Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: *Sistema para la gestión de reglas de negocio para el framework .NET 4.0 y la tecnología WWF 3.5*

Autores: *Isunnis Barreras Vento*

Dainer Mesa Lima

Tutor:

Ing. Reynier Blanco Zambrano.

Ciudad de La Habana, Cuba

Curso 2011-2012

"Si una persona es perseverante, aunque sea dura de entendimiento, se hará inteligente; y aunque sea débil se transformará en fuerte."

Leonardo Da Vinci

Declaración de autoría

Declaramos que somos los únicos autores del presente trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de ____ del año ____.

_____	_____
Isunnis Barreras Vento	Dainer Mesa Lima
Autor	Autor

_____

Ing. Reynier Blanco Zambrano

Tutor

Datos de contacto

Tutor: Ing. Reynier Blanco Zambrano

Email: rblanco@uci.cu

Ingeniero en Ciencias Informáticas, UCI 2009, Título de Oro.

Arquitecto del proyecto Sistema Único de Identificación Nacional perteneciente al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas.

Dedicatoria

Dedico este trabajo a toda mi familia y a todos los que de una forma u otra me apoyaron para que este sueño se hiciera realidad, pero en especial a mi bisabuela Hidefelia que hoy no está con nosotros pero sé que desde el cielo me está viendo y se siente muy orgullosa de mí.

Isunnis Barrera Vento

Dedico mi tesis a mis padres motor impulsor de toda mi vida, las personas más importantes y que gracias a ellos soy la persona que soy. A mi familia que me ha brindado todo su apoyo y amor. A todos los que de una forma u otra ayudaron para que este sueño se hiciera realidad. A la Revolución Cubana por darme la oportunidad de estudiar y a nuestro Comandante en Jefe por la oportunidad de haberlo hecho en esta universidad.

Dainer Mesa Lima

Agradecimientos

Mis más grandes agradecimientos a toda la familia en general por ser ese motor impulsor que siempre me ha guiado por el camino correcto y apoyarme siempre cuando más lo he necesitado. Le doy gracias a la vida por ser tan afortunada y tener más de una madre, primeramente esa personita que ha sido para mí madre, padre, amiga, mi querida abuela Neni, a ti Noemí mis más grandes agradecimientos por ser una madre para mí y no puedo dejar de mencionar a quien me trajo al mundo que sin ella no estaría aquí en estos momentos. A mi papá que aunque hoy no puede estar físicamente aquí sabe que siempre está conmigo, a mi hermano, a todos mis tíos en especial a mi segundo papá, bárbaro, a todos mis primos Dayelin, Disneydis, Yordany, Yoandy y Danaysi.

A mi novio por todo ese apoyo y comprensión que me ha dado aun estando lejos de mí, por ser cada día más paciente, por aguantar mis malcriadeces día a día, por darme esa fuerza que en muchas ocasiones he necesitado, gracias por todo tu amor te kiero. A mi segunda familia por toda su ayuda y paciencia en especial a Rafaela.

A todos los que han compartido conmigo en estos 5 años que nunca olvidaré, especialmente a aquellos que me han soportado desde primer año, Eliatne, Greybi, Laura, Jorge, Damaris, Mariena, Yelaysi y a los que compartí con ellos en años posteriores, a Geni por ser esa persona tan madura que me ha ayudado muchísimo, Wendy, Gisel, Deyanira, Nairelis, las melli, el chino, Roberto, María, Madiela, Lianny en fin a todos porque sería injusto que se me olvidara alguno.

A mi compañero está de más darte las gracias por haberme apoyado y soportado todos estos meses. Al tutor por habernos apoyado cuando lo necesitamos. En fin a todos los profesores del laboratorio Soluciones Integrales que de una forma u otra nos ayudaron a realizar este sueño.

Isunnis Barrera Vento

Agradecimientos

Les agradezco a mis padres por todo el apoyo compresión y amor que me han dado en todo este tiempo así como por las cosas buenas que me han enseñado durante toda mi vida, por ser mis guías en momentos difíciles. A mis hermanos por el apoyo que me han brindado en todo este tiempo. A los más pequeños de la familia, a mis sobrinitos, por enseñarme a tener un poco más de paciencia y por todo su amor. A mis tías que siempre se preocupan por mí y me han ayudado en todo lo que han podido. A Miriam que es como mi segunda madre, ya que me cuido desde los dos meses de nacido, siempre voy a estar en deuda contigo. A Cari que es como mi madrina aunque no estoy bautizado.

A la persona que desde hace un año y algo más de 4 meses lleva conmigo apoyándome en todo. A ti mi vida gracias por todo tu amor, comprensión y paciencia. Gracias por estar ahí siempre para mí en las buenas y en las malas, te amo una pila.

A todos mis amistades de estos cinco años. Los que han estado desde primero a Yendry, Julio, Ivan, Jose Carlos, a todo el hace 5 años 1104. A los que vinieron después Adonis, Valladares, Sandy, Yunier, Yosvani, Tony, a todos los del grupo 1503. A una amiga muy especial que conocí y que estoy en deuda con ella a Dayana muchas gracias por haberme brindado tu amistad. A los profesores que me formaron e hicieron que hoy estuviera graduándome, para ellos muchas gracias. A los profe que no me dieron clase pero que me apoyaron en mi vida como estudiante y personal muchas gracias para ti Geidis, Damián, Eilen, Matilde, Mayelin, Joel en fin a todos muchas, pero muchas gracias. A todas las personas que conozco porque de todos aprendí algo bueno o malo pero aprendí.

A mi compañera de tesis por todo este tiempo que me ha apoyado y ayudado para ti mil y un gracias. Al tutor que desde el 4to año que nos dieron el tema de tesis está apoyándonos en todo. A los profesores y trabajadores del proyecto que han apoyado el desarrollo de este trabajo durante todo el curso. Para todos ustedes también muchas, muchas gracias.

Dainer Mesa Lima

Resumen

El Sistema Único de Identificación Nacional actualmente no cuenta con un sistema para la gestión de reglas de negocio. Razón por la cual los desarrolladores se encuentran declarando las reglas en el código fuente de la aplicación y para realizar un cambio en una de estas reglas tienen que reescribir el código y recompilar toda la aplicación para actualizar los cambios realizados.

El presente trabajo describe el desarrollo de un sistema para automatizar la gestión de las reglas de negocio para el Sistema Único de Identificación Nacional. Posibilitando que un cambio realizado a una o varias reglas solo afecte a esta y no a la aplicación.

El sistema se implementó para aplicaciones desarrolladas sobre el *framework*.NET 4.0 y la tecnología WWF 3.5, utilizando la herramienta de desarrollo *Visual Studio Team System* 2010 y el lenguaje de desarrollo C Sharp. El despliegue de la aplicación beneficiará a la institución en el tiempo de adaptación en el funcionamiento y la usabilidad del sistema.

Palabras claves:

Framework .NET 4.0, Reglas de negocio, Tecnología WWF 3.5

Índice

Índice

Índice de tablas

Índice de figuras

Introducción

En la actualidad es común que ocurran rápidos y constantes cambios en los ambientes de negocios. Estos cambios afectan tanto al negocio como a las aplicaciones que le dan soporte. Como resultado, las aplicaciones requieren renovaciones dinámicas y adaptaciones para cumplir con las necesidades reales del mismo. Una tendencia que aborda dicha problemática es la conocida por "enfoque de reglas de negocio"(1). La idea principal de este enfoque consiste en una técnica de dominio del negocio basada en reglas explícitas que son escritas y expresadas en lenguaje claro, cercano al natural y separadas de los datos de la aplicación. Dichas reglas son motivadas por factores importantes e identificables que se desea que guíen o influyan en el comportamiento del negocio, y deberán ser de fácil acceso, modificación y administración por parte de los desarrolladores así como de los usuarios del negocio(2).

Las reglas de negocio constituyen un pilar fundamental en el desarrollo de aplicaciones más flexibles según la comunidad de los Sistemas de Información (SI)(3). Una regla de negocio es "una sentencia que define o restringe algún aspecto del negocio, el cual se pretende imponga la estructura o controle o influya en el comportamiento del mismo"(4), según el BRG[1], evitando que se realicen acciones no válidas.

Tradicionalmente, las reglas de negocio fueron incluidas directamente en el código por el programador, en los objetos de base de datos de la aplicación, o en ambos(3). Como consecuencia, ante algún cambio de una o varias reglas, era necesario cambiar el código del programa para poder actualizar estos cambios, teniendo que recompilar toda la aplicación, lo cual era muy engorroso ya que se dependía del equipo de desarrollo.

Las reglas de negocio se determinan como análisis de los procesos que se identifican en el levantamiento de requisitos de una aplicación, donde estas se definen, y explican su funcionamiento. Para automatizar estas es necesario un sistema encargado de darles funcionalidad, logrando implementarlas en un lenguaje y mantenerlas almacenadas para su manipulación. Con esta automatización se evitan fallas, y de este modo, se logran importantes reducciones de tiempo y costo.

Estudios actuales tienden a almacenar las reglas en un repositorio, que son manejadas por un motor de reglas independiente al SI del negocio. Por tanto las restricciones no se codifican en el sistema y este sólo puede realizar accesos necesarios al repositorio(3).

[1] **Grupo de Reglas de Negocio:** acrónimo de *Business Rules Group*.

En estos últimos años se ve una tendencia a gestionar de forma centralizada y sistemática las reglas de negocio, para su fácil manejo, entendimiento y cambio. Con el auge que han tenido los sistemas orientados a procesos, ha sido necesario realizar sistemas que se encarguen del manejo de las reglas de negocio dentro de los procesos. Existen varios motores de reglas como *Ilog Rule*, *External Rule Set ToolKit* y *Drools*, sin embargo no pueden ser utilizados debido a que estos sistemas no utilizan el sistema gestor de base de datos Oracle, no tienen un administrador web y no son genéricos.

El Sistema Único de Identificación Nacional (SUIN) es un sistema orientado a procesos el cual utiliza la tecnología de *Windows Workflow Foundation*[2] (WWF) 3.5. Dicho proyecto no cuenta con un sistema para la gestión de las reglas de negocio. Razón por la cual surge la necesidad de administrar de forma centralizada e independiente dichas reglas, de manera que las mismas puedan ser gestionadas por personal autorizado. Estas reglas pueden variar en el tiempo y se necesita que estos cambios se reflejen en el comportamiento de la aplicación en el menor tiempo posible, deben poder utilizarse en cualquier aplicación desarrollada en la *framework*.NET 4.0, las reglas y sus tipos de datos deben almacenarse en base de datos.

A partir de la situación antes descrita y de la necesidad de facilitar los cambios en la lógica del negocio con un mínimo de alteración en el sistema y en el menor tiempo posible, se define el siguiente **problema científico**: ¿Cómo gestionar reglas de negocio para el SUIN utilizando el *framework* .NET 4.0 y la tecnología WWF 3.5de manera que mejore el tiempo de adaptación en el funcionamiento y la usabilidad del sistema?

Para dar solución al problema científico se define como **objeto de estudio**: reglas de negocios para sistemas orientados a procesos.

Se plantea como **objetivo general**: desarrollar un sistema que permita la gestión de reglas del negocio en aplicaciones desarrolladas sobre el *framework* .NET 4.0 y la tecnología WWF 3.5de manera que mejore el tiempo de adaptación en el funcionamiento y la usabilidad del sistema.

La investigación está enmarcado en el **campo de acción**: reglas de negocio para sistemas orientados a procesos utilizando el *framework* .NET 4.0 y la tecnología WWF 3.5.

El trabajo tiene como **hipótesis**: si se implementa un sistema para la gestión de reglas de negocio se mejorará el tiempo de adaptación en el funcionamiento y la usabilidad de los sistemas desarrollados sobre el *framework* .NET 4.0 y la tecnología WWF 3.5.

Para dar cumplimiento al objetivo planteado se definen las siguientes **tareas de la investigación**:

[2]**Windows WorkflowFoundation:** es básicamente una librería constituida por artefactos de ejecución, reglas, actividades, servicios de *runtime* y un diseñador que permite el diseño gráfico de los *workflows*.

- ✓ Análisis de los principales conceptos asociados a los sistemas de gestión de reglas de negocio, así como los sistemas que los informatizan, para adquirir los conocimientos necesarios y llevar a cabo la investigación.
- ✓ Análisis de sistemas similares a nivel internacional para verificar que no existe un sistema que resuelva los problemas por los cuales se plantea la investigación y adquirir diferentes conocimientos sobre los ya existentes.
- ✓ Estudio de las herramientas a utilizar en el desarrollo de la aplicación para verificar que las herramientas son capaces de satisfacer las necesidades del sistema.
- ✓ Descripción las herramientas, tecnologías y metodología más adecuadas para el desarrollo del sistema con el objetivo de profundizar en sus características y funcionamiento.
- ✓ Identificación de los procesos de negocio para lograr un mejor entendimiento del negocio.
- ✓ Especificación de los requisitos para describir el funcionamiento del sistema.
- ✓ Realización del modelo de datos para la persistencia de los datos en un sistema gestor de base de datos.
- ✓ Realización de los prototipos de interfaz de usuario, para lograr un mayor entendimiento y comunicación con el cliente.
- ✓ Implementación del sistema para la gestión de reglas de negocio en aplicaciones desarrolladas sobre el *framework* .Net, para llevar a cabo la propuesta de solución.
- ✓ Aplicación de pruebas unitarias al sistema de gestión de reglas de negocio, para validar las funcionalidades con las que cuenta el sistema.
- ✓ Diseño de casos de prueba, como guía para realizar las pruebas a las interfaces del sistema.
- ✓ Aplicación de las pruebas de aceptación de software al sistema para la gestión de reglas de negocio para aplicaciones desarrolladas sobre el *framework* .Net, con el objetivo de validar el correcto funcionamiento de la propuesta de solución.

Para la realización de este trabajo se utilizaron los siguientes **Métodos Teóricos,** los cuales estudian y explican teóricamente las características del problema que no se pueden observar a simple vista.

Analítico-Sintético: para un mejor entendimiento y estudio del problema se ha descompuesto el mismo en varios elementos, de esta forma se ha determinado cuáles son los fenómenos que lo generan. Luego de haber realizado el análisis de todos los fragmentos del problema se hace una unión del mismo posibilitando definir sus características generales y las relaciones que existen entre las partes analizadas.

Modelado: se utilizó como herramienta para comprender el problema y crear un modelo abstracto del mismo que permita el diseño de la solución del Sistema de Gestión de Reglas de Negocio.

Histórico-Lógico: se realizó la investigación con apoyo de este método investigativo para encontrar las problemáticas existentes, se realiza un análisis de los procesos de negocio durante toda su trayectoria histórica. Además se hace un análisis de los antecedentes históricos de las reglas de negocio.

Métodos Empíricos

Revisión documental: se utilizó para consultar la bibliografía referente a las reglas de negocio y la información existente en Internet para llevar a cabo las tareas de investigación.

Justificación de la investigación

A partir de la necesidad que existe hoy en el SUIN con el manejo de las reglas de negocio, que actualizar los cambios en una regla implica recompilar la aplicación, se decide crear una aplicación que gestione las reglas de negocio para aplicaciones desarrolladas sobre el *framework* .Net 4.0 y la tecnología WWF 3.5, garantizando una mayor adaptación del sistema ante algún cambio.

Para garantizar el cumplimiento de todos los elementos planteados anteriormente, la presente investigación estará compuesta por tres capítulos que serán resumidos a continuación.

El capítulo I "**Fundamentación teórica**": se definen temas relacionados con las reglas de negocio, se describen los conceptos fundamentales que serán tratados a lo largo de la investigación, así como la selección de la metodología, técnicas y herramientas que serán utilizadas.

El capítulo II "**Concepción de la propuesta de solución**": se exponen las principales características del sistema, se realiza por consiguiente una propuesta general del sistema y el análisis de cada una de las funcionalidades, y procesos del sistema.

El capítulo III "**Implementación y validación**": se describe a profundidad la construcción de la propuesta de solución. Se realiza un análisis de las características y funcionalidades que se desean desarrollar. Así como los resultados de las pruebas realizadas al mismo.

Capítulo 1: Fundamentación Teórica

El objetivo fundamental de este capítulo es hacer un análisis sobre los diferentes aspectos teóricos relacionados con el tema. Se abordan detalladamente los conceptos fundamentales, el estado del arte del sistema y se describen las herramientas y la metodología que se utilizarán para el desarrollo del mismo. Se hace un estudio de las tecnologías que puedan ser adecuadas para el desarrollo del sistema.

1.1Reglas de negocio

Evolución de las reglas de negocio

La primera aparición de la frase regla de negocio fue en el año 1984, cuando *Daniel Appelton* escribió un artículo, llamado “Reglas de negocio: el enlace perdido”. *Appelton* discutió problemas que eran causados por la falta de estandarización de términos de negocios. Su punto de vista consistía en que los analistas del negocio no podían proporcionar soluciones comunes si los usuarios usaban términos que variaban en significado de un departamento a otro dentro de una misma organización.(5)

Existen investigaciones relacionadas con las reglas de negocio en varias áreas del saber. Sus inicios se remontan al campo de la Inteligencia Artificial (IA), donde las reglas le proporcionan a las aplicaciones la capacidad de tener una representación del conocimiento.

Las **reglas de negocio** reciben especial atención tanto desde el punto de vista académico como de la industria, y existen varias definiciones de las mismas por ejemplo:

- ✓ Descripción muy específica que permite controlar y tomar una decisión en una empresa o una organización, describen lo que se debe hacer desde el punto de vista de un experto en un ámbito concreto. Estas reglas pueden ser expresadas como reglas simples (de tipo SI <<Condiciones>> ENTONCES <<Acciones>>), como cuadro de decisión o como árbol de decisión.(6)
- ✓ Afirmación que define o restringe algún aspecto del negocio, pero, en contraste con las sentencias de regla de negocio, no puede ser descompuesta en otras reglas de negocio más detalladas.(7)
- ✓ Componente clave en cómo se toman las decisiones en una empresa. En efecto, cada vez que se toma una decisión dentro de una organización es porque se ha consultado reglas definidas, las cuales en muchos casos se encuentran escritas en manuales de políticas que los responsables de decisiones deben conocer para desarrollar su gestión cotidiana.(8)

Después de un análisis de los diferentes conceptos de reglas de negocio se ha definido para este trabajo que las regla son aquellas condiciones, validaciones y normas que se deben cumplir y controlar

dentro de la organización y que son definidas de acuerdo con el comportamiento esperado del negocio y de la organización.

Las reglas de negocio deben ser:

- ✓ Declarativas.
- ✓ Atómicas.
- ✓ Construidas de manera independiente y distinta.
- ✓ Expresadas en lenguaje natural.
- ✓ Orientadas al negocio.
- ✓ Estar en una fuente única (ej. repositorio de reglas)(9)

Un **contexto** está determinado por el proceso. Es decir, dependiendo del proceso la información será guardada y presentada de manera distinta. Por lo tanto es determinante saber el contexto del proceso con el fin de ser capaz de crear formas, reglas de negocio y asignar los recursos de un proceso.

La **fórmula de la regla** permite construir de manera gráfica la lógica de cada regla. Los módulos que se presentan pueden ser combinados, agrupados o anidados para conseguir el código de la lógica que se desea expresar.

Una **variable** es un elemento de una fórmula, proposición o algoritmo, que puede ser sustituido o puede adquirir un valor cualquiera dentro de su universo.

La **condición** de regla de negocios determina la condición que debe cumplir para que se realicen una o varias acciones.

La **acción** de una regla de negocios es el resultado de la evaluación de la condición de la regla si se cumple o no.

Operadores: el marco de trabajo de las reglas de negocio admite el uso de operadores lógicos y aritméticos en la creación de las reglas de negocio, entre los operadores lógicos encontramos Y, O, NO y dentro de los aritméticos agregar, restar, multiplicar, dividir y resto.(10)

Para el establecimiento de procedimientos y estándares de creación y uso de las reglas de negocio, han aparecido y siguen apareciendo propuestas que se centran, bien en la semántica y sintaxis de las expresiones, bien en sistemas de definiciones de terminologías ligadas a repositorios de datos que puedan ser transformables a lenguaje XML y otros para la intercomunicación entre empresas.

Con herramientas disponibles públicamente para construir y ejecutar sistemas expertos, muchos departamentos IT[3] y unidades de negocio reconocieron la posibilidad de crear aplicaciones basadas en conocimiento sin la necesidad de adquirir software comercial. Igualmente, por las fuerzas del mercado,

[3]**Tecnología de Información:** *acrónimo de Information Technology.*

muchas empresas surgieron con el fin de crear sus propios motores para implementar soluciones a la medida, con el pasar del tiempo, la competencia entre diferentes proveedores provocó que muchos de éstos incluyeran características adicionales: manejo de bases de datos para almacenar las reglas, editores gráficos para representarlas, flujos de trabajo para publicarlas, entre otras. Poco a poco fueron evolucionando de simples herramientas y ejecución de reglas a complejos sistemas para administrar su ciclo de vida, incluyendo monitoreo y detección de errores. Así surgieron los ahora denominados Sistemas de Gestión de Reglas de Negocio.

Tipos de reglas de negocio

Reglas de asignación: almacenan el valor en el operando derecho en la ubicación designada por el operando izquierdo. No obstante, para que pueda tener lugar el proceso de asignación, es posible que sea necesario realizar procesos de cálculo complejos o de recuperación de datos para obtener el valor de un operado.

Reglas de datos: son las reglas a cumplir por el valor que adquiere la tipología de datos que tratamos en la actividad. Por ejemplo, el alumno no puede tener un año de nacimiento negativo.

Reglas de restricción: estas implican el control de valor que puede adquirir un dato en función a otros datos. Por ejemplo, el alumno no puede matricularse en más de X créditos en un mismo año.

Reglas de acceso: son aquellas que establecen los controles de acceso oportunos a la información y a los procesos. Por ejemplo al expediente de un alumno solo tiene acceso el usuario con perfil de secretaria.(11)

Enfoque de las reglas de negocio

El enfoque de reglas de negocio está dirigido a proporcionar a las personas del negocio un control directo sobre el funcionamiento del mismo, logrando así que el personal pueda definir las reglas y normativas de su trabajo, manteniendo un lenguaje uniforme, actualizando con gran facilidad los sistemas de cómputo que le sirven de soporte en su trabajo.

Con el enfoque de reglas de negocio la tecnología está avanzando al punto en que estos costos no necesitan ser tolerados, y se devuelve el control a las personas del negocio.

Vías para implantar y gestionar reglas

Se pueden distinguir principalmente tres vías para implantar y gestionar reglas:

- ✓ *Hardcode*: Es el sistemas más tradicional y ya casi obsoleto. Las reglas se implementan directamente en el código fuente de los sistemas de la organización. El principal problema de

esta práctica es que el mantenimiento es complicado y costoso y las organizaciones no obtienen la agilidad y eficiencia necesarias.(12)

- ✓ *BRE*[4]: Nacieron para independizar las reglas de negocio del resto del código fuente. Se componen básicamente de un repositorio de reglas y un motor de ejecución. Representaron un importante avance sobre la opción "*hardcode*" ya que los *BRE* suelen permitir desplegar reglas en caliente y sin necesidad de modificar el resto del código. Aun así, son sistemas muy orientados a IT y no consiguen que las áreas de negocio puedan definir sus reglas de manera independiente.(12)
- ✓ *BRMS*[5]: Son la evolución natural de los *BRE*. Incorporan un IDE gráfico y funcionalidades orientadas a usuarios no técnicos con el objetivo de cubrir el ciclo de vida completo de las reglas. Así, los *BRMS* se han convertido en la herramienta completa para gestionar las reglas de negocio y favorecer la colaboración entre IT y negocio.(12)

Categorías de las reglas de negocio

Los sistemas de reglas de negocio tienen tres componentes fundamentales: la estructura, la potencia y el control. La estructura es proporcionada por los conceptos del negocio y sus relaciones, la potencia por los procesos y el control por las reglas de negocio.(13)Una regla de negocio cae en una de las siguientes cuatro categorías:

- ✓ Términos: el elemento básico de una regla de negocio es el lenguaje usado para expresarla. La propia definición de un término es en sí una regla de negocio que describe cómo las personas piensan y hablan acerca de las cosas. Por tanto, definir un término es establecer una categoría de regla de negocio. Los términos han sido tradicionalmente documentados en glosarios o como entidades en el modelo elegido.
- ✓ Hechos: relacionan términos con otros: la naturaleza o estructura operativa de una organización puede ser descrita basándose en los hechos que relacionan términos con otros términos. Los hechos pueden ser documentados como sentencias en lenguaje natural o como interrelaciones, atributos y estructuras de generalización en un modelo gráfico.
- ✓ Restricciones o sentencias de acción: cada empresa restringe el comportamiento de alguna manera.

[4]**Motor de reglas**:acrónimo de *Business Rules Engines*.

[5]**Sistema de gestión de reglas de negocio:**acrónimo de Business Rules Management System.

- ✓ Derivaciones: las reglas de negocio definen cómo el conocimiento en una forma puede ser transformado en otro conocimiento, posiblemente en una forma diferente.(5)

Uso de las reglas de negocio

El uso de las reglas de negocio se clasifica en cinco grupos. A continuación se encuentra la descripción de cada uno de estos grupos.

- ✓ Reglas asociadas a transiciones: las reglas de transición, son aquellas reglas que permiten verificar que en algún punto del proceso se está cumpliendo una condición específica y el proceso continúa por el camino asociado a la condición que se cumplió. Estas reglas siempre deben retornan verdadero o falso.
- ✓ Reglas de condiciones de visibilidad, obligatoriedad o edición de campos o atributos de una forma: estas reglas son usadas para validar información, es decir, para garantizar que la información se guarde adecuadamente para la ejecución del proceso, permitiendo hacer visible, requerido o editable un atributo en una forma de acuerdo a una condición. Al Igual que las reglas de transición, tienen la característica que deben retornar verdadero o falso.
- ✓ Reglas usadas en eventos (acciones) dentro de una figura: estas reglas permiten realizar una acción al inicio, al guardar o al salir de una actividad. Las actividades automáticas son definidas por estas reglas, en este caso debe crear la regla en el evento “entrar” de la actividad o al “salir”.
- ✓ Reglas de asignación de actividades: las asignaciones se componen, de dos reglas. Una que determina la condición para realizar la condición y la otra que determina el usuario específico que debe realizarla.

  Las reglas que determinan la condición para realizar la asignación siempre retornan verdadero (true) o falso (false). Las reglas de asignación determinan el usuario o el perfil específico (ubicación, área, cargo, rol y/o habilidad) del usuario que debe realizar la actividad dentro del proceso.(9)

Lenguaje de las reglas de negocio

Las reglas de negocio pueden ser escritas en un lenguaje de programación de sintaxis, por tanto pueden ser administrados por los analistas de negocio, sin conocimientos de programación. Destacando que el lenguaje PEL6 está diseñado específicamente para crear reglas de negocio.

[6]**PEL**: P*erformance Point Expression Language.*

Los lenguajes de programación orientada a objetos o de procedimiento tradicional, como C, C++ y *Microsoft Visual Basic*, están orientados a los programadores. Incluso los lenguajes avanzados de programación orientada a objetos, como Java y C#, siguen siendo principalmente lenguajes de programadores. El ciclo de desarrollo tradicional de software basado en diseño, desarrollo, compilación y comprobación requiere mucho tiempo y coordinación, y no permite a quienes no son programadores participar en el mantenimiento de directivas empresariales automatizadas. El marco de trabajo de reglas de negocios corrige este problema, facilitando un entorno de desarrollo que permite la rápida creación de aplicaciones sin el extenso ciclo de programación tradicional de aplicaciones.

Entre otros lenguajes podemos encontrar los que utiliza Drools, para especificar las reglas (DRL[7]), para las condiciones, acciones y funciones de las mismas, las cuales se puede expresar con distintos lenguajes, como Java y MVEL.(14)Luego son guardadas en archivos de texto con la extensión drl. Aparte del DRL se pueden especificar lenguajes específicos de dominio (DSL[8]), los cuales se asocian a un drl, y también existe la opción de especificar las reglas en una planilla de cálculo, como Excel.

1.2Sistema de Gestión de Reglas de Negocio

Los sistemas de gestión de reglas de negocio nacen con el objetivo de cubrir principalmente una necesidad en la infraestructura de las organizaciones como son facilitar la definición, despliegue y seguimiento de las reglas de negocio de una compañía. Es lo que se conoce como gestión del ciclo de vida de las reglas.

Los *BRMS* son sistemas informáticos que permiten facilitar la comunicación entre los administradores de negocios y los que manejan la tecnología de la informática,(15)dando a los primeros control de la lógica e incluso del código de definición de la aplicación. Estas aplicaciones se pueden ver como un sistema de n-capas que contiene una combinación de web server, servidores de aplicación y base de datos. El *BRMS* ocupa una cuarta capa que pone la lógica en manos de los analistas y ejecutivos claves del negocio.

Una de las herramientas más actuales en Gestión Empresarial es el *BRMS* que está compuesto por un motor de reglas. Además se relaciona con otras herramientas de Gestión, interactuando a distintos niveles para asistir en la toma de decisiones.

[7]**Lenguaje de reglas de Drools:***acrónimo deDrools Rule Language*

[8]**Lenguajes específicos de dominio:***acrónimo de DomainSpecificLanguage*

Básicamente, el sistema permite al usuario definir sus reglas de negocio, de forma que se van aplicando automáticamente en la toma de decisiones. En su relación con otras herramientas, se conecta con los sistemas *BPM*[9] (*Business Process Management*) y con los de *Business Intelligence*[10].

Entre las ventajas de estos sistemas encontramos las siguientes:

- ✓ Los analistas del negocio toman control directo de las reglas que gobiernan cómo se maneja y comporta la empresa.
- ✓ Se pueden hacer alteraciones directamente sin necesidad de esperar por soporte del área de informática.
- ✓ Proveen una solución muy importante para cierto tipo de negocios que requieran agilidad en la toma de decisiones y a la vez rapidez de adaptación en el funcionamiento de sus sistemas por los cambios constantes del medio.
- ✓ Permiten que los analistas del negocio puedan ver y entender las reglas sin tener que depender del departamento de informática para realizar las modificaciones.
- ✓ El costo de mantenimiento se reduce al no tener que recodificar, reprobar, recompilar y reinstalar el aplicativo por cada cambio menor.
- ✓ Los cambios requeridos por los usuarios pueden ser implementados sin cambios en el código, aislando el cambio y probando sólo la regla que ha sido modificada.

Las reglas de negocio a menudo están codificadas dentro del software, dificultando su acceso y modificación e incrementando los costes. Gracias a que permite hacer una gestión explícita de las reglas de negocio independientemente del software de las aplicaciones, el *BRMS* se ha convertido en la tecnología clave para las arquitecturas orientadas a servicios (SOA[11]),(16) la gestión de los procesos de negocio (*BPM*) y otras iniciativas tecnológicas innovadoras.

Con las herramientas actuales, el usuario dispone de una interfaz amigable para reprogramar por sí mismo sus reglas de negocio.

Los *BRMS* permiten a las aplicaciones basadas en reglas:

- ✓ Capturar definiciones y reglas que son sujeto de cambios frecuentes.
- ✓ Implementar esos cambios rápidamente y fácilmente dentro de una aplicación
- ✓ Gestionar y escribir reglas en un lenguaje de negocio familiar.

[9] **Gestión de procesos del negocio**: acrónimo de *Business Process Management.*

[10] **Inteligencia de negocio**: acrónimo de *Business Intelligence.*

[11] **Arquitectura orientada a servicios**: acrónimo de *Service-Oriented Architecture.*

- ✓ Usuarios expertos pueden actualizar la funcionalidad de acuerdo a sus proyectos en vez de que lo haga la tecnología.

Objetivos y características de los *BRMS*

Objetivos:

- ✓ Controlar y dar visibilidad de las decisiones del negocio por parte del analista de negocio.
- ✓ Identificar las reglas de negocio con el fin de darles un correcto tratamiento.
- ✓ Optimizar la ejecución de reglas de negocio mediante motores de reglas específicos.

Características:

- ✓ Sentencia básica de una regla
- ✓ Motor de reglas
- ✓ Repositorio de reglas
- ✓ Plantilla de reglas
- ✓ Chequeo de sintaxis de reglas

Sentencia básica de una regla: las reglas en los *BRMS* son caracterizadas por ser

- Declarativas, no procedimentales
- Indican cuando una expresión es verdadera, no cómo calcularlo
- Usualmente tienen la forma: *IF A THEN B*
- Puede tener más de una sentencia combinadas por *AND y OR*
- Puede tener más de una acción resultante

Motor de reglas: es un sistema que se configura para dar servicio a las necesidades del negocio a través de la definición de objetos y reglas de negocio, el software se rige por flujos que derivan responsabilidades a los distintos cargos de la empresa repartiendo así el trabajo equitativamente y cuantitativamente, indicando cuándo, quién y dónde tiene que desempeñar la tarea asignada.

Repositorio de Reglas: lugar centralizado donde todas las reglas de negocio son guardadas, creando una plataforma de fácil migración y transferencia durante los múltiples desarrollos de sistemas. Además de crear la posibilidad de almacenar diferentes versiones de definición de reglas y guardar la historia de los cambios soportando el versionamiento de reglas. Además de permitir la habilidad para auditar reglas y la definición de múltiples versiones de reglas.

Las Plantillas o *templates:* son patrones predefinidos para las reglas. Una plantilla de regla de negocio representa una regla definida parcialmente que contiene espacios para completar información. Pueden ser usados para crear múltiples reglas con estructura similar, donde sólo varía el valor llenado.

Chequeo de sintaxis de reglas: un buen *BRMS* ofrece facilidades para chequear la sintaxis de las reglas en tiempo real respecto a como si fuese un lenguaje estructurado.(17)

Generalidades de los *BRMS*

La tecnología BRMS constituye una evolución de los llamados *BRE* o "motores de reglas" (*Business Rules Engines*), un tipo de software que, a partir de una entrada inicial de datos, es capaz de decidir qué reglas deben aplicarse a cada caso concreto. Los programas *BRMS* pretenden acercar la gestión de reglas de negocio que hasta ahora se llevaba a cabo desde las áreas técnicas a los especialistas en esta área.

"La aplicación de la tecnología *BRMS* supone una ayuda sustancial a la toma de mejores decisiones con mayor rapidez con el objetivo de manejar los cambios y su complejidad asociada", en palabras de Bertrand.(18)

“Las ventajas que se derivan de esta tecnología se aprecian en dos grandes vertientes. En primer lugar, en la toma de decisiones la capacidad de explicar y tomar una decisión respecto a una determinada política, capacidad de trasladar el proceso a una máquina liberando la capacidad de los profesionales para que puedan dedicarse a otras tareas”, explica Bertrand, quien añade que, por otra parte, el sistema se convierte en un mecanismo de mejora continua en tanto que "las funciones de la máquina pueden enriquecerse cada vez que se descubren y formulan nuevas reglas de negocio".(18)

Las soluciones *BRMS* aparecen como el medio idóneo para materializar una aptitud de búsqueda permanente de una mayor eficiencia, que se traduce en un grado más elevado de automatización y transparencia.

Tareas que facilitan los *BRMS*

Con un BRMS es posible realizar cualquiera de las siguientes tareas:

- ✓ Encapsulamiento: en vez de tener una serie de *IF* anidados en el código que hacen engorrosa su modificación, se les simplifica al definirlos como estructuras atómicas e independientes del resto de la aplicación.
- ✓ Agregación: con el *BRMS*, se generan flujos de trabajo mediante conjuntos coherentes de reglas, estableciendo jerarquías de ejecución de acuerdo al dominio del problema. Es decir, ya no sólo se realiza una gestión de reglas, sino de procesos de negocio más complejos.
- ✓ Externalización: se publican estas reglas de manera individual o en conjuntos previamente definidos en forma de servicios que serán consumidos por terceros. Así, se incrementa la agilidad de respuesta al cambio.

1.3Soluciones existentes a nivel Internacional

El estudio de algunos de los sistemas que se dedican a la gestión reglas de negocio se presenta a continuación.

ILOG Rules for .NET es un sistema de gestión de reglas de negocio completamente funcional diseñado específicamente para tecnologías *Microsoft* y la plataforma .NET. Brinda funcionalidad para construir e implementar aplicaciones basadas en reglas para entornos *.NET* y *SOA*. Permite la gestión de las reglas de negocio a lo largo del ciclo de vida de las aplicaciones de negocio de misión crítica. Para los analistas y otros usuarios de negocio, proporciona herramientas de creación y gestión integradas en *Microsoft Office System* y una interfaz basada en la Web.(19)Para los arquitectos y desarrolladores de *software*, proporciona un conjunto de herramientas dentro de *Microsoft Visual Studio* .NET para integrar la tecnología de reglas de negocio de *ILOG* directamente en las aplicaciones .NET. Además de permitir que las reglas de negocio sean cambiadas rápidamente y reestructuradas en sin cambiar el código de la aplicación. El archivo no puede ser accedido desde Cuba, por lo que no pudo ser evaluado.

La herramienta *External Rule Set Toolkit* aplica números de versión principal y secundaria a las Reglas, permitiéndole mantener simultáneamente y almacenar varias versiones, la herramienta no proporciona ningún bloqueo u otras características de gestión de la configuración además de la función de administración de la versión. Con la herramienta, puede crear nuevas versiones de Reglas o eliminar las versiones existentes.(20)Presenta varias deficiencias dentro de las cuales se encuentran que puede ser utilizada únicamente para aplicaciones con *workflow* ya que está anclada a las actividades de los mismos, está realizada para el gestor de base de datos *SQL*[12], no tiene un administrador de web y no permite guardar el contexto de la regla en base dato.

Como última herramienta se analizó *Drools* .NET el cual es un software con licenciamiento libre, distribuido según los términos de la licencia apache. Está basado en la especificación JSR-94 (*Java Rule Engine* API), convirtiéndolo por tanto en un estándar. Actualmente forma parte de los productos de la corporación *JBoss Inc*. bajo el nombre de *JBoss Rules*. Dispone de un motor de reglas basado en inferencia de encadenamiento hacia adelante, más conocido como sistema de reglas de producción, usando una implementación avanzada del algoritmo Rete con un alto rendimiento, optimización, y por tanto, eficiencia. Aunque existen otros productos BRMS de código abierto como es el caso de Open Rules, *Drools* es un producto sólido con la garantía de formar parte de uno de los productos que conforman *JBoss*. (21)Esto lo convierte en un producto estrella entre otros *BRMS* de código abierto del

[12]**Lenguaje de consulta estructurado**: acrónimo de *Structured Query Language.*

mercado. Como todo sistema presenta inconvenientes como el hecho de no tener un administrador de reglas, las mismas tienen que ser creadas solamente por quien tenga conocimientos de programación y si existen cambios habría que cambiar la versión desplegada.

En la investigación realizada se encontraron los sistemas antes mencionados que realizan la gestión de reglas de negocio, sin embargo no pueden ser utilizados porque no cumplen con las necesidades y especificaciones indicadas por el Sistema Único de Identificación Nacional.

1.4Tecnologías, metodologías y lenguajes utilizados

Plataforma .NET

La plataforma .NET de *Microsoft* provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el producto principal en la oferta de *Microsoft*, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma *Windows*. Es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios *web XML*[13].

Algunas de las ventajas que ofrece la plataforma.NET son las siguientes:

- ✓ Administración de código: se realiza control automático del código haciendo que este sea seguro.
- ✓ Lenguajes interoperables: en una misma solución se puede utilizar cualquier lenguaje o incluso varios a la vez siempre que sean compatibles con .NET.
- ✓ Recolección de basura: el *CLR*[14] detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. El programador no tiene que preocuparse por liberar la memoria aunque si lo desea puede hacerlo manualmente.
- ✓ Rendimiento: Todos los códigos que se ejecutan en el ambiente .NET son compilados, lo cual proporciona un gran rendimiento a diferencia de versiones interpretadas.(22)

Framework.NET 4.0

.NET *Framework* es el modelo de programación completo y coherente de Microsoft para compilar aplicaciones que ofrezcan una sensacional experiencia visual del usuario, comunicación perfecta y segura, y la capacidad de modelar una amplia gama de procesos empresariales. Está formado

[13]**Lenguaje de marcas extensible:** acrónimo de *Extensible Markup Language*.

[14]**Entorno en tiempo de ejecución de lenguaje común:** acrónimo de*Common Language Runtime.*

básicamente por dos elementos, el entorno de ejecución de aplicaciones fundamentales y las bibliotecas base.(23)

Microsoft .NET Framework 4.0 proporciona nuevas mejoras y características que la hacen diferente a sus versiones anteriores, aunque posee la capacidad de funcionar en paralelo con ellas.

Entre sus características encontramos:

- ✓ Mejoras en CLR y la BCL[15].
- ✓ Innovaciones en los lenguajes *Visual Basic* y *CSharp*.
- ✓ Mejoras en el acceso a datos y el modelado.
- ✓ Mejoras en *ASP.NET*.
- ✓ Mejoras en *Windows Presentation Foundation* (WPF).
- ✓ Mejoras en *Windows Workflow* (WW) que permiten a los desarrolladores interactuar con flujos de trabajo.
- ✓ Mejoras en *Windows Communication Foundation (WCF)*.

ASP.NET

ASP.NET es un *framework* para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web *XML* con el código mínimo. *ASP.NET* forma parte de .NET *Framework* y al codificar las aplicaciones *ASP.NET* se tiene acceso a las clases en .NET *Framework*. Está construido sobre el *Common Language Runtime*, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET *Framework*. Las páginas de *ASP.NET*, conocidas oficialmente como *web forms* o formularios web, son el principal medio de construcción para el desarrollo de aplicaciones web. (24)

ASP.NET trae diversas mejoras entre las cuales se destacan:

- ✓ Rendimiento: la aplicación compila en una sola vez al lenguaje nativo, y luego, en cada petición tiene una compilación *Just In Time*, es decir se compila desde el código nativo, lo que permite mucho mejor rendimiento.
- ✓ Rapidez en programación: mediante diversos controles, se puede con unas pocas líneas mostrar toda una base de datos y hacer rutinas complejas.
- ✓ Servicios web: trae herramientas para compartir datos e información entre distintos sitios.
- ✓ Seguridad: tiene diversas herramientas que garantizan la seguridad de las aplicaciones.

[15] **Biblioteca de clases base:** *acrónimo de Base Class Library.*

1.4.1Metodología de desarrollo: MSF for CMMI

MSF[16] *for CMMI*[17] define cinco fases durante el ciclo de vida del proyecto que encapsula flujos de actividades y actividades. Las fases son inicio, planificación, construcción, estabilización y despliegue. Cada fase concluye con un punto de control. Cada punto de control proporciona una oportunidad para autorizar el trabajo, continuar en el proyecto, o cancelar o suspender el proyecto.(25)

Entre los diferentes roles definidos por esta metodología se encuentran el líder, jefe de desarrollo, jefe de producto, el arquitecto de software, desarrollador, analista, diseñadores, probador, integrador, documentador-capacitador, administrador de la calidad, entre otros. Los miembros del equipo de trabajo pueden desempeñar roles distintos en las diferentes fases del ciclo del vida del proyecto y son responsables de cumplir con las actividades y de generar la documentación. Esta metodología define que ningún rol es más importante que otro. Uno de los beneficios de implementar la metodología de *MSF for CMMI* es contar con una evaluación estándar por medio de la cual se puede validar la habilidad de desarrollar software en una organización.

Es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Características:

- ✓ Adaptable: es parecida a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- ✓ Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.
- ✓ Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.

Microsoft Solution Framework

La metodología *MSF* es del tipo de metodologías ágiles, está enfocada a dirigir proyectos o soluciones de innovación, en ella no se detalla ni se hace énfasis de la organización ni el tamaño del equipo de desarrollo, está más bien centrada en la gestión y administración del proyecto para lograr el impacto deseado.(26) La misma cuenta de cinco fases las cuales serán explicadas a continuación:

[16]**Framework de Soluciones de Microsoft**: acrónimo de *Microsoft Solution Framework.*

[17]**Integración de Modelos de Madurez de Capacidades**: acrónimo de*CapabilityMaturityModelIntegration.*

Visión: en esta fase se debe realizar un estudio de lo que se pretende que en el futuro que haga la aplicación o proyecto, para ello debe realizarse un documento de estrategia y alcance donde debe quedar pactada la necesidad de funcionalidad y servicio que se debe contar en la solución.
Planificación: se debe concretar claramente la estructura dela solución para ello se debe crear un documento de planificación y diseño de la arquitectura, diseñar las pruebas de concepto donde se plantean los diferentes escenarios para probar la validez de los criterios utilizados para el diseño, además de las métricas.
Desarrollo: se debe codificar las aplicaciones y realizar las configuraciones necesarias para un correcto funcionamiento, además de hacer pruebas continuamente para verificar la calidad del producto a lo largo del desarrollo y no únicamente al final del proceso.
Estabilización: se debe seleccionar el entorno de prueba piloto, lo que se pretende con esto es identificar las deficiencias con un grupo reducido de usuarios para corregirlas y así en el futuro no tener problemas cuando se use la solución por todos, se debe crear un plan de gestión de incidencias, realizar una revisión de documentación final de la arquitectura y elaboración de plan de despliegue o implementación.
Despliegue o Implementación: en esta etapa final ya se ha comprobado la calidad de la solución por lo cual está lista para ser publicada, en este sentido se debe liberar la solución y crear un registro de mejoras y sugerencias, revisar las guías y manuales y entrega de proyecto final.

Beneficios
MSF ayuda a implantar soluciones de tecnología:

- ✓ Permitiendo desarrollar siguiendo una filosofía de reutilización de múltiples componentes, lo cual reduce el tiempo necesario para desarrollar nuevas aplicaciones, garantiza la uniformidad e interoperabilidad entre las mismas, y las hace mucho más flexibles para incorporar los cambios que sean necesarios en el futuro.
- ✓ Estableciendo un equipo de trabajo balanceado, con tareas y objetivos claramente definidos, que permitan no solo desarrollar buenos sistemas, sino también saber en todo momento cuál es el grado real de avance del proyecto, y cuáles son los riesgos que se corren si se decide introducir modificaciones al mismo una vez que el desarrollo ya ha sido iniciado.

Modelo de Capacidad y Madurez (CMMI)
CMMI es un modelo de mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software. El mismo es una nueva generación de modelos de madurez, que basado en los principios de calidad total, propone un conjunto de buenas prácticas que enseñan qué hacer para

implantar procesos productivos más efectivos. Este modelo plantea cinco posibles niveles de madurez que puede alcanzar una organización. A su vez, cada nivel con excepción del inicial queda caracterizado por un conjunto de áreas de proceso que agrupan prácticas que, al ser ejecutadas colectivamente, permiten cumplir con algún objetivo que es considerado importante para el modelo.
Los niveles de *CMMI* son:

- ✓ Inicial.
- ✓ Administración básica de proyectos.
- ✓ Proceso estandarizado.
- ✓ Proceso administrado cuantitativamente.
- ✓ Proceso en mejora continua. (27)

Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. Permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

Lenguaje C#

El lenguaje de programación C# (pronunciado en inglés "*C Sharp*" o en español "C sostenido") es un lenguaje de programación orientado a objetos. (Lenguaje de Programación C#, 2006) "Es una evolución de los lenguajes C y C++ e incorpora las ventajas o mejoras que tiene el lenguaje JAVA.(28)
Algunas de las características del lenguaje de programación C# se fundamentan en su código que trata íntegramente como un objeto. Su sintaxis es muy similar a la de JAVA. Es un lenguaje orientado a objetos y a componentes. Permite el ahorro de tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada. Está estructurado en tres partes fundamentales, las cuales son, una librería estándar, un programa compilador y un preprocesador.
Entre sus principales características se destacan:
Sencillez: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:

- ✓ El código escrito en C# es auto contenido. No necesita de ficheros adicionales al propio fichero fuente tales como ficheros de cabecera o ficheros IDL[18].

[18]**Lenguaje de descripción de interfaz**: acrónimo de *Interface Definition Language*.

- ✓ El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile.
- ✓ Incorpora elementos útiles para el desarrollo de aplicaciones como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits.
- ✓ La instrucción *foreach* permite recorrer colecciones con facilidad y es ampliable a tipos de datos definidos por el usuario.

Eficiente: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador *unsafe*) y podrán usarse en ellas punteros de forma similar a como se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grande. (29)

Dentro de las características fundamentales que lo hacen un lenguaje robusto se encuentran las siguientes:

- ✓ Provee el beneficio de un ambiente elegante y unificado.
- ✓ El manejo de errores está basado en excepciones.
- ✓ Soporta los conceptos como encapsulación, herencia y polimorfismo de la programación orientada a objetos.
- ✓ Soporta los modificadores de acceso *private, protected, public* y agrega un cuarto modificador *internal*.
- ✓ Permite la interoperabilidad con APIs[19] al estilo C y DLLs[20], esta característica para acceder a APIs nativas es llamada *Platform Invocation Services* (PInvoke).
- ✓ No soporta herencia múltiple, solamente el *runtime* .NET permite la herencia múltiple en la forma de interfaces, las cuales no pueden contener implementación.

Se utilizó el lenguaje C Sharp porque es nativo de la plataforma. NET, es un lenguaje robusto, orientado a objetos, que posee gestión de excepciones y comprobación de tipos, estos ayudan a tener un software menos propenso a errores, ya que muestra donde está el error para que sea corregido.

[19] **Interfaz de programación de aplicaciones**: acrónimo de *Application programing interface*.

[20] **Lenguaje de definición de datos**: acrónimo de *Data Definition Language*.

Lenguaje de modelado: UML

UML[21] es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Fue concebido por *Grady Booch*, *Ivar Jacobson* y*Jim Rumbaugh*. Estos autores fueron contratados por la empresa *Rational Software Company* para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de *UML* han participado, no obstante, otras empresas de gran peso en la industria como *Microsoft*, Oracle o IBM, así como grupos de analistas y desarrolladores.

Los principales objetivos en el diseño de *UML* fueron: obtener un lenguaje simple pero suficientemente expresivo, que permitiese modelar aplicaciones en cualquier dominio, obtener un lenguaje legible, puesto que sería un lenguaje utilizado por las personas y permitir la generación automática de código.

El UML provee beneficios significativos para los ingenieros de software y las organizaciones al ayudarles a construir modelos rigurosos y trazables, que soporten el ciclo de vida de desarrollo de software completo.(30)UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- ✓ Mayor rigor en la especificación.
- ✓ Permite realizar una verificación y validación del modelo realizado.
- ✓ Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa, es decir a partir del código fuente generar los modelos.

Este lenguaje ha mejorado el desarrollo de software no sólo al establecer un estándar común que simplifica la comunicación entre desarrolladores de software. Sus principios fundamentales son fáciles de entender y de aprender. Hoy en día, es el lenguaje de la ingeniería de software. Es utilizado no sólo para la especificación de un sistema sino también para propósitos de comunicación entre las personas involucradas en el desarrollo de un sistema (ingenieros, científicos del área de computación, administradores, líderes y otros), o para la documentación de software existente.

Hypertext Markup Language (XML)

XML es una tecnología muy sencilla que tiene a su alrededor otras tecnologías que la complementan y la hacen mucho más grande, con unas posibilidades enormes y básicas para la sociedad de la información.(31) Su principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Además de permitirle al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas

[21]**Lenguaje unificado de modelado**: acrónimo de *Unified Modeling Language*.

tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello.
XML no es un lenguaje, sino varios lenguajes, no es una sintaxis, sino varias y no es una manera totalmente nueva de trabajar, sino una manera más refinada que permitirá que todas las anteriores se puedan comunicar entre sí sin problemas, ya que los datos cobran sentido.
Es importante en el mundo de internet debido a que existen muchos sistemas distintos que tienen que comunicarse entre sí, interesa por igual a todas las ramas de la informática y el tratamiento de datos, permite muchos avances a la hora de trabajar con ellos. Las ventajas de usar *XML* son las que a continuación se mencionan:

- ✓ Son datos plenamente personalizados según las necesidades puntuales.
- ✓ Son fáciles de automatizar y gestionar.
- ✓ Admite la definición de vocabularios específicos.
- ✓ Separa contenido del procesamiento y visualización.
- ✓ Aumenta la seguridad mediante la validación de documentos.

Se utilizó el lenguaje XML para almacenar las reglas de una forma serializada en la base de datos. Teniendo en cuenta que permite almacenar los datos basándose en estructuras y las características antes expuestas.

Cascade Style Sheet (CSS)

CSS[22] es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML[23] y XHTML[24]. Es la mejor forma de separar los contenidos y su presentación además de ser imprescindible para la creación de páginas web complejas. CSS es la forma más recomendable de dar formato a archivos de lenguaje decodificación de hipertexto, para poder mantenerlos y actualizarlos de forma sencilla. Este lenguaje permite trabajar por separado el formato y el contenido de un documento,(32)
La separación de los contenidos y su presentación presentan numerosas ventajas, obliga a crear documentos HTML/XHTML bien definidos y con significado completo, también llamados documentos semánticos. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

[22] **Hojas de Estilo en Cascada**: acrónimo de *Cascade Style Sheet.*

[23] **Lenguaje de marcado de hipertexto**: acrónimo de *HypertextMarkupLanguage.*

[24] **Lenguaje de marcado de hipertexto extensible:** acrónimo de *Extensible Hypertext Markup Language.*

CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, en fin permiten controlar la apariencia de una página *Web*.

Sus características más importantes son:

- ✓ Controlan todos los elementos de la presentación de un documento *HTML*.
- ✓ Permiten agrupar varios formatos en nuevas clases de texto, para evitar tener que repetir las mismas etiquetas *HTML* en diferentes partes del documento.
- ✓ La presentación final de un servicio de información se puede alterar sin más que alterar una hoja de estilo común que compartan todos los documentos.

Dentro de los beneficios que brinda dicho lenguaje se destacan los siguientes:

- ✓ Separación entre contenido y estructura.
- ✓ Control más preciso de la apariencia.
- ✓ Páginas más pequeñas y ágiles.
- ✓ Gestión de cambios más rápido y fácil.
- ✓ Transparente respecto al navegador.
- ✓ Los documentos que usan Hojas de Estilo generalmente resultan más compactos.
- ✓ Las hojas de estilo pueden aplicarse de varias maneras y combinarse formando una cascada de estilos con la información de cada una.
- ✓ Pueden usarse con otros lenguajes de programación (como JavaScript) para conseguir efectos dinámicos en las páginas.

Se utilizó el estilo CSS para definir la presentación, además que permite un control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo y los beneficios previamente mencionados.

JavaScript

JavaScript es un lenguaje que puede ser utilizado por profesionales y por quienes se inician en el desarrollo y diseño de sitios *Web*. No requiere de compilación ya que funciona del lado del cliente. Es soportado por la mayoría de los navegadores como *Internet Explorer*, *Netscape*, *Opera*, *Mozilla Firefox*, entre otros. Además de poner a disposición del programador todos los elementos que forman la página Web, para que éste pueda acceder a ellos y modificarlos dinámicamente. Este lenguaje es es usado para crear pequeños programas dentro de una página web que permiten interactuar con el usuario(33)

En este lenguaje no es necesario declarar los tipos de variables que van a utilizarse porque realiza una conversión automática de tipos, además las referencias a objetos se comprueban en tiempo de ejecución debido a que no es un lenguaje compilado.

Java Script destaca su capacidad de integrarse a la perfección con el sistema operativo y con la mayoría de los navegadores web, además que puede combinarse con otras herramientas de desarrollo web, como hojas de estilo CSS y PHP.

1.5Herramientas

Sistema gestor de base de datos: Oracle

Sistemas de Gestión de Bases de Datos (SGBD)

Un SGBD es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando la seguridad e integridad de los mismos. Además permite crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.(34)

Oracle es un RDBMS[25] desarrollado por *Oracle Corporation*. Se considera uno de los sistemas gestores de bases de datos más completos, destacando:

- ✓ Soporte de transacciones
- ✓ Estabilidad
- ✓ Escalabilidad
- ✓ Soporte multiplataforma

Oracle *Database* 11g proporciona nuevas e innovadoras funcionalidades que garantizan alto rendimiento, alta escalabilidad, fiabilidad y seguridad mediante el uso de plataformas, asegurando altos niveles de calidad de servicio e incrementos de la flexibilidad de negocio reduciendo además los costes de explotación. Incorpora seguridad de archivos que permite la gestión de todo tipo de datos, incluyendo imágenes, ficheros de texto o tipos avanzados de datos soportados de manera nativa, como XML, imágenes médicas y objetos en 3D.

Oracle Database 11g es el primer gestor de base de datos del mundo en incluir funcionalidades que permiten hacer pruebas de cambios en aplicaciones simulando las cargas reales generadas por los usuarios en los entornos de producción. Con su gran arsenal de nuevas y mejores características,

[25]**Sistema Gestor de Base de Datos Relacional**: acrónimo de *Relational Data Base Management System.*

ofrece soluciones concretas a las demandas de los usuarios respecto a las bases de datos actuales y de próxima generación.(35)

Incorpora un nuevo compilador Java que facilita un alto rendimiento en la ejecución de procedimientos almacenados en Java sin necesidad de utilizar un compilador de terceros. Introduce un conjunto de funcionalidades en la línea de la auto-administración de la base de datos. Estas funcionalidades incluyen la generación automática de sentencias SQL, asistentes para reducir las tareas de particionamiento y ciclo de vida de los datos que proporciona una interfaz simple para visualizar incidencias en la base de datos.

Como características básicas tiene:

- ✓ Soporte: se encuentra disponible sobre varias plataformas (*Windows*, *Linux*, *Unix*).
- ✓ Fiabilidad: continúa disponibilidad de las aplicaciones y datos.
- ✓ Escalabilidad: condición que tienen los sistemas de adaptarse al crecimiento de trabajo de manera fluida y hacerse más grande sin perder calidad en los servicios ofrecidos.
- ✓ Seguridad y protección de datos: características de seguridad que permite compartir la red de recursos de una empresa con la confianza de que la privacidad se mantiene.
- ✓ Auto-gestión: *Oracle* automatiza muchas de las funciones de infraestructura de modo que un solo administrador puede administrar cientos de servidores.

El SUIN utiliza para la gestión de los datos el sistema Oracle 11g,debido a que el sistema implementar es para el SUIN se utilizará el gestor de base de datos antes mencionado, además de las características previamente explicadas.

Visual Studio 2010

Microsoft Visual Studio 2010 incluye potentes herramientas que simplifican todo el proceso de desarrollo de aplicaciones. Los equipos pueden observar una mayor productividad y ahorro de costes al utilizar características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que le ayudarán a crear siempre un código de gran calidad.(36)

Esta herramienta está acompañada por .NET *Framework* 4.0. Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo obteniendo un sistema completo formado por la plataforma de servicios.NET, los compiladores de varios lenguajes, editores y diseñadores específicos para cada tipo de tarea, análisis de código, de modelado de sistemas y aplicaciones, de desarrollo y ejecución de pruebas, administración de bases de datos, entre otros.

Microsoft Visual Studio 2010 ofrece muchas funcionalidades importantes para el desarrollo de software entre las que se pueden citar, utilizar el *Framework* .NET 4.0 y poder programar para las versiones anteriores (2.0, 3.0, 3.5); conjunción con *XAML*[26] (*Extensible Applications Markup Language* por sus siglas en inglés); un diseñador para *Windows Presentation Foundation* y *Workflow Foundation* que son parte del *Framework* .NET 4.0 *IntelliSense* para *JavaScript*; el nuevo Lenguaje *LINQ*[27]siendo este un agregado a los lenguajes *Visual Basic* y *Visual C#* para la realización de consultas *SQL*.

Microsoft Visual Studio Team System 2010

*Microsoft Visual Studio TeamSystem*2010 mantiene los propósitos de *Microsoft* de facilitar a los desarrolladores y equipos de desarrollo la rápida creación de aplicaciones interconectadas, con experiencias de usuario atractivas para *Windows Vista*, *Office System* 2010, dispositivos móviles e Internet. Presenta una versión renovada del servidor *Team Foundation Server* el cual ofrece un gestor de pruebas mejorado y nuevas herramientas para el servidor. Incluye elementos nuevos como *Lab Management*, que permite crear y gestionar entornos virtuales. Brinda fabulosas herramientas como *Expression* y *SharePoint* presentes en versiones anteriores pero que ahora son más maniobrables y con muchas más novedades. Además el entorno de trabajo está desarrollado en WPF, se incluye el soporte para varios monitores y presenta muy buena compatibilidad con el código existente de versiones anteriores.(37)

Como entorno integrado de desarrollo se utilizará Visual Studio 2010 porque permite a los desarrolladores crear aplicaciones de alta calidad con gran rapidez, más seguras, confiables y administrables en cualquier entorno que soporte la plataforma .NET.

Herramienta de modelado de base de datos: Embarcadero ER/Studio

Embarcadero ER/Studio 8.0 es una herramienta de base de datos que le ayuda a diseñar, generar y mantener aplicaciones de base de datos de calidad y alto rendimiento, desde un modelo lógico de sus requerimientos de información y reglas del negocio que definen su base de datos, hasta un modelo físico optimizado por las características específicas de su base de datos de destino.(38) Permite visualizar la estructura adecuada, los elementos clave y un diseño optimizado de su base de datos. Incorpora nuevas capacidades colaborativas de modelado y administración, así como mayor soporte en la integración de almacenamiento de datos diseñados para visualizar, documentar y compartir el conocimiento.

[26]**Lenguaje extensible de formato para aplicaciones**: acrónimo de *Extensible Applications Markup.*

[27]**Consultas integradas en el lenguaje**: acrónimo de *LanguageIntegratedQuery.*

Para el diseño y modelado de bases de datos físicas incluye capacidades de planeación de modelado seguros. Permite la creación y consolidación de modelos de proyecto en un modelo global. Para soportar la transición de lógico a físico, provee a los administradores de base de datos dos nuevas características: capacidad de planificación y modelado seguro. La capacidad de planificación permite a los diseñadores comunicar la forma en que esperan que la base de datos crezca, y la funcionalidad de modelado seguro permite a los administradores de la información desarrollar roles de acceso en el modelo lógico, que pueden ser transferidos al modelo físico. Además ayuda a las empresas a descubrir, documentar y re-utilizar los activos de datos, ofrece además:

- ✓ Entorno de diseño dirigido por el modelo
- ✓ Soporte al ciclo de vida completo de las bases de datos
- ✓ Gestión de modelos empresarial
- ✓ Capacidades de comunicación empresariales
- ✓ Almacén de datos y soporte a la integración
- ✓ Diseños de calidad de bases de datos
- ✓ Documenta y mejoralas base de datos existentes.

Herramienta de modelado: AltovaUModel2009

"*UModel* 2009 incluye las más altas funcionalidades para potenciar a los usuarios con las más completas ventajas del desarrollo de software UML".(39)

Dentro de las características que posee se encuentran, el soporte para los 14 tipos de diagramas UML, modelado de esquemas *XML* en diagramas UML, diagramas de procesos de negocio, ingeniería inversa de código fuente y ficheros binarios Java, C# y VB.NET, Sincronizado de modelo y código a través de ingeniería de ida y vuelta, generación de documentación personalizable de proyecto, integración con sistemas de control de versiones además de una estrecha integración con *Visual Studio y Eclipse*.

Se utilizó Altova UModel 2009 para el modelado del diagrama de clases, modelo de proceso actual y proceso mejorado, diagrama de despliegue y componentes.

PLSQL

PL/SQL[28]permite definir procedimientos utilizando sentencias *SQL*, su estructura es similar a la de cualquier otro lenguaje con procedimientos como C o C++. Provee una manera muy cómoda de relacionar los conceptos de bases de datos y manejarlos mediante ciertas estructuras de control, dentro del contexto de una herramienta de programación. Su utilización es dentro del administrador de

[28]**PL/SQL**: acrónimo de *Procedural Language/Structured Query Language.*

bases de datos Oracle y sus principales características son la posibilidad que brinda de utilizar sentencias *SQL* para manipular datos en Oracle y sentencias de control de flujo para organizar esta manipulación de datos.(40)Dentro del lenguaje, es posible declarar constantes y variables, definir procedimientos y funciones y atrapar errores en tiempo de ejecución. Así visto, PL/SQL combina la el poder de la manipulación de datos, con *SQL*, y las facilidades del procesamiento de los mismos, tal como en los más modernos lenguajes de programación.

Dentro de los beneficios que brinda este lenguaje se destacan los siguientes:

- ✓ Permite la integración de procedimientos construidos con *SQL*.
- ✓ Permite especificar no sólo que hacer con un comando *SQL*, sino cómo hacerlo, dando un mejor control para su ejecución.
- ✓ Mejora la performance, ya que a través de *PLSQL* se pueden unir varias sentencias *SQL* en un solo programa, por lo tanto se envía un solo bloque a la base de datos en lugar de enviar una sentencia *SQL* a la vez.
- ✓ Portable: *PLSQL* corre en cualquier servidor Oracle.
- ✓ Manejo de excepciones, permite manejar eficientemente las excepciones.

1.6 Navegadores

Mozilla Firefox

Es un navegador de Internet, libre, de código abierto y con gran potencia, con interfaz gráfica de usuario, desarrollado por la Corporación *Mozilla* y un gran número de voluntarios externos. Cualquier usuario corriente de PC puede usar el navegador sin tener la necesidad de cambiar su sistema operativo. *Firefox*, es multiplataforma y está disponible en versiones para *Microsoft Windows*, *Mac OS X* y *GNU/Linux.*(41)

Es recomendable utilizar *Mozilla Firefox* para visualizar el sistema a realizar, ya que se encuentra disponible para distintas plataformas, utilizando las mismas características en cada una de ellas.

Internet Explorer

Internet Explorer o IE es un navegador web desarrollado por *Microsoft*. Funciona en el sistema operativo *Windows*. La popularidad de *Internet Explorer* se debe en gran parte, a que es el navegador oficial de *Windows* por lo que viene integrado en su sistema operativo. Permite la búsqueda de cualquier cosa a través de su navegador.(42) Posee un cómodo sistema de pestañas que permite tener muchas páginas en una misma ventana, las pestañas se pueden mover horizontalmente para organizar mejor la navegación. También memoriza las páginas más visitadas para hacer más fácil la búsqueda y sugiere páginas a medida que se anotan en la barra de direcciones.

Otra característica interesante es que *Internet Explorer* reduce automáticamente las páginas que se desean imprimir. Así, no se pierde el contenido por problemas de espacio. Además, en las opciones de impresión se pueden usar márgenes ajustables, diseños para personalizar, prescindir de encabezados y pies de páginas innecesarios o también imprimir sólo lo que se seleccione.

Chrome

Google Chrome es un navegador web desarrollado por *Google* y compilado con base en componentes de código abierto. Es el tercer navegador más utilizado de Internet.(43)Esta nueva herramienta simplifica los pasos para visitar las páginas, sitios online, y acelera los procesos de búsqueda, los cuales son bastiones indispensables para la utilización de la web 2.0, con base en las nuevas plataformas y aplicaciones más utilizadas en la red, *Chrome* se concibió como un facilitador para el usuario, y no sólo una interfaz entre la red y el cibernauta. Dentro de sus ventajas se destacan la barra multidireccional, buscador libre de 'marcas', plataformas independientes y más rápido, más fuerte, más libre.

Conclusiones

Se logró un mayor dominio sobre el negocio y las funcionalidades del sistema a partir del estudio de los conceptos asociados.

El estudio de los diferentes sistemas de gestión de reglas de negocio evidenció la necesidad de realizar un sistema que permitiera la aplicación de cambios continuos a las reglas de negocio sin tener que modificar el código.

El análisis realizado a las diferentes herramientas, lenguajes y tecnologías permitió profundizar los conocimientos necesarios para el desarrollo del sistema de gestión de reglas de negocio, y ayudó a determinar según las necesidades de la solución el ambiente de desarrollo.

Capítulo 2: Propuesta de solución

A partir de la selección realizada en el capítulo anterior se presenta la aplicación de la metodología, tecnologías y herramientas para el diseño de la arquitectura. En el presente capítulo se establecen las metas y restricciones que se deben cumplir para un mejor funcionamiento del sistema, así como la descripción del diseño de la arquitectura.

2.1 Modelo de negocio

Modelo de negocio actual

En el SUIN las reglas de negocio se realizan a mano por el programador en el código fuente. Ante un cambio en alguna de las reglas los clientes del negocio avisan a los programadores del cambio, estos tienen que realizar este cambio en el código fuente del programa. Seguidamente tienen que compilar la aplicación, para después realizarle las pruebas con el nuevo cambio. Después generan una nueva versión de la aplicación para poder publicar la aplicación en los servidores del cliente. Esto presenta una gran deficiencia ya que implica un gasto de tiempo adicional, lo que provoca un atraso en el cronograma de entrega del producto.

Modelo del proceso mejorado

Una estrategia para resolver los problemas que existen actualmente en el SUIN, es separar las reglas de negocio del código, emplear los motores de reglas de negocio y herramientas que permitan su mantenimiento. Estos motores de reglas realizan la evaluación de las mismas en cierto contexto. El programador ahora solo tiene que declarar en el código fuente del programa el nombre de la regla, para que el motor de regla se encargue de su evaluación y ejecución. Ante un cambio en alguna de las reglas los clientes del negocio pueden modificar la regla a través de una interfaz gráfica web, y este cambio se pondría en funcionamiento en el momento que este termine de realizar el cambio, por lo cual no depende del equipo de desarrollo para realizar las modificaciones en la regla de negocio. En este proceso la demora ante un cambio es bastante rápido ya que no implica tener que recompilar la aplicación y solo se tiene que probar la regla modificada. A continuación se muestra la vista global del sistema en la figura 1,2 y 3.

Vista global del sistema

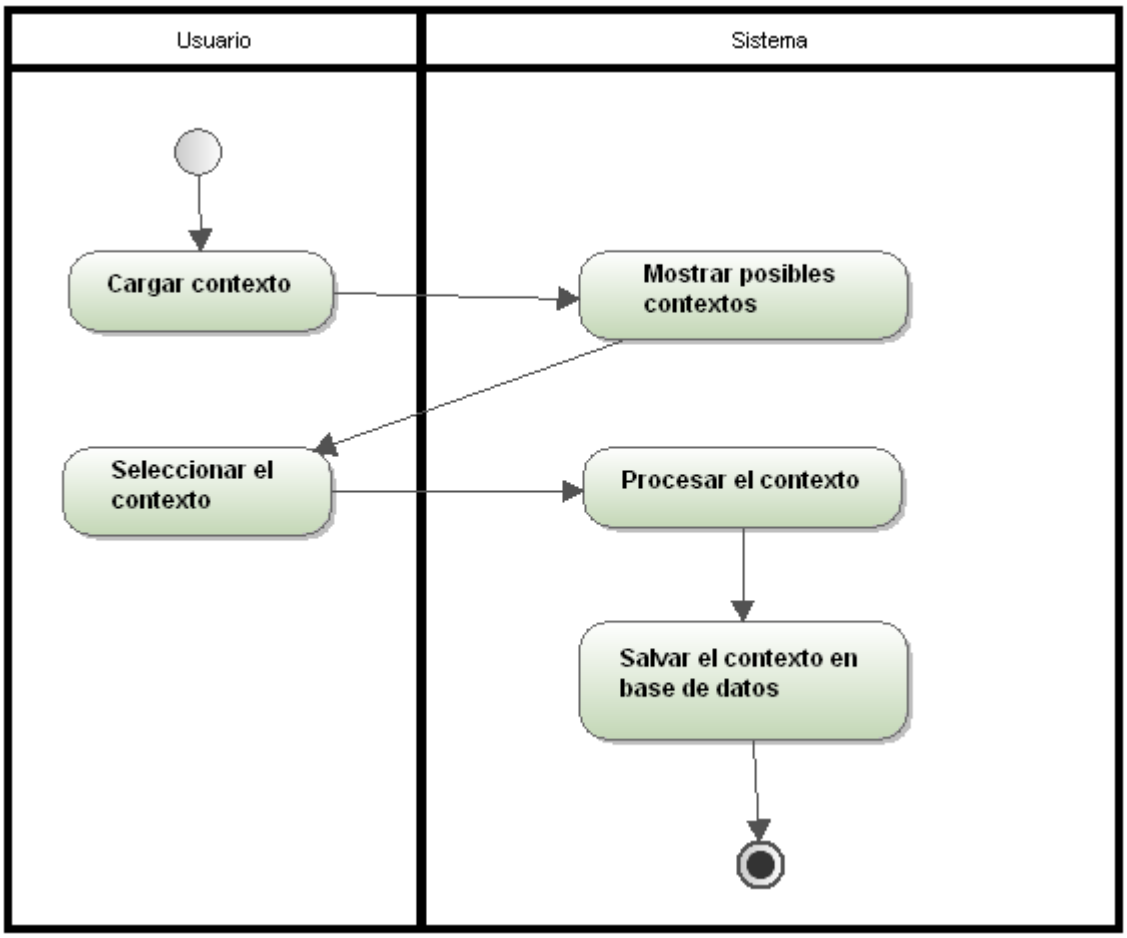


Figura 1 Vista global del sistema del proceso cargar contexto.

- Cargar contexto: es la actividad de seleccionar un contexto.
- Mostrar posibles contextos: se refiere a mostrar un listado con los contextos.
- Seleccionar el contexto: es la acción que realiza el usuario después de haber encontrado en el directorio el contexto deseado.
- Procesar el contexto: es la acción que realiza el sistema luego del usuario haber cargado dicho contexto.
- Salvar el contexto en base de datos: es la acción de guardar en la base de datos el contexto.

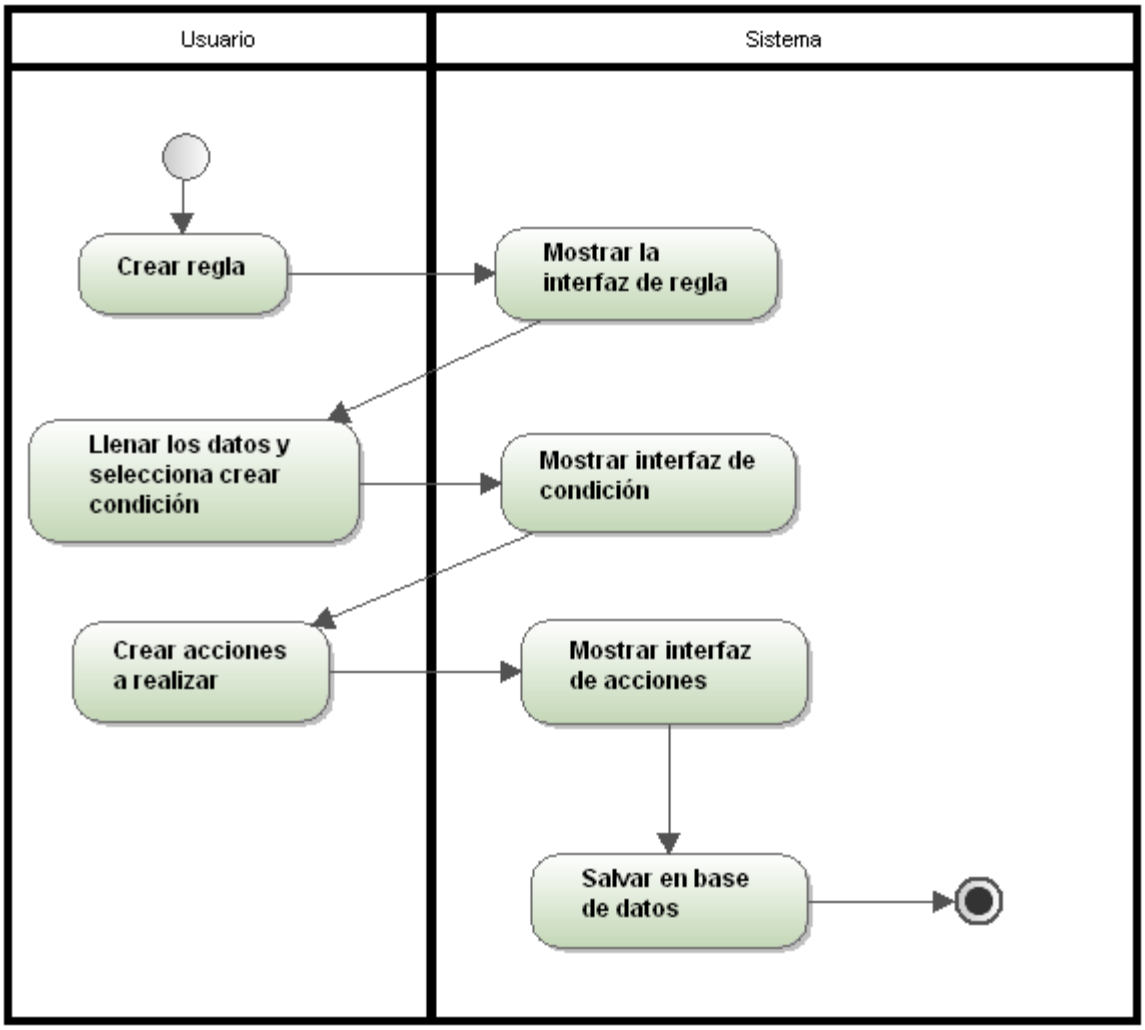


Figura 2 Vista global del sistema del proceso crear regla.

- Crear regla: es el encargado de crear una regla.
- Mostrar la interfaz de regla: es la actividad de mostrar la interfaz de una regla.
- Llenar los datos y selecciona crear condición: es la acción que realiza el usuario para crear una nueva regla.
- Mostrar interfaz de condición: es la actividad de mostrar la interfaz de una condición.
- Crear acciones a realizar: es cuando el usuario selecciona crear acción.
- Mostrar interfaz de acciones: es la acción de mostrar la interfaz de las acciones.
- Salvar en base de datos: es la acción de guardar en la base de datos la regla creada.

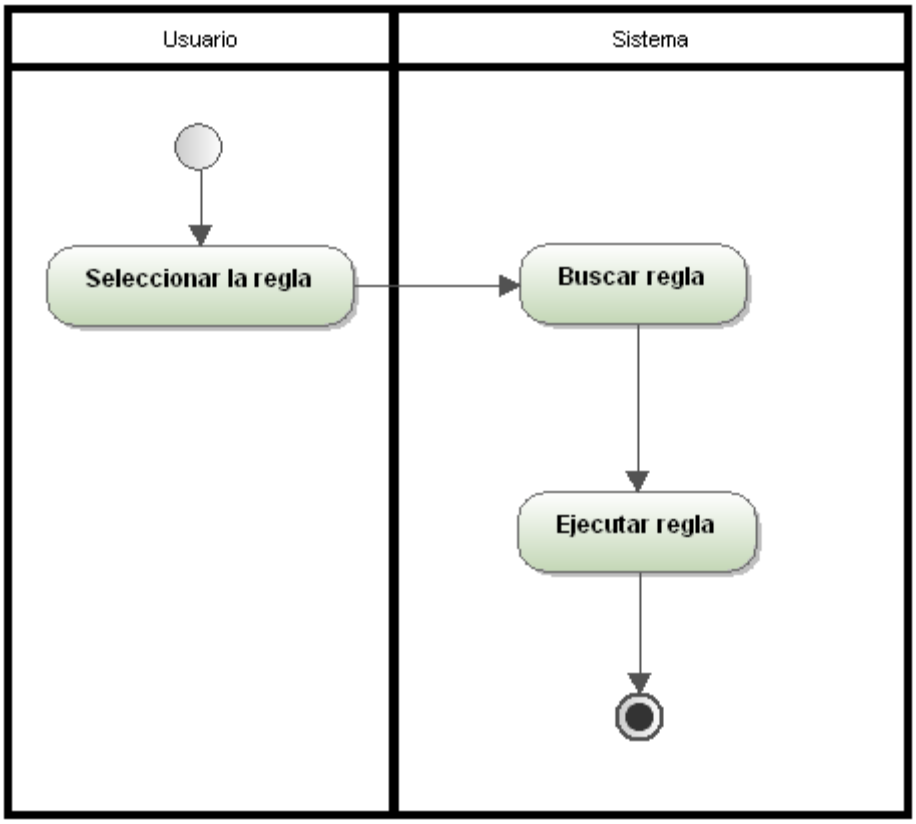


Figura 3 Vista global del sistema del proceso ejecutar regla.

- Seleccionar la regla: el la usuario es el encargado de seleccionar regla que desea ejecutar.
- Buscar regla: es la actividad de buscar la regla en la base de datos.
- Ejecutar regla: es la acción que se realiza después de haber obtenido la regla de la base de datos evaluando y ejecutando las acciones.

Descripción de roles

Tabla 1 Descripción de roles.

Rol	Objetivo
Usuario	Tiene los permisos para crear, modificar y eliminar reglas.
Administrador	Tiene todos los permisos incluyendo la eliminación de contexto.

2.2 Especificación de requisitos de software.

Catálogo de requisitos

El objetivo de la especificación de requisitos es definir de forma precisa y completa las funcionalidades y restricciones del sistema que se desea. A continuación se expone un resumen del catálogo de los requisitos funcionales del sistema.

Tabla 2 Catálogo de requisitos.

Requerimiento	Nombre	
RF1	Gestionar fórmula	• Adicionar fórmula
		• Mostrar fórmula
		• Eliminar fórmula
RF2	Gestionar contexto	• Cargar contexto
		• Mostrar contexto
		• Eliminar contexto
RF3	Gestionar condición	• Adicionar condición
		• Mostrar condición
		• Modificar condición
RF4	Gestionar acción	
RF5	Gestionar regla	
RF6	Evaluar regla	
RF7	Ejecutar regla	

Descripción de los requisitos funcionales

El glosario de términos de ingeniería de software de IEEE[29]define un requerimiento o requisito como:

- ✓ Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- ✓ Condición o capacidad que tienen que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- ✓ Una representación documentada de una condición o capacidad.(44)

Los requisitos pueden ser clasificados en funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir y no alteran la funcionalidad del sistema, esto quiere decir que los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

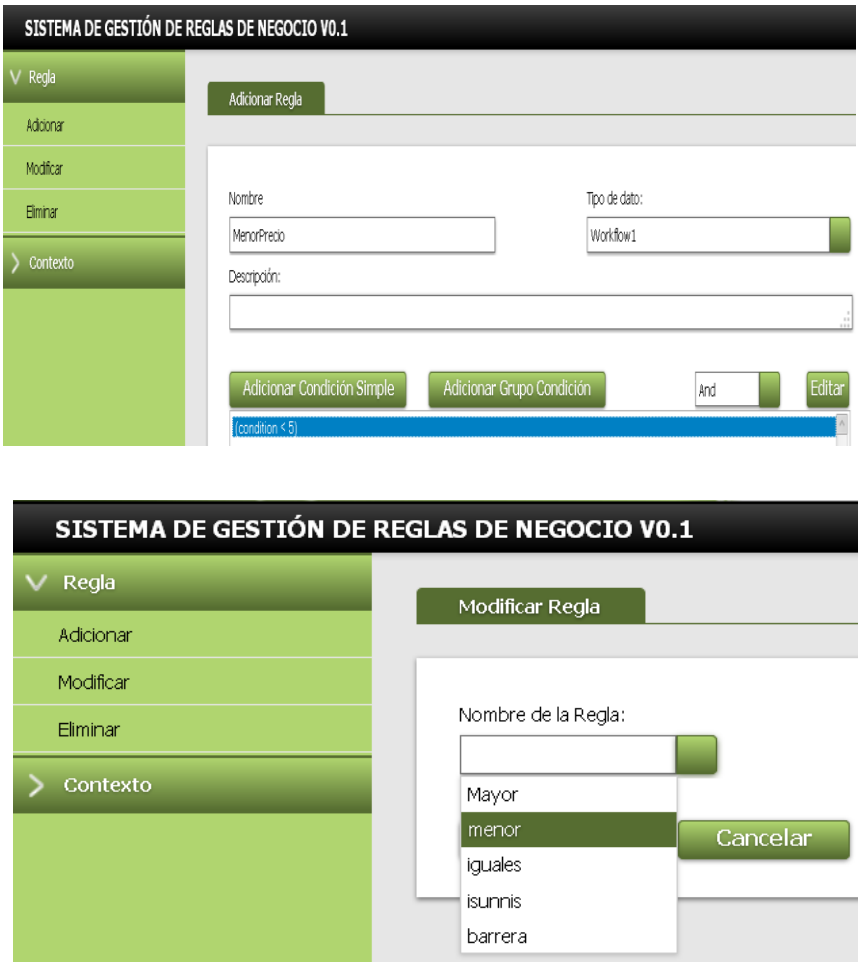
A continuación se presenta la descripción del requisito funcional Gestionar regla en la tabla 3.Para la descripción del resto ver Anexo 1 Descripción de requisitos funcionales.

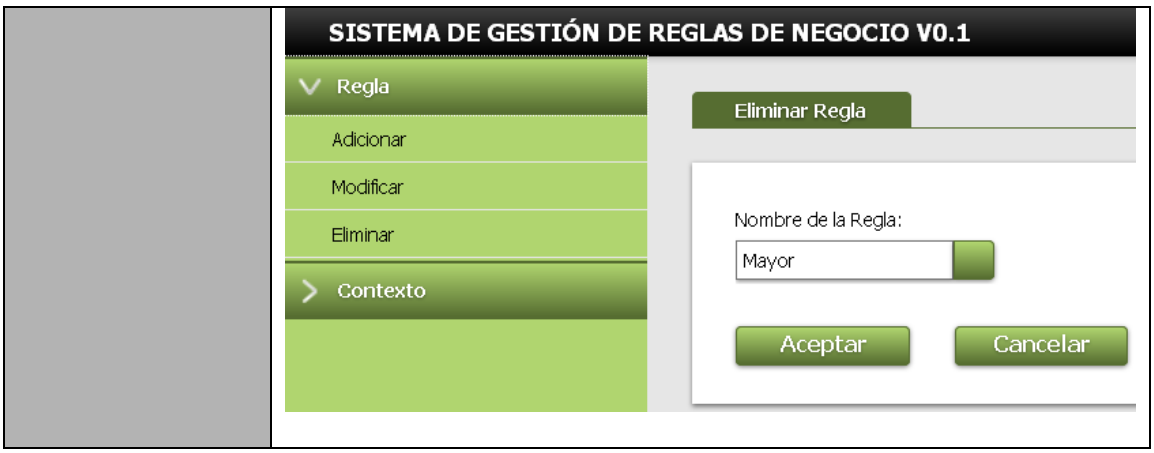
[29]**Instituto de ingenieros eléctricos y electrónicos**: acrónimo de *Institute of Electrical and Electronics Engineers*.

RF 5 Gestionar regla

Tabla 3 Descripción del requisito funcional Gestionar regla.

Propósito	Permite gestionar regla.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	a. Regla b. Contexto c. Condición d. Acción	a. Nombre b. Contexto c. Descripción d. Condición e. Lista de acciones (si es verdadera) f. Lista de acciones (si es falsa)
Descripción	5.1 Adicionar regla 5.1.1 Seleccionar la opción adicionar regla. 5.1.2 Mostrar una interfaz donde se muestran los campos necesarios. a. Nombre b. Contexto c. Descripción d. Condición e. Lista de acciones (si es verdadera) f. Lista de acciones (si es falsa) 5.1.3 Seleccionar el botón aceptar 5.2. Modificar regla 5.2.1 Seleccionar la opción modificar regla. 5.2.2 Seleccionar la regla que desea modificar. 5.2.3 Mostrar una interfaz con los parámetros que contiene la regla. 5.2.4 Modificar los valores deseados. 5.2.5 Seleccionar la opción aceptar.	

	5.3. Mostrar regla 5.3.1 Realizar los pasos del 5.2.1 al 5.2.3 de modificar regla. 5.3.2 Seleccionar la opción aceptar. 5.4. Eliminar regla 5.4.1 Seleccionar la opción eliminar regla. 5.4.2Seleccionar la regla. 5.4.3 Seleccionar la opción eliminar regla. 5.4.4Mostrar una ventana de confirmación. 5.4.5 Seleccionar la opción “Aceptar”.
Validaciones	El campo nombre, contexto y condición no pueden ser nulos.
Postcondiciones	Se guarda la regla en base de datos.
Prototipo	



Modelo conceptual

Un modelo conceptual es una representación de los conceptos significativos en un dominio del problema. Contiene las entidades fundamentales definidas atendiendo a su importancia dentro del proceso a modelar. Además permite identificar a partir de los requisitos funcionales los conceptos fundamentales, relacionarlos y dar una visión de las futuras entidades del sistema con sus atributos y relaciones. En la figura 4 se muestra el modelo antes mencionado.

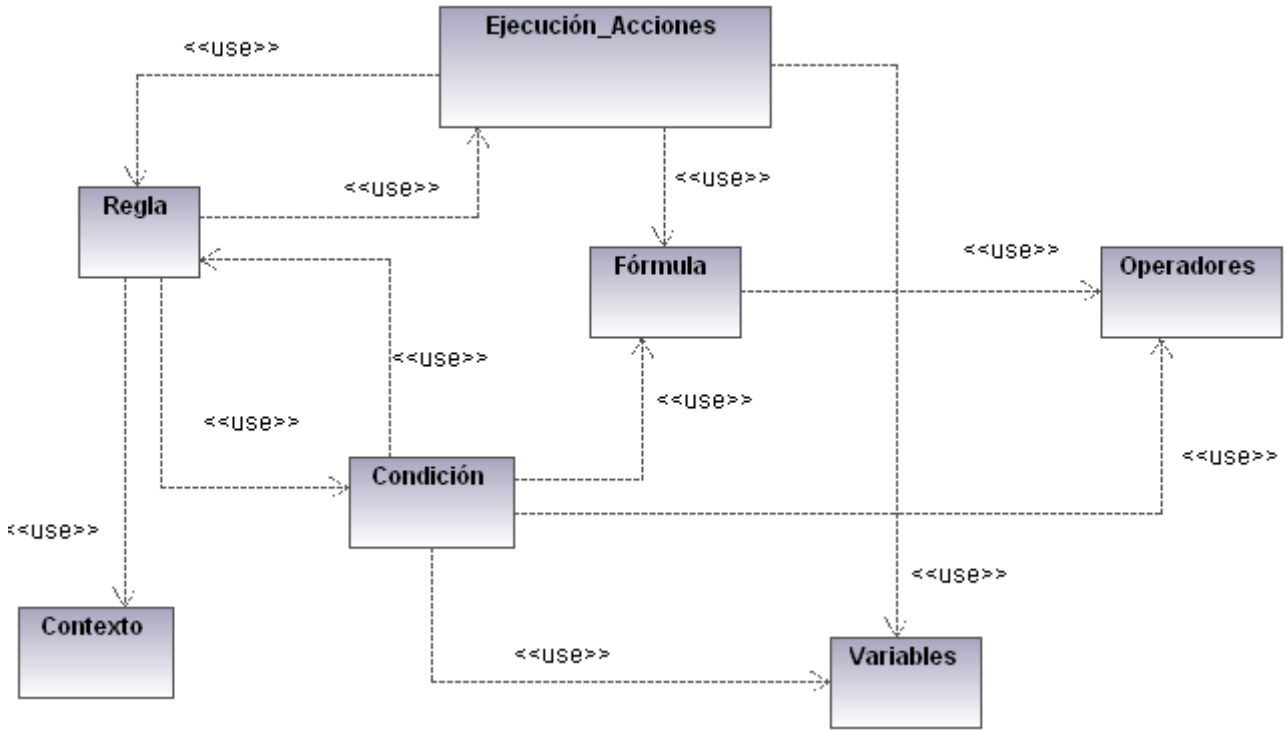


Figura 4 Modelo conceptual.

Especificación de requisitos no funcionales

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar.

Usabilidad

RnF.1 El sistema podrá ser utilizado por cualquier usuario con las siguientes características:

- ✓ Conocimientos básicos relativos al uso de una computadora.
- ✓ Conocimientos básicos del sistema operativo Windows.
- ✓ Conocimientos sólidos relativos a los procesos de negocio de acorde al rol que desempeñe.

RnF.2 El sistema será distribuido en idioma español, aunque debe contar con soporte multilenguaje.

Restricciones de diseño

RnF.3 El sistema debe implementarse usando el lenguaje C#, sobre la plataforma ASP.NET.

RnF.4 El gestor de base de datos, será Oracle Enterprise Edition11gR2.

RnF.5 El sistema debe desarrollarse usando el IDE Visual Studio 2010.

Soporte:

RnF.6 El sistema debe propiciar su mejoramiento y la incorporación de otras opciones en el futuro.

2.3Arquitectura de la solución

La arquitectura de un software es determinante para su éxito pues se encarga de establecer los fundamentos para que los miembros del equipo de desarrollo en sus diferentes roles se guíen por una línea de trabajo común y así alcanzar los objetivos de manera exitosa.

El sistema está basado en la arquitectura n-capas la cual brinda escalabilidad, flexibilidad y mejoras en las posibilidades de mantenimiento, debido a que cada capa es independiente de la otra, los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo. En su vista más abstracta se define una arquitectura en 4 capas como se muestra en la figura 5, la arquitectura propuesta y la descripción de las cuatro capas que lo componen.

Capa de Presentación: está regida por el componente *BRM*, que se comunica con la capa de negocio a través servicios y el protocolo http. Además en esta capa se incluyen las interfaces con las cuales el usuario interactuará para enviar y solicitar datos, que se encuentran estandarizadas mediante el componente UIToolsBox.

Capa de negocio: se denomina capa de negocio o lógica del negocio porque es aquí donde se encuentran todas las funcionalidades que dan respuesta a los requisitos del sistema. Estará presente

el componente *Rule Engine* que guía el flujo de procesos en la aplicación. Además se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de acceso a datos con el objetivo de intercambiar con el gestor de base de datos para almacenar o recuperar datos de él.

Capa de acceso a datos: esta capa se utiliza con el objetivo de estandarizar el acceso a datos, y eliminar el acoplamiento de los componentes de negocio al gestor de base de datos, a través de los conectores. Es la única capacitada para interactuar con la capa de base de datos a través del protocolo TNS y a su vez expone servicios a la capa de negocio.

Capa de base de datos: está constituida por todo el conjunto de tablas y procedimientos que permiten el almacenamiento de la información recolectada y procesada en un gestor de base de datos Oracle. La única capa encargada de interactuar con ésta es la capa de acceso a datos.

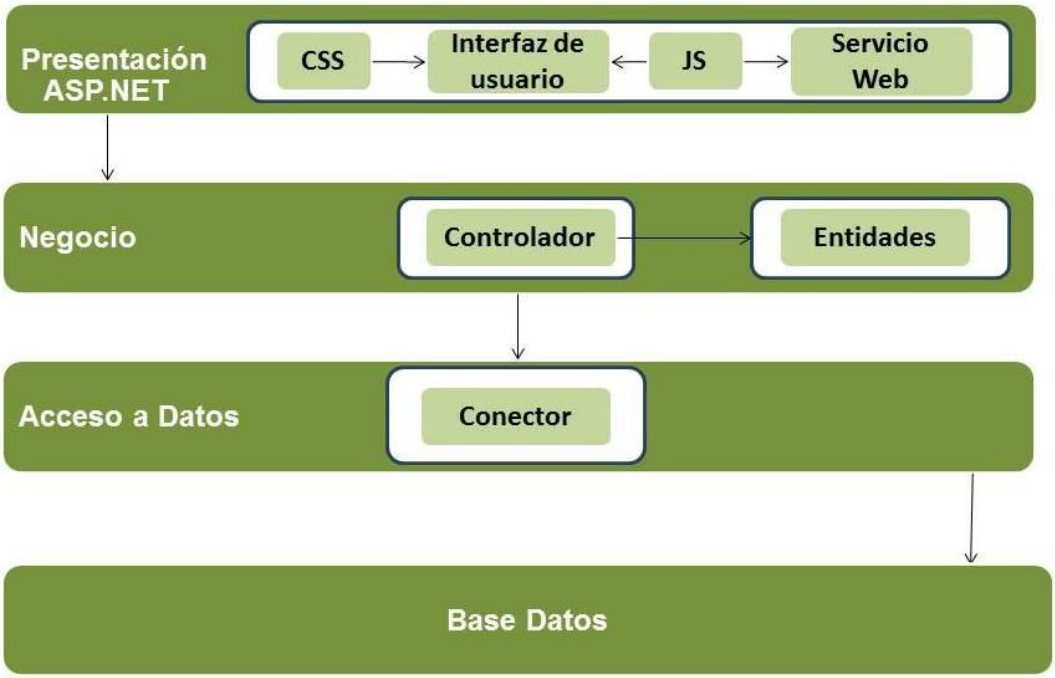


Figura 5 Vista lógica de la arquitectura del software.

2.4Patrones de diseño

Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles. Un patrón de diseño es también:

- ✓ Una solución estándar para un problema común de programación.

- ✓ Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- ✓ Un proyecto o estructura de implementación que logra una finalidad determinada.
- ✓ Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- ✓ La forma de un diagrama de objeto o de un modelo de objeto.

Para el desarrollo se han tenido en cuenta varios patrones que permiten la flexibilidad y no constituyen grandes cambios en el rendimiento del mismo, los cuales se describen a continuación:

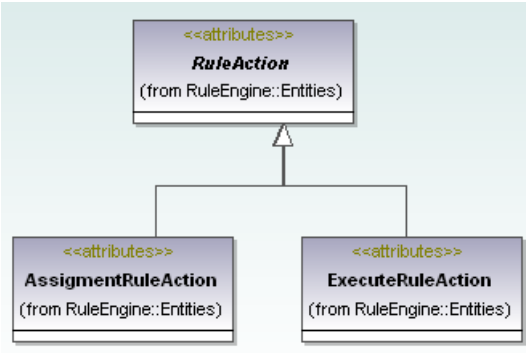
Encapsulación: sugiere esconder algunos componentes, permitiendo sólo accesos estilizados al objeto. Se hace uso de este patrón en casi todas las clases que componen al sistema, permitiendo que éstas solo posean como elementos públicos aquellos que son exclusivamente necesarios.(45) Ejemplo:

```
[PathExcludeProperty]
public Guid IdRule { get; set; }

[PathExcludeProperty]
public string RuleName { get; set; }
```

Fábrica: provee una interfaz para crear familias de objetos relacionados o dependientes sin especificar los tipos concretos de clases. Su uso se encuentra centrado a la creación de los conectores correspondientes al acceso a datos que se esté utilizando, así como en la obtención de los servicios a utilizar.(46)

Subclases: propone heredar miembros por defecto de una superclase, seleccionando la implementación correcta a través de resoluciones sobre qué implementación debe ser ejecutada. Se puede encontrar este patrón con más fuerza en las entidades de negocio, por su conceptualización las funciones y la información que almacenan pueden estar diferenciadas en cierta medida. Ejemplo:



Excepciones: indica introducir estructuras de lenguaje para interceptar y arrojar excepciones. Se identificaron los diferentes tipos de errores a tratar dentro del sistema creando una clase que permite identificar cada tipo de error en el momento de ejecución.

```
[Serializable]
public class RuleException :Exception
{
    public RuleException()
        : base("message")
    {
    }

    public RuleException(string message)
        : base(message)
    {
    }
}
```

2.5 Especificación de clases

Una especificación describe, a nivel abstracto, el comportamiento de un método (procedimiento) o clase. Las especificaciones de clase describen una visión abstracta de los contenidos y del comportamiento de una clase. (47)

Para la propuesta de solución se definieron diferentes tipos de clases: clases entidades, clases conectoras, clases abstractas e interfaces.

Clase entidad: se refieren a las tablas de la Base de Datos, o sea son aquellas donde se almacenan información, son las encargadas de modelar la información y manejan actividades específicas.

Tabla 4 Descripción de la clase entidad BusinessRule.

Nombre de la entidad	*BusinessRule*	
Descripción	Entidad que representa los datos de la regla.	
Nombre del atributo	**Tipo**	**Descripción**
RuleName	*string*	Nombre de la regla.
Condition	*condition*	Condición de la regla.
ContextDescription	*RuleContext*	Tipo de dato de la regla.
ActionIf	*List<RuleAction>*	Acciones si se cumple la condición.
ActionElse	*List<RuleAction>*	Acciones si no se cumple la condición.
Description	*string*	Descripción de la regla.

Clase conectora: son las que manejan directamente el acceso a la base de datos, cuentan con la estructura de la lógica de acceso a datos permitiendo una independencia total del gestor de base de datos a utilizar.

Tabla 5 Descripción de la clase conectora OracleRuleManagerConnector.

Nombre	*OracleRuleManagerConnector*
Descripción	Clase que permite la conexión del acceso a datos con el negocio.
Métodos	**Descripción**
SaveRule	Método que salva una regla.
LoadRule	Método que devuelve una regla a partir de su nombre.
GetAllRule	Método que devuelve en un diccionario todas las reglas y su descripción.
UpdateRule	Método que actualiza una regla.
DeleteRule	Método que elimina una regla a partir de su nombre.
SaveContext	Método que salva un tipo de datos.
LoadContext	Método que devuelve un tipo de datos a partir de su nombre.

Clase abstracta: es aquella que forzosamente se ha de derivar si se desea que se puedan crear objetos de la misma. Representan los escalones más elevados de algunas jerarquías de clases, suelen ser utilizadas en aquellos casos en que se quiere que una serie de clases mantengan una cierta característica o interfaz común.

Tabla 6 Descripción de la clase conectora Condition.

Nombre	*Condition*	
Descripción	Clase que permite la creación de condiciones para la regla.	
Nombre del atributo	**Tipo**	**Descripción**
LeftMember	*Member*	Miembro derecho de la condición.
RightMember	*Member*	Miembro izquierdo de la condición.
Operator	*CondOperator*	Operador de la condición.

2.6Diagrama de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema.(48)

En las figuras6 y 7se muestra un fragmento del diagrama de clases. De la clase *Condition* heredan dos clases, en esta clase se almacena la información básica de una condiciones el caso de *BusinessRule* almacena información de la regla de negocio y *Formule* almacena la información de la fórmula. Para una mejor comprensión de lo antes expuesto se recomienda analizar el anexo 2 donde se encuentra el resto del diagrama.

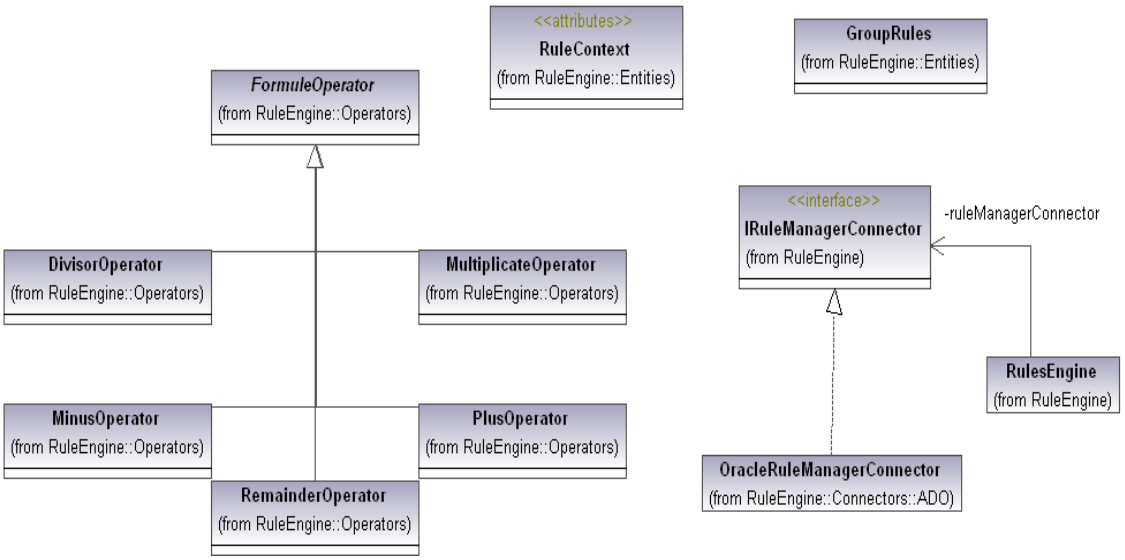


Figura 6 Diagrama de clases del diseño (I).

2.7 Modelo de datos

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos, las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos. (49)

En la figura 8 se muestra el modelo de datos de las clases persistentes que almacenan los datos de las reglas de negocio. Este diagrama de base de datos está compuesto por las entidades que permitirán almacenar los datos comunes y específicos de una regla de negocio.

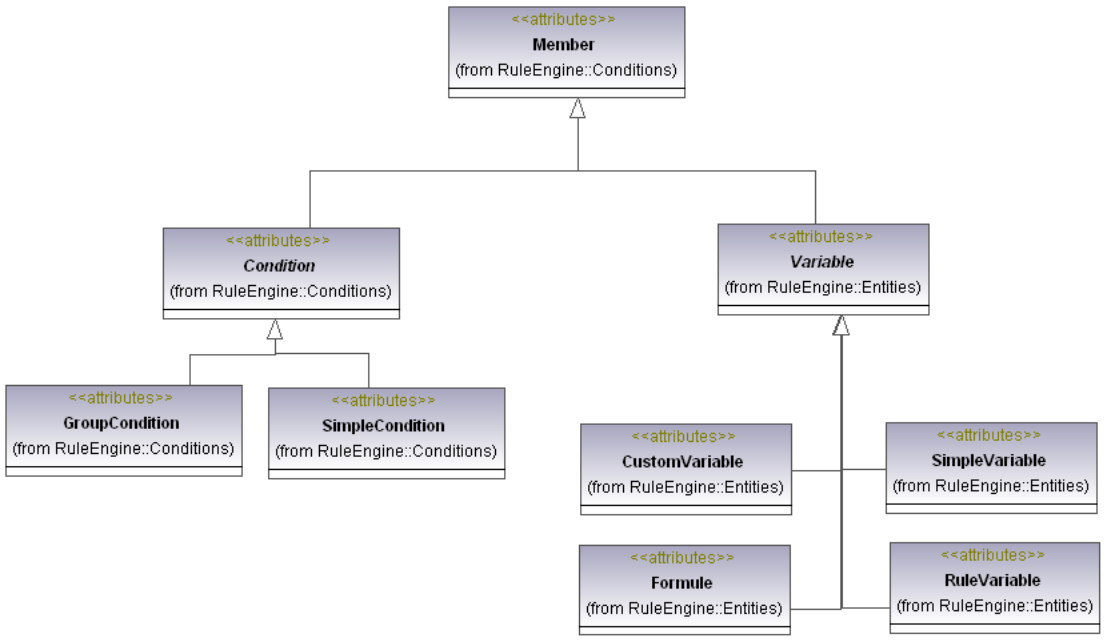


Figura 7 Diagrama de clases del diseño (II).

Figura 8 Modelo de datos: vista lógica

Conclusiones

Con la representación del proceso mejorado se logró un mejor entendimiento de los principales conceptos tratados en el desarrollo del sistema.

La arquitectura definida permite una mayor flexibilidad entre los componentes que interactúan en el sistema, garantizando un alto grado de genericidad.

Los patrones de diseño permiten lograr un mayor entendimiento en los diagramas de clases.

La definición del modelo de datos permitió definir las estructuras y restricciones necesarias para salvar la información necesaria en la base de datos.

Capítulo 3: Implementación y validación

En el presente capítulo se describen primeramente algunos estándares de codificación que permiten la creación de un código fácil de entender y mantener. Se exponen los diagramas de componentes, despliegue e interfaces correspondientes al sistema. Se describen las pruebas realizadas tanto unitarias como las de sistema y los resultados que arrojaron las mismas.

3.1 Estándares de codificación

Los estándares de codificación conducen a una mayor coherencia del código, esto a su vez permite generar código más fácil de entender, facilitará su desarrollo y mantenimiento, además reduce el costo total de las aplicaciones. Un estándar de codificación completo comprende todos los aspectos de la generación de código. Con el objetivo de facilitar el mantenimiento del software se emplearon los siguientes estándares de codificación:

Pascal: la primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Puede utilizar los identificadores en caso de tres o más caracteres.

- ✓ Los nombres de las clases se escribieron en mayúsculas, siguiendo el estilo Pascal por ejemplo BusinessRule.

Camell: la primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra concatenada es mayúscula.

- ✓ Los nombres de los atributos con estilo Camell. Por ejemplo ruleName.

Mayúscula

Todas las letras en el identificador se capitalizan. Esta convención se utilizará sólo para los identificadores que constan de dos o menos letras. Por ejemplo:

- ✓ System.IO
- ✓ System.Web.UI

Sensibilidad a mayúsculas: Para evitar confusiones y garantizar la interoperabilidad entre lenguajes, se siguieron las siguientes reglas sobre el uso de mayúsculas y minúsculas:

- ✓ No se deberá utilizar nombres o identificadores que requieran ser entre mayúsculas y minúsculas.
- ✓ No se deberá crear dos espacios de nombre que se diferencien solo en el uso de las mayúsculas.
- ✓ No crear funciones con nombres de parámetros que se diferencian solo en el uso de la mayúscula.

- ✓ No se deberá crear espacios de nombre para identificarlas clases que se diferencien solo en el uso de las mayúsculas.
- ✓ No crear clases con propiedades que se diferencien solo en el uso de las mayúsculas. No crear clases con métodos que se diferencien solo en el uso de las mayúsculas.

Para nombrar las clases no se recomienda usar los nombres de los espacios de nombres y otras clases comúnmente usadas, por ejemplo *System*, *Collections* o *Forms*. La utilización de nombres significativos para las clases, que expresen su significado y la declaración de una clase con la letra I al inicio indica que es una clase interfaz. Por ejemplo IRuleManagerConnector. Se definieron además otras reglas las cuales están definidas en el Anexo 3.

3.2 Tratamiento de errores

Para garantizar una mayor integridad y confidencialidad en los datos que utiliza el sistema se adoptan las siguientes estrategias para el tratamiento de errores:

- ✓ Los mensajes de error que emitirá el sistema se mostrarán en un lenguaje de fácil compresión para los usuarios.
- ✓ Cuando el usuario introduzca la información incorrecta o deje campos vacíos se mostrará un mensaje en color rojo indicando el error.
- ✓ Se implementan clases para la captura de Excepciones.

En la figura 9 se muestra un ejemplo del tratamiento de errores en el sistema.

Figura 9 Tratamiento de errores.

3.3 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Pueden ser utilizados para modelar sistema de software de cualquier tamaño y complejidad.(50) En la figura 10 se muestran los componentes del sistema.

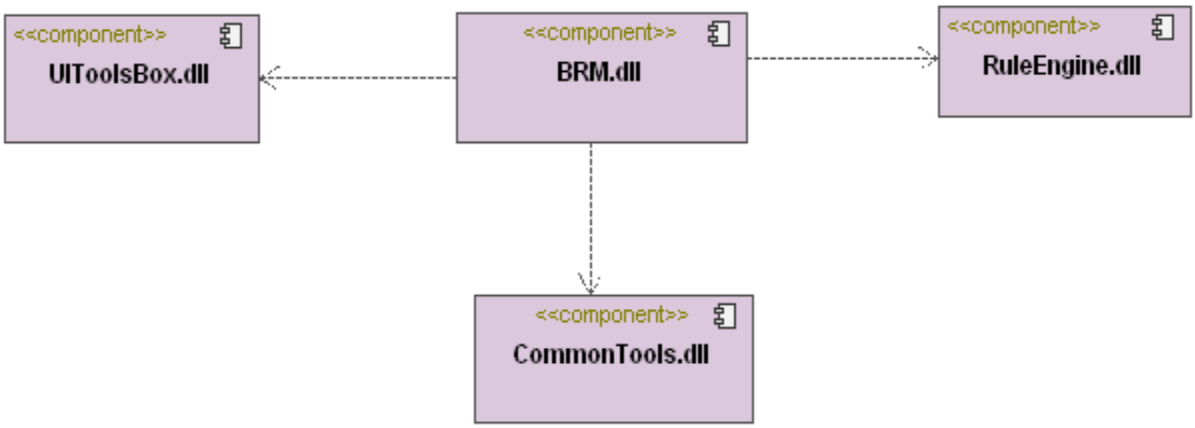


Figura 10 Diagrama de componentes.

El diagrama anterior muestra los principales componentes para la arquitectura del proyecto general. El paquete *BRM* recoge todos los componentes, los cuales están agrupados en un proyecto individual, allí existe un paquete con todos los módulos y en cada uno de ellos se encuentran los llamados *UserControls*, ficheros estructurados <nombre.ascx> que tienen los componentes gráficos y los formularios de las páginas web. El flujo de interfaces es gestionado por la capa de negocio, posibilitando que cualquier cambio que se realice no afecte directamente a esta capa, el paquete es el *RuleEngine*. Por último el *CommonTools*, es un componente externo pero relacionado con el proyecto. Es el componente que rige el comportamiento del sistema, el mismo se encarga de la autenticación y la autorización, esta última controlada desde el menú, maneja el comportamiento de la navegación, el bloqueo de las instancias de procesos, el filtrado de procesos y flujos de trabajos a partir de la información del usuario autenticado.

3.4 Interfaces del sistema

Para lograr un buen diseño del sistema se deben tener una interfaz amigable, cómoda y sencilla de entender por el usuario, esto garantiza el buen funcionamiento de la empresa donde será implantado. Se deben tener en cuenta varios aspectos dentro de los cuales se encuentran que la información

importante se muestre con solo cargar la página, tener presente el contraste de color entre el texto y el fondo, además de no utilizar una carga excesiva de imágenes.

Teniendo en cuenta estos aspectos y las definiciones de diseño establecidas por el proyecto, se muestran a continuación algunas de las pautas empleadas en el Sistema de Gestión de Reglas de Negocio:

Página principal: estará compuesta por el área de identificación, los íconos complementarios y la información utilitaria, el menú lateral, el área de trabajo, el pie de página y el fondo.

Botones: para la tipografía de los botones se usará *Tahoma* regular 10 puntos, los mismos serán de 20 px alto y 10 px de aire entre los extremos del texto dentro del botón y los lados del mismo. Justificándose siempre que sea posible a la derecha del contenido, a una separación de este de 5 px y 10 px entre ellos, excepto en los casos de los botones ubicados al final de la página como son Aceptar y Cancelar.

Tipografía: se utilizará la *Tahoma* en sus distintas variantes y puntajes acorde al contenido.

Subtítulos primarios: identifican secciones lógicas de contenido. Pueden agrupar a varios etiquetas, combos entre otros.

Titulares o *labels* de los cuadros de textos: se utilizará la tipografía Tahoma 11 puntos en su versión normal, justificados a la derecha, color negro.

A continuación se muestra una de las interfaces de usuario del sistema en la figura 11, el resto se encuentran en el anexo4.

Figura 11 Interfaz modificar regla.

3.5 Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Estos diagramas describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones.(51)

En la figura 12 se muestra el despliegue del sistema.

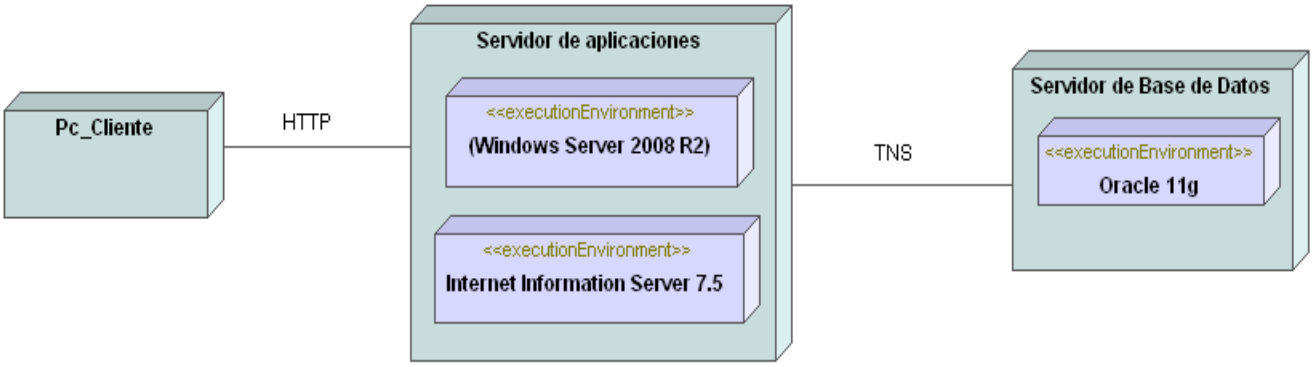


Figura 12 Diagrama de despliegue.

El nodo Pc Cliente es cualquier computadora en la cual trabaje los usuarios del negocio, estas solo tienen que tener instalado un navegador web. El servidor de aplicaciones se refiere a la computadora sobre en la cual se publicará la aplicación web, el mismo debe tener instalado Windows Server 2008 R2 e *Internet Information Server* 7.5. La comunicación entre las pc clientes y el servidor de aplicaciones se realiza por el protocolo HTTP[30]. El servidor de base de datos es la computadora encargada de tener la base de datos, para esto deberá tener instalado el servidor de base de datos Oracle 11g. La comunicación entre el servidor de aplicaciones y el servidor de base de datos se realizará por TNS[31].

3.6 Pruebas

La etapa de pruebas permite evaluar el producto que se ha desarrollado, es decir, verifican el correcto funcionamiento del sistema cumpliendo con los requerimientos establecidos por el cliente. Las pruebas

[30] **Protocolo de Transferencia de Hipertexto**: *acrónimo de Hypertext Transfer Protocol*.

[31] **Sustrato de red transparente**: *acrónimo de Transparent Network Substrate*.

son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa.
Para la validación de la propuesta de solución se utilizaron pruebas unitarias (pruebas de caja blanca) y pruebas de sistema a las funcionalidades.

Pruebas unitarias: caja blanca

La prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se utilizan para ejecutar otro código fuente llamando directamente a los métodos de una clase y pasándole los parámetros necesarios.(52) Además se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos.
La herramienta que se utilizó para realizar las pruebas unitarias fue *Visual StudioTeamSystem2010*. Los métodos de pruebas unitarias residen en clases *Test*, que se almacenan en archivos de código fuente. *Test Edition* permite crear, editar, administrar y ejecutar pruebas así como obtener y almacenar los resultados de las pruebas. Estas pruebas se le realizan a las funcionalidades más críticas.
Realizar pruebas unitarias permite comprobar el correcto funcionamiento de un segmento de código de la aplicación para determinar si el estado real coincide con el esperado. Las pruebas realizadas a los algoritmos de la solución fueron satisfactorias, por lo cual se demuestra que los métodos utilizados cumplen con los resultados esperados, demostrando su correcto funcionamiento. A continuación se muestran en las figura 13 y 15 las pruebas que se le realizaron a las funcionalidades *Evaluate* y *Execute* respectivamente. En la figura 14 y 16se muestran los resultados de dichas pruebas, las cuales indican que las funcionalidades probadas están correctamente implementadas. El resto de las pruebas se encuentran en el anexo 5.

Pruebas de sistema: caja negra

Las pruebas de sistema (pruebas de caja negra) son aquellas que se enfocan directamente en el exterior del módulo, sin importar el código, son pruebas funcionales en las que se trata de encontrar fallas en las que éste no se atiene a su especificación, como ser interfaz con el usuario, apariencia de los menús, control de las teclas, etc. Una prueba de caja negra examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software.
El método de la caja negra se centra en los requisitos fundamentales del software y permite obtener entradas que prueben todos los requisitos funcionales del programa.(53) Con este tipo de pruebas se intenta encontrar:

- ✓ Funciones incorrectas o ausentes.

```
public void EvaluateTest()
{
    SimpleCondition a = new SimpleCondition { Name = "b", Type = PrimitiveType.Integer,
    Operator = CondOperator.GreaterEqual, LeftMember = (new SimpleVariable { Name = "f", Value = 16 }),
    RightMember = (new SimpleVariable { Name = "f", Value = 16 }) };
    BusinessRule target = new BusinessRule { Condition = a, RuleName = "w1" };
    ExcutionContext excution = new ExcutionContext();
    object context = typeof(ExampleBusinessRuleWorkflow.Workflow1);
    bool expected = true;
    bool actual;
    actual = target.Evaluate(excution, context);
    Assert.AreEqual(expected, actual);
}
```

Figura 13 Prueba unitaria realizada a la funcionalidad Evaluate().

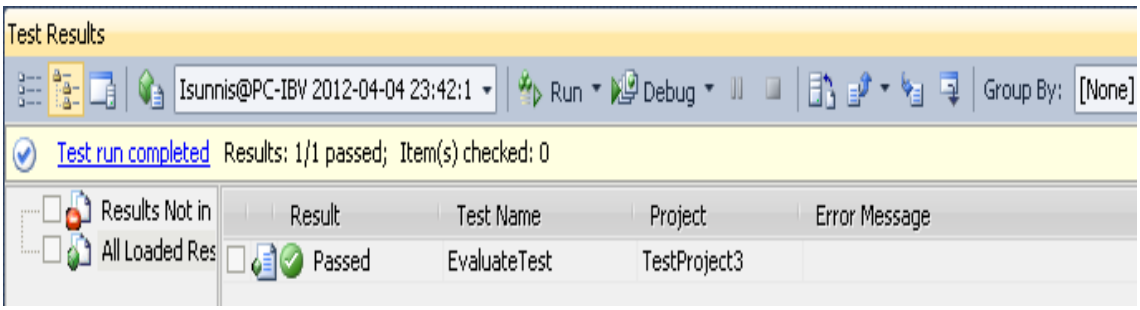


Figura 14 Resultado de la prueba unitaria realizada a la funcionalidad Evaluate ().

- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a la base de datos externa.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Diseños de casos de pruebas

Los casos de prueba definen un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo específico. Cada técnica de prueba proporciona unos criterios distintos para generar estos casos o datos de prueba. El objetivo es diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo. Cada caso de prueba va acompañado del resultado que ha de producir el software al ejecutarlo para detectar un posible fallo en el programa. A continuación se muestra el diseño de caso de prueba de la funcionalidad Gestionar regla. Para ver el resto de los casos de prueba anexo 6.

En cada iteración de prueba se genera un resumen de todas las no conformidades que son entregadas al desarrollador, el mismo es el encargado de erradicarlas. A continuación se muestran los resultados referentes a la segunda iteración de pruebas. Para ver el resto de las iteraciones ver anexo 7.

```
public void ExcuteTest()
{
    SimpleCondition a = new SimpleCondition
    {
        Name = "b",
        Type = PrimitiveType.Integer,
        Operator = CondOperator.GreaterEqual,
        LeftMember = (new SimpleVariable { Name = "f", Value = 16 }),
        RightMember = (new SimpleVariable { Name = "f", Value = 16 })
    };
    BusinessRule target = new BusinessRule
    {
        RuleName = "mayor",
        Condition = a,
        ActionIf = new List<RuleAction>{ new AssigmentRuleAction
        {
            Name = "condition",
            Type = PrimitiveType.Integer,
            Variable = new SimpleVariable { Name = "v", Value = 1 }
        }
        },
        ActionElse = new List<RuleAction>{ new AssigmentRuleAction
        {
            Name = "condition",
            Type = PrimitiveType.Integer,
            Variable = new SimpleVariable { Name = "v", Value = 0 }
        }
        },

    };
    ExcutionContext rules = new ExcutionContext();
    object context =  (new ExampleBusinessRuleWorkflow.Workflow1()).GetWorkflow();
    bool expected = true;
    bool actual;
    actual = target.Excute(rules, context);
    Assert.AreEqual(expected, actual);
}
```

Figura 15 Prueba unitaria realizada a la funcionalidad Excute ().

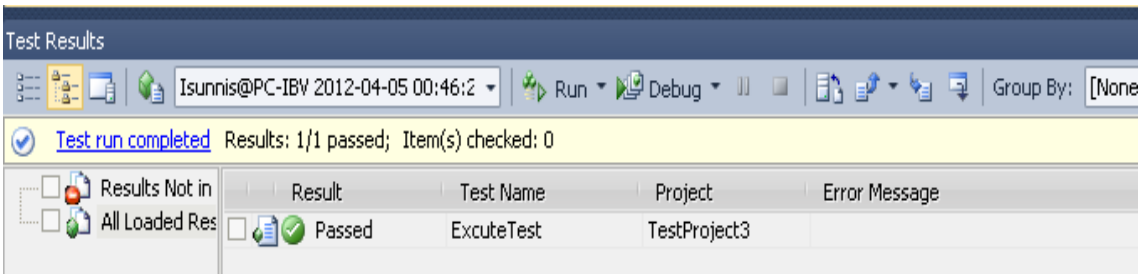


Figura 16 Resultado de la prueba unitaria realizada a la funcionalidad Excute ().

Tabla 7 Caso de prueba a la funcionalidad Gestionar regla.

Escenario	Descripción	Nombre	Tipo de dato	Condición Simple	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción adicionar regla correctamente.	*El usuario adiciona correctamente una regla.*	*V/Mayor Edad*	*V/Workflow1*	*Ver RF3*	*V/letras*	*El sistema procede a guardar la regla en base de datos.*	*Inicio/Regla /Adicionar*

EC 1.2 Opción adicionar regla incorrectamente.	*El usuario adiciona incorrectamente una regla.*	*V/Mayor Edad*	*V/Workflow1*	*Ver RF3*	*V/letras*	*El sistema muestra un mensaje indicando que ha ocurrido un error*	*Inicio/Regla/Adicionar*
EC 1.3 Opción adicionar regla dejando campos vacíos.	*El usuario adiciona una regla dejando campos vacíos.*	*V/vacío* *V/Mayor Edad* *V/Mayor Edad*	*V/Workflow1* *V/vacío* *V/Workflow1*	*Ver RF3* *Ver RF3* *Ver RF3*	*V/letras* *V/letras* *V/vacío*	*El sistema muestra un mensaje indicando que hay campos vacíos.*	*Inicio/Regla/Adicionar*
EC 1.4 Opción Modificar.	*El usuario selecciona esta opción para modificar las reglas ya existentes.*	*V/Mayor Edad*	*V/Workflow1*	*Ver RF3*	*V/letras*	*El sistema actualiza los campos después del usuario haber modificado la regla.*	*Inicio/Regla/Modificar*
EC 1.5 Opción Eliminar.	*El usuario selecciona esta opción para eliminar una regla seleccionada.*	*V/Mayor Edad*	*V/Workflow1*	*Ver RF3*	*V/letras*	*El sistema elimina de la base de datos la regla seleccionada por el usuario.*	*Inicio/Regla/Eliminar*
EC 1.5 Opción Cancelar.	*El usuario selecciona la opción Cancelar.*	*V/Mayor Edad*	*V/Workflow1*	*Ver RF3*	*V/letras*	*El sistema muestra un mensaje de alerta con las opciones "Si" y "No" una vez que oprima cancelar* *Seleccionar la opción "Si", en caso de que desee terminar.* *Seleccionar la*	*Inicio/Regla/Eliminar/Cancelar*

						opción "No", en caso de que no desee finalizar.	

Tabla 8 Resultado de la segunda iteración de pruebas.

Iteración 2					
Elemento	No	No Conformidad	Aspecto correspondiente	Etapa de detección del error	Importancia
RF3	1	Falta validar los campos (valor y value).	Adicionar condición simple.	Al insertar los elementos en la condición.	Significativa.
RF1	2	Cambiar el término (Terminar) en el botón por Cancelar.	Adicionar fórmula simple.	Al seleccionar la opción Terminar.	No significativa.
RF5	3	Agregar el botón Cancelar a la interfaz.	Adicionar una regla.	Al mostrar la interfaz.	No significativa.
RF1	4	Validar los campos valor.	Adicionar una fórmula simple.	Al enviar los valores.	Significativa.
RF5	5	Debe permitir insertar el nombre de una regla con tilde.	Adicionar una regla.	Al insertar el nombre de la regla.	Significativa.
RF2	6	Se debe poner el nombre seleccionar al botón (Button).	Cargar contexto.	Al buscar la ubicación del contexto.	No significativa.
RF4	7	Validar el campo valor.	Adicionar una acción.	Al enviar los valores.	Significativa.
		Mostrar una			

RF2	8	ventana de confirmación al eliminar un contexto.	Eliminar un contexto.	Al eliminar un contexto.	No significativa.

Resultado de las pruebas de sistema

Se realizaron tres iteraciones de pruebas donde se detectaron varias no conformidades que fueron mitigadas durante las etapas de pruebas. En la primera iteración se encontraron un total de 16 no conformidades, que en la mayoría de los casos eran por problemas de validación. Posteriormente en una segunda iteración se validó la corrección de los errores encontrados en la primera iteración, aunque se detectaron 8 nuevas no conformidades. Por último se realizó una tercera iteración que arrojó como resultado 3 no conformidades, mostrando la reducción del número de errores en la aplicación. En el transcurso de las etapas de pruebas se fueron mitigando las no conformidades detectadas como se muestra en la figura 17, con el objetivo de corregir los errores encontrados y garantizar la calidad de la aplicación.

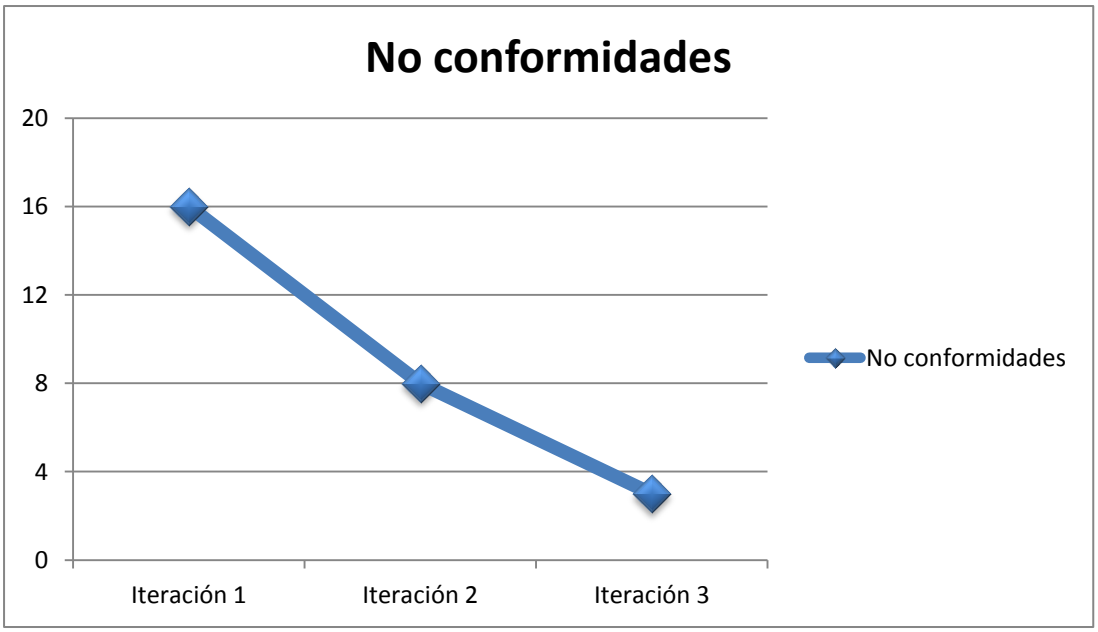


Figura 17 Resultado de las pruebas

3.7Análisis del tiempo de respuesta.

Se realizó una prueba al sistema donde se obtuvo que el tiempo de respuesta ante un cambio de los workflow sin el sistema de gestión de reglas de negocio demora un promedio de 4 horas, luego de probar estos cambios utilizando el sistema de gestión de reglas de negocio se demostró que el tiempo

promedio de adaptación es 4 minutos. El sistema se encuentra en despliegue y certificado anexo 8 en la Dirección de Identificación y Registro (DIR).

Tabla 9 Funcionamiento del SUIN sin el sistema de gestión de reglas de negocio.

Sistema	Versión	Regla	Tiempo
SUIN	0.7.1	EdadMaximaArribo	3hrs
	0.7.3	EdadMinimaArriba	5hrs
	0.7.4	EdadMinimaCI	2hrs
	0.7.6	PrecioSalPrisCI	6hrs

Tabla 10 Funcionamiento del SUIN utilizando el sistema de gestión de reglas de negocio.

Sistema	Versión	Regla	Tiempo
SUIN	0.7.1	EdadMaximaArribo	4min
	0.7.3	EdadMinimaArriba	3min
	0.7.4	EdadMinimaCI	5min
	0.7.6	PrecioSalPrisCI	4min

3.8 Beneficios que brinda el sistema

Con la implementación del Sistema de gestión de reglas de negocio para el Sistema Único de Identificación Nacional obtendrán varios beneficios los cuales se muestran a continuación:

- ✓ Rapidez de adaptación en el funcionamiento del sistema.
- ✓ Se reduce el costo de mantenimiento al no tener que recompilar por cada cambio menor.
- ✓ Los cambios requeridos pueden ser implementados sin cambios en el código, aislando el cambio y probando sólo la regla que ha sido modificada.

Conclusiones

Los estándares de codificación permitieron darle mayor claridad a la implementación de las funcionalidades.

Un correcto tratamiento de las excepciones permitió identificar los diferentes errores que fueron encontrados a lo largo de las iteraciones y al mismo tiempo considera las mismas como parte del negocio del sistema.

La correcta realización de las pruebas unitarias y de sistema permitió identificar las diferentes no conformidades y validar las funcionalidades acorde a los requisitos planteados.

Conclusiones Generales

Con la realización del presente trabajo de diploma se obtuvo la primera versión del Sistema de Gestión de Reglas de Negocio para el Sistema Único de Identificación de Nacional. A través de las diferentes etapas se pudo observar que:

El desarrollo de una aplicación web demostró que era posible administrar de forma centralizada las reglas, lo cual facilitó el manejo y actualización de las mismas.

Las pruebas realizadas al sistema permitió comprobar su correcto funcionamiento y validar que el tiempo de adaptación ante un cambio fue de aproximadamente 4 minutos mientras que antes se realizaba en un aproximado de 3 horas.

El desarrollo del sistema de gestión de reglas de negocio permitió mejorar el tiempo de adaptación en el funcionamiento y la usabilidad en aplicaciones desarrolladas sobre el *framework* .NET 4.0 y la tecnología WWF 3.5.

Recomendaciones

Se recomienda:

- ✓ Implementar un lenguaje para la validación de la estructura de las reglas, mediante un compilador para mejorar la sintaxis en la creación de las reglas.

Bibliografía

1. **Hendryx, S.** business rules and standards. *a fair isaac white paper.* 2003.

2. **Bajec, M. K.** using business rules technologies to bridge the gap between business and business applications, in proc. of the ifip 16th world computer congress 2000, information technology for business management. Pekin, China : g.e. rechnu, 2000.

3. *Traducción De Un Patrón De Reglas De Negocio En Bases De Datos Relacionales.* **Boggiano Castillo, Martha Beatriz, y otros, y otros.** Villa Clara : Latin American and Caribbean Conference for Engineering and Technology, 2009.

4. **BRG.** defining business rules ~ what are they really? . 2000.

5. **María Elena Martínez del Busto, Isel Moreno Montes de Oca,Paulino Hernández Hernández.***Modelo de hechos genérico para procesar reglas de negocio.* Cristóbal, Venezuela : s.n., June 2-5,2009.

6. Eurodecision. *Eurodecision.* [Online] 2009-2012. [Cited: noviembre 17, 2011.] http://www.eurodecision.es/systemes-a-base-de-regles.

7. **Barbara von Halle, Larry Goldberg and Guest.***The Business Rule Revolution: Running Business the Right Way.* Silicon Valley,California,USA : s.n., 2006.

8. Cientec. *Cientec.* [Online] [Cited: noviembre 18, 2011.] http://www.cientec.com/management/management-brms.html.

9. **The Business Rules Group.***Manifiesto de Reglas de Negocio Los Principios de la Independencia de las Reglas .* 2003.

10. MSDN. *MSDN.* [Online] Microsoft, 2012 . [Cited: noviembre 18, 2011.] http://msdn.microsoft.com/es-es/library/aa954061%28v=bts.10%29.aspx.

11. **Carmen de Pablos, José Joaquín López-Hermoso Agius,Santiago Martín-Romo Romero,Sonia Medina Salgado.***Informática y Comunicaciones en la Empresa.* Madrid : ESIC.

12. Polymita. *Polymita.* [Online] 2002-2012. [Cited: noviembre 20, 2011.] http://www.polymita.com/portal/es/bpm/Business_Rules_Management.

13. **María Elena Martínez del Busto, Nestor Ibargollín Pérez,Martha Beatriz Boggiano, Alain Pérez Alonson,Luisa González González.***Edición de reglas de negocio sobre una ontología que permita su implementación independiente de las bases de datos.* Universidad Central de Las Villas, Cuba. : s.n., ENERO-JUNIO, 2009.

14. **Gabriel Sosa, Lucas Dima,Rafael Urdaneta,Gabriela Esperón.***Inteligencia Artificial aplicada al desarrollo de Evaluaciones Matemáticas.* 2011.

15. **Freund-Ruecker-Hitpass, Jakob Freund, Bernd Rucker, Bernhard Hitpass.***BPMN 2.0 Manual de referencia.* Santiago de Chile. : s.n., 2011.

16. **Graham, Ian.***Business Rules Management and Service Oriented Architecture.*

17. **Universidad Peruana Unión.** Escuela de Posgrado. *Escuela de Posgrado.* [Online] 2011. [Cited: noviembre 29, 2011.] http://www.posgrado.upeu.edu.pe/epgvirtual/documentos/guia1.ppt.

18. **Business Publications Spain .** Business Publications Spain . *Business Publications Spain .* [Online] [Cited: noviembre 22, 2011.] http://www.computing.es/negocios/reportajes/1012747002201/brms-tecnologia-mejora-constante.1.html.

19. **Ting Yu, Sushil Jajodia.***Secure Data Management in Decentralized Systems.* North Carolina State University : s.n.

20. **Microsoft.** Kit de herramientas de RuleSet externo. *Kit de herramientas de RuleSet externo.* [Online] Microsoft, 2011. [Cited: 11 27, 2011.] http://msdn.microsoft.com/es-es/library/ee960221.aspx.

21. **Becerra, Rodolfo Adrián.***Sistemas Expertos para la realización de diagnóstico de trastornos neuromusculares con electromiografía.* Mendoza : s.n., 2011.

22. **QTX.** Qualitrain. [Online] 2001-2011. [Cited: noviembre 20, 2011.] http://www.qualitrain.com.mx/index2.php?option=com_content&do_pdf=1&id=149.

23. *El Lenguaje .NET I.*

24. **Fernando Berzal, Francisco José Cortijo,Juan Carlos Cubero.***Desarrollo Profesional de Aplicaciones Web con ASP.NET.*

25. **Microsoft Corporation.** MSF for CMMI Process Improvement. *MSF for CMMI Process Improvement.* [Online] 2005. [Cited: noviembre 26, 2011.] http://guides.brucejmack.biz/MSF%20for%20CMMI%20Process%20Improvement/Process%20Guidance/Supporting%20Files/ProcessGuidance.htm.

26. **Sánchez., María A. Mendoza.***Metodologías De Desarrollo De Software.* Perú S.A.C. : s.n., 2004.

27. **Gracia, Joaquin.** IngenieroSoftware. *IngenieroSoftware.* [Online] 2003. [Cited: noviembre 30, 2011.] http://www.ingenierosoftware.com/calidad/cmm-cmmi.php.

28. **Gunnerson, Eric.***A programmer's introduction to C#.* s.l. : Apress, 2000.

29. **Miguel Ángel Gómez Vicente, Manuel Martín Mohedano.***Introducción a C#.* Universidad de Salamanca. : s.n., 2005.

30. **Sparks, Geoffrey.***Una Introducción al UML.* Australia : s.n.

31. **Peter Rob, Carlos Coronel.***Sistemas de bases de datos: Diseño, implementación y administración.* 2004.

32. **García, Carlos Egea.***Diseño web para tod@s I.* Barcelona : Icaria Editorial, s.a.

33. **Rodríguez, Ana Nieves.***Bases de Javascript.* Universidad Nacional de Tucumán : s.n.

34. **Cuéllar, Javier Fernández.***CALIDAD Y SEGURIDAD A NIVEL DE FILAS EN BBDD ORACLE.* Universidad Carlos III de Madrid : s.n., 2009.

35. **Oracle .***Oracle database 11g Press Kit.*

36. **Nick Randolph, David Gardner,Michael Minutillo,Chris Anderson.***Professional Visual Studio 2010.*

37. **Microsoft Corporation.** msdn. *msdn.* [Online] 2011. [Cited: noviembre 26, 2011.] http://blogs.msdn.com/b/bharry/archive/2009/04/09/team-system-2010-overview.aspx.

38. **Marco A. Guevara Injoque, César R. Flores Nazario.***AllFusion Erwin Data Modeler.* 2002.

39. **GmbH, Altova.***Altova UModel 2009 User & Reference manual.* 2009.

40. **Mercedes Marqués Andrés, José I. Aliaga Estellés,Salvador García Gil,Gregorio Quintana Ortí.***SQL y desarrollo de aplicaciones en Oracle 8.* Universitat Jaume: s.n., 2001.

41. **Villenueve, Louise.***Mozilla Firefox(Windows y Linux).* Barcelona : ENI, octubre 2005.

42. **Pearson Education DeutschlandGmbH.***Windows 2000 profesional.* Barcelona : Boixareu , 2000.

43. **Andreu, Joaquin.***Servicios en Red.* Madrid : Editex, S.A.

44. **Pressman, Roger S.***Ingieneria de software un enfoque práctico.* s.l. : McGraw-Hill Interamericana de España S.L, 2010.

45. **Jackson , Daniel and Devadas, Srinivas.**Curso práctico en Ingeniería de Software. [Online] [Cited: diciembre 7, 2011.] http://mit.ocw.universia.net/6.170/6.170/f01/pdf/lecture-12.pdf..

46. **Frank Buschmann, Kevlin henney, douglas C. Schmidt.***Pattern-Oriented Software Architecture.* England : s.n., 2007.

47. Mitopencourseware. *Mitopencourseware.* [Online] [Cited: enero 15, 2012.] http://mit.ocw.universia.net/6.170/6.170/f01/related-resources/specifications.html.

48. **Falgueras, Benet Campderrich.***Ingenieria del software.* Barcelona : UOC.

49. **Behrouz A. Forouzan, Sophia Chung Fegan.***Foundations of Computer Science: From Data Manipulation to Theory of Computation.* s.l. : S.A, 2003.

50. **Falgueras, Benet Campderrich.***Ingeniería del software.* Barcelona : UOC.

51. **López, Francisco Javier Moliner.***Informática Bloque Específico.* Valencia : Mad, S.L.

52. **Antonio Molina Marco, Pedro Sánchez Palma, Patricio Letelier Torres, Juan Sánchez Díaz.***Metodología Y Tecnología de la Programación.* Universidad Politécnica de Valencia. : s.n.

53. **Isabel Ramos Román, José Javier Dolado Cosín.***Técnicas Cuantitativas para la Gestión en la Ingeniería del Software.* Spain : Netbiblo, S.L, 2007.

54. **Microsoft.** Microsoft. *Microsoft.* [Online] 2011. [Cited: noviembre 28, 2011.] http://technet.microsoft.com/es-es/library/bb490157.aspx.

55. **Collison, Simon.***Beginning CSS Web Development: From Novice to Professional.* 2006.

56. **Michaelis, Mark.***Essential C# 4.0.* 2010.

57. **Laura Alemany de Aguinaga, Víctor Manuel Arias Torrijos,Alberto Sánchez Carrillo.***Depurador Declarativo para Plataforma .NET.* Universidad Complutense de Madrid : s.n., 2007-2008.

58. **María Elena Martínez del Busto, Isel Moreno Montes de Oca,Abel Rodríguez Morffi, Manuel Castro Artiles, Luisa González González.***Business vocabulary of kidney transplant with ontological approach for a generic fact model.* Santa Clara, Cuba. : s.n., 2010.

59. **Juan Carlos Bustamante, Fabricio Echeverría,José Lucio,José Peña.***Sistemas de Información Gerencial BRMS (Business Rule Management System).* Guayaquil,Ecuador : s.n., 2006.

60. **Martha Beatriz Boggiano Castillo, Alaín Pérez Alonso,Ramiro Pérez Vázquez,María Elena Martínez Del Busto,Luisa González González.***Traducción de un patrón de Reglas de Negocio en bases de datos relacionales.* 2009.

61. **Montaldo, Diego Fernando.***"Patrones de Diseño de Arquitecturas de Software Enterprise".* Universidad de Buenos Aires : s.n., Noviembre 2005.

62. **Erich Gamma, Richard Helem,Ralph Johnson,John Vlissides.***Design Patterns.Elements of Reusable Object-Oriented Software.* 1995.

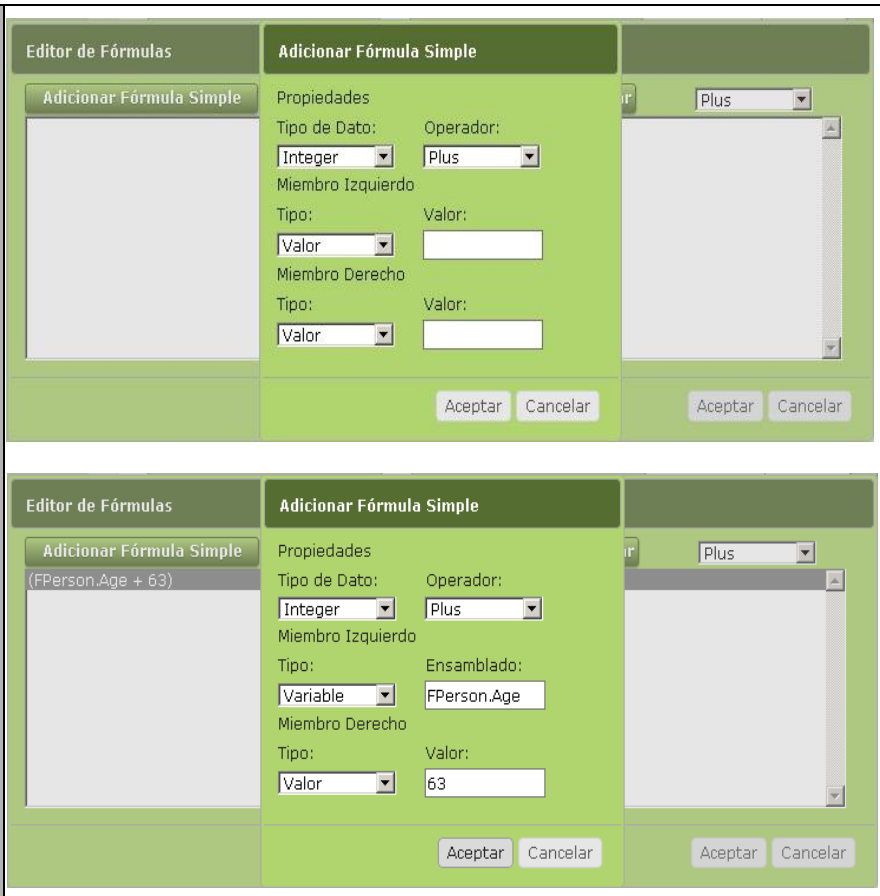
Anexos

Anexo1 Descripción de requisitos funcionales.

RF 1 Gestionar fórmula

Tabla 11 Descripción del requisito funcional Gestionar fórmula.

Propósito	Permite gestionar una fórmula.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	Fórmula	
Descripción	1. Adicionar fórmula 1.1 El usuario selecciona la opción adicionar fórmula. 1.2 Se muestra una interfaz con los campos que se necesitan para adicionar una fórmula. a. Propiedades b .Miembro izquierdo c. Miembro derecho 1.3 El usuario introduce los datos necesarios. 1.4 Se selecciona el botón aceptar. 2. Modificar fórmula 2.1 El usuario selecciona la opción modificar fórmula. 2.2 Se muestra una interfaz con las fórmulas existentes. 2.3 El usuario selecciona la fórmula que desea modificar. 2.3 Se muestra dicha fórmula con sus propiedades. 2.4 El usuario modifica los valores deseados y selecciona el botón aceptar.	
Validaciones	Debe seleccionar una fórmula.	
Postcondiciones	Se guarda la fórmula en base de datos.	

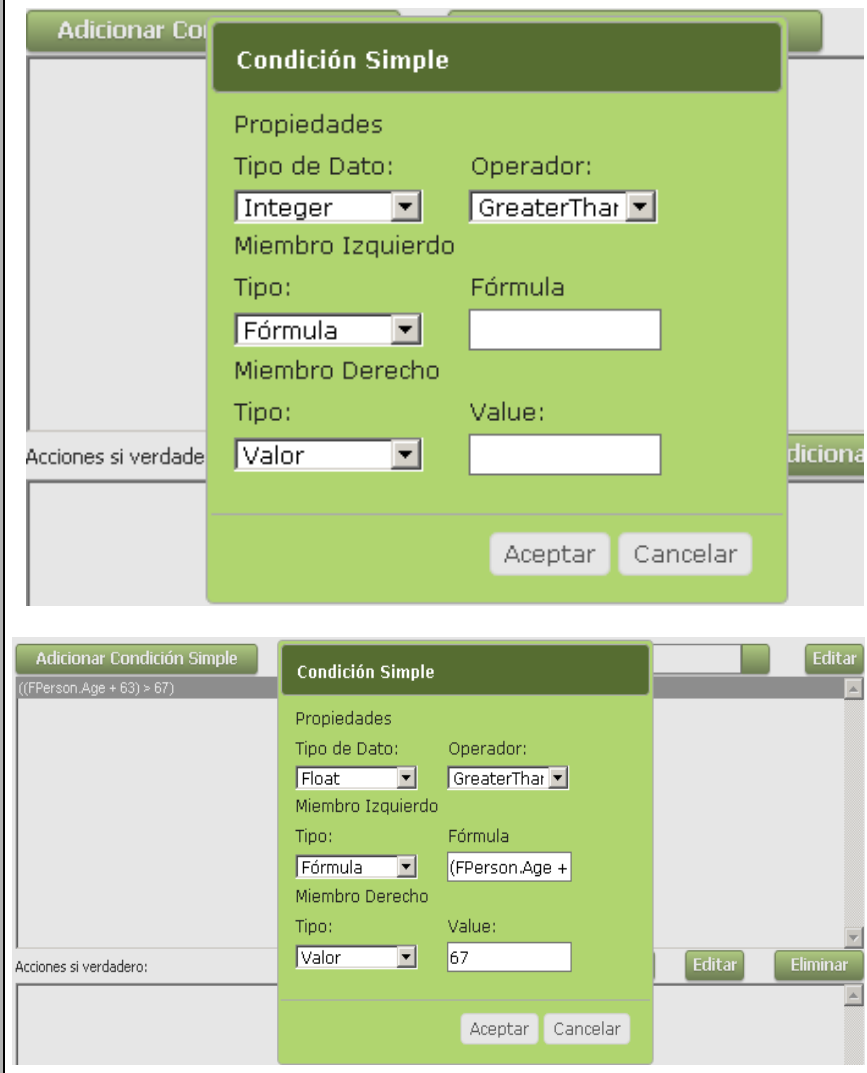
Prototipo	

RF 3 Gestionar condición

Tabla 12 Descripción del requisito funcional Gestionar condición.

Propósito	Permite gestionar una condición.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	Condición	
Descripción	1. Adicionar condición 1.1 Adicionar condición simple 1.1.1 El usuario selecciona la opción adicionar condición simple. 1.1.2 Se muestra una interfaz con los campos que se necesitan para adicionar una condición. a. Propiedades	

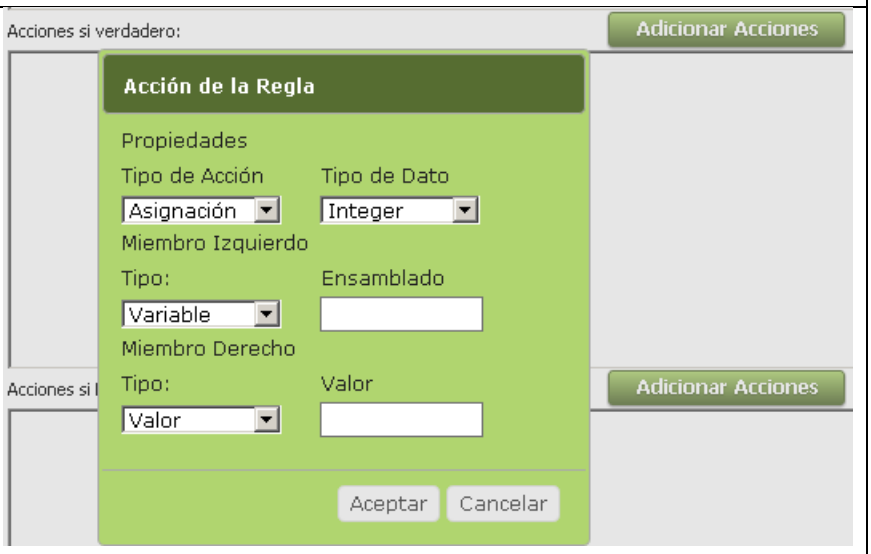
	b. Miembro izquierdo c. Miembro derecho. 1.1.3 El usuario introduce los datos necesarios. 1.1.4 Se selecciona el botón aceptar. 1.2 Adicionar grupo de condición. 1.2.1 Selecciona la opción adicionar grupo de condición. 1.2.2 Para adicionar un grupo de condición deben existir al menos dos condiciones simples. 1.2.3 El usuario selecciona la opción aceptar. 2. Modificar condición 2.1 El usuario selecciona la opción modificar condición. 2.2 Se muestra una interfaz con las condiciones existentes. 2.3 El usuario selecciona la condición que desea modificar. 2.4 Se muestra una interfaz con las propiedades de dicha condición. 2.5 El usuario modifica los campos deseados y selecciona el botón aceptar.
Validaciones	Debe seleccionar una condición.
Postcondiciones	Se guarda la condición en base de datos.

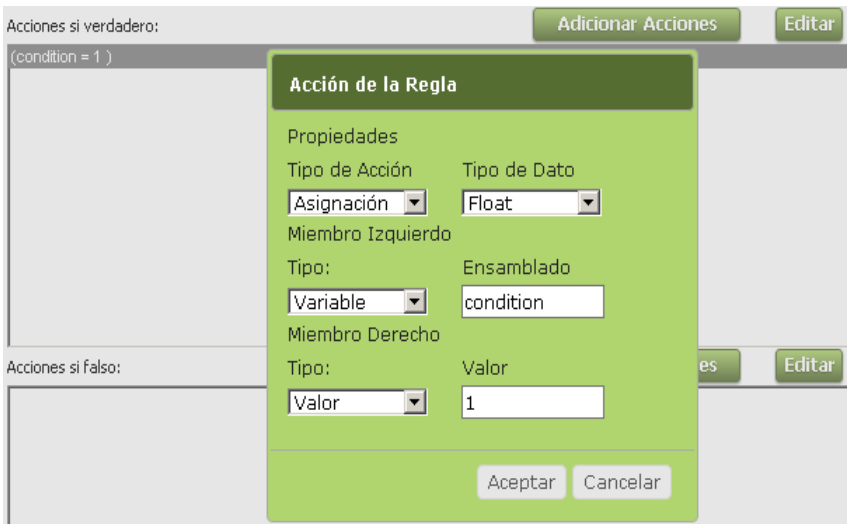
Prototipo	

RF 4 Gestionar acción

Tabla 13 Descripción del requisito funcional Gestionar acción.

Propósito	Permite gestionar una acción.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	Acción	
Descripción	1. Adicionar acción 1.1 El usuario selecciona la opción adicionar acción.	

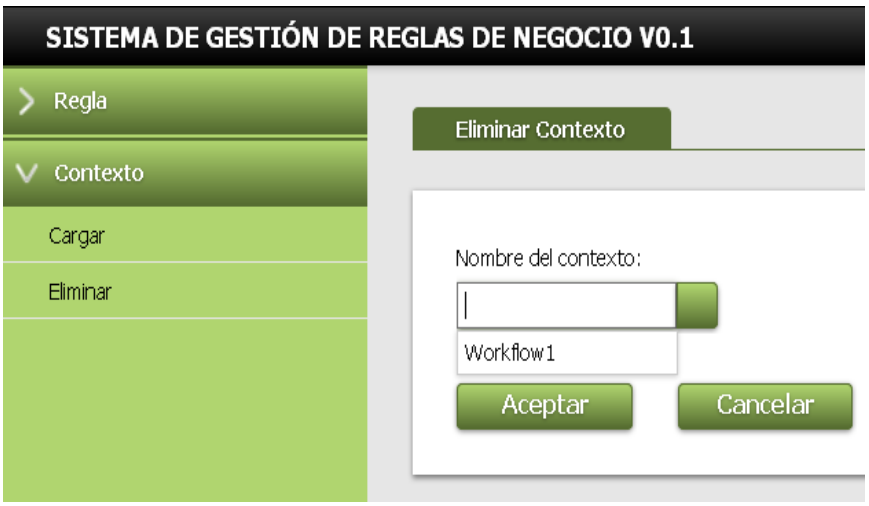
	1.2 Se muestra una interfaz con los campos que se necesitan para adicionar una acción. a. Propiedades b. Miembro izquierdo c. Miembro derecho 1.3 El usuario introduce los datos necesarios. 1.4 Selecciona el botón aceptar. 2. Modificar acción 2.1 El usuario selecciona la opción modificar acción. 2.2 Se muestra una interfaz con las propiedades de dicha acción. 2.3 El usuario modifica los valores deseados. 2.4 Selecciona la opción aceptar. 3. Eliminar acción 3.1 El usuario selecciona la acción que desea eliminar. 3.2 Selecciona la opción eliminar.
Validaciones	Debe seleccionar una condición.
Postcondiciones	Se guarda la acción en base de datos.
Prototipo	



RF 2 Gestionar contexto

Tabla 14 Descripción del requisito funcional Gestionar contexto.

Propósito	Permite gestionar un contexto.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	Contexto	
Descripción	1. Cargar contexto 1.1 El usuario 1.2 Se muestra una interfaz con un botón para seleccionar el directorio donde se encuentra el contexto. 1.3 Se muestra una interfaz donde se puede buscar y seleccionar el contexto. 1.4 Si el contexto tiene dependencias se muestra una interfaz para cargar dichas dependencias. 1.5 Se muestra un listado con todas las propiedades que posee el contexto. 1.6 Se selecciona la propiedad que se desee cargar para utilizar en la regla. 1.7 Se selecciona el botón aceptar.	

	2. Mostrar contexto. 2.2 El usuario tiene que estar en el escenario crear acción o crear condición para que se le muestre el contexto. 3. Eliminar contexto 3.1 El usuario selecciona la opción eliminar contexto. 3.2 Se muestra una interfaz con los contextos existentes. 3.3 El usuario selecciona el contexto que desea eliminar. 3.4 Se muestra una ventana de confirmación. 3.5 El usuario selecciona la opción “Aceptar”.
Validaciones	Debe seleccionar un directorio válido.
Postcondiciones	Se guarda el contexto en base de datos.
Prototipo	 

RF 6 Evaluar regla

Tabla 15 Descripción del requisito funcional Evaluar regla.

Propósito	Permite evaluar una regla.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	Regla	
Descripción	1. Evaluar regla 1.1 El usuario debe especificar el nombre y el contexto de la regla.	
Validaciones	Que exista la regla en base de datos.	
Postcondiciones		
Prototipo	No procede	

RF 7 Ejecutar regla

Tabla 16 Descripción del requisito funcional Ejecutar regla.

Propósito	Permite ejecutar una regla.	
Roles	Usuario y administrador.	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Conceptos tratados	**Concepto**	**Atributos**
	Regla	
Descripción	1. Ejecutar regla 1.1 El usuario debe especificar el nombre de la regla.	
Validaciones	Que exista la regla en base de datos.	
Postcondiciones		
Prototipo	No procede	

Anexo 2 Diagrama de clases del diseño

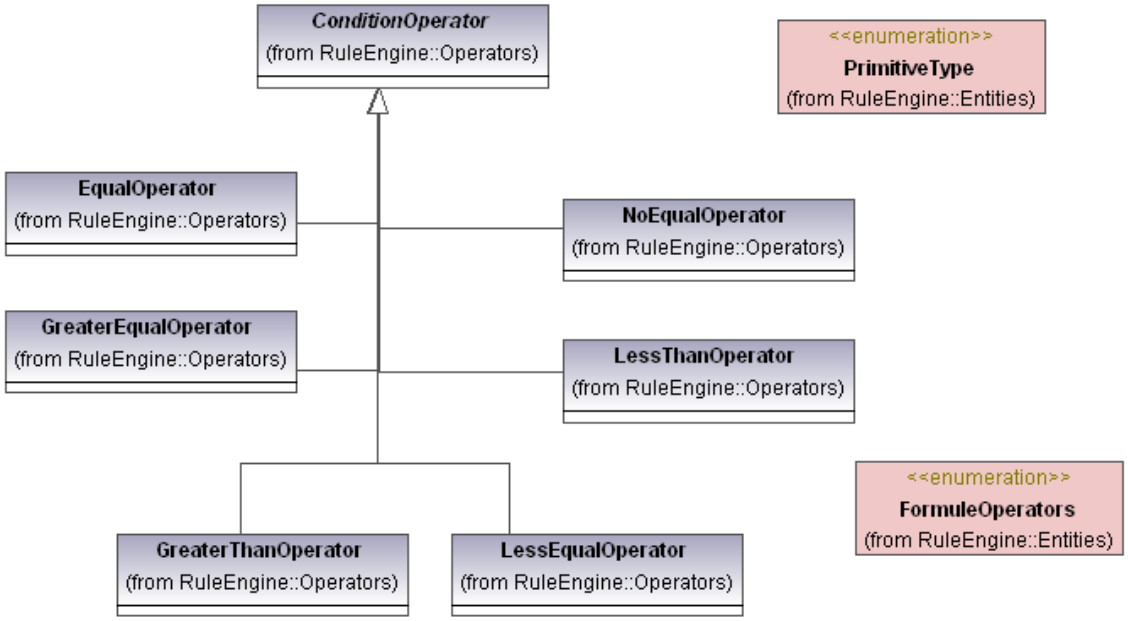


Figura 18 Diagrama de clases del diseño (III).

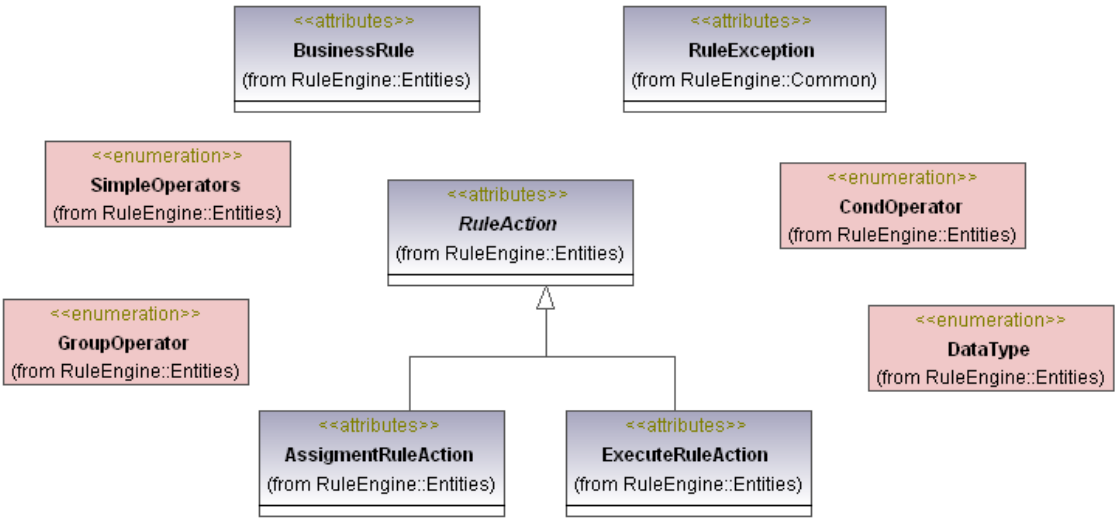


Figura 19 Diagrama de clases del diseño (IV).

Anexo 3 Estándares de codificación.

La siguiente tabla resume las reglas de capitalización y provee ejemplos para cada tipo de identificador.

Tabla 17 Reglas de capitalización.

Identificador	Case	Ejemplo
Parámetro	Camello	**typeName**
Atributo Protegido	Camello	**redValue** **Nota**: Una propiedad es recomendada al uso de un atributo Protected.
Clases	Pascal	**AppDomain**
Enumerador	Pascal	**ErrorLevel**
Valores de Enumerador	Pascal	**FatalError**
Evento	Pascal	**ValueChange**
Clases Excepción	Pascal	**WebException** **Note**: Siempre termina con el sufijo Exception.
Atributo estático de solo lectura	Pascal	**RedValue**
Interface	Pascal	**IDisposable** **Note**: Siempre comienza con el prefijo I.
Método	Pascal	**ToString**
Namespace	Pascal	**System.Drawing**
Propiedad	Pascal	**BackColor**
Atributo Publico	Pascal	**RedValue** **Nota**: Una propiedad es recomendada al uso de un atributo Public.

Anexo 4 Interfaces

Figura 20 Interfaz cargar contexto.

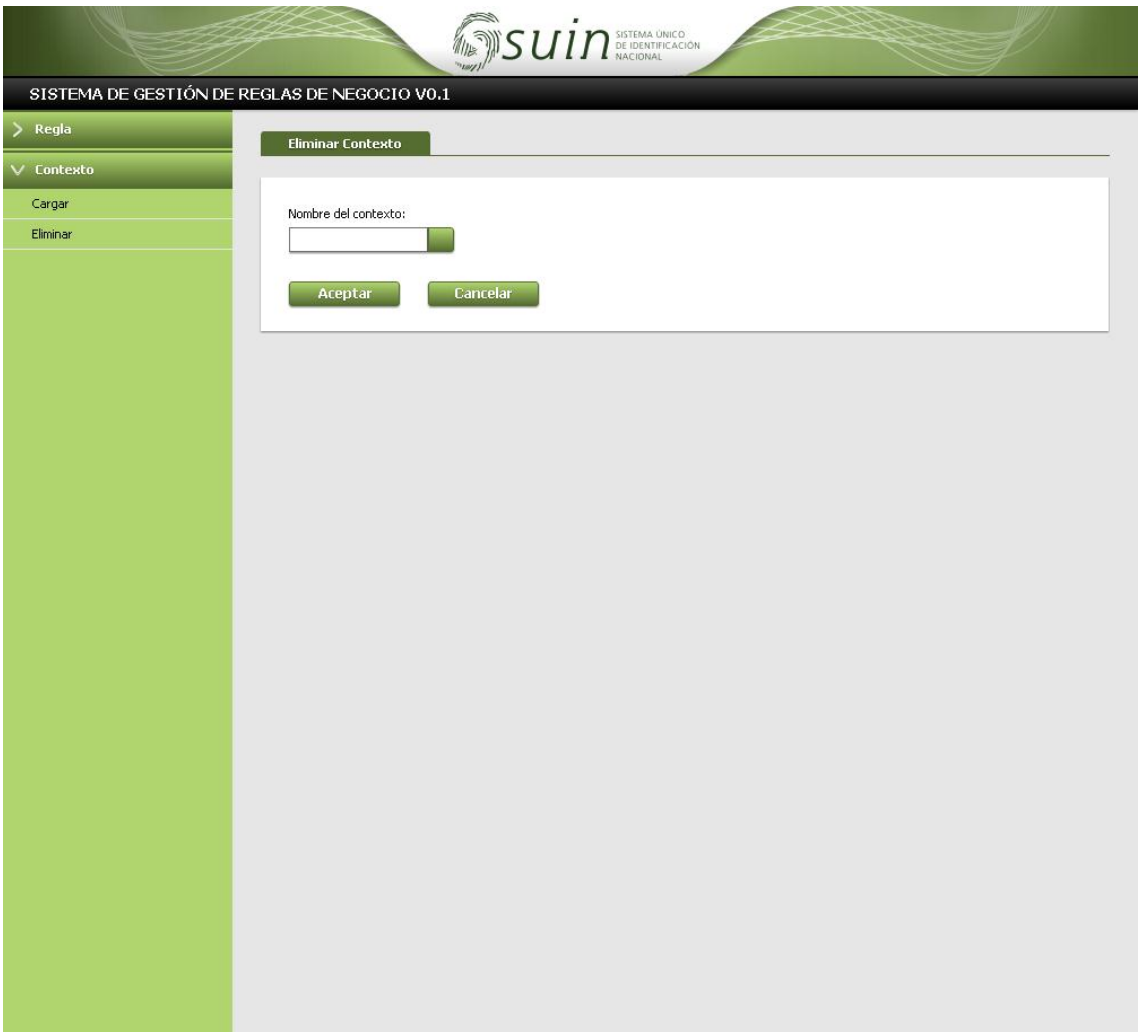


Figura 21 Interfaz eliminar contexto.

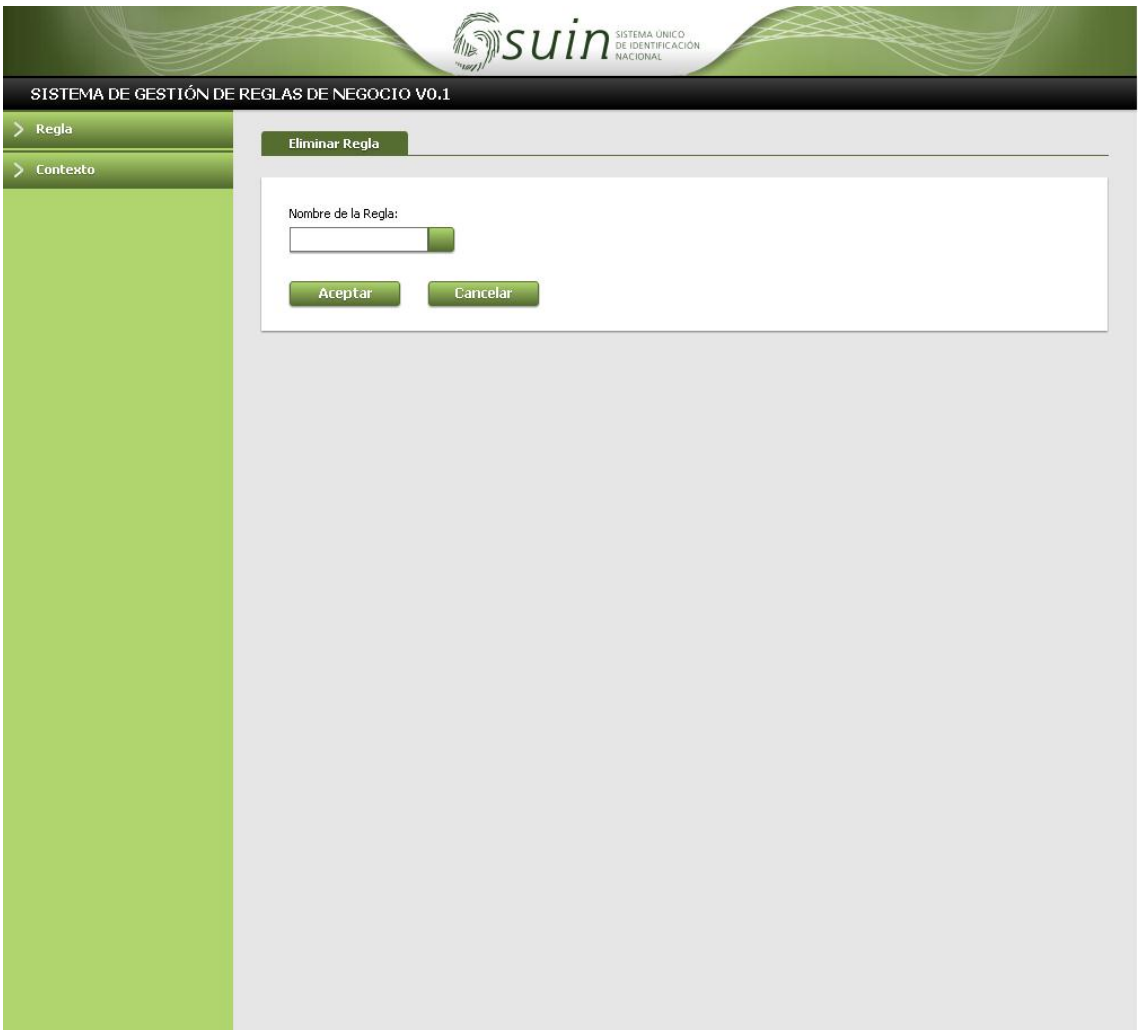


Figura 22 Interfaz eliminar regla.

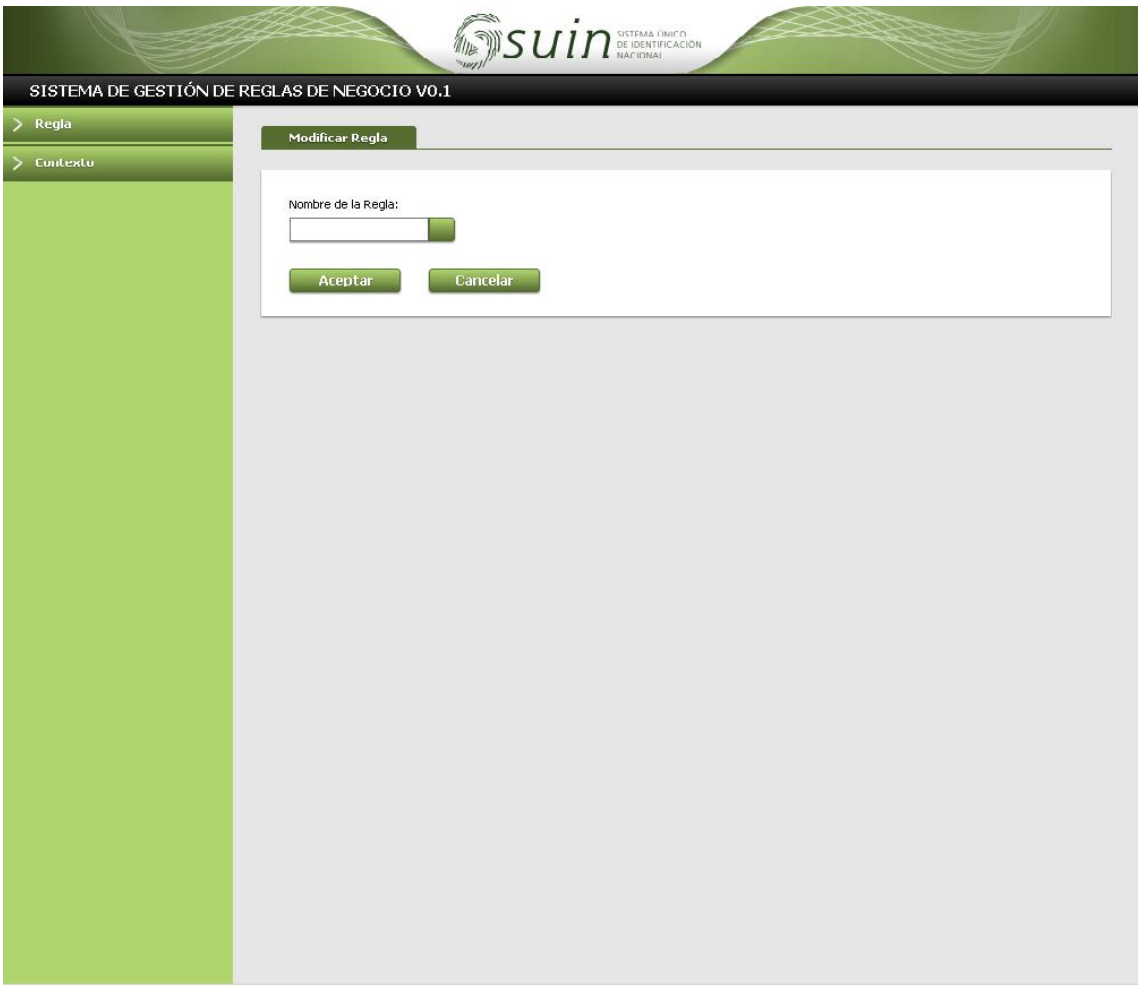


Figura 23 Interfaz modificar regla.

Anexo 5 Pruebas unitarias

```
/// <summary>
///A test for DeleteContext
///</summary>
[TestMethod()]
public void DeleteContextTest()
{
    RulesEngine target = new RulesEngine(); // TODO: Initialize to an appropriate value
    Guid contextID = target.LoadContext("Workflow1").ContextID; // TODO: Initialize to an appropriate value
    target.DeleteContext(contextID);
    //Assert.Inconclusive("A method that does not return a value cannot be verified.");
}
```

Figura 24 Prueba unitaria realizada a la funcionalidad DeleteContext().

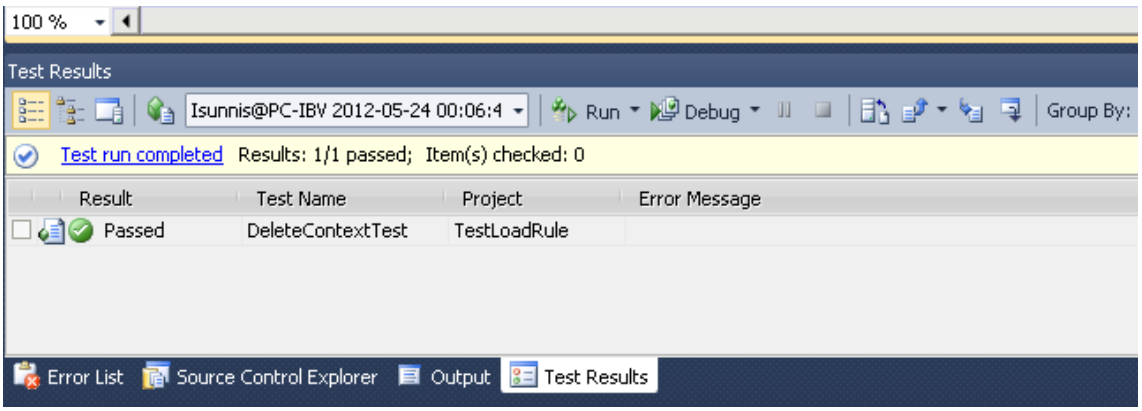


Figura 25 Resultado de la prueba realizada a la funcionalidad DeleteContext().

```
/// <summary>
///A test for DeleteRule
///</summary>
[TestMethod()]
public void DeleteRuleTest()
{
    RulesEngine target = new RulesEngine(); // TODO: Initialize to an appropriate value
    string rule = "iguales"; // TODO: Initialize to an appropriate value
    target.DeleteRule(rule);
    //Assert.Inconclusive("A method that does not return a value cannot be verified.");
}
```

Figura 26 Prueba unitaria realizada a la funcionalidad DeleteRule().

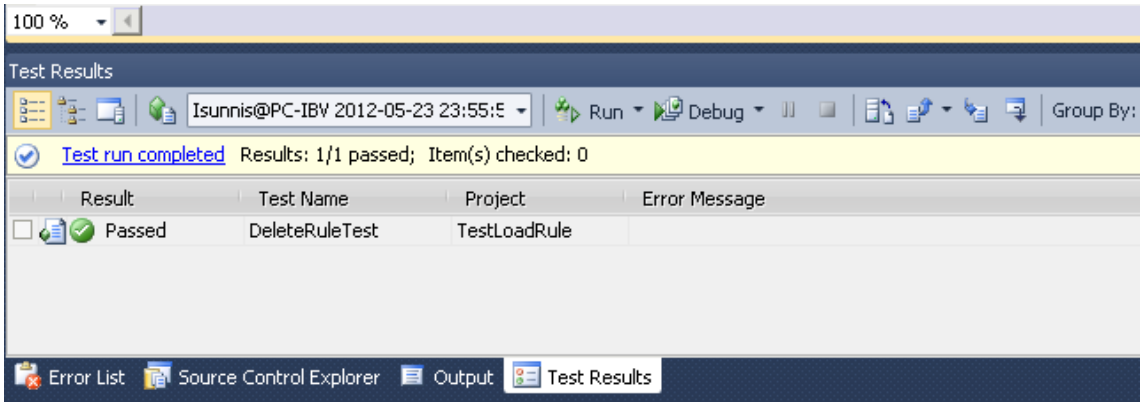


Figura 27 Resultado de la prueba unitaria realizada a la funcionalidad DeleteRule().

```
/// <summary>
///A test for UpdateRule
///</summary>
[TestMethod()]
public void UpdateRuleTest()
{
    RulesEngine target = new RulesEngine(); // TODO: Initialize to an appropriate value
    BusinessRule rule = target.LoadRule("Mayor"); // TODO: Initialize to an appropriate value
    Guid contexId = new Guid(); // TODO: Initialize to an appropriate value
    target.UpdateRule(rule, rule.ContextDescription.ContextID);
    //Assert.Inconclusive("A method that does not return a value cannot be verified.");
}
```

Figura 28 Prueba unitaria realizada a la funcionalidad UpdateRule().

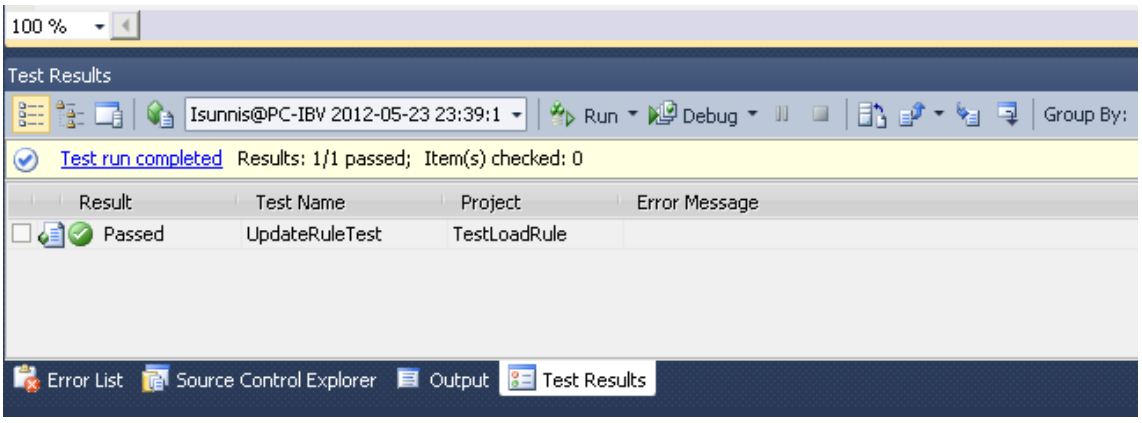


Figura 29 Resultado de la prueba unitaria realizada a la funcionalidad UpdateRule().

Anexo 6 Casos de prueba

Tabla 18 Caso de prueba realizado al RF3 Gestionar condición.

Escenario	Descripción	Tipo de dato	Operador	Tipo MI	Valor MI	Tipo MD	Valor MD	Respuesta del sistema	Flujo central
EC 1.1 Opción adicionar condición correctamente.	*El usuario adiciona una condición correctamente.*	*V/integer*	*V/lessThan*	*V/valor*	*V/5*	*V/variable*	*V/Fperson.Age*	*El sistema procede a adicionar la condición.*	*Inicio/Regla/AdicionarRegla/AdicionarCondición*
EC 1.2 Opción adicionar condición incorrectamente.	*El usuario adiciona una condición incorrectamente.*	*V/string*	*V/lessThan*	*V/valor*	*V/25*	*V/variable*	*V/Fperson.Age*	*El sistema muestra un mensaje indicando el error.*	*Inicio/Regla/AdicionarRegla/AdicionarCondición*
EC 1.3 Opción adicionar condición dejando campos vacíos.	*El usuario adiciona una condición dejando campos vacíos.*	*V/double*	*V/vacío*	*V/valor*	*V/6*	*V/valor*	*3*	*El sistema muestra un mensaje indicando que hay campos vacíos.*	*Inicio/Regla/AdicionarRegla/AdicionarCondición*
		V/integer	*V/greaterThan*	*V/vacío*	*V/89*	*V/variable*	*V/Fperson.Age*		
		V/integer	*V/lessThan*	*V/valor*	*V/vacío*	*V/valor*	*V/36*		
		V/string	*V/greaterThan*	*V/variable*	*V/Fperson.Age*	*V/vacío*	*V/8*		
		V/integer	*V/greaterThan*	*V/valor*	*47*	*V/variable*	*V/vacío*		
EC 1.4 Opción modificar.	*El usuario modifica la condición seleccionada.*	*V/integer*	*V/lessThan*	*V/valor*	*V/5*	*V/variable*	*V/Fperson.Age*	*El sistema guarda las modificaciones realizadas por el usuario.*	*Inicio/Regla/AdicionarRegla/Editar*

Tabla 19 Caso de prueba realizado al RF2 Gestionar contexto.

Escenario	Descripción	Seleccionar	Respuesta del sistema	Flujo central
EC 1.1 Opción Cargar contexto.	*El usuario carga el contexto de la regla.*	*V/Workflow1*	*El sistema muestra los contextos existentes.*	*Inicio/Contexto/CargarContexto*
EC 1.2 Opción Eliminar.	*El usuario selecciona esta opción para eliminar el contexto seleccionado.*	*V/Workflow1*	*El sistema elimina de la base de datos el contexto seleccionado por el usuario.*	*Inicio/Contexto/EliminarContexto*
EC 1.4 Opción Cancelar.	*El usuario selecciona la opción Cancelar.*	*V/workflow1*	*El sistema muestra un mensaje de alerta con las opciones "Si" y "No" una vez que oprima cancelar Seleccionar la opción "Si", en caso de que desee terminar. Seleccionar la opción "No", en caso de que no desee finalizar.*	*Inicio/Contexto/EliminarContexto/Cancelar*

Tabla 20 Caso de prueba realizado al RF1 Gestionar fórmula.

Escenario	Descripción	Tipo de dato	Operador	Tipo MI	Valor MI	Tipo MD	Valor MD	Respuesta del sistema	Flujo central
EC 1.1 Opción adicionar fórmula correctamente.	*El usuario adiciona correctamente una fórmula.*	*V/Intiger*	*V/Plus*	*V/valor*	*V/123*	*V/valor*	*V/456*	*El sistema procede a adicionar la fórmula.*	*Inicio/Regla/adicionar/AdicionarCondiciónSimple/Fórmula*
EC 1.2 Opción adicionar fórmula incorrectamente.	*El usuario adiciona incorrectamente una fórmula.*	*V/Intiger*	*V/Plus*	*V/valor*	*V/letras*	*V/variable*	*V/65*	*El sistema envía un mensaje indicando que se ha producido un error.*	*Inicio/Regla/adicionar/AdicionarCondiciónSimple/Fórmula*
EC 1.3 Opción adicionar fórmula dejando campos vacíos.	*El usuario adiciona una fórmula dejando campos vacíos.*	*V/vacío* *V/Intiger* *V/Intiger*	*V/Minus* *V/vacío* *V/div*	*V/valor* *V/variable* *V/vacío*	*V/256* *V/condition* *V/3*	*V/valor* *V/valor* *V/valor*	*V/5* *V/85* *V/56*	*El sistema indica que hay campos vacíos.*	*Inicio/Regla/adicionar/AdicionarCondiciónSimple/Fórmula*

		V/float *V/double* *V/Intiger*	*V/mult* *V/Plus* *V/div*	*V/valor* *V/valor* *V/variable*	*V/vacío* *V/5* *V/Fperson. Age*	*V/valor* *V/vacío* *V/valor*	*V/12* *V/3* *V/vacío*		
EC 1.4 Opción Modificar	*El usuario seleccion a la fórmula que desea modificar .*	*V/float*	*V/Min us*	*V/valo r*	*v/12*	*V/valo r*	*V/6*	*El sistema guarda las modificacion es realizadas por el usuario.*	*Inicio/R egla/Mo dificar/C ondición /Fórmul a/Editar*

Anexo 7 Resultado de las pruebas.

Tabla 21 Resultado de la primera iteración de pruebas.

Iteración 1					
Elemento	No	No Conformidad	Aspecto correspondiente	Etapa de detección del error	Importancia
RF4	1	Agregar el botón Cancelar a la interfaz.	Al adicionar una acción.	Al mostrar la interfaz.	No significativa.
RF4	2	Cambiar el término Accion de la regla por Acción de la regla.	Al adicionar una acción.	Al mostrar la interfaz.	No significativa.
RF4	3	Validar el campo valor.	Al modificar una acción.	Al enviar los datos.	No significativa.
RF1	4	Cambiar el término Modificar formula por Modificar Fórmula	Al modificar una fórmula.	Al mostrar la interfaz.	No significativa.
RF5	5	Mostrar una ventana de confirmación.	Al eliminar una regla.	Al eliminar la regla seleccionada.	Significativa.
RF	6	Agregar el botón Cancelar a la interfaz.	Al eliminar la regla.	Al mostrar la interfaz.	No significativa.
RF1	7	Cambiar el término Eliminar	Al eliminar una fórmula	Al mostrar la interfaz.	No significativa.

		formula por Eliminar fórmula			
RF1	8	Cambiar el término Terminar en el botón por Cancelar.	Al adicionar una fórmula.	Al mostrar la interfaz.	No significativa.
RF4	9	Agregar el botón Cancelar a la interfaz	Al adicionar una acción.	Al mostrar la interfaz Navegación de Reglas.	No significativa.
RF3	10	Cambiar el término Navegacion de reglas por Navegación de Reglas.	Al adicionar una condición.	Al mostrar la interfaz Navegación de Reglas	No significativa.
RF4	11	Cambiar el término finalizar en el botón por Cancelar.	Al modificar una acción.	Al mostrar la interfaz.	No significativa.
RF3	12	Cambiar el término Condicion Simple por Condición Simple.	Al modificar una condición.	Al mostrar la interfaz.	No significativa.
RF5	13	Cambiar el término tipoDato por	Al adicionar una regla.	Al mostrar la interfaz.	No significativa.

		Tipo de dato.			
RF5	14	Cambiar el término AddRule por Adicionar regla.	Al adicionar una regla.	Al mostrar la interfaz.	No significativa.
RF2	15	Adicionar el botón Cancelar a la interfaz.	Al eliminar el contexto.	Al mostrar la interfaz.	No significativa.
RF4	16	Mostrar una ventana d confirmación al eliminar la acción.	Al eliminar una acción.	Al eliminar la acción seleccionada.	No significativa.

Anexo 8 Acta de aceptación

Acta de Aceptación

Proyecto: Modernización del sistema de Identificación, Inmigración y Extranjería de Cuba (Identidad Cuba).

Producto: Sistema Único de Identificación Nacional de la Población de la República de Cuba (SUIN).

Categoría de las pruebas: Revisión a la aplicación.

Fecha de conciliación: 22 de Mayo de 2012

Observaciones del proceso:

Durante el proceso de pruebas de aceptación se detectaron un conjunto de No Conformidades (NC) y Pedidos de Cambio (PC) que fueron registrados adecuadamente en los correspondientes informes diarios de NC y PC, con sus respectivas observaciones.

Teniendo en cuenta que fue validada por el cliente la corrección de la mayoría de las No Conformidades y gestionados los Pedidos de Cambios y valorando que las No Conformidades pendientes de solución no se consideran críticas por parte del cliente, para la etapa de despliegue del sistema, se ha tomado el acuerdo de Aceptar la aplicación Sistema Único de Identificación Nacional de la Población de la República de Cuba (SUIN) en su versión 1.0 y el documento "Manual de Usuario" en su versión 1.0, todos con fecha 21 de Mayo de 2012.

El conjunto de incidencias pendientes, de las que fueron detectadas durante el proceso de pruebas de aceptación han sido especificadas en los documentos Informe Final de No Conformidades (NC) e Informe Final de Pedidos de Cambio (PC), con sus respectivas observaciones.

Página 1 de 2

Acta de Aceptación

Para que conste la Aceptación de los resultados de las pruebas y por tanto la Aceptación de los entregables especificados, dando fe del acuerdo, se extiende la presente **Acta** en tres (3) ejemplares, rubricados por los principales **Representantes de las Partes**.

Coronel Juan Manuel López Acevedo
Jefe de Dirección de Identificación y Registros

Tte Coronel Magnolia Soto Bernal
Jefe de Departamento CIRP

MsC. Tayché Capote García
Representante Centro Nacional de Calidad de Software (CALISOFT)

MsC. Yudenia Ramírez Mastrapa
Jefa del proyecto