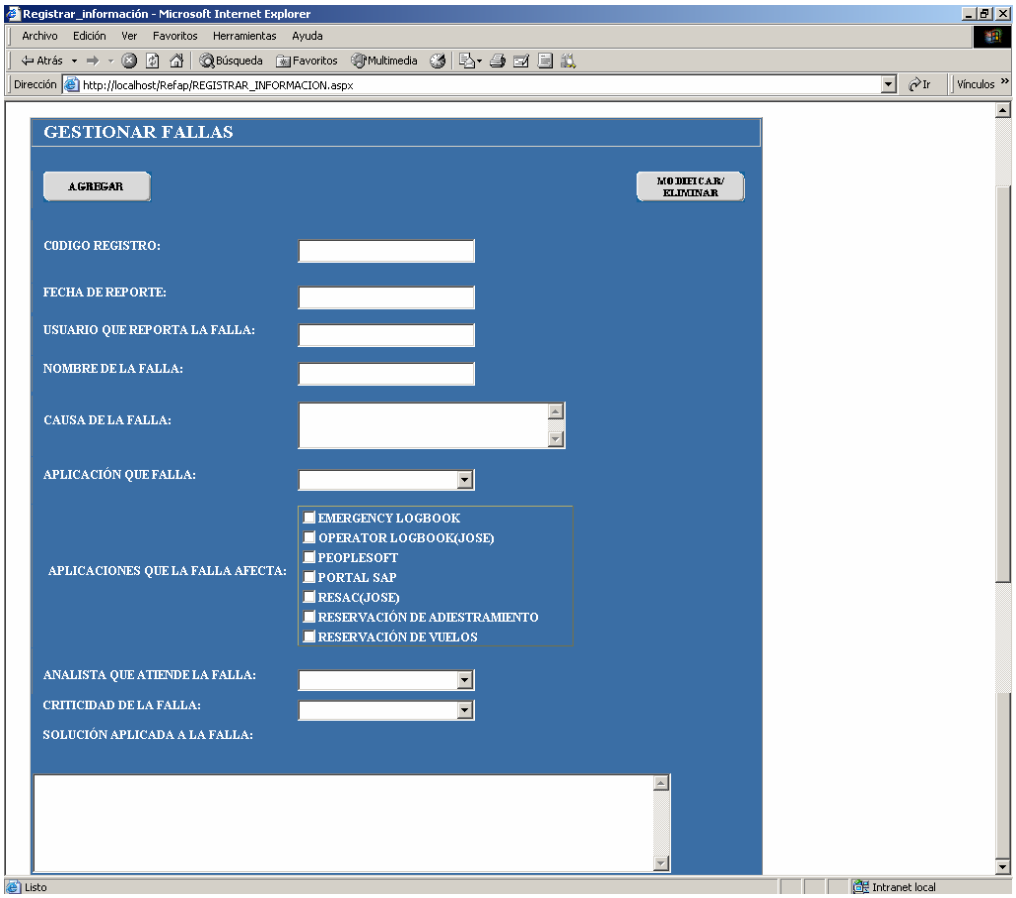
permiso que le haya otorgado el administrador del sistema. Luego aparecerá la ventana con el menú principal, tal como se muestra a continuación:

El usuario presionando (clic con el Mouse) sobre la opción que desee del menú, accederá al módulo seleccionado.

3 GESTIONAR FALLAS

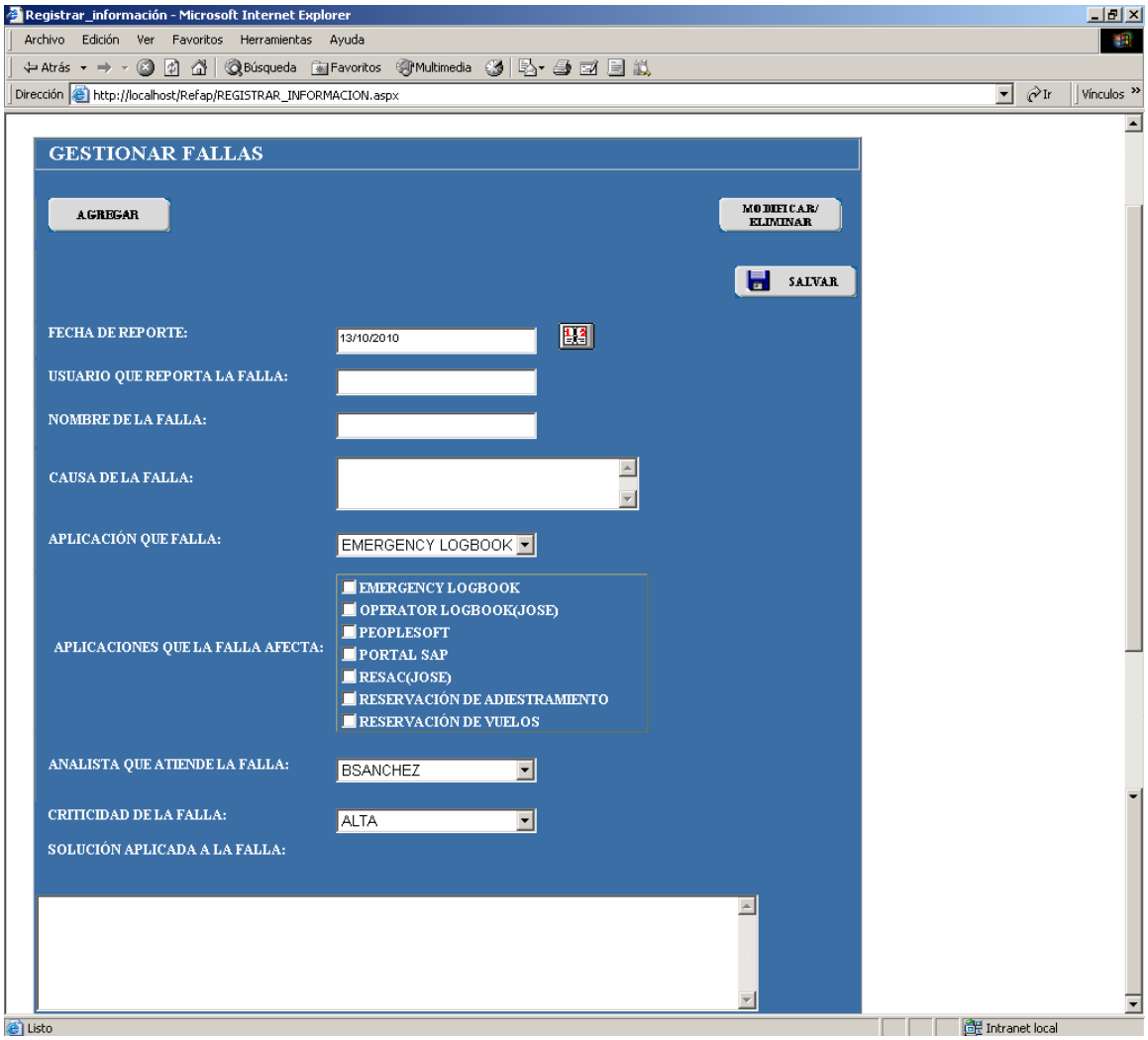
Si presiona la opción Gestionar Fallas, se desplegará la ventana de Gestionar Fallas que contiene todos los campos que se requieren llenar para documentar una falla, tal como se muestra a continuación:

En la figura anterior no se puede observar completamente el formulario de Gestionar Fallas, debido al espacio que abarca. A continuación se muestra el formulario en cuestión.

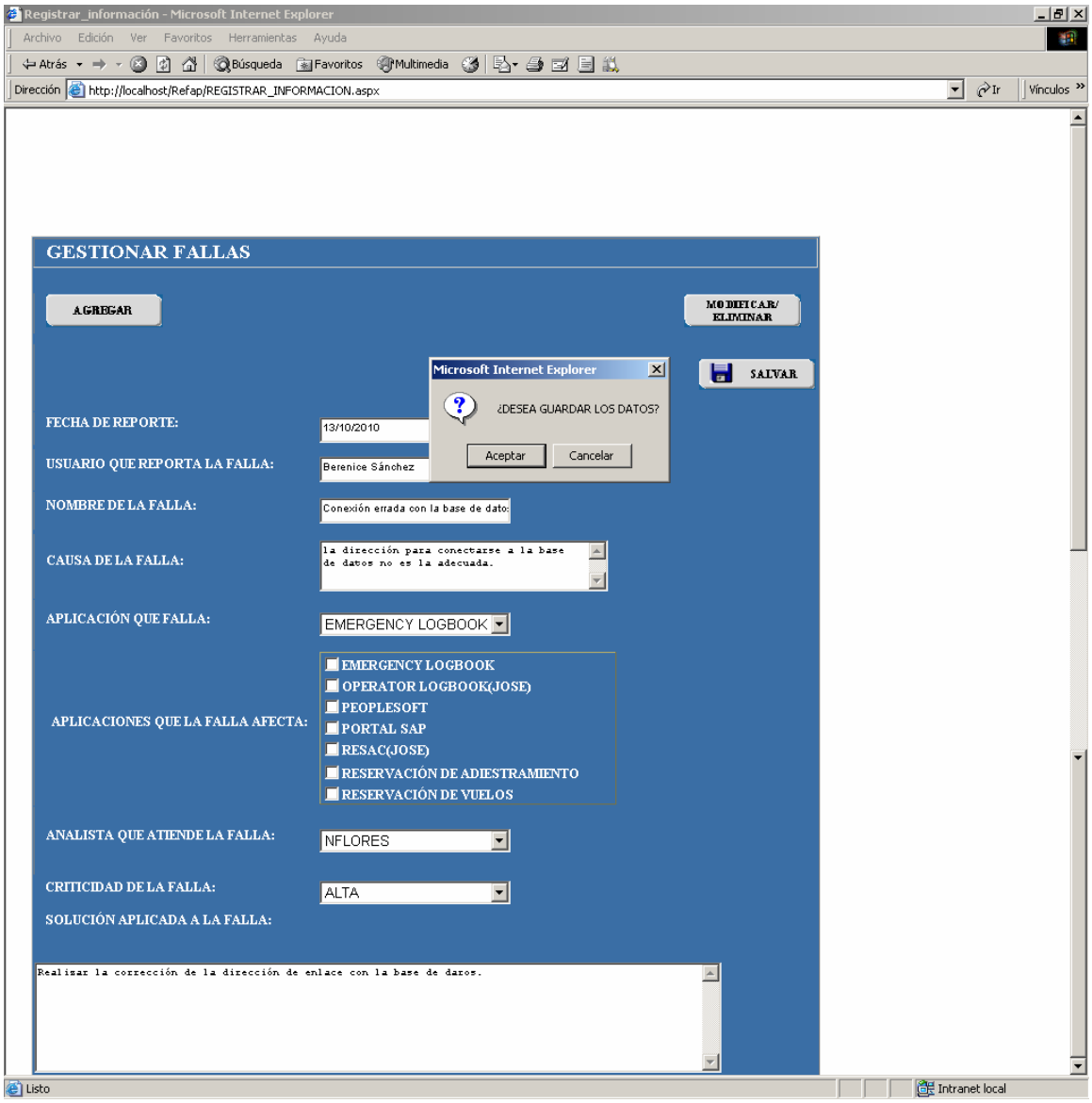


3.1 Gestionar Fallas (Agregar)

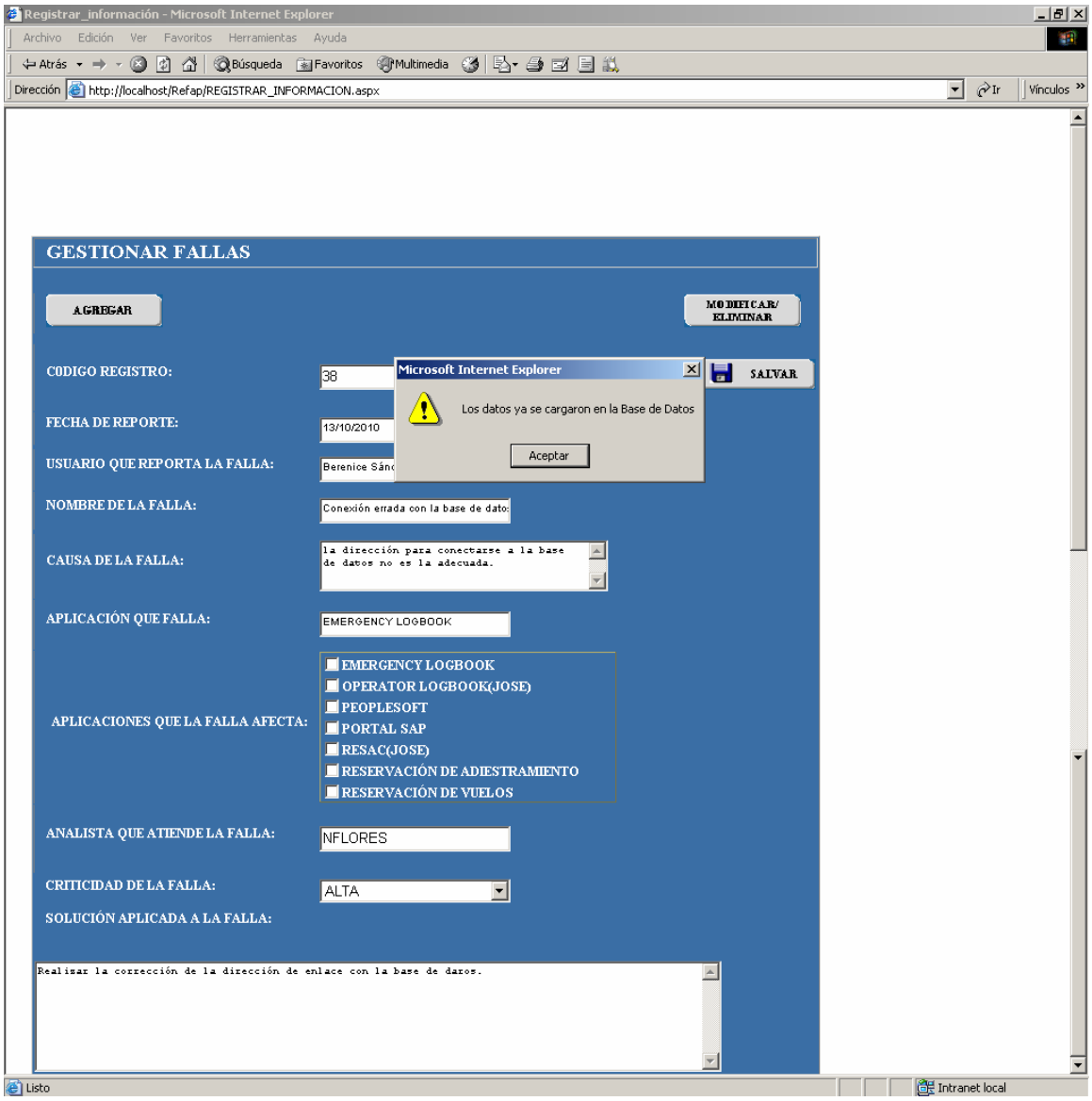
Se debe presionar sobre el botón agregar para ingresar una nueva falla, se visualizará el botón de salvar y la información que es estática en el sistema.



Se deben llenar los campos que se encuentran vacíos y seleccionar la información adecuada de las listas que están previamente cargadas y presionar el botón salvar, aparece un mensaje de confirmación.



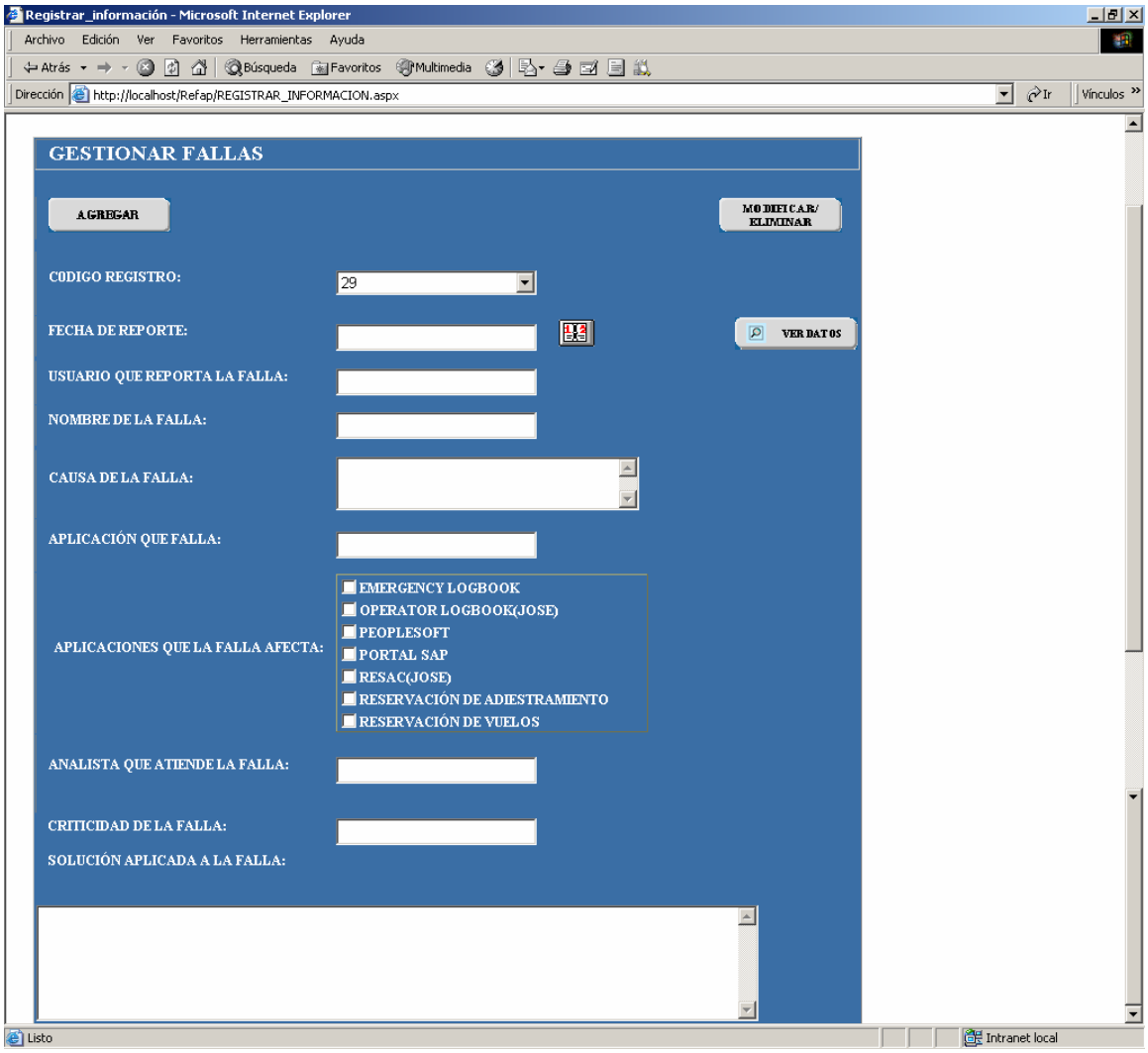
Se debe presionar Aceptar, para guardar los datos en la base de datos refap.



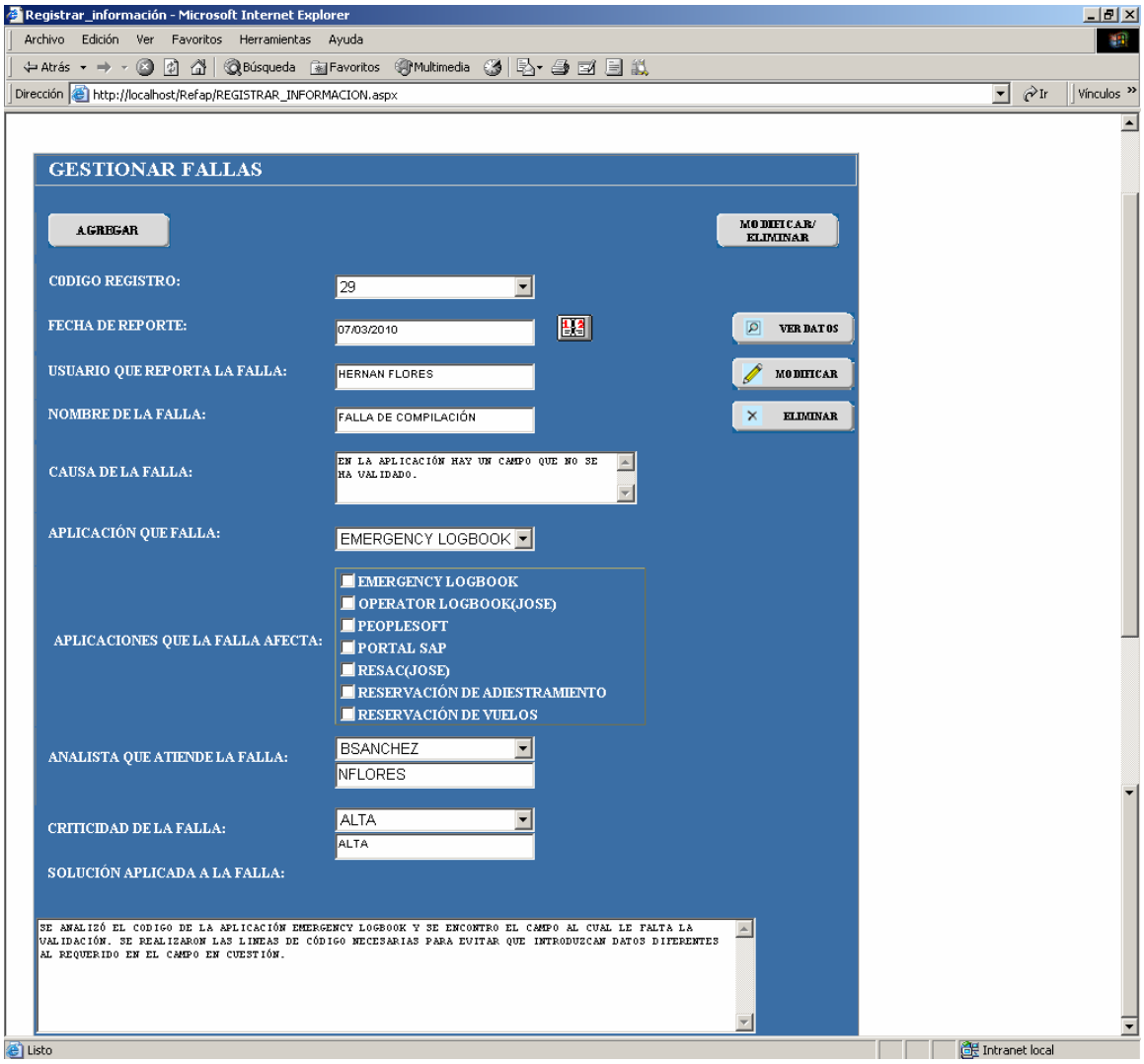
Se puede observar que al guardar los datos se le genera un código de registro a la falla.

3.2 Gestionar Fallas (Modificar/Eliminar)

Al presionar sobre el botón Modificar/Eliminar aparece la ventana de Gestionar Fallas con una lista de todos los códigos de registros que se han generado, como se visualiza a continuación:

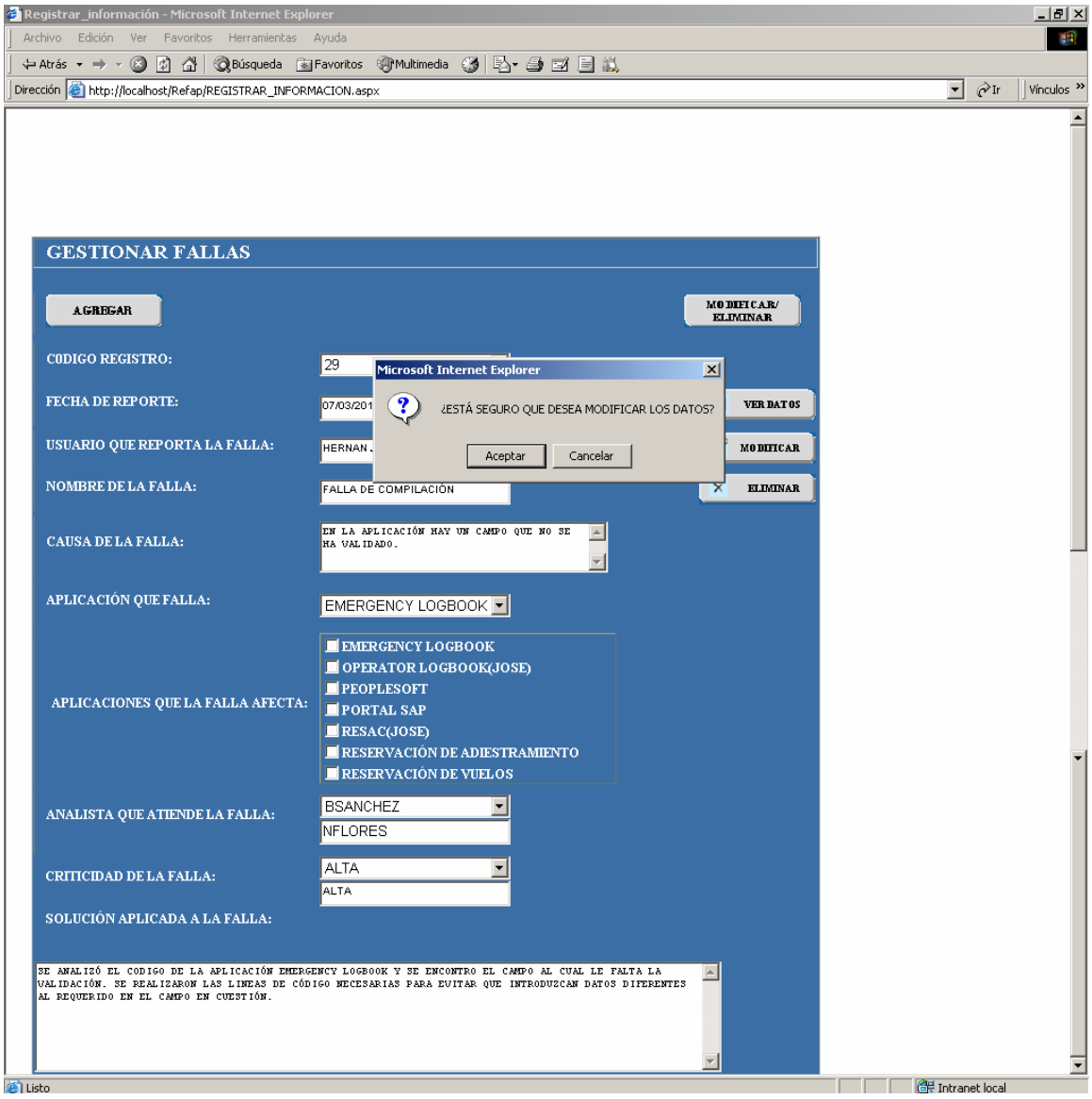


Se debe seleccionar el código de registro que se desee consultar y presionar en el botón ver datos. Se desplegará toda la información almacenada sobre dicho registro, además se habilitan los botones de modificar y eliminar.

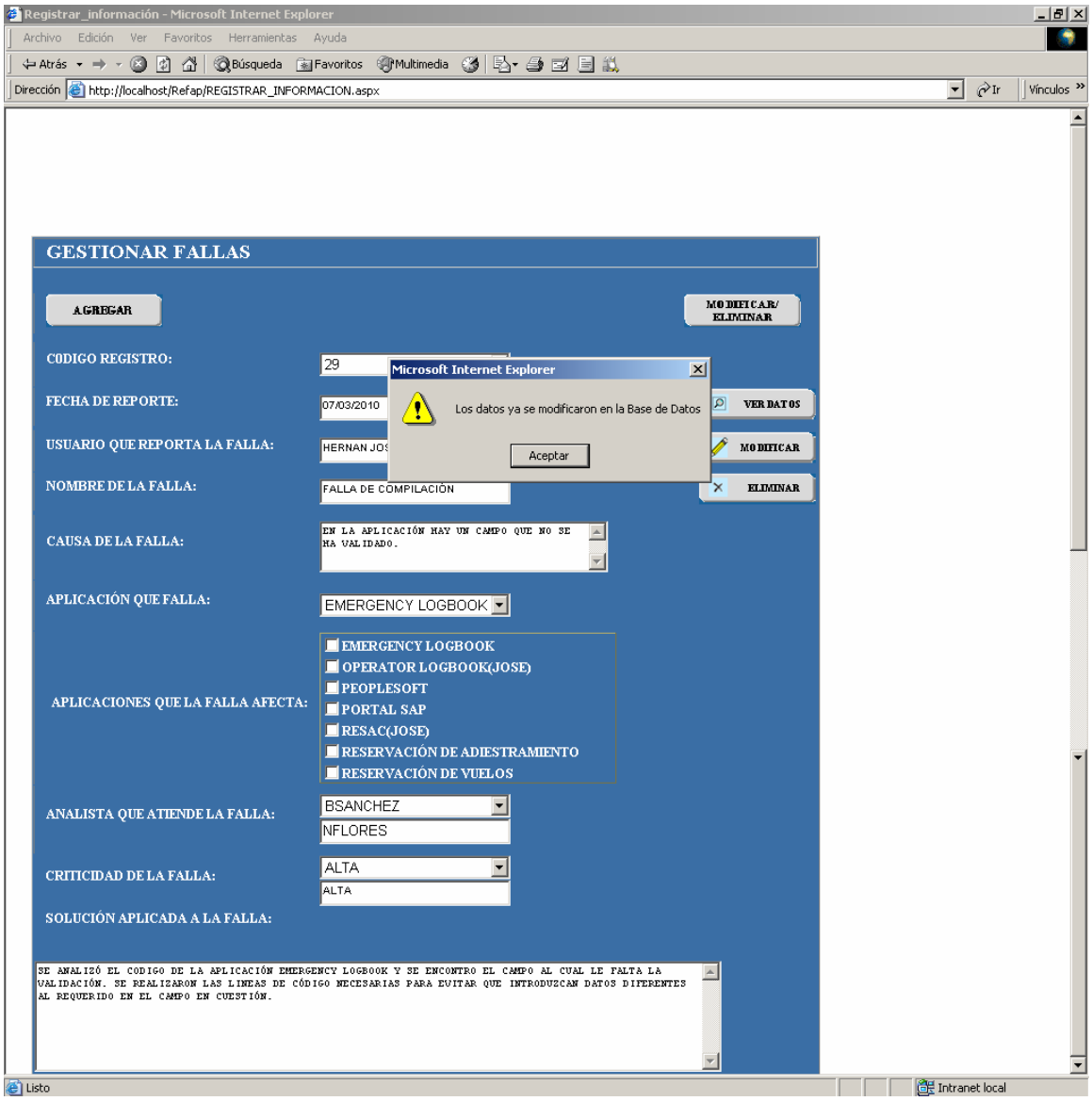


3.2.1 Gestionar Fallas (Modificar un registro)

Para modificar algún campo del registro, se realizan los cambios que desee y presionar el botón de modificar, aparecerá un mensaje de confirmación.



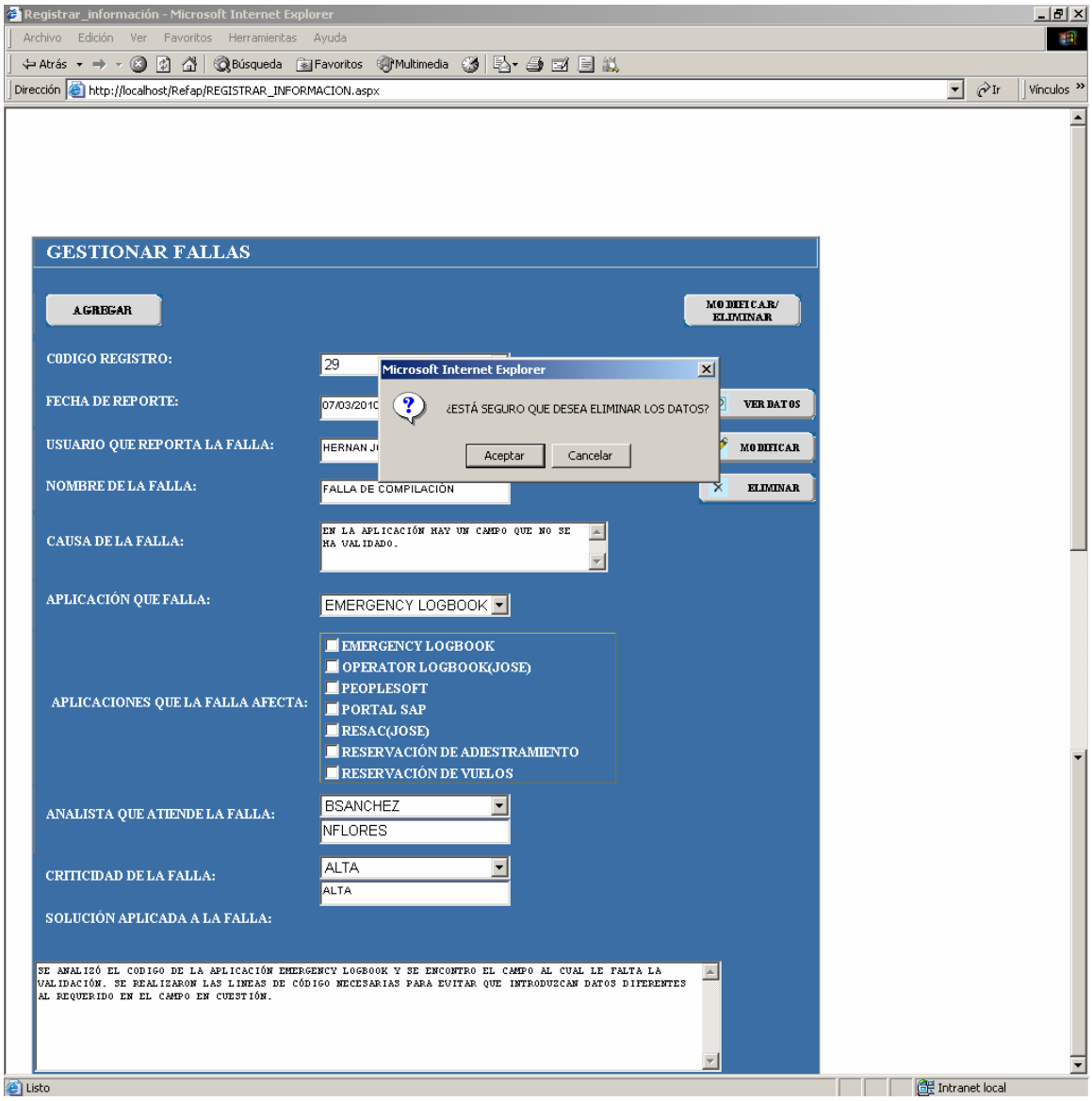
Presionar el botón OK, para que se guarden los cambios en la base de datos refap**.**



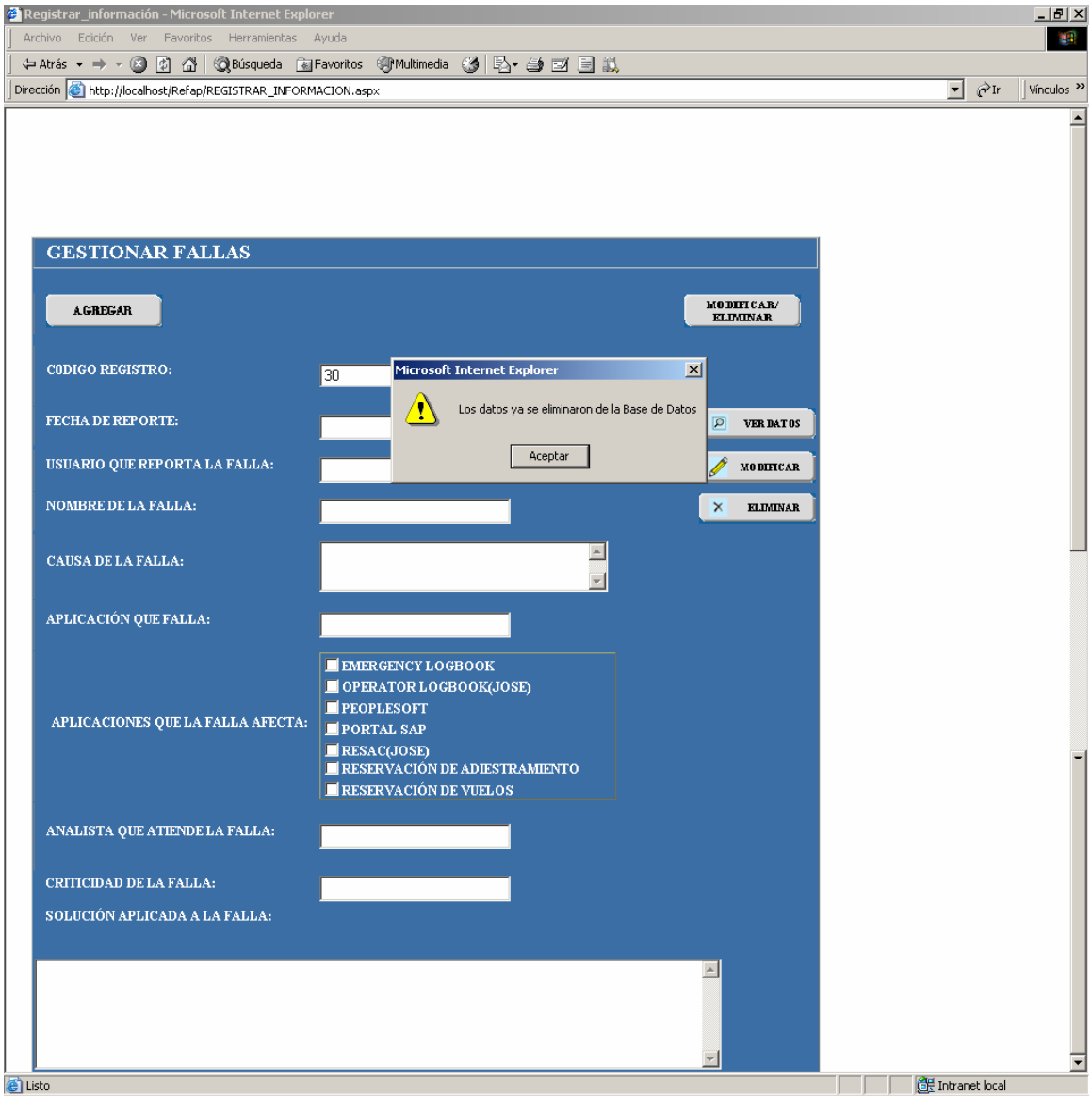
Los datos se modifican en la base de datos refap, tal como lo indica el mensaje anterior mostrado en la pantalla de Gestionar Fallas.

3.2.2 Gestionar Fallas (Eliminar un registro)

Para eliminar un registro se debe seleccionar el registro a eliminar y presionar el botón eliminar.



Se debe presionar el botón OK; para eliminar el registro. Se observará el mensaje que ratifica que los datos fueron eliminados de la base de datos refap, como se muestra a continuación:

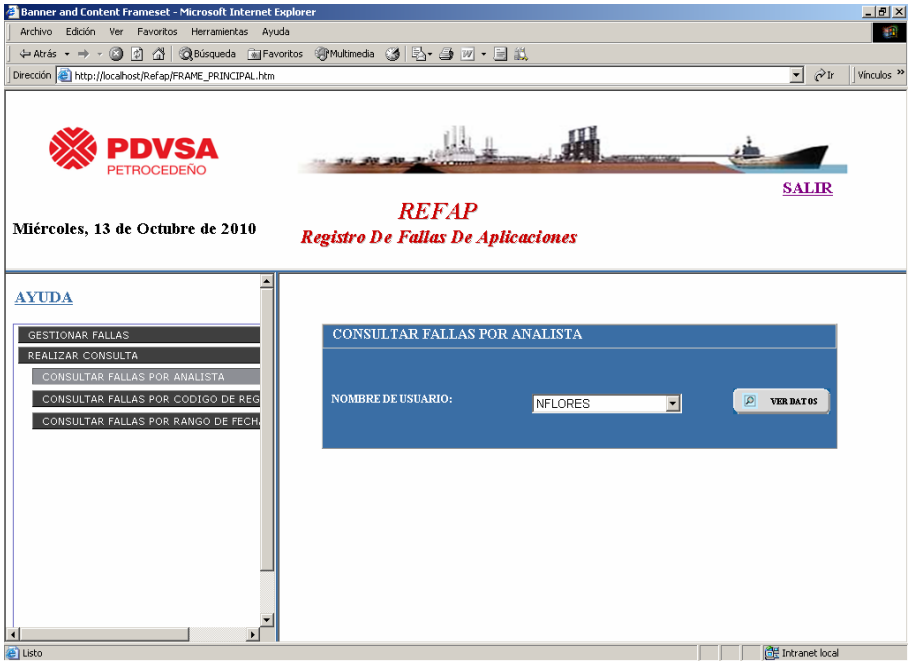


4 REALIZAR CONSULTA

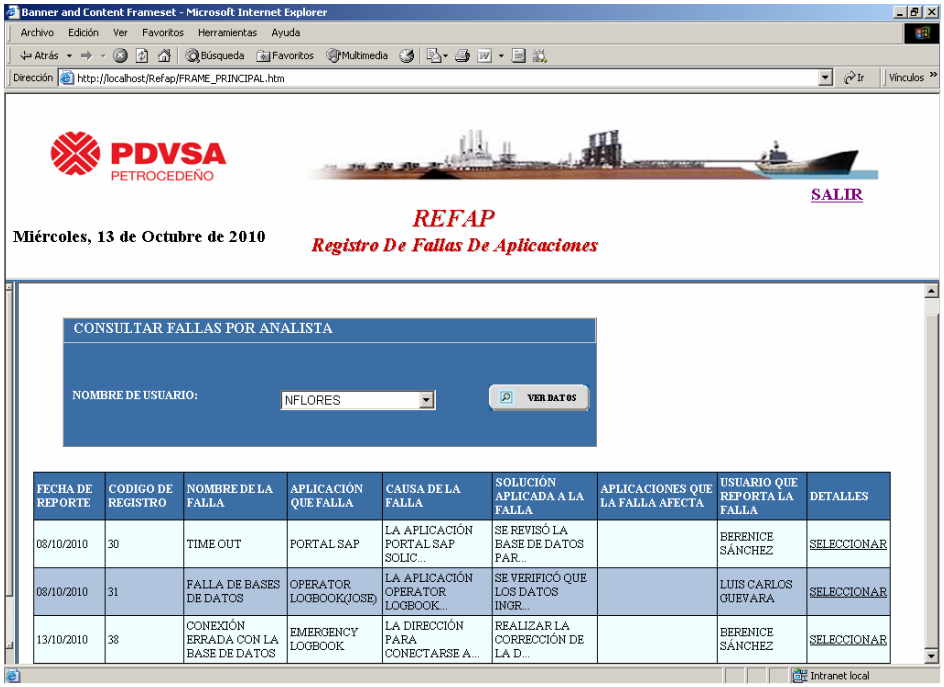
Si presiona la opción Realizar Consulta se desplegará el submenú que contempla dicho módulo, como se muestra a continuación:

4.1 Consultar fallas por Analista

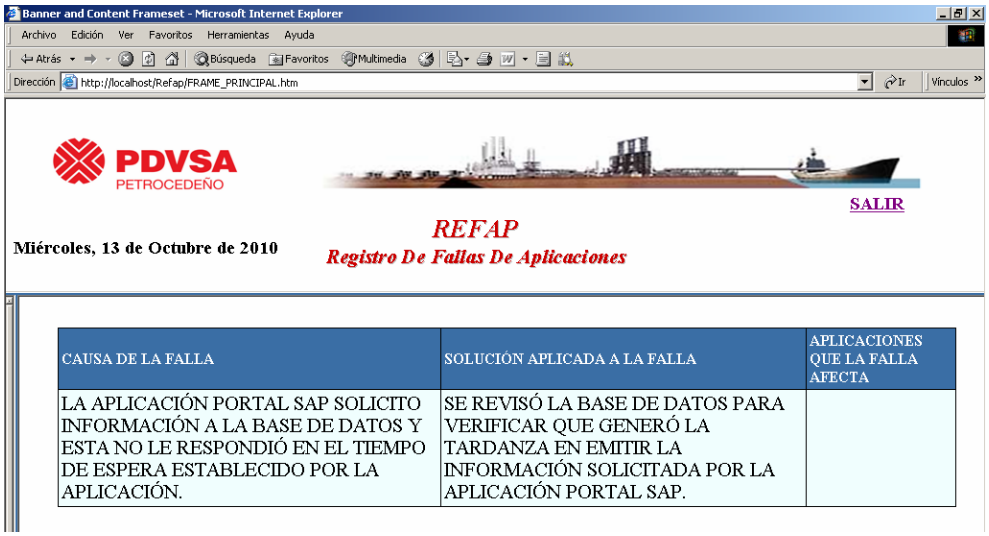
Si selecciona la opción consultar fallas por analista, se visualizará la siguiente pantalla:



Se escoge el analista a consultar y se presiona el botón ver datos, se desplegarán todas las fallas atendidas por dicho analista, tal como se puede observar a continuación:



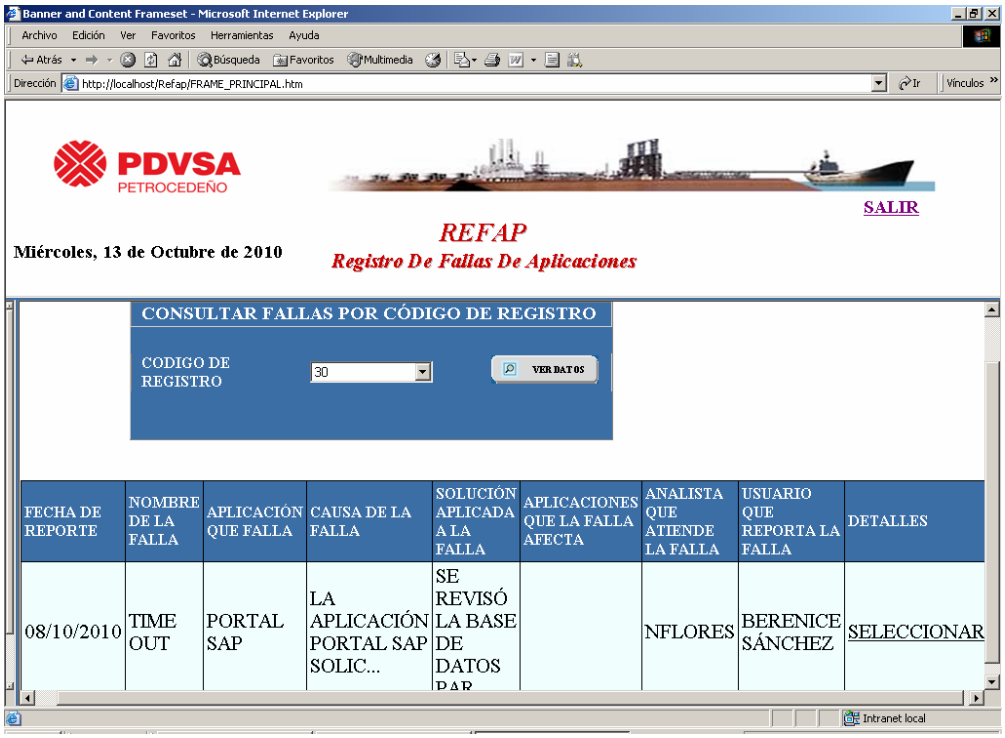
Como se puede apreciar la tabla desplegada contiene una columna denominada detalles, la cual posee un link seleccionar, mediante éste se puede observar en otra tabla de forma más amplia la información de los campos causa de la falla, solución aplicada a la falla y aplicaciones que la falla afecta.



4.2 Consulta por código de registro

Al seleccionar ésta opción se observará la siguiente pantalla:

Se selecciona el código de registro a consultar y se presiona el botón ver datos, se desplegarán toda la información de la falla al cual pertenece dicho código, tal como se puede apreciar a continuación:

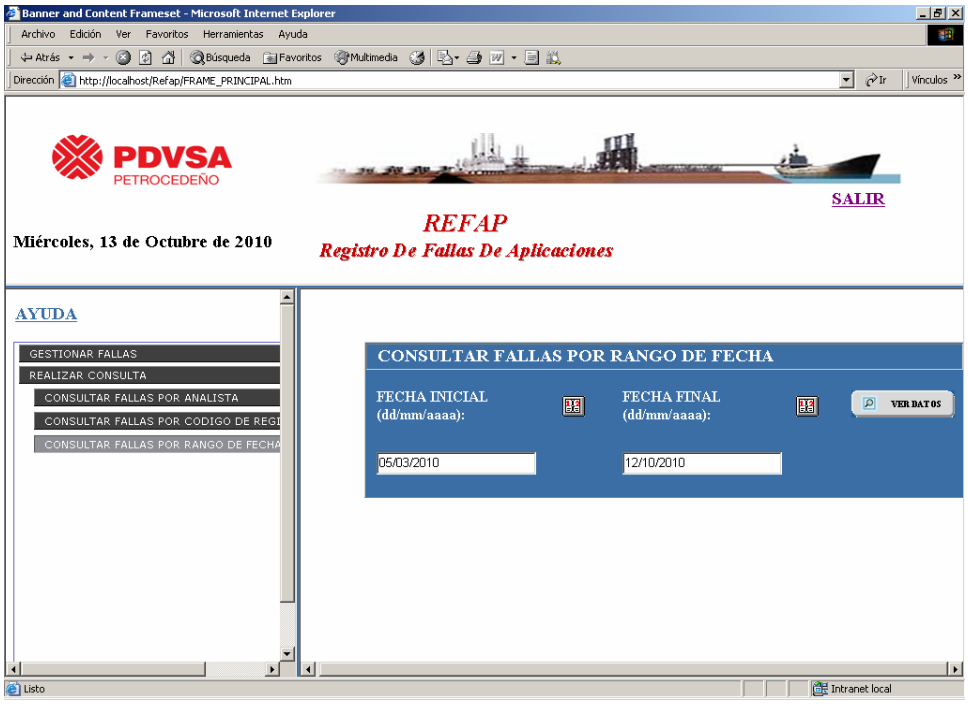


El link seleccionar de la columna detalles funciona igual que en la consulta por analista.

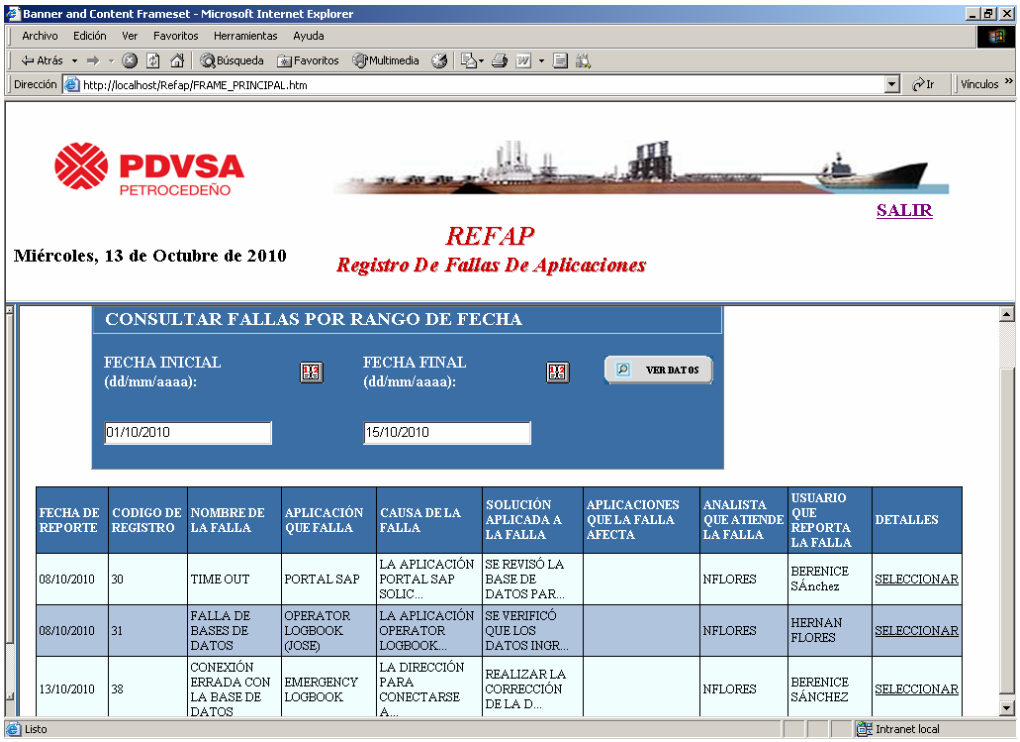
4.3 Consultar fallas por rango de fechas

Al seleccionar consultar fallas por rango de fechas se observará la siguiente ventana:

Presionando con un clic sobre los botones que se encuentran al lado de las fechas, aparecerá el calendario para que seleccione el rango de fechas que desea consultar.



Presionar con un clic sobre el botón ver datos para observar todas las fallas con su respectiva documentación, que se encuentran registradas en el intervalo de fechas seleccionado.

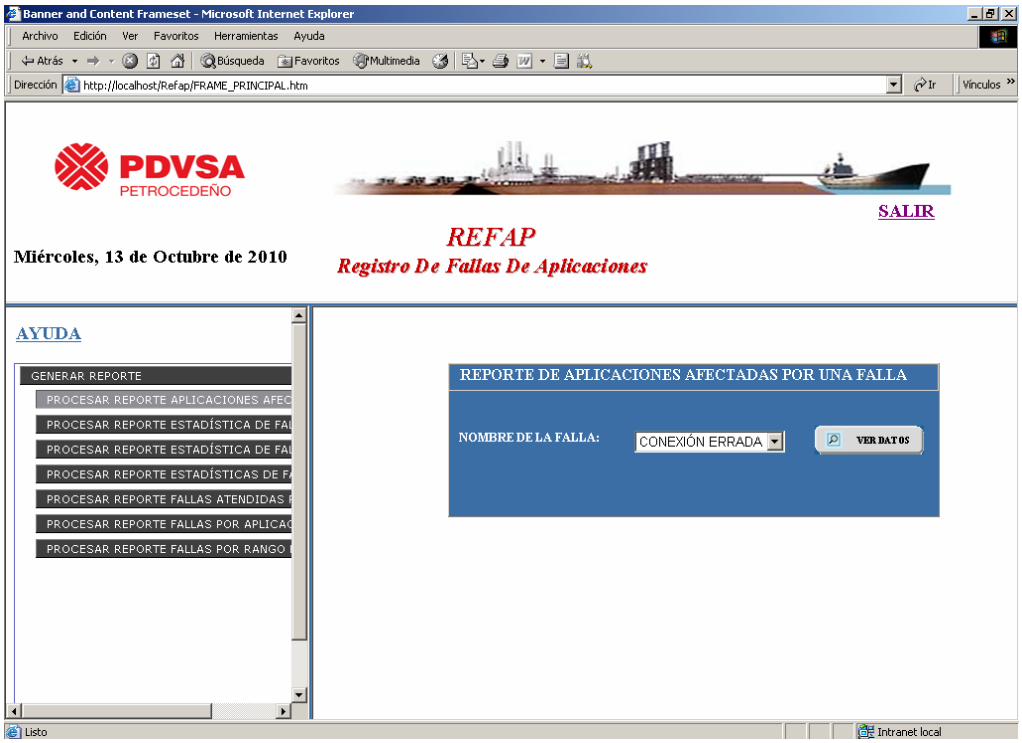


5 GENERAR REPORTES

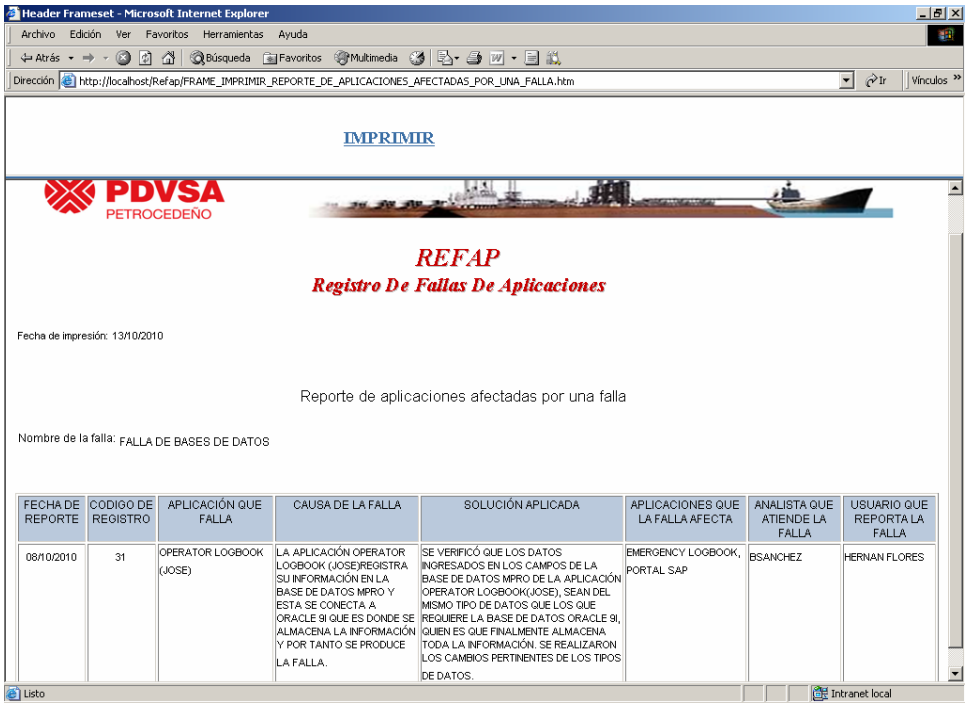
Si presiona la opción Generar Reportes se desplegará el submenú que contempla dicho módulo, como se muestra a continuación:

5.1 Procesar Reporte Aplicaciones afectadas por una falla

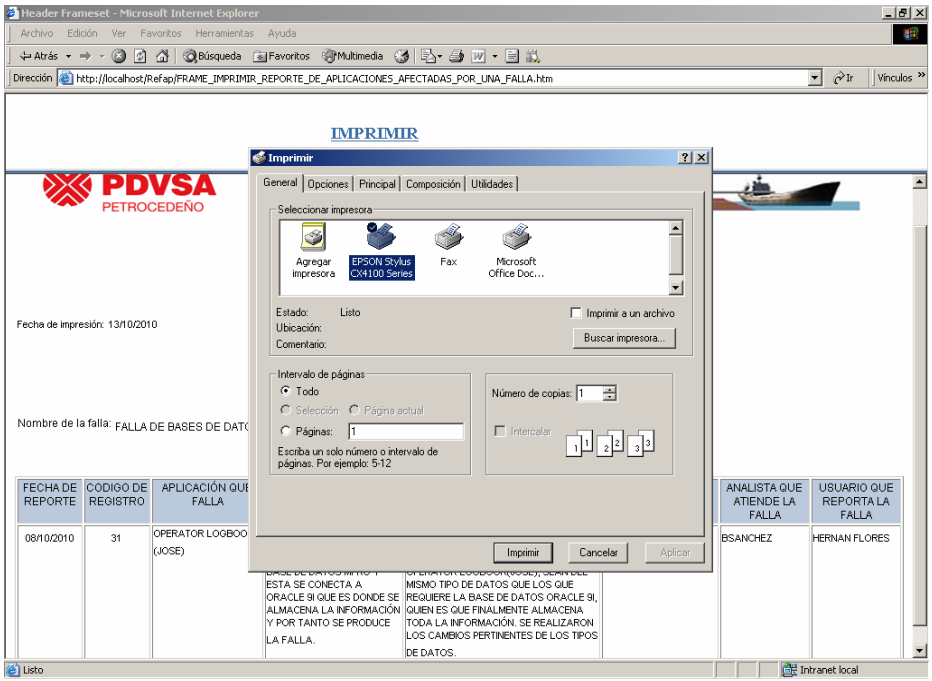
Al seleccionar la ésta opción, se observará la siguiente ventana:



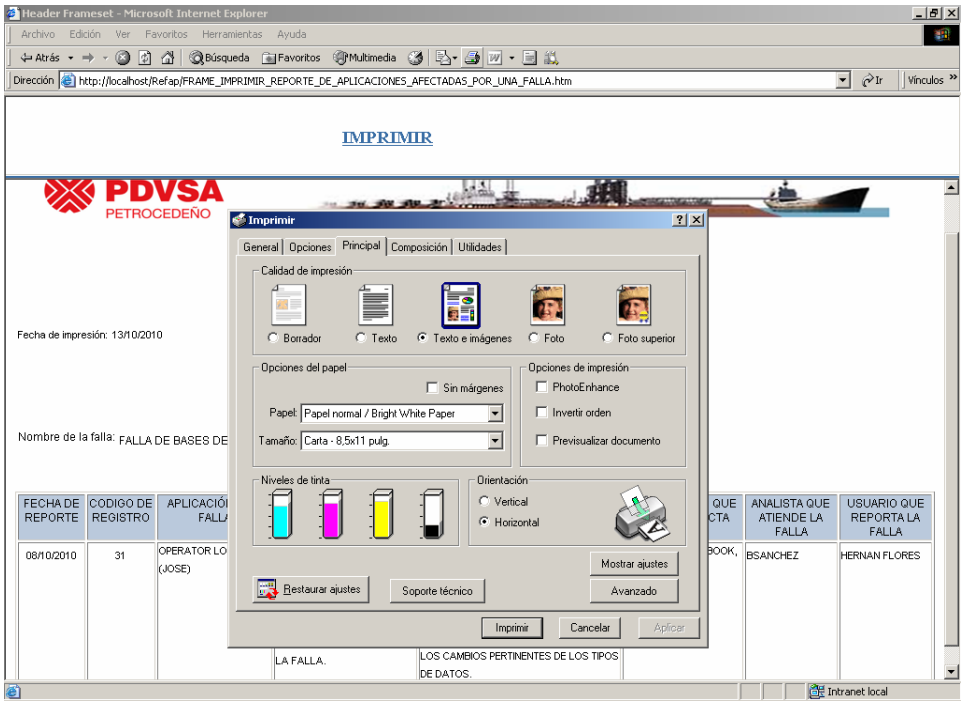
Se debe presionar el botón ver datos y se desplegará el reporte que contiene la información de las aplicaciones que son afectadas cuando se produce una determinada falla.



Presionando el link imprimir aparecerá la siguiente ventana:



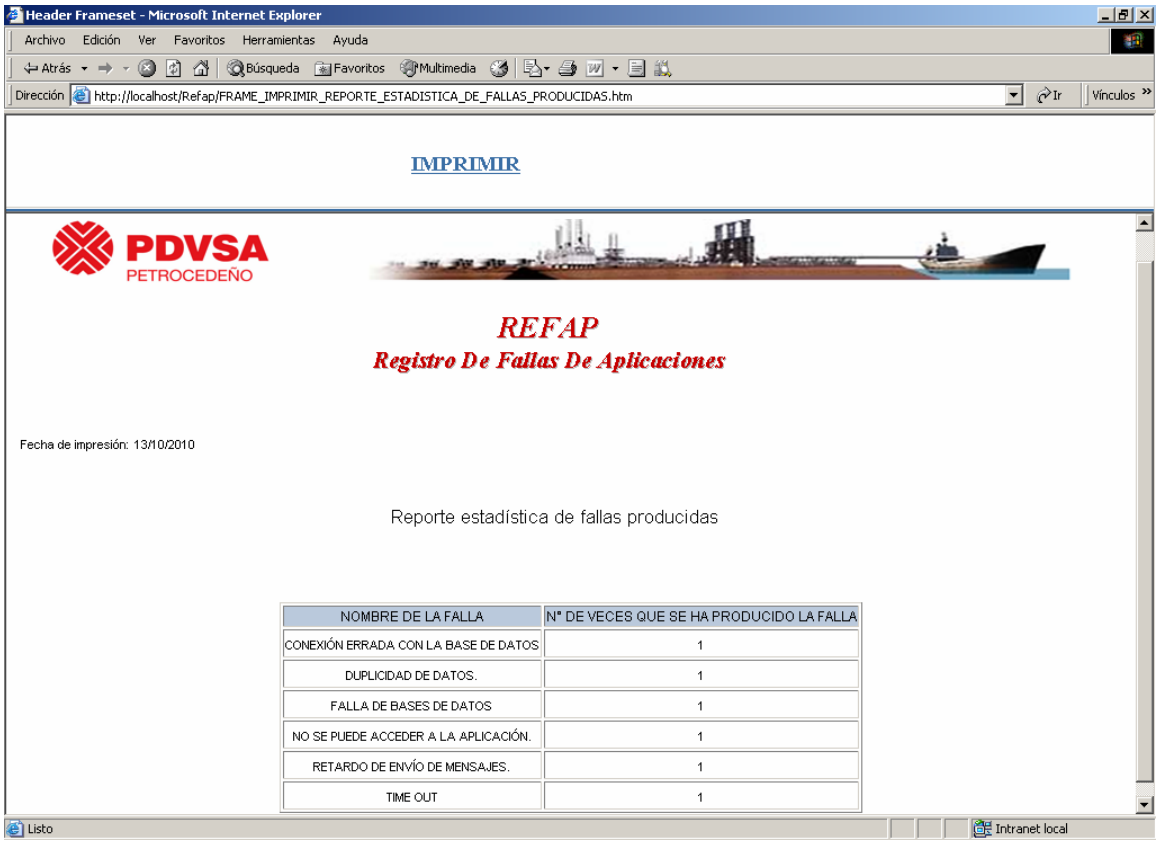
Configurar la página presionando la opción Principal. Elegir el tamaño de papel y seleccionar Orientación Horizontal, como se muestra a continuación:



Presionar el botón Imprimir.

5.2 Estadística de fallas producidas

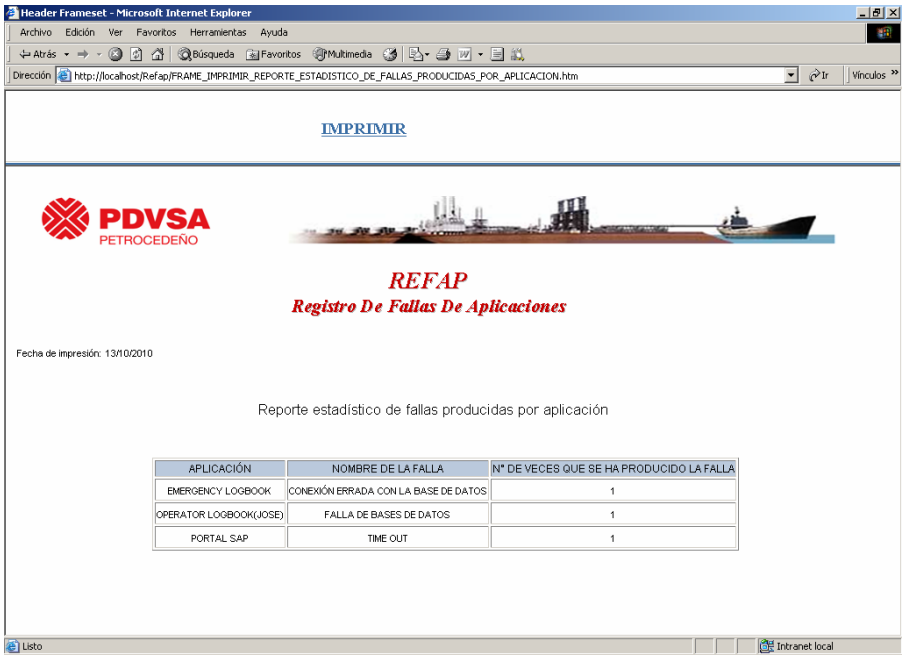
Seleccionando ésta opción se visualizará el reporte que refleja las veces que se ha producido una determinada falla.



Presionar el link imprimir para imprimir el reporte.

5.3 Estadísticas de fallas producidas por aplicación

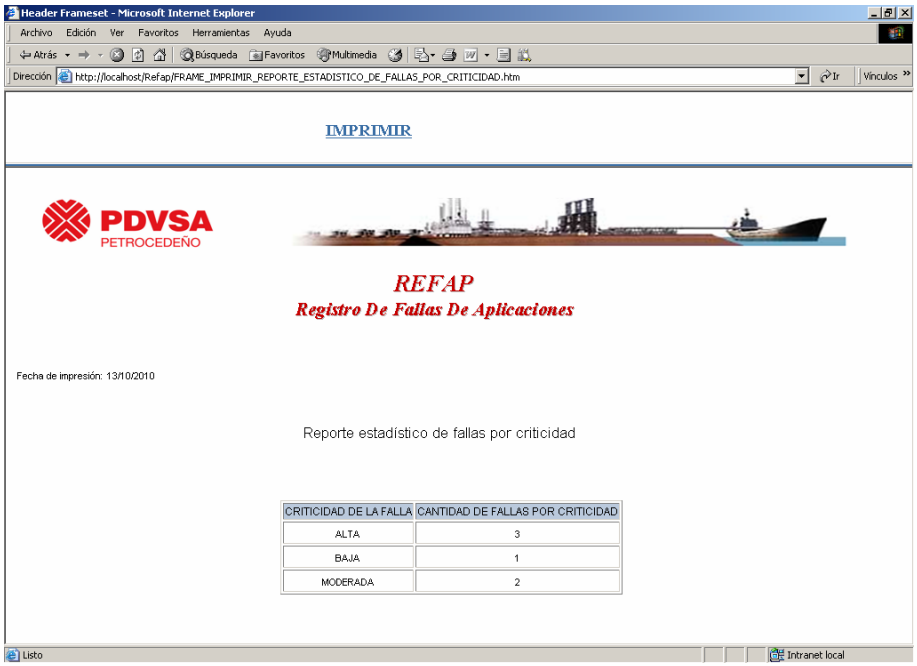
Mediante esta opción se reflejará el reporte con todas las aplicaciones y las fallas que se han producido en cada una de ellas, al igual que la cantidad de veces que dicha falla se ha presentado en la aplicación, tal como se observa en la siguiente pantalla:



Para imprimir el reporte se debe presionar el link imprimir.

5.4 Estadística de fallas por criticidad

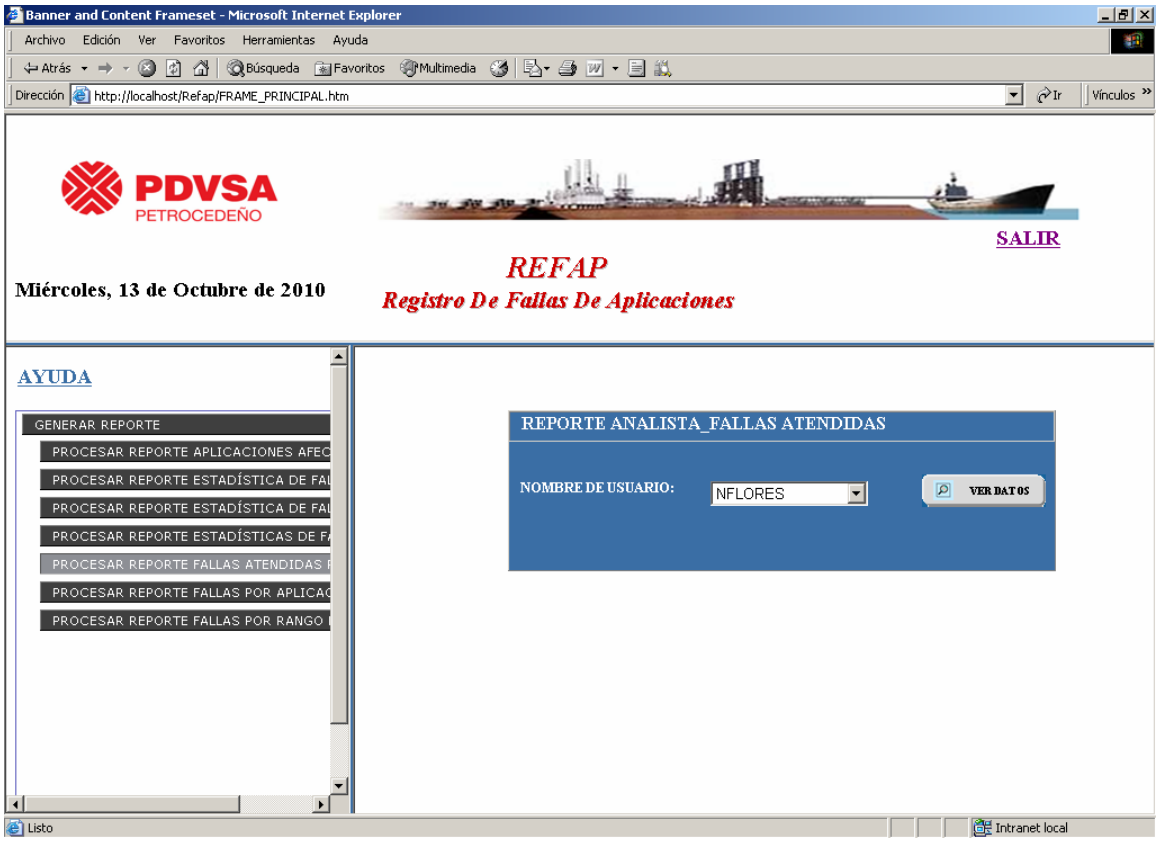
Seleccionado esta opción podrá visualizar el reporte con la estadística de las fallas dependiendo del tipo de criticidad que posea.



Para imprimir presionar el link correspondiente.

5.5 Fallas atendidas por analistas

Al seleccionar esta opción se visualizará la siguiente ventana:



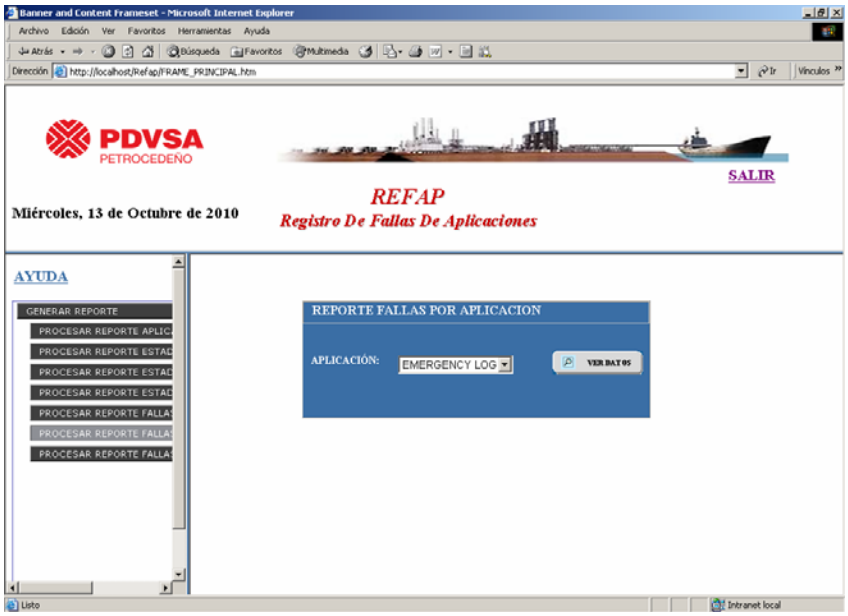
Presionar el botón ver datos y se visualizará el reporte con todas las fallas atendidas por el analista seleccionado.

FECHA DE REPORTE	CODIGO DE REGISTRO	NOMBRE DE LA FALLA	APLICACIÓN QUE FALLA	CAUSA DE LA FALLA	SOLUCIÓN APLICADA	APLICACIONES QUE LA FALLA AFECTA	USUARIO QUE REPORTA LA FALLA
13/10/2010	38	CONEXIÓN ERRADA CON LA BASE DE DATOS	EMERGENCY LOGBOOK	LA DIRECCIÓN PARA CONECTARSE A LA BASE DE DATOS NO ES LA ADECUADA.	REALIZAR LA CORRECCIÓN DE LA DIRECCIÓN DE ENLACE CON LA BASE DE DAROS.		BERENICE SÁNCHEZ
13/10/2010	39	RETARDO DE ENVÍO DE MENSAJES.	EMERGENCY LOGBOOK	LA BASE DE DATOS ESTÁ SOBRECARGADA DE INFORMACIÓN.	DEPURAR LA BASE DE DATOS Y AUMENTAR LA CAPACIDAD DE ALMACENAJE DE LA MISMA.		HECTOR MÁRQUEZ
13/10/2010	40	DUPLICIDAD DE DATOS.	PEOPLESOFT	LA BASE DE DATOS NO ESTÁ NORMALIZADA.	APLICACIÓN DE LAS NORMAS DE NORMALIZACIÓN A LA BASE DE DATOS.	RESERVACIÓN DE ADIESTRAMIENTO, RESERVACIÓN DE VUELOS	CARLOS BLANCO
13/10/2010	41	NO SE PUEDE ACCEDER A LA APLICACIÓN.	EMERGENCY LOGBOOK	SE ELIMINARON ARCHIVOS DE LA CARPETA DE LA APLICACIÓN EMERGENCY LOGBOOK.	RESTAURAR LOS ARCHIVOS DE LA APLICACIÓN.		HERNELIS FLORES

Para imprimir el reporte presionar el link correspondiente.

5.6 Fallas por aplicación

Al seleccionar esta opción se observará la siguiente ventana:



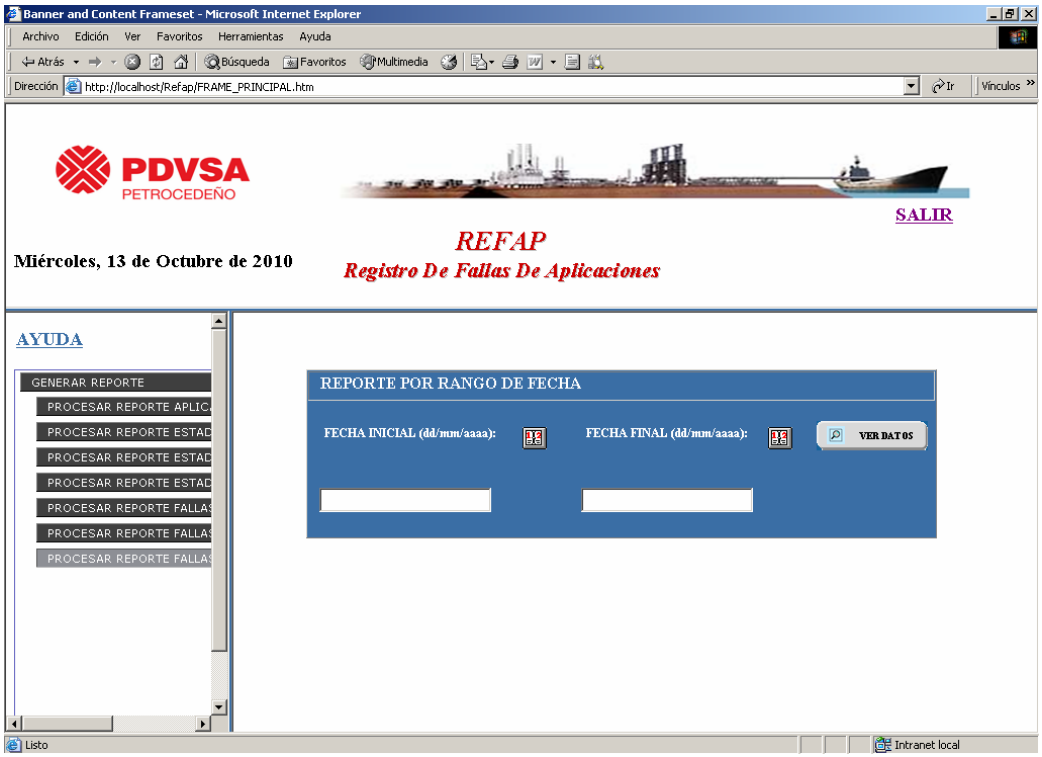
Seleccione la aplicación que desee y presione el botón ver datos. Se visualizará el reporte que contiene todas las fallas que se han producido en la aplicación previamente seleccionada.

FECHA DE REPORTE	CODIGO DE REGISTRO	NOMBRE DE LA FALLA	CAUSA DE LA FALLA	SOLUCIÓN APLICADA	APLICACIONES QUE LA FALLA AFECTA	ANALISTA QUE ATIENDE LA FALLA	USUARIO QUE REPORTA LA FALLA
13/10/2010	38	CONEXIÓN ERRADA CON LA BASE DE DATOS	LA DIRECCIÓN PARA CONECTARSE A LA BASE DE DATOS NO ES LA ADECUADA.	REALIZAR LA CORRECCIÓN DE LA DIRECCIÓN DE ENLACE CON LA BASE DE DAROS.		NFLORES	BERENICE SÁNCHEZ
13/10/2010	39	RETARDO DE ENVÍO DE MENSAJES.	LA BASE DE DATOS ESTÁ SOBRECARGADA DE INFORMACIÓN.	DEPURAR LA BASE DE DATOS Y AUMENTAR LA CAPACIDAD DE ALMACENAJE DE LA MISMA.		NFLORES	HECTOR MÁRQUEZ
13/10/2010	41	NO SE PUEDE ACCEDER A LA APLICACIÓN.	SE ELIMINARON ARCHIVOS DE LA CARPETA DE LA APLICACIÓN EMERGENCY LOGBOOK.	RESTAURAR LOS ARCHIVOS DE LA APLICACIÓN.		NFLORES	HERNELIS FLORES

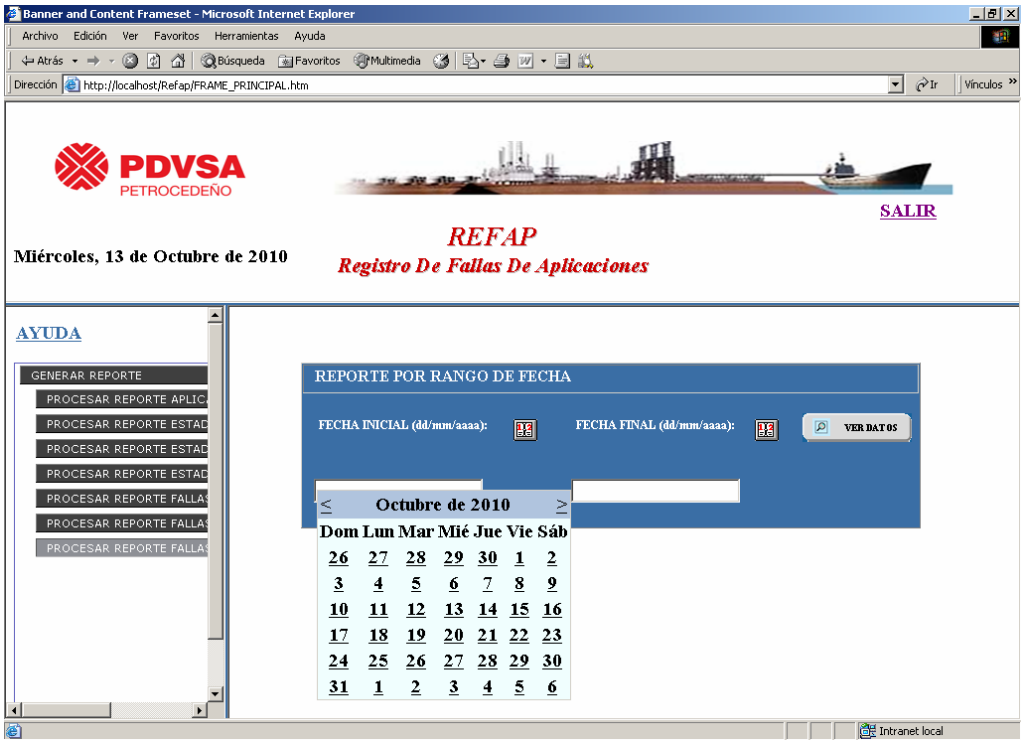
Presionar el link imprimir para imprimir el reporte.

5.7 Por rango de fechas

Seleccionando esta opción se visualizará la ventana que se muestra a continuación:



Presionando sobre los botones que se encuentran a un lado de las etiquetas de las fechas se puede seleccionar del calendario la fecha inicial y final del rango a visualizar.



Presionar ver datos, se mostrará el reporte que contempla todas las fallas registradas en el rango de fechas ingresado, ordenadas por fecha y código de registro.

FECHA DE REPORTE	CODIGO DE REGISTRO	NOMBRE DE LA FALLA	APLICACIÓN QUE FALLA	CAUSA DE LA FALLA	SOLUCIÓN APLICADA	APLICACIONES QUE LA FALLA AFECTA	ANALISTA QUE ATIENDE LA FALLA	USUARIO QUE REPORTA LA FALLA
13/10/2010	38	CONEXIÓN ERRADA CON LA BASE DE DATOS	EMERGENCY LOGBOOK	LA DIRECCIÓN PARA...	REALIZAR LA CORRECCIÓN DE LA DIRECCIÓN DE ENLACE CON LA BASE DE DAROS.		NFLORES	BERENICE SÁNCHEZ
13/10/2010	39	RETARDO DE ENVÍO DE MENSAJES.	EMERGENCY LOGBOOK	LA BASE DE DATOS...	DEPURAR LA BASE DE DATOS Y AUMENTAR LA CAPACIDAD DE ALMACENAJE DE LA MISMA.		NFLORES	HECTOR MÁRQUEZ
13/10/2010	40	DUPLICIDAD DE DATOS.	PEOPLESOFT	LA BASE DE DATOS...	APLICACIÓN DE LAS NORMAS DE NORMALIZACIÓN A LA BASE DE DATOS.	RESERVACIÓN DE ADIESTRAMIENTO, RESERVACIÓN DE VUELOS	NFLORES	CARLOS BLANCO
13/10/2010	41	NO SE PUEDE ACCEDER A LA APLICACIÓN.	EMERGENCY LOGBOOK	SE ELIMINARON ARC...	RESTAURAR LOS ARCHIVOS DE LA APLICACIÓN.		NFLORES	HERNELIS FLORES

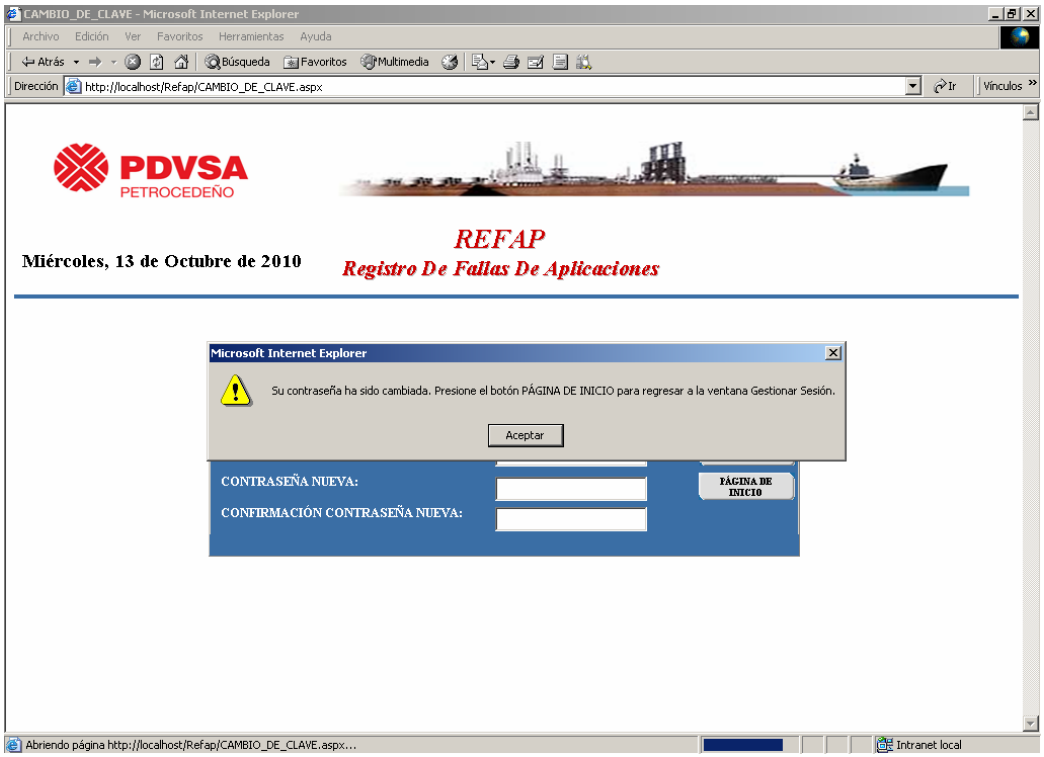
Presionar el link imprimir para imprimir el reporte.

6 CAMBIAR CONTRASEÑA

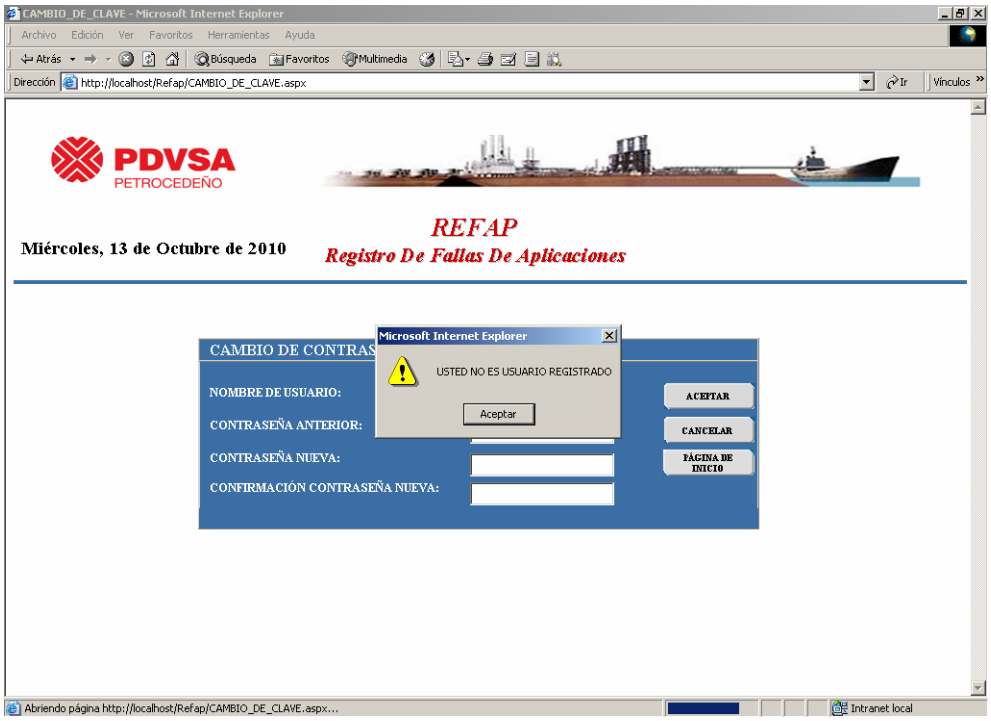
La ventana de Gestionar Sesión presenta la opción cambiar clave, mediante la cual los usuarios pueden cambiar sus contraseñas en el momento que lo consideren necesario.

Al presionar el botón de cambiar clave se observara la ventana de cambio de contraseña, en la cual se deben llenar los campos solicitados y presionar el botón Aceptar como se muestra a continuación:

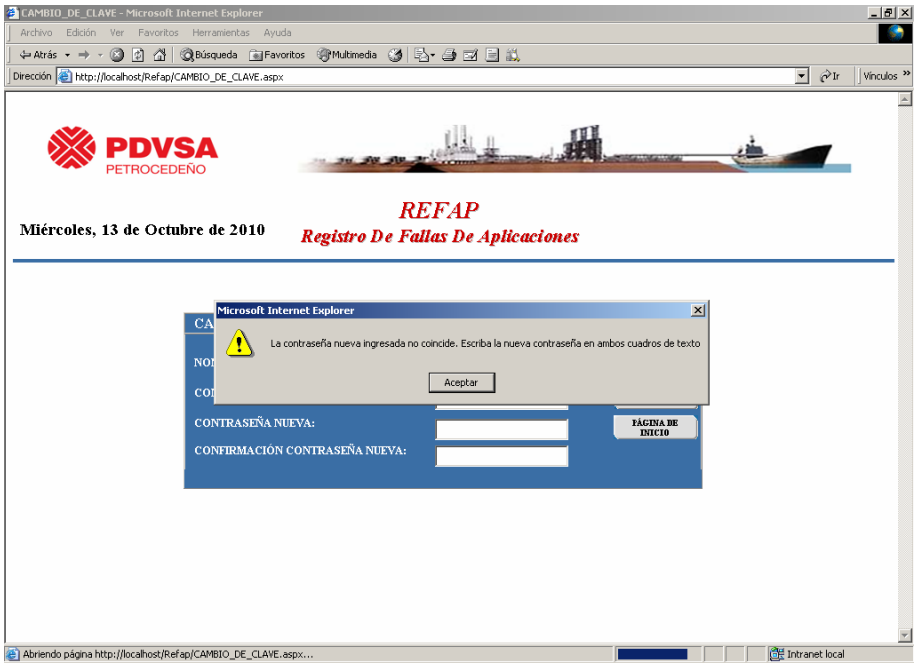
Si el usuario y la clave introducidos son correctos se visualizará el siguiente mensaje:



De lo contrario, se visualizará el siguiente mensaje:



Si la contraseña nueva y la confirmación de la contraseña nueva no coinciden, se visualizará el siguiente mensaje:



Para regresar a la ventana Gestionar Sesión, se debe presionar el botón PÁGINA DE INICIO.

Si presiona el botón CANCELAR, se limpiarán todos los cuadros de texto.

APÉNDICE B

MANUAL DEL ADMINISTRADOR

REGISTRO DE FALLAS DE APLICACIONES

(REFAP)

CONTENIDO

1 INTRODUCCIÓN

Refap es una aplicación que permite registrar información concerniente a las fallas que se generan en las diferentes aplicaciones existentes en el Mejorador PDVSA PETROCEDEÑO.

El objetivo de Refap es mantener de forma automatizada y organizada un respaldo de las soluciones que los analistas de la coordinación IS aplican para resolver las fallas de aplicaciones, brindándoles comodidad a la hora de buscar información sobre dichas fallas.

REFAP cuenta con una interfaz gráfica amigable y de fácil interactividad para los usuarios.

REFAP está constituido por 4 módulos que se identifican a continuación:

- Gestionar Administración.
- Gestionar Fallas. (Ver detalles en el manual de usuario).
- Realizar Consultas. (Ver detalles en el manual de usuario).
- Generar Reportes. (Ver detalles en el manual de usuario).

Este manual está dirigido únicamente al administrador del sistema, con la finalidad de mostrarle los pasos a seguir en el registro, modificación y eliminación de toda la información que permanece estática en el sistema; es decir, dando a conocer la forma de manejar las aplicaciones, criticidad de las fallas y usuarios, los cuales están contemplados en el módulo de Gestionar Administración. También se especifican los detalles sobre la base de datos refap.

2 BASE DE DATOS

La base de datos refap esta constituida por siete tablas, en la cual se almacena todos los datos que se ingresan a través del sistema Refap.

Las tablas están relacionadas por medio de las claves que posee cada una de ellas, como puede observarse en la figura 1.1 que se muestra a continuación:

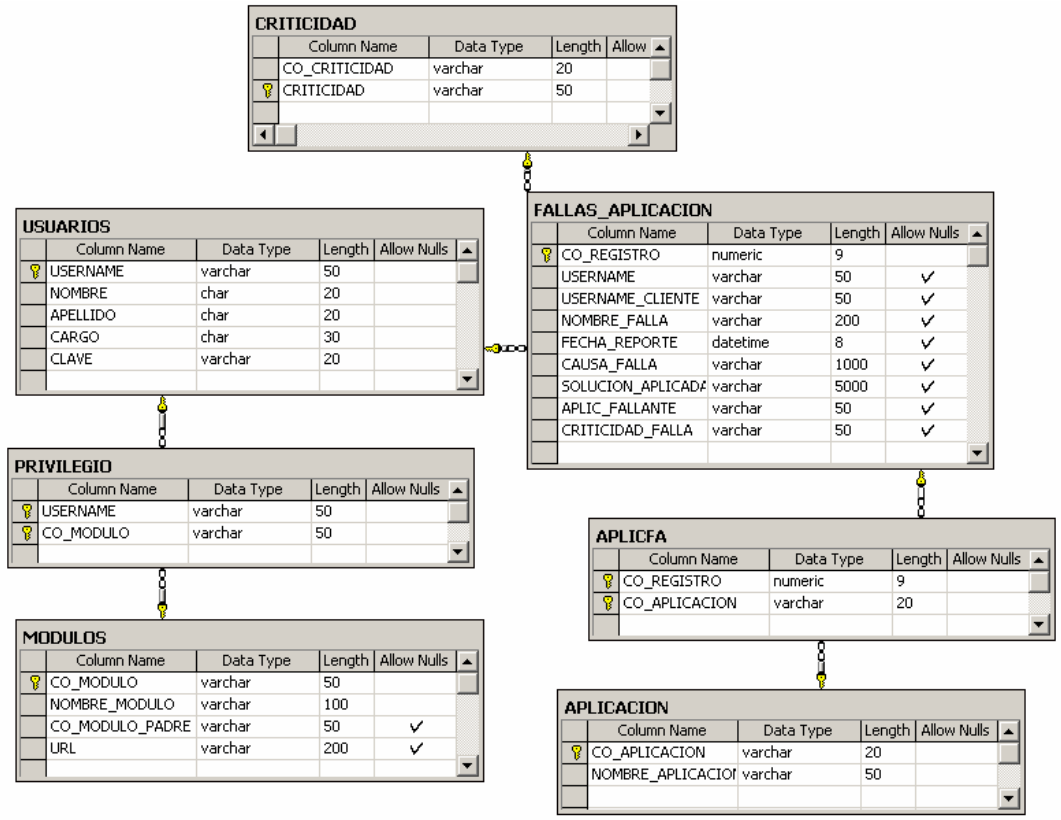


Figura 1.1, Base de datos del sistema REFAP

2.1 Descripción de las tablas:

FALLAS_APLICACION: es la tabla principal, donde se carga toda la información concerniente a las fallas que se generan en las diferentes aplicaciones existentes en el

Mejorador PDVSA PETROCEDEÑO. Contiene una clave principal (CO_REGISTRO), dicho campo es autonumérico y se genera cada vez que se registra una falla.

USUARIOS: almacena toda la información de los usuarios y su clave principal es el campo USERNAME.

PRIVILEGIO: almacena todos los códigos de los módulos al cual puede acceder un usuario; por lo tanto tiene dos claves el campo USERNAME y el campo CO_MODULO, los datos de dichos campos se obtienen a través de la tabla USUARIOS Y MODULOS respectivamente, como puede apreciarse en la figura 1.1 que muestra la relación existente entre estas tablas y la tabla PRIVILEGIO.

MODULOS: almacena todos los datos concernientes a los módulos de la aplicación. La clave principal de esta tabla es el campo CO_MODULO.

APLICACIÓN: almacena los datos de todas las aplicaciones existentes en el mejorador PDVSA PETROCEDEÑO. La clave principal de esta tabla es el campo CO_APLICACION.

APLICFA: almacena aquellas aplicaciones que son afectadas por una determinada falla, los dos campos que contempla son sus claves, el CO_REGISTRO y CO_APLICACION, dichos campos son llenados por medio de los datos obtenidos de las tablas FALLAS_APLICACION y APLICACION respectivamente, de ahí que se deriva la relación existente entre estas tablas y APLICFA.

CRITICIDAD: almacena los datos de la criticidad de la falla. Su campo clave es CRITICIDAD.

3 INICIO DE SESIÓN

Refap es una aplicación Web; por lo tanto; su acceso es a través de la dirección electrónica: http://sup-36/appsnet/Refap/INICIAR_SESION.aspx en la cual se visualizará la siguiente ventana:

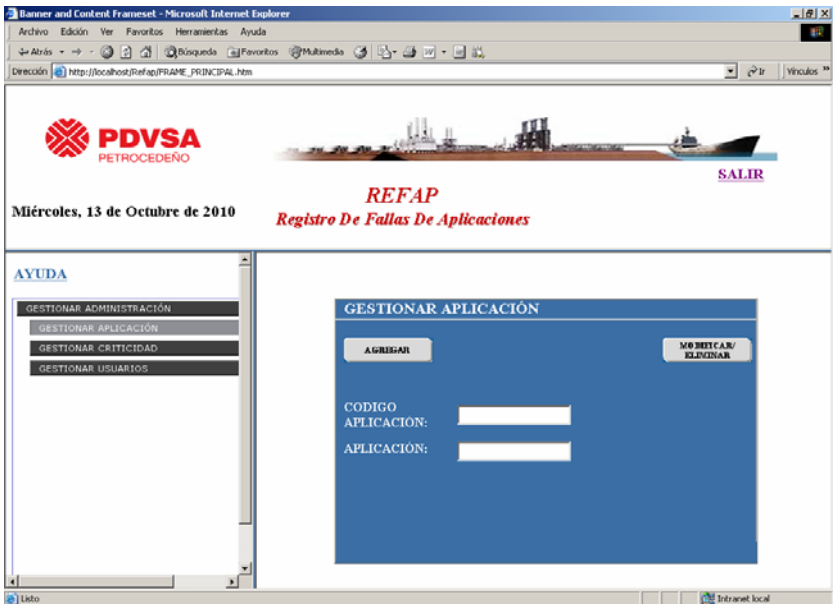
Una vez ingresado al sistema, se visualiza el menú que contiene el módulo Gestionar Administración, al cual puede acceder el Administrador.

4 GESTIONAR ADMINISTRACIÓN

El administrador debe seleccionar la opción GESTIONAR ADMINISTRACIÓN. Se desplegará el submenú que contempla dicho módulo, como se puede apreciar a continuación:

4.1 Gestionar Aplicación

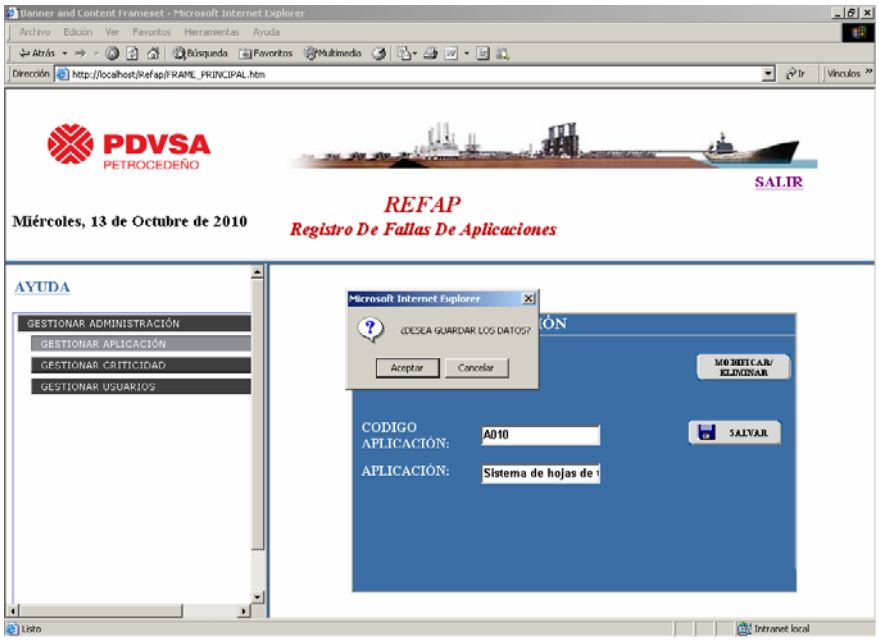
Al seleccionar la opción Gestionar Aplicación, se observará la ventana que contiene el formulario para cargar todas las aplicaciones existentes en el mejorador PDVSA PETROCEDEÑO.



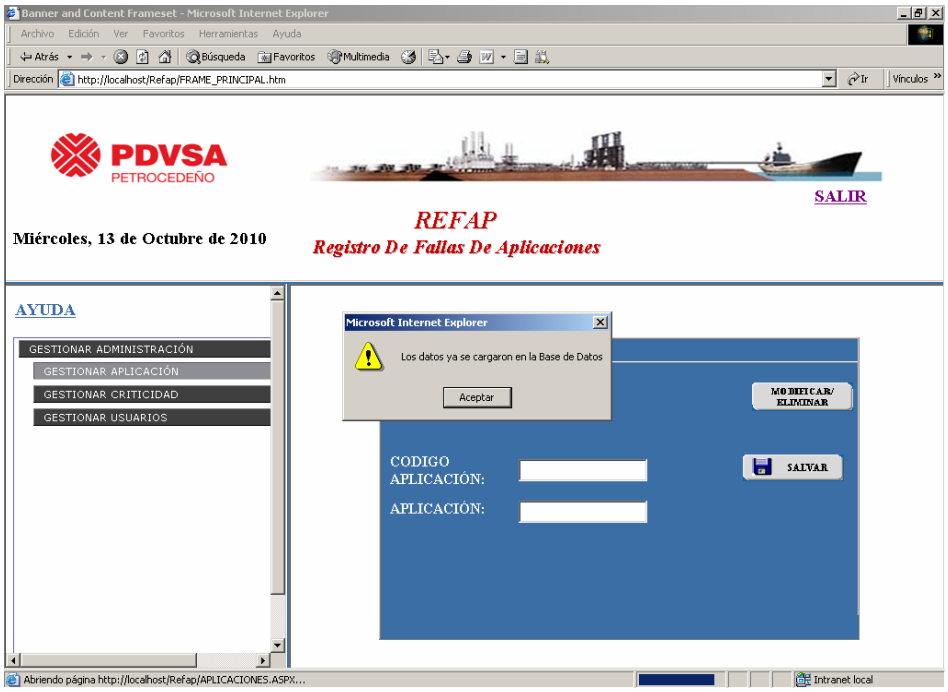
4.1.1 Gestionar Aplicación (Agregar)

Presionar sobre el botón AGREGAR para ingresar una aplicación. Se activaran los cuadros de textos y se visualizará el botón SALVAR, como se puede observar a continuación:

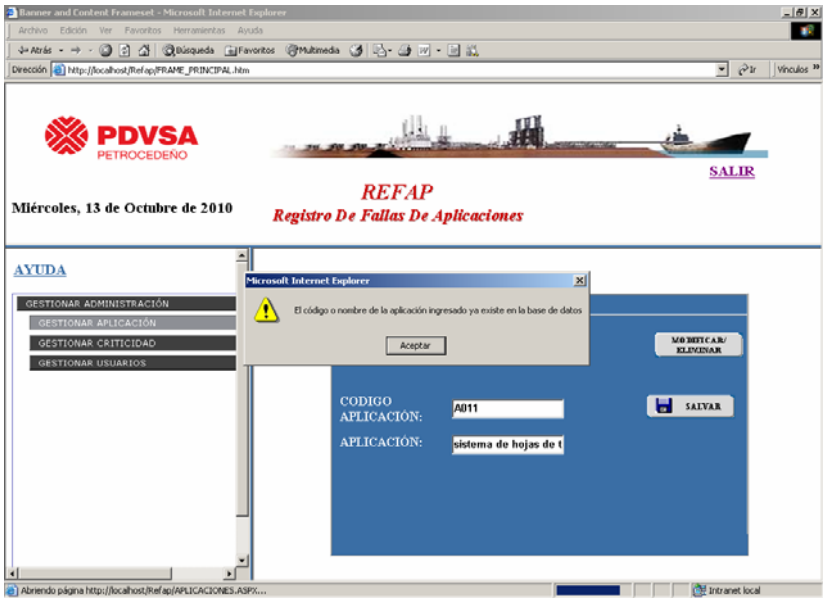
Ingresar el código y nombre de la aplicación, presionar el botón SALVAR. Aparecerá un mensaje de confirmación, tal como se muestra a continuación:



Presionar el botón Aceptar para guardar la información en la base de datos Refap, se visualizará el siguiente mensaje:

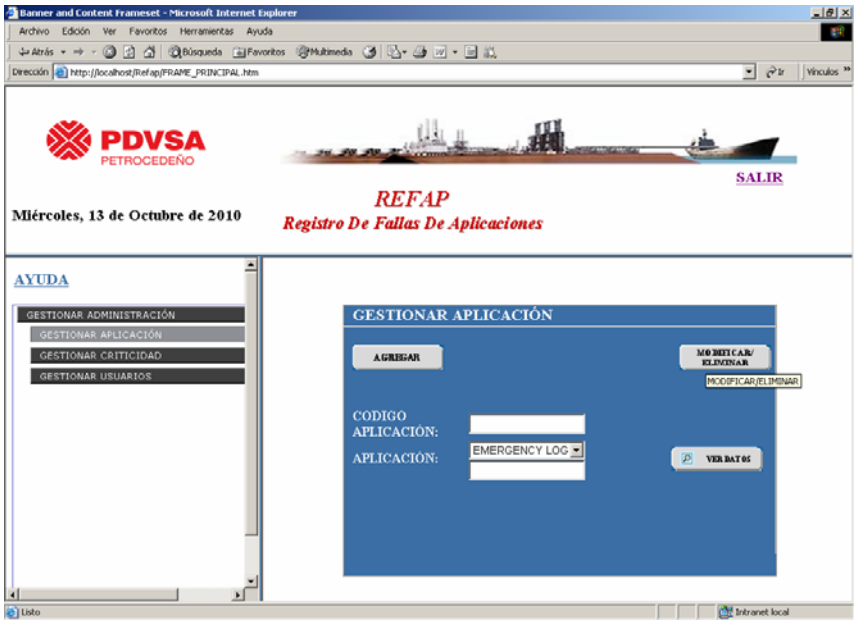


Si el código de la aplicación o el nombre ya han sido cargados anteriormente, aparecerá el siguiente mensaje.

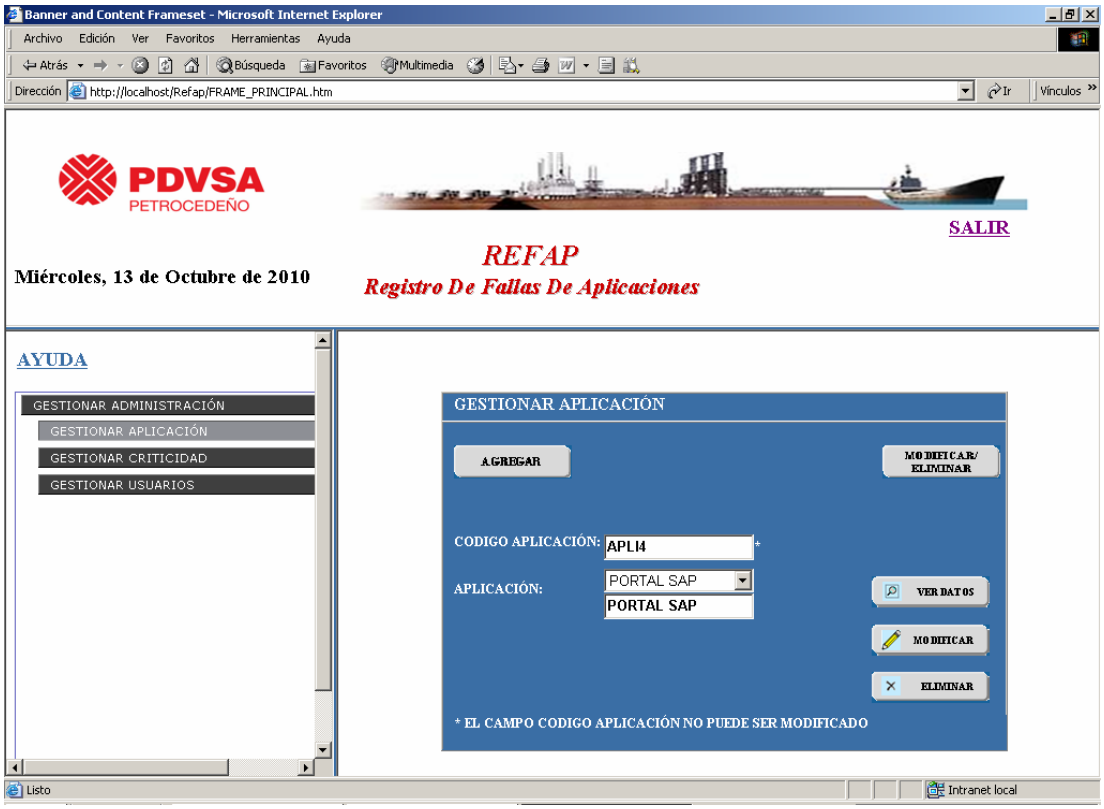


4.1.2 Gestionar Aplicación (Modificar/Eliminar)

Presionando esta opción se activará el botón de ver datos.

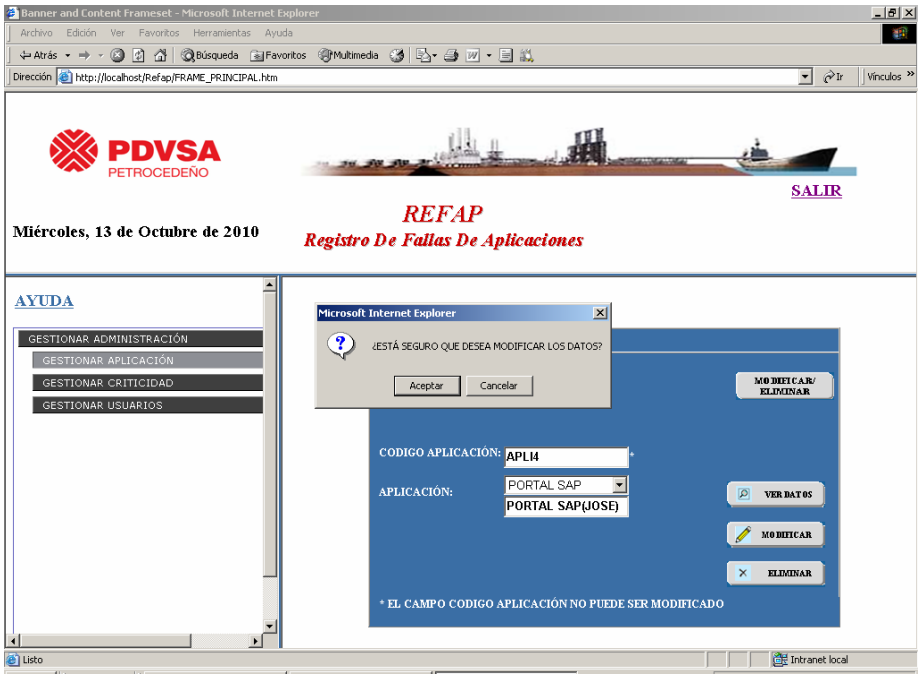


Seleccionar la aplicación que desee modificar o eliminar y presionar el botón VER DATOS. Se observará la siguiente ventana:

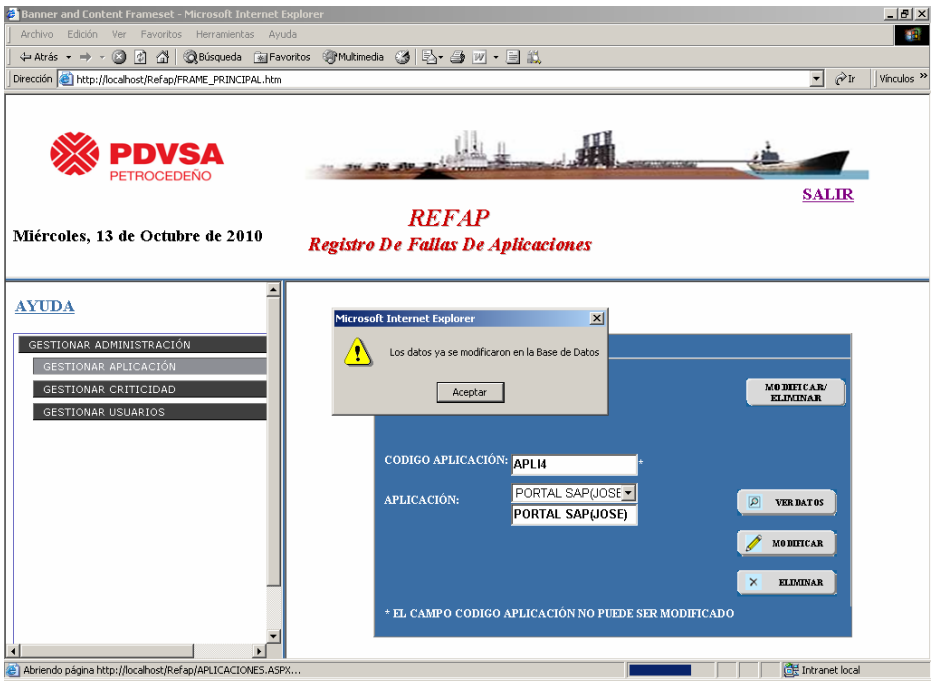


4.1.2.1 Modificar

Para Modificar, realicé los cambios en el cuadro de texto Aplicación y presione el botón MODIFICAR. El código de la aplicación no puede ser modificado tal como se muestra en la ventana anterior.

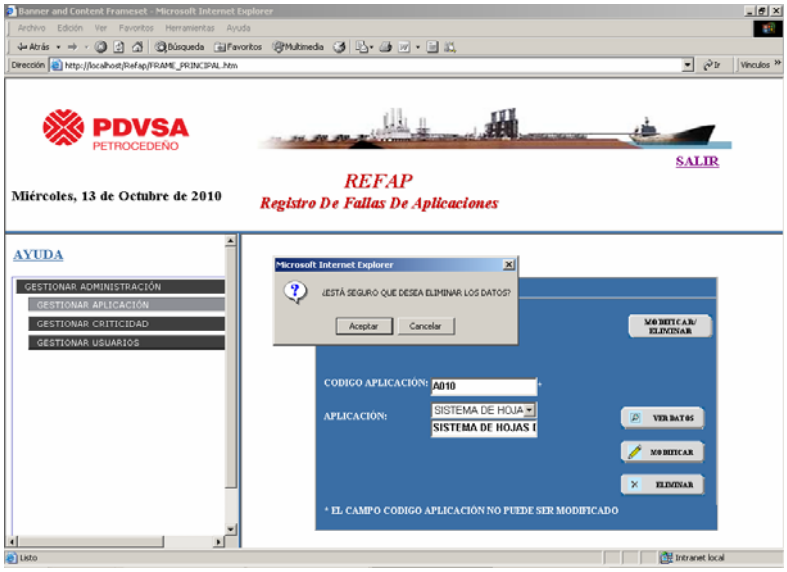


Presionar Aceptar para guardar los cambios en la base de datos. Aparecerá el siguiente mensaje:

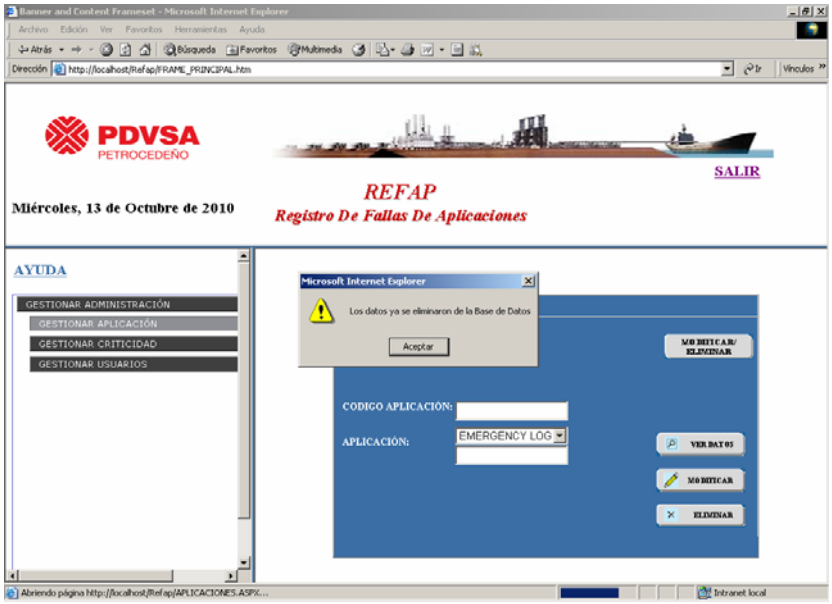


4.1.2.2 Eliminar

Para eliminar una aplicación, seleccione la aplicación que desee eliminar y presione el botón ELIMINAR.



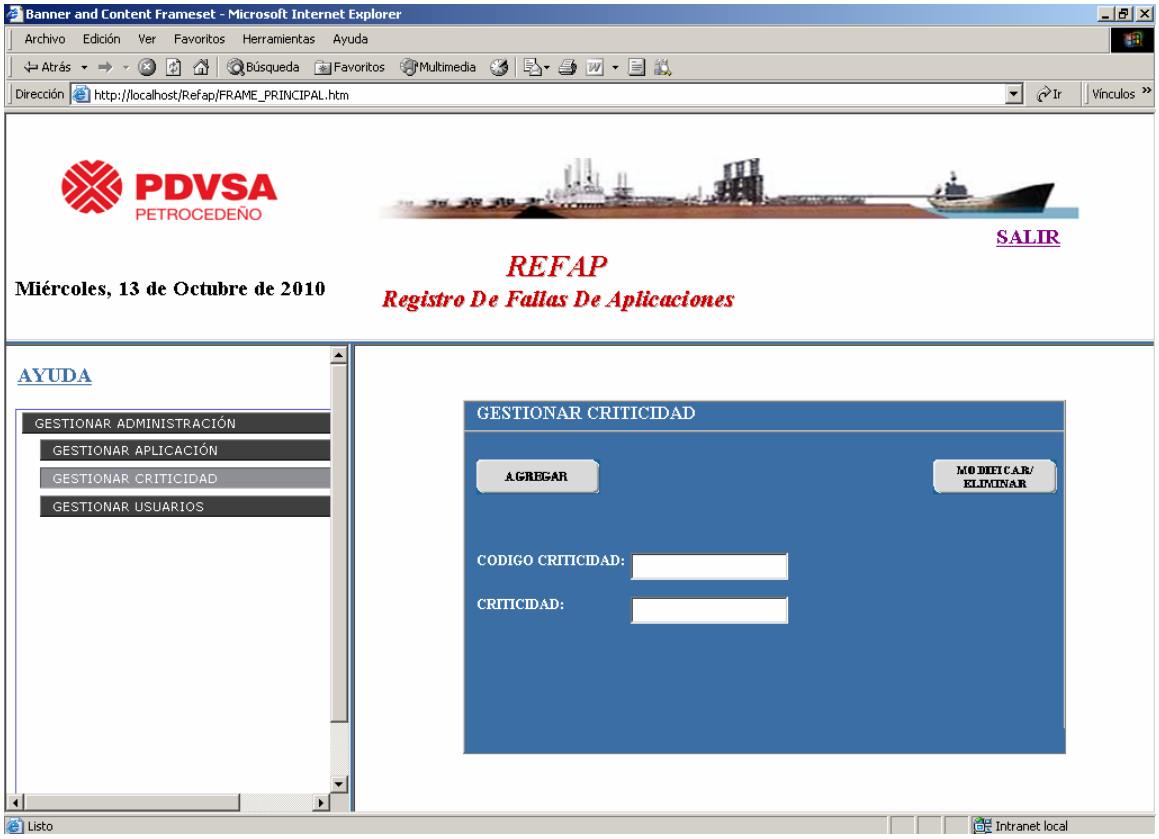
Presionar Aceptar, para eliminar la aplicación. Aparecerá el siguiente mensaje:



Presionar el botón cancelar, para no eliminar la aplicación.

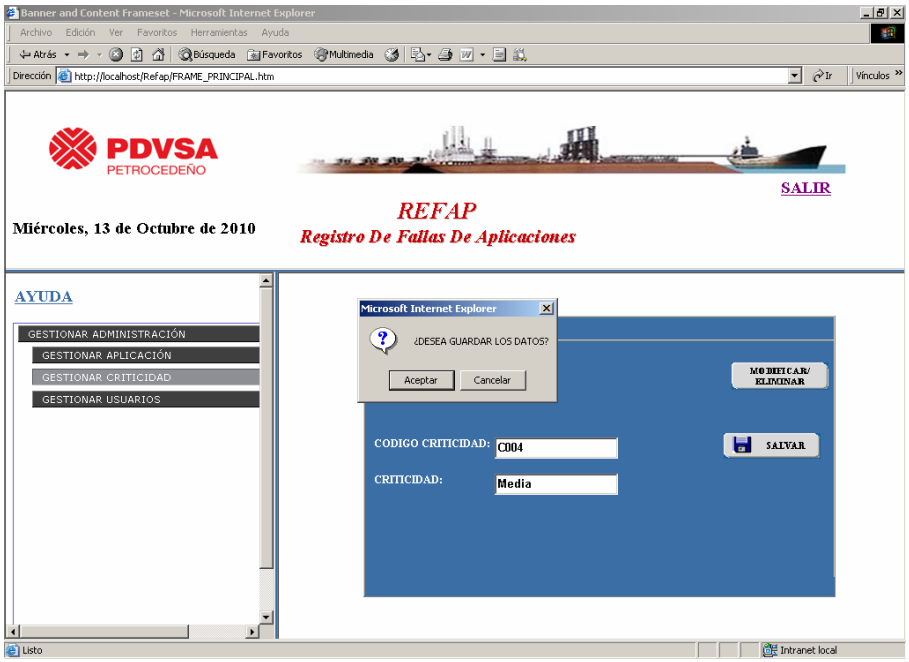
4.2 Gestionar Criticidad

Seleccionando esta opción se accederá a la ventana Gestionar Criticidad, por medio de la cual se cargaran los distintos tipos de criticidad que se presentan en las fallas generadas en las aplicaciones.

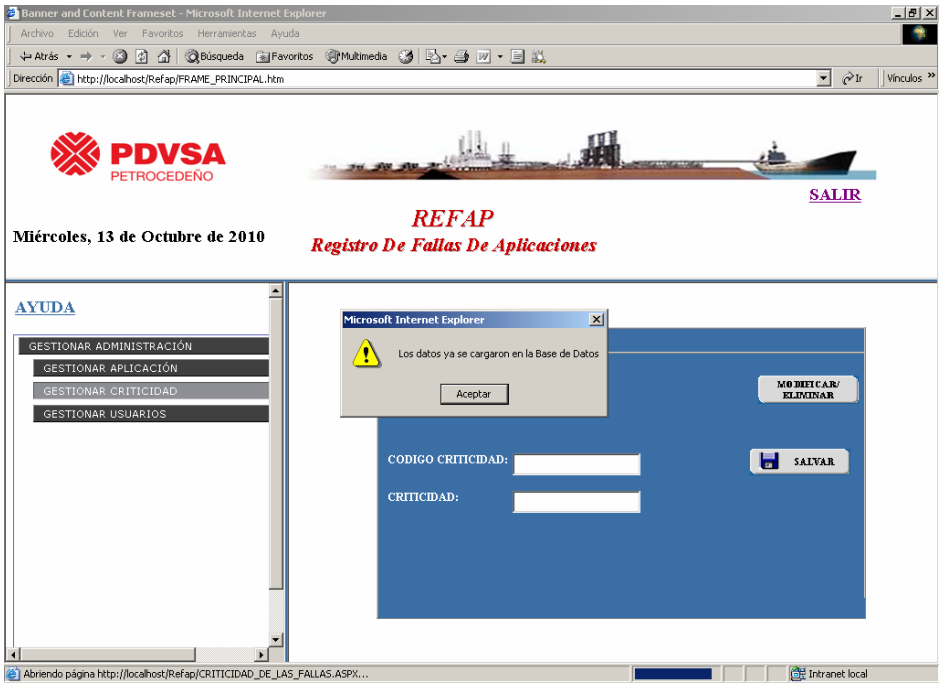


4.2.1 Gestionar Criticidad (Agregar)

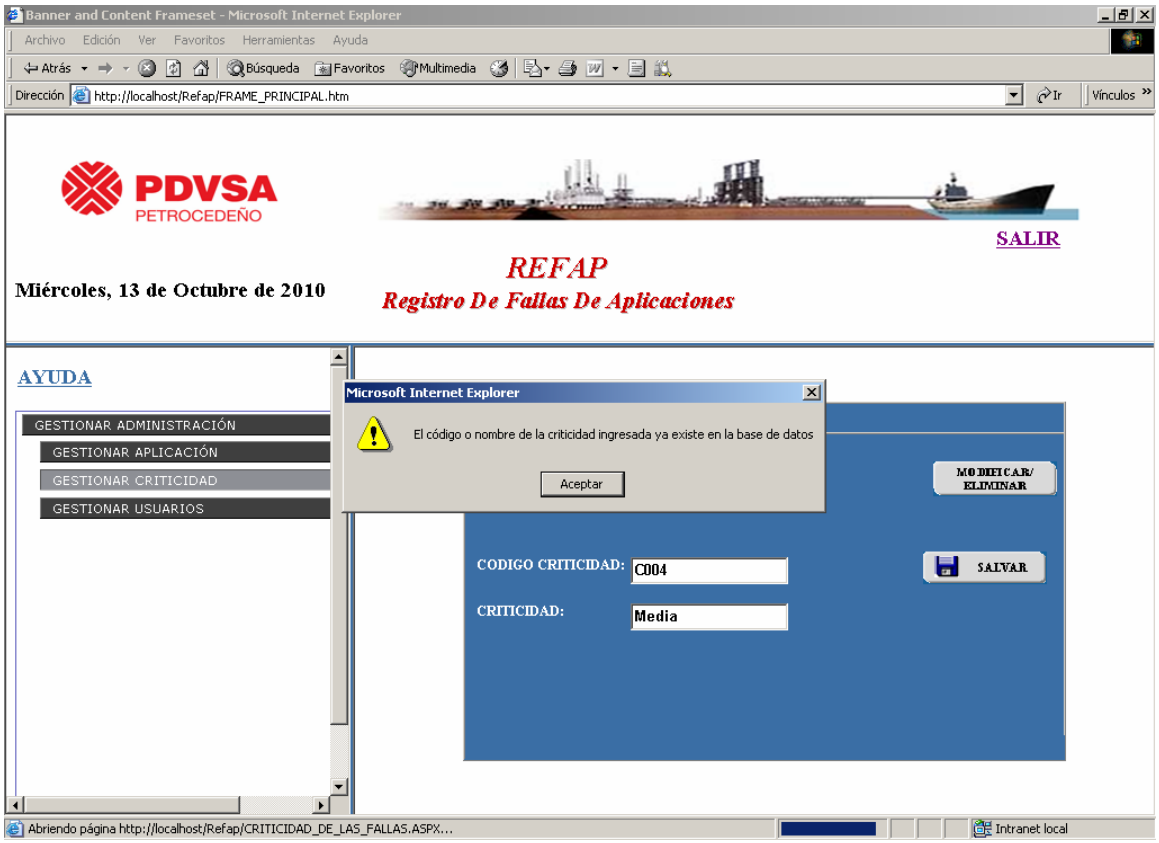
Elegir AGREGAR, se visualizará el botón SALVAR. Introducir los datos y presionar SALVAR para guardar el registro de la criticidad en la base de datos.



Presionar Aceptar, aparecerá el mensaje ratificando que los datos han sido cargados en la base de datos.



Si el código de la criticidad o la criticidad ingresada ya están cargados en la base de datos, se visualizará el siguiente mensaje:

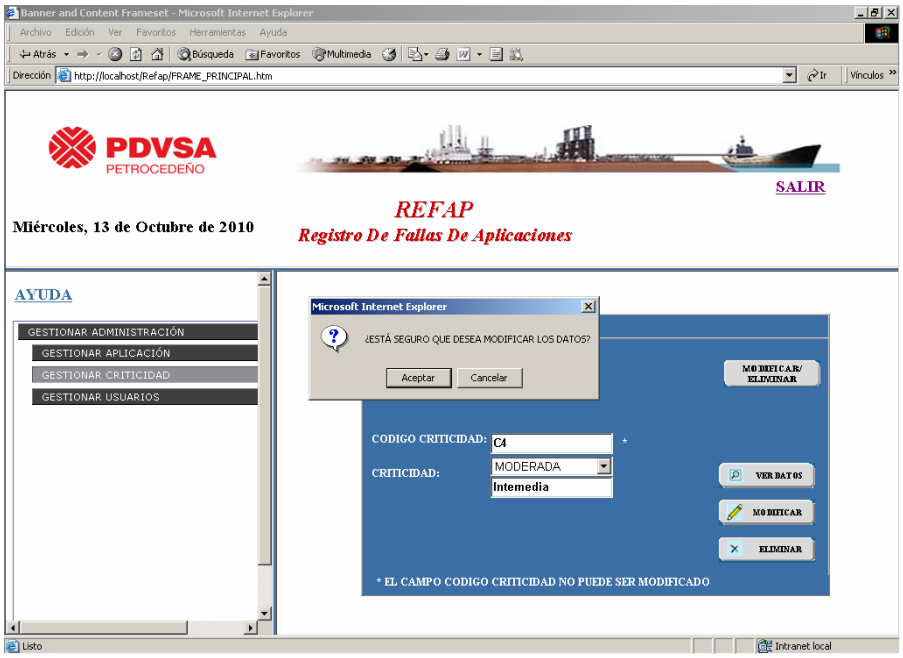


4.2.2 Gestionar Criticidad (Modificar/Eliminar)

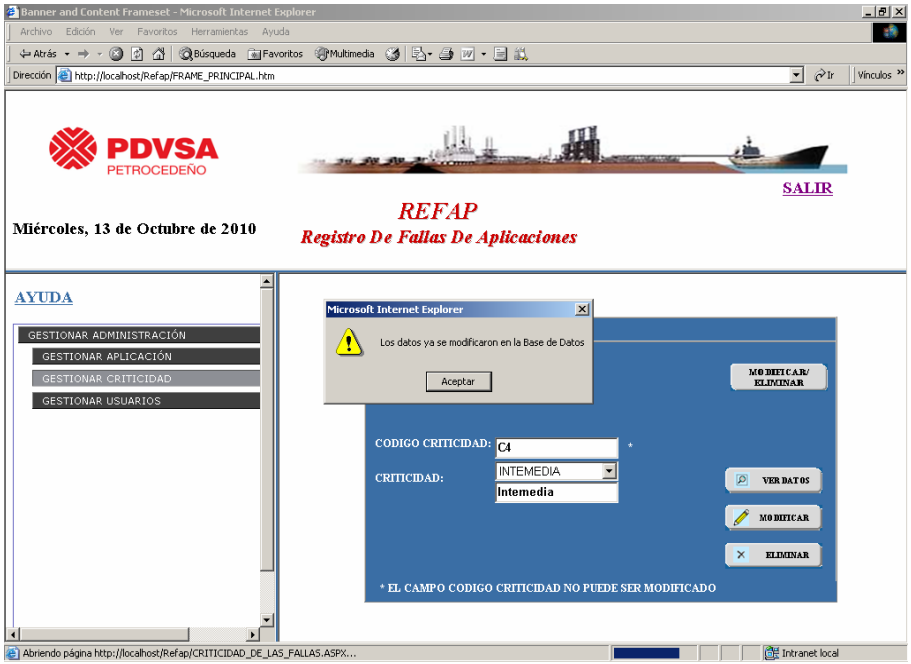
Al seleccionar esta opción se activara el botón ver datos.

4.2.2.1 Modificar

Para Modificar, realicé los cambios en el cuadro de texto Criticidad y presione el botón MODIFICAR. Se observara la siguiente ventana:



Presionar el botón Aceptar para modificar los datos. Aparecerá la siguiente ventana:

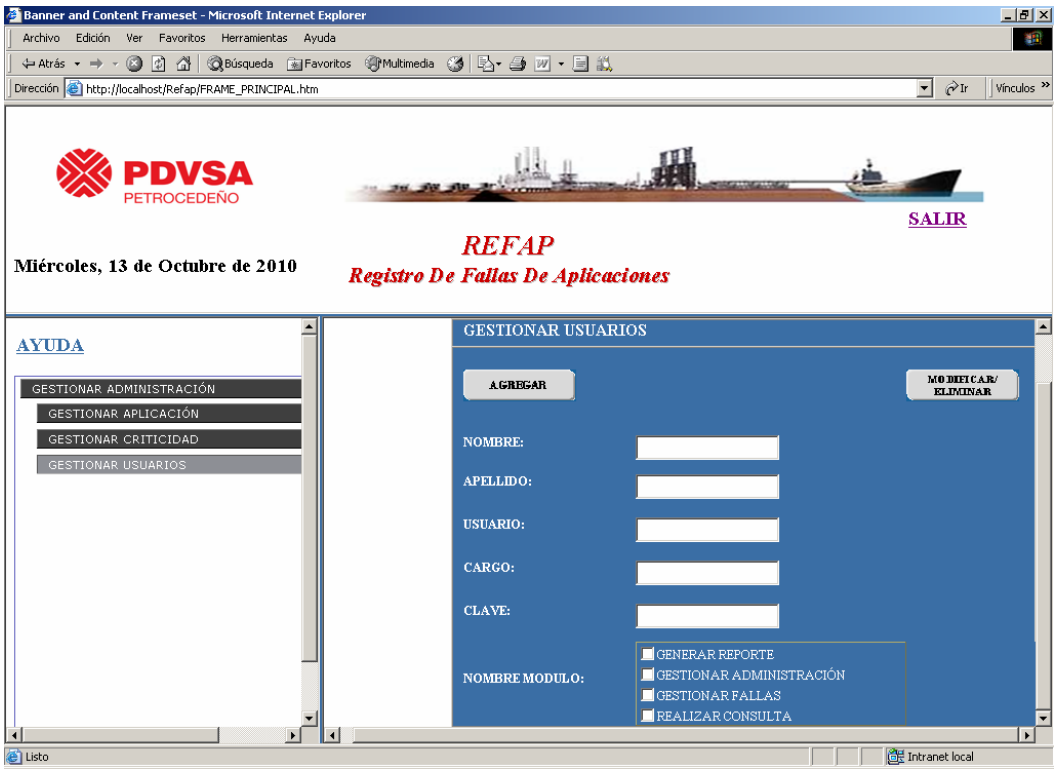


4.2.2.2 Eliminar

Para eliminar una Criticidad, selecciónela y presione el botón ELIMINAR. Confirmar que desea eliminar dicha Criticidad y automáticamente el sistema le indicará que los datos han sido eliminados.

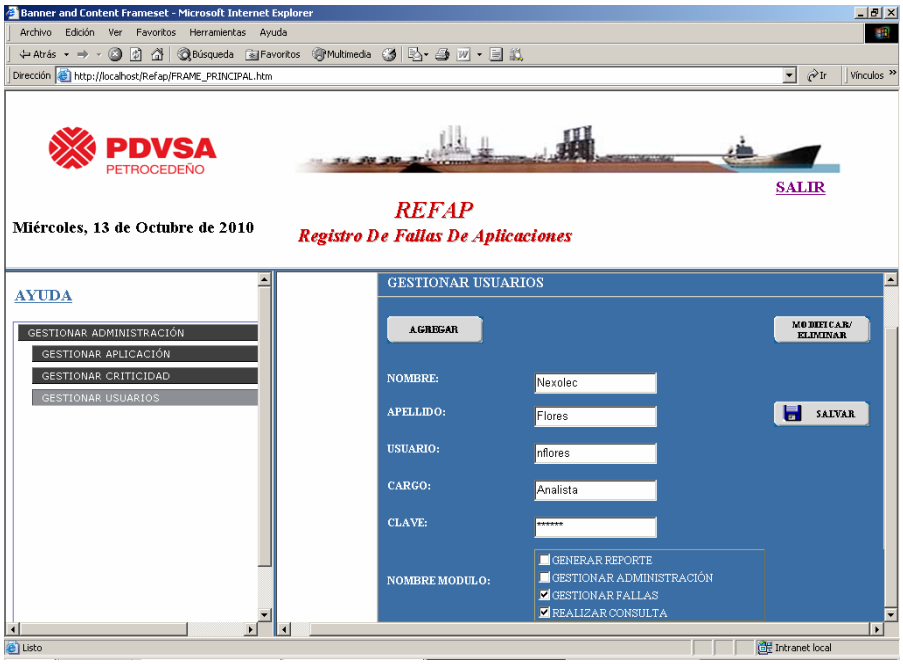
4.3 Gestionar Usuarios

Seleccionando esta opción se accede a la ventana Gestionar Usuarios, en la cual se puede registrar un nuevo usuario, modificar o eliminar un usuario ya existente.

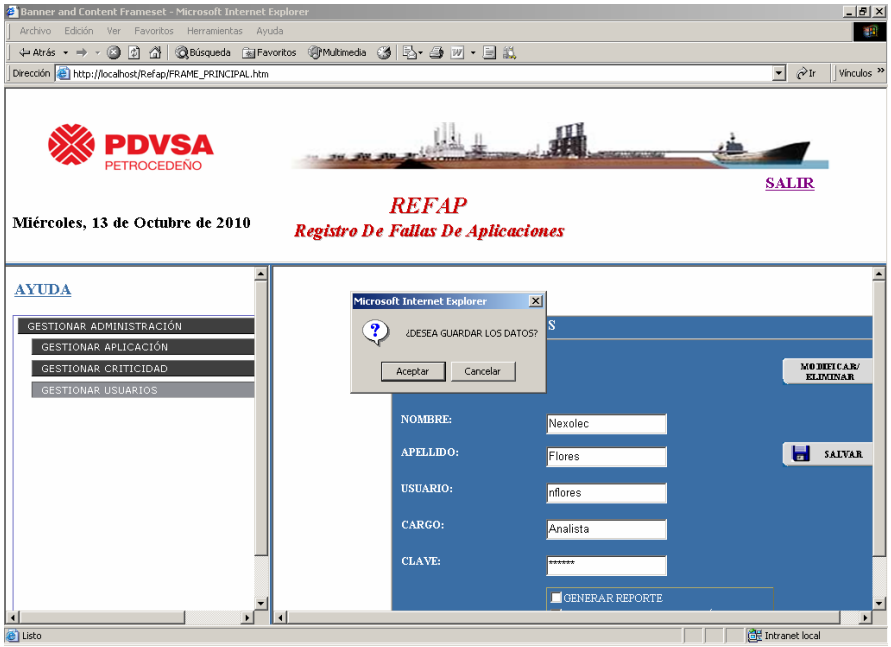


4.3.1 Gestionar Usuarios (Agregar)

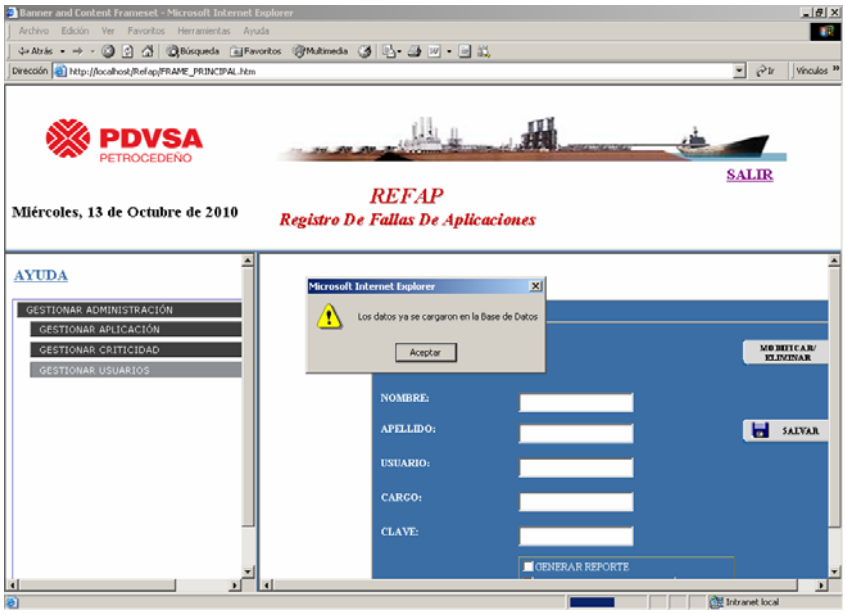
Presionando el botón AGREGAR se visualiza el botón de SALVAR y se activan los cuadros de texto, se deben llenar los campos Nombre, Apellido, Usuario, Cargo, Clave y seleccionar los módulos al cual puede acceder el usuario a ingresar.



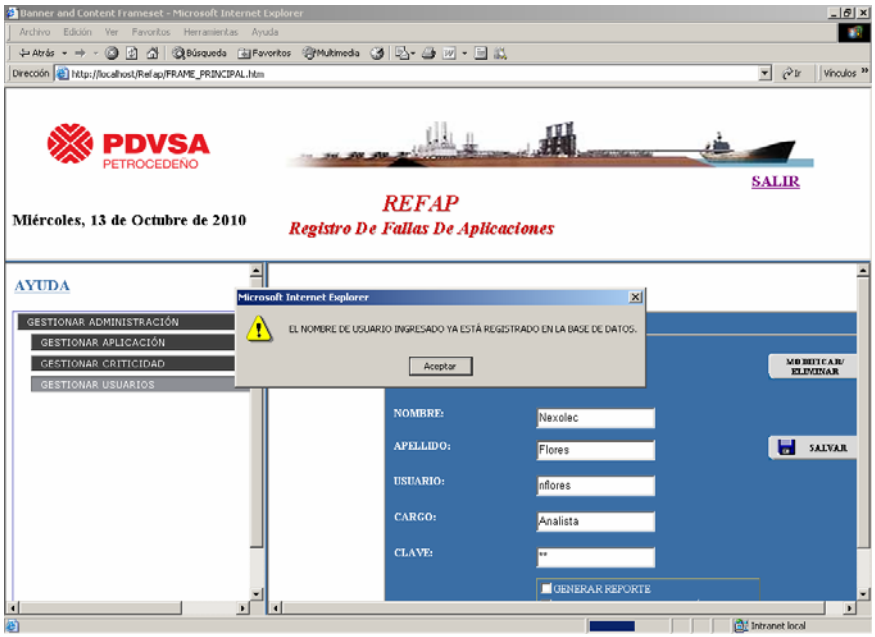
Una vez llenado todos los campos, presionar el botón SALVAR, aparecerá el siguiente mensaje:



Presionar Aceptar, para guardar los datos.

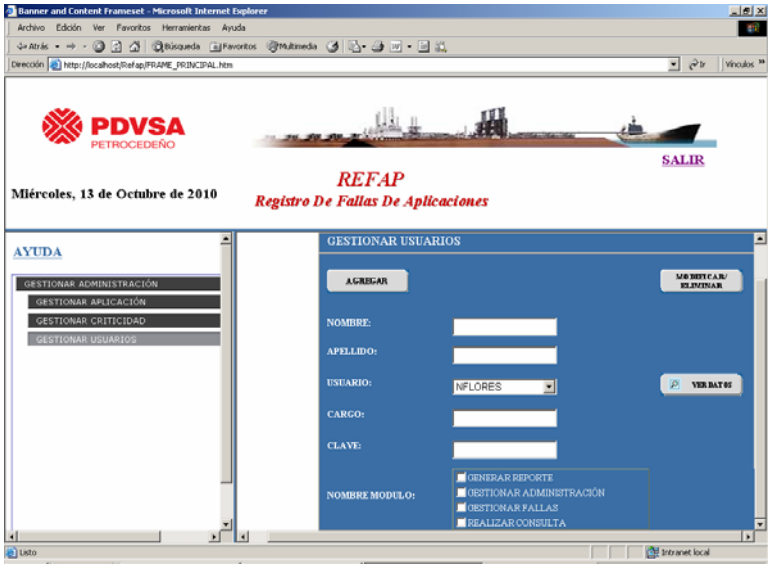


Si el usuario a ingresar ya ha sido agregado anteriormente, se visualizará el siguiente mensaje:



4.3.2 Gestionar Usuarios (Modificar/Eliminar)

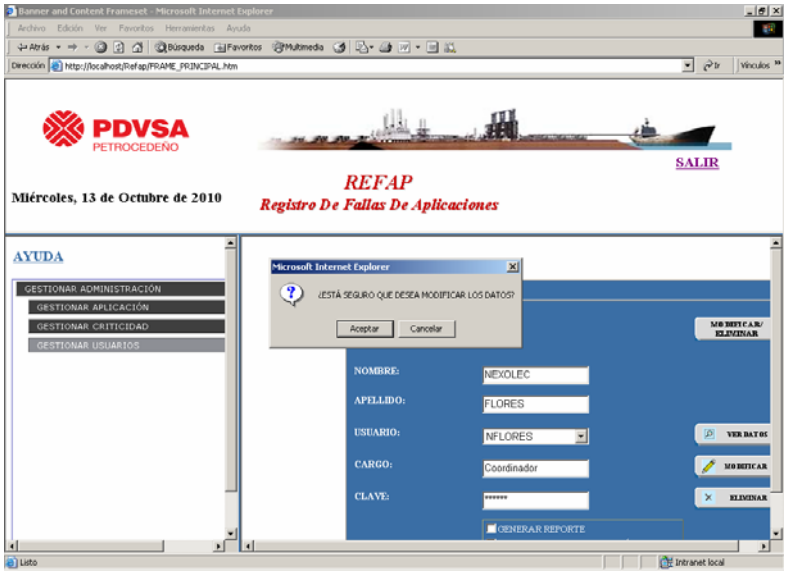
Presionando el botón Modificar/Eliminar, se visualiza el botón VER DATOS y la lista con todos los usuarios que han sido ingresado a la base de datos REFAP, como se muestra a continuación:



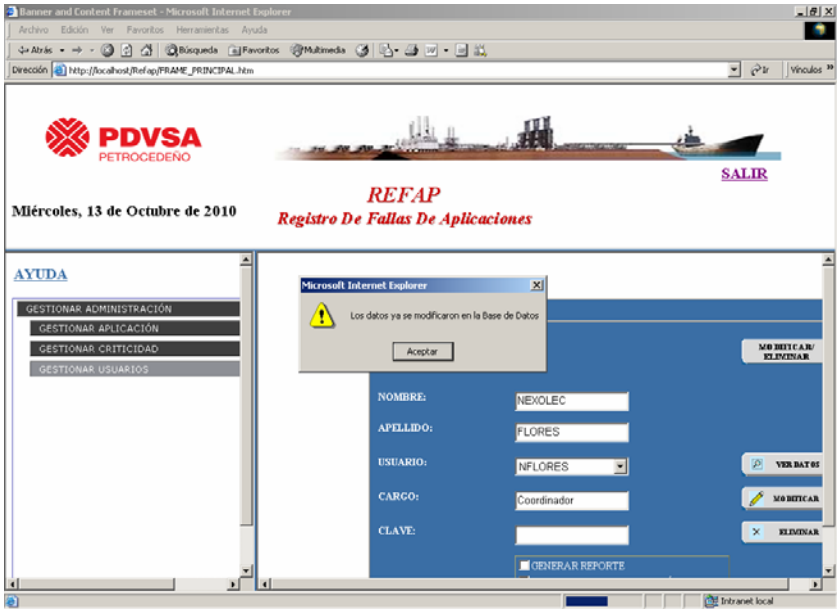
Seleccionar el usuario y presionar el botón VER DATOS.

4.3.2.1 Modificar

Se deben realizar los cambios que desee hacerle al usuario y presionar el botón MODIFICAR.

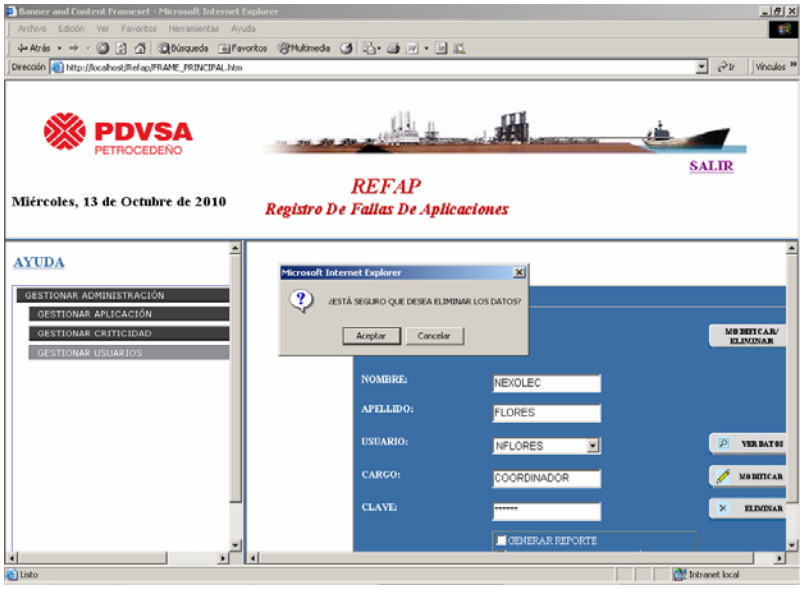


Presionar Aceptar, para modificar los datos.

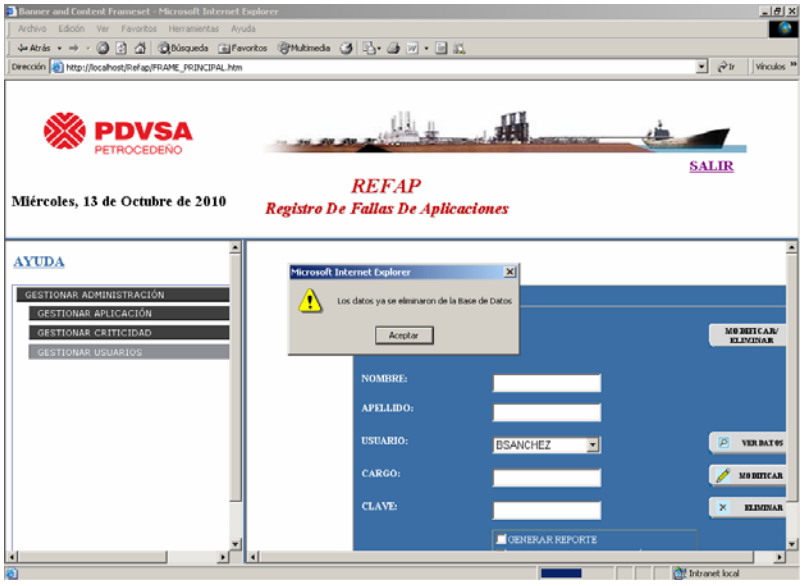


4.3.2.2 Eliminar

Para eliminar un usuario se debe seleccionar de la lista el usuario y presionar el botón ELIMINAR. Aparecerá un mensaje de confirmación como se muestra a continuación:



Presionar el botón Aceptar, para eliminar el usuario.



METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	**"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".**
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
Flores S., Nexolec del V.	**CVLAC:** 15.114.201 **E MAIL:** nexfs@hotmail.com
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

Desarrollo de Software.

Aplicación Web.

Diseño de Base de Datos (BDD).

Lenguaje de Modelado Web (WebML).

Lenguaje de Modelado Unificado (UML).

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÀREA
Ingeniería y Ciencias Aplicadas	**Ingeniería en Computación**

RESUMEN (ABSTRACT):

La Coordinación IS de PDVSA PETROCEDEÑO, se encarga de solucionar las necesidades y/o fallas que se generan en las Aplicaciones existentes en el Mejorador. Para resolver las fallas los Analistas aplican sus conocimientos y conservan un soporte de la implementación que realizaron, y así emplearlo cuando vuelva a ocurrir la misma falla. Los soportes se guardan en físico o en el correo electrónico, lo que resulta incómodo a la hora de buscarlos. Con la finalidad de contar con una herramienta de trabajo en la cual se pueda documentar las fallas generadas en las aplicaciones, la Coordinación IS, plantea la necesidad de una aplicación que cubra sus requerimientos, por lo cual nace la siguiente propuesta: **"DESARROLLO DE UN SOFTWARE, PARA EL REGISTRO DE LAS FALLAS QUE SE GENERAN EN LAS DIFERENTES APLICACIONES EXISTENTES EN EL MEJORADOR DE PDVSA PETROCEDEÑO, QUE SON REPORTADAS A LA COORDINACIÓN IS".** Éste Software brindará comodidad, efectividad y seguridad al buscar información, además estadísticas sobre las fallas. Para diseñar el Software se utilizó el Proceso Unificado, UML y Webml. Se implementó SQL SERVER 2000 como sistema manejador de Base de Datos, para la codificación se usó Visual Basic .Net 2003, XML, XSLT.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Dorta., Pedro	**ROL**	**CA**	**AS**	**TU** X	**JU**
	CVLAC:	12.914.617			
	E_MAIL	dortap@hotmail.com			
	E_MAIL				
Bastardo., José	**ROL**	**CA**	**AS**	**TU**	**JU** X
	CVLAC:	6.890.832			
	E_MAIL	josebastardo@gmail.com			
	E_MAIL				
Carrasquero., Manuel	**ROL**	**CA**	**AS**	**TU**	**JU** X
	CVLAC:	7.374.987			
	E_MAIL	manuelscm@hotmail.com			
	E_MAIL				
	ROL	**CA**	**AS**	**TU**	**JU**
	CVLAC:				
	E_MAIL				
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2010	08	13
AÑO	**MES**	**DÍA**

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
TESIS.SistemaDeInformaciónWeb_REFAP.doc	Application/msword

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: 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. 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. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____ (**OPCIONAL**)

TEMPORAL: _____ (**OPCIONAL**)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

Ingeniero en Computación

NIVEL ASOCIADO CON EL TRABAJO:

Pre-Grado

ÁREA DE ESTUDIO:

Departamento de Computación y Sistemas

INSTITUCIÓN:

Universidad de Oriente – Núcleo Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

De acuerdo al artículo 41 del Reglamento de Trabajos de Grado:

"Los Trabajos de Grado son exclusiva propiedad de la Universidad de Oriente y sólo podrán ser utilizadas para otros fines con el consentimiento del Consejo de Núcleo respectivo, quien lo participará al Consejo Universitario"

Nexolec del Valle Flores Sánchez

AUTOR

Ing. Pedro Dorta	Ing. José Bastardo	Ing. Manuel Carrasquero
TUTOR	**JURADO**	**JURADO**

Ing. José Luis Bastardo

POR LA SUBCOMISION DE TESIS