Universidad de las Ciencias Informáticas
Facultad 1

"Sistema Calculador de Métricas para evaluar la calidad de un software en el ERP- Cuba"

Trabajo diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Alexey Talavera Calás

Tutor: Ing. Elianys Hurtado Sola

Co-Tutor: Ing. Yoandry Morejón Borbón

Ciudad de la Habana, Junio de 2010

Pensamiento

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert

Einstein.

Declaración de autoría

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Alexey Talavera Calás.

_____

Ing. Elianys Hurtado Sola.

Agradecimientos

Durante la vida de estudiante son muchas las personas que contribuyen con nuestra formación profesional y para la vida, profesores, amigos y la familia fundamentalmente. En este momento cumbre de nuestras vidas, donde nos convertimos en profesionales les agradezco sinceramente, con el mayor deseo de estar a la altura de las enseñanzas y principios que compartieron conmigo.

A Elianys Hurtado Sola, mi tutora, por su ayuda, su paciencia y su infinito apoyo. Por haber estado siempre dispuesta para mostrarme el camino.

A Noel Jesús Rivero por estar siempre dispuesto a ayudarme a aclarar mis dudas, que por cierto, no eran pocas y por su amistad.

A mi padre, por ser mi mejor amigo y mi eterno guía. A mi madre, que nunca ha dejado de enseñarme y que siempre me ha mostrado el camino para ser mejor. A mis amigos en general que supieron darme fuerzas en todo momento.

Dedicatoria

En el presente trabajo quiero hacer un humilde reconocimiento a todos aquellas mujeres y hombres que han contribuido en la formación profesional y humana a lo largo de mi vida, la misma que no sería igual sin las enseñanzas y atenciones, tanto en las aulas como fuera; los desvelos, paciencia, el perdón, y el apoyo frente a las adversidades para continuar el camino de frente, sabiendo que una caída implica la oportunidad de soportarla con esperanza, con el constante aprender de los errores propios; así como de la amistad, la cual no puedo definir pero si definir su grandeza y valía. No sería el mismo sin todos aquellos a quienes he conocido, a quienes aprecio y admiro, de quienes he recibido grandes lecciones. Por ello, aseguro que no sería mejor de lo que soy sin ellos. Agradezco en particular:

A las personas más importantes en mi vida, mi madre (Franci) y mi padre (Rogelio) cuyos esfuerzos han hecho posible este logro.

A mis abuelos (Nolberto, Marta y Lorenza).

A mis tías Rosa y Marina.

A mis primos Yoan, Yoanne.

A mi novia Bárbara.

A mis amigos.

A todos, muchas gracias, pues en el momento en el que las palabras no son suficientes para expresar lo que el alma desea, rebasa un tono, simplemente queda decir aquello que por su significado extenso y sin límites es, GRACIAS.

RESUMEN

La Industria Cubana del Software está llamada a convertirse a corto plazo, en uno de los principales renglones económicos del país. Para esto, producir un producto con excelente calidad, es la meta a seguir por las empresas productoras de software del país y también por la Universidad de las Ciencias Informáticas.

Uno de los aspectos de mayor impacto en la calidad del software son las métricas de software, las que tienen entre sus objetivos ayudar a entender qué ocurre durante el desarrollo y el mantenimiento del mismo.

Este trabajo realizado en la Universidad de las Ciencias Informáticas, brinda una solución informática a modo de aplicación web, que evalúa la calidad de software a través del cálculo de métricas. Este sistema gestiona los atributos de calidad, las métricas de software y las escalas, que son necesarios para realizar la evaluación. Además, el mismo gestiona configuraciones mediante las cuales realiza la evaluación y genera un reporte con el resultado de la misma. Está basado en tecnología PHP y utiliza PostgreSQL como gestor de base de datos. La aplicación fue desarrollada a raíz de la necesidad por parte del Grupo de Calidad del proyecto ERP- Cuba, de contar con una herramienta que le permita mantenerse informado constantemente sobre el estado de los productos que en el proyecto se desarrollan.

Para obtener la aplicación en cuestión se trazaron un conjunto de tareas necesarias para la culminación del sistema con calidad, entre las que figuran, el estudio del estado del arte sobre el tema a tratar, la realización de una propuesta para dar solución al problema, el modelado e implementación de la solución, y la validación de la misma utilizando métricas de software, lo que permitió finalmente, hacer las recomendaciones necesarias para la incorporación de futuras funcionalidades a la propuesta elaborada.

Índice

Contenido

Índice de Tablas

Capítulo 2

Capítulo 3

Índice de Imágenes

Capítulo 2

Capítulo 3

Anexos

INTRODUCCIÓN

La realización de software ha avanzado grandemente a nivel mundial, por lo que existe la necesidad de garantizarlos con calidad. La evaluación de métricas mediante un sistema automatizado representa una variante para comprobar el cumplimiento de las normas de calidad y el buen funcionamiento de una aplicación. Las métricas consisten en un grupo de medidas que ayudan al desarrollador a tener una continua información del estado del componente o proyecto en general, que se está realizando. Esto brinda la posibilidad de ir reparando los errores que ocurran en el proyecto en las distintas etapas de su creación y evita la acumulación de problemas luego de terminado el sistema. De esta forma el desarrollador puede conocer el tipo de problema que está teniendo su implementación y en qué lugar (dígase a nivel de atributo, clase, componente o sistema en general) y tiene la posibilidad de poder corregirlo con más facilidad y en el momento preciso, para continuar con el desarrollo del software.

De forma general, la calidad del software es una necesidad para todos. La calidad del software es medible y varía de un sistema a otro o de un programa a otro, puede medirse después de elaborado el producto, pero puede resultar muy costoso, por lo que es recomendable tenerla en cuenta durante todas las etapas del ciclo de vida del software, es decir, debe marchar junto con el proceso de desarrollo de software.

Según Norman E. Fenton, profesor de Ciencias de la Computación en Queen Mary (Universidad de Londres), una métrica es “una asignación de un valor a un atributo de una entidad de software, ya sea un producto o un proceso”. (1)

Las métricas son un buen medio para entender, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento. Pueden ser utilizadas para que los profesionales e investigadores puedan tomar las mejores decisiones.

La Universidad de las Ciencias Informáticas (UCI) es uno de los centros surgidos a raíz de la Batalla de Ideas. La UCI no es solo una entidad educativa, sino también productiva, que realiza múltiples proyectos destinados a la informatización de la sociedad cubana, la comercialización de software, así como a otros programas solidarios que ejecuta el país a nivel mundial. En el desarrollo de los productos, la evaluación de las métricas se realiza de forma manual, lo cual hace el trabajo más engorroso e ineficiente.

En la universidad se desarrolla actualmente el sistema Cedrux, que no es más que un sistema de planificación de recursos empresariales. Cedrux es un sistema ERP que se desarrolla con el objetivo de gestionar los recursos de una empresa, que consiste básicamente en un software capaz de manejar la

información de las empresas de manera eficaz, debido a que en el año 2008 fue reconocida por la Oficina Nacional de Informatización la existencia, en empresas cubanas, de sistemas informáticos desarrollados sobre plataformas antiguas, con pocos criterios de seguridad y casi ninguno bajo conceptos de informática multicapa y distribuida en la red. Una gran parte de estos sistemas fueron implementados utilizando arquitectura cliente-servidor, desarrollados sobre un ambiente multiusuario y casi todos, extranjeros.

Los sistemas ERP se encuentran operando actualmente en todo tipo de empresas. Se caracterizan fundamentalmente por estar conformados por múltiples elementos de una misma aplicación; compras, logística, ventas, contabilidad, gestión de proyectos, producción, sistemas de información geográfica, inventarios, control de almacenes, pedidos, nóminas, entre otras, forman el conjunto que definen a este sistema tan utilizado en el presente.

En la actualidad ya los sistemas ERP muestran al mundo ventajas explícitas y grandes posibilidades empresariales:

- Aumento de productividad de la planta o negocio.
- Reducción de inventarios.
- Incremento en ventas por tiempo de respuesta a clientes.
- Disminución de compras.
- Disminución de comisiones bancarias por cheques expedidos por órdenes.

En el proyecto ERP que se realiza actualmente en la universidad, se evalúa la calidad del los productos que se desarrollan, de igual forma que para los demás sistemas que se han desarrollado en la UCI, es decir, la evaluación de las métricas se realiza de forma manual, haciéndose necesaria la automatización del proceso de cálculo de métricas en el mismo y en aquellos proyectos de gestión que utilicen el marco de trabajo Sauxe, con el fin de acortar el tiempo de las entregas finales a los usuarios.

Debido a las ventajas que ofrece el uso de las métricas en el proceso de medición de la calidad de software y lo anteriormente abordado surge la necesidad de buscar una solución al siguiente **problema**: en el proceso de evaluación de la calidad de los productos que se desarrollan con el marco de trabajo Sauxe, la aplicación de las métricas se realiza de forma manual lo que trae como consecuencia la

ocurrencia de errores humanos, que el trabajo se vuelva más engorroso e ineficiente, además de pérdida de tiempo en la entrega de los productos finales.

El **objeto de estudio** recae sobre los atributos de calidad y métricas de software así como las herramientas que realizan el proceso de cálculo de métricas de software desarrollados con tecnología PHP, tomando como **campo de acción** el proceso de Cálculo de Métricas para el servidor de Integración Continua.

Se define como **objetivo general** del presente trabajo: desarrollar una herramienta que permita analizar la calidad de software mediante el cálculo de métricas.

Objetivos Específicos

1. Analizar los sistemas existentes que utilicen el cálculo de métricas para evaluar la calidad de software, así como las herramientas y tecnologías que se utilizarán para el desarrollo de la solución.
2. Desarrollar una herramienta que realice el cálculo y reporte de métricas de software Orientado a Objeto.
3. Probar y validar la solución.

Idea a Defender.

Con el desarrollo de una herramienta que permita analizar la calidad del software mediante el cálculo de métricas, se podría mejorar la eficiencia del proceso de evaluación de la calidad en el proyecto ERP y se podría evitar la pérdida de tiempo al entregar los productos finales.

Tareas de la Investigación.

1. Estudio de las métricas que se utilizan para obtener la calidad de los software.
2. Estudio de las soluciones realizadas respecto a los cálculos de métricas en aplicaciones.
3. Estudio de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
4. Definición de requisitos y escenarios arquitectónicos.
5. Diseño e implementación de las interfaces.
6. Diseño e implementación de la capa de negocio que dé respuesta a los requisitos.

7. Diseño e implementación de la capa de acceso a datos.
8. Implementación de las validaciones, excepciones e integración al marco de trabajo del ERP.
9. Validación de la implementación de los componentes obtenidos en la herramienta a través de métricas.
10. Realización de pruebas de concepto a los componentes obtenidos.

Para la realización de las tareas de investigación se emplearon los métodos siguientes:

Métodos Teóricos. Permiten estudiar las características del objeto de investigación que no son observables directamente.

Inductivo- Deductivo. Se realiza un estudio general de las herramientas existentes capaces de realizar el cálculo de métricas a programas desarrollados con tecnología PHP, para desarrollar un programa que permita calcular la calidad de un software mediante dicho cálculo.

El contenido de este trabajo se encuentra estructurado en tres capítulos de la siguiente forma:

En el capítulo 1, “Fundamentación teórica”, se realiza un análisis acerca del uso e importancia de las métricas, así como de las diferentes herramientas que se usan en este campo actualmente. Se enuncian las tendencias y tecnologías actuales seleccionadas a emplear en el desarrollo de la propuesta y se fundamenta el por qué de su utilización.

El capítulo 2, “Descripción y análisis de la propuesta de solución”, describe la situación actual, define la propuesta del sistema. Se definen además las funcionalidades del sistema, a través de los requerimientos funcionales y la descripción de los escenarios arquitectónicos. También aborda sobre la elaboración de la propuesta de solución y se modela la vista lógica de los procesos involucrados así como los diagramas de clases del diseño.

En el capítulo 3, “Implementación, prueba y validación de la solución” se procede con la construcción de la propuesta de solución. Se modela el diagrama de despliegue y los diagramas de componentes y se valida la propuesta mediante casos de prueba y métricas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se realiza un proceso investigativo de aspectos teóricos y conceptuales relacionados con los procesos de medición y el uso de las métricas en el desarrollo del software necesarios para la elaboración y concepción del Trabajo de Diploma. Se estudian y definen los conceptos para comprender el dominio del problema. Además se realiza un análisis de los sistemas existentes utilizados actualmente en Cuba y el mundo, relacionados con el proceso de medición de software a través del cálculo de métricas. Se muestra un estudio del lenguaje de modelado gráfico a usar y el modelo de desarrollo escogido para guiar el avance del software. Por último se realiza un análisis de las herramientas y tecnologías seleccionadas para llevar a cabo la elaboración del sistema.

1.2 Métricas de Software.

Una métrica de software no es más que la aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos para suministrar información relevante a tiempo, así el desarrollador junto con el empleo de estas técnicas mejorará el proceso y sus productos.

En general, la medición persigue tres objetivos fundamentales:

- Entender qué ocurre durante el desarrollo y el mantenimiento.
- Controlar qué es lo que ocurre en los proyectos.
- Mejorar los procesos y productos. (2)

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver los problemas. La informática aporta herramientas y procedimientos sobre los que se apoya la ingeniería de software, la cual tiene entre sus principales objetivos producir ya sea un sistema, un producto o una aplicación con alta calidad. En el cumplimiento de este objetivo las métricas juegan un papel primordial, pues evalúan el comportamiento de un atributo de calidad en el producto que se está desarrollando.

1.3 Tipos de métricas

Según el contexto en que se aplican las métricas de Software se clasifican en:

Métricas de proceso:

Se recopilan de todos los proyectos y durante un largo periodo de tiempo y se caracterizan por el control y ejecución del proyecto así como la medición de tiempos de las fases.

Métricas de proyecto:

Permiten evaluar el estado del proyecto y seguir la pista de los riesgos.

Métricas de producto:

Se centran en las características del software y no en cómo fue producido. También son productos los artefactos, documentos, modelos y componentes que conforman el software. Se miden elementos como el tamaño, la calidad, la totalidad, la volatilidad y el esfuerzo. (3)

1.4 Atributos de Calidad.

Es aquella característica diferencial que posea el producto como rango distintivo de otro producto similar y cuyo proceso de elaboración y condiciones finales de calidad, cumplan las normas establecidas en el protocolo correspondiente. (4) Entre los atributos de calidad se encuentran:

- **Disponibilidad** define la proporción del tiempo que el sistema es funcional y trabaja. Puede ser medido como un porcentaje del tiempo total en que el sistema estuvo caído en un periodo predefinido. La disponibilidad puede verse afectada por errores del sistema, problemas de infraestructura, ataques o carga del sistema.
- **Integridad Conceptual** define la consistencia y coherencia del diseño total. Esto incluye la forma en que los componentes o módulos han sido diseñados, así como factores como el estilo de codificación y la nomenclatura de las variables.
- **Flexibilidad** es la habilidad del sistema para adaptarse a ambientes y situaciones variables y para soportar cambios en políticas de negocios y reglas de negocio. Un sistema flexible es uno que es fácil de reconfigurar o que se adapta en respuesta a los diferentes requerimientos de usuarios y del sistema.

- **Interoperabilidad** es la habilidad de que diversos componentes de un sistema diferentes sistemas funcionen correctamente al intercambiar información, comúnmente por medio de servicios. Un sistema interoperable hace fácil intercambiar y usar información interna y externamente.
- **Capacidad** de mantenimiento es la habilidad de un sistema para permitir cambios en sus componentes, servicios, características e interfaces en la medida en que dichos cambios son requeridos cuando se adiciona o cambia la funcionalidad, se corrigen errores o se suplen nuevos requerimientos de negocios. Entro otros.

1.5 Sistemas existentes usados en la medición y aplicación de las métricas.

El uso de las métricas está presente en casi todas las esferas de la industria, por lo que medir ha dejado de ser algo sencillo. Ante tal situación se hizo necesario buscar algunos sistemas que brindaran una serie de mediciones cuantitativas sobre un producto de software determinado, para extraer una serie de factores de calidad que permitan al usuario conocer el estado del software analizado en diferentes niveles.

A nivel internacional

1.5.1 Process Dashboard.

Una de las herramientas utilizadas en el campo de las mediciones es el Process Dashboard (Proceso de Software Escritorio). Esta herramienta brinda a todos los miembros del equipo de desarrollo la posibilidad de medir su tiempo de trabajo de una forma sencilla, es de escritorio y solo puede ser ejecutada por una persona a la vez. Además es una aplicación para recoger métricas ya definidas, de modo que no permite la definición de nuevas métricas por parte de los usuarios finales del software. (4)

1.5.2 PHP_ CodeSniffer:

Es un script de PHP5 que señaliza y "olfatea" código PHP para detectar violaciones de un conjunto definido de normas de codificación. Se trata de un instrumento de desarrollo esencial que garantiza que el código sigue siendo limpio y coherente. Incluso puede ayudar a prevenir algunos de los errores semánticos cometidos por los desarrolladores. Dentro de PHP_ CodeSniffer pueden existir múltiples

normas de codificación que se encargan de “oler” una parte del código en especial. Es por ello que el modo de instalación se puede utilizar a través de múltiples proyectos.

Principales Ventajas:

- Indican la necesidad de aplicar la refactorización.
- Ayuda a incrementar la retroalimentación.
- Reduce el esfuerzo de comentario manual de código.
- Permite conocer el tamaño del código.

Desventajas:

- No precisa si la aplicación funciona bien, solo si el código PHP está correcto.
- Mensajes de error poco claros.
- Poca documentación oficial. (5)

1.5.3 PHPUnit:

Es una migración completa de Junit 3.8.1 a PHP5. Es utilizado por Zend Framework para la gestión de pruebas unitarias y como framework de pruebas en el Servidor de Integración Continua Xinc; puede ser anexado, además, al Servidor de Integración Continua CruiseControl para gestionar aplicaciones desarrolladas con el lenguaje de programación PHP. Definido en el “Test Anything Protocol”, creado para facilitar la lectura de los resultados de las pruebas.

Principales Ventajas:

- Soporte para Mock Objects (objeto que representa a otro y lo sustituye en funcionalidad).
- Soporte para pruebas sobre base de datos
- Forma parte del grupo de frameworks de xUnit.
- Se integra con varias aplicaciones de test.
- Soporte para realizar cálculo de métricas.

Desventajas:

- Las pruebas deben ser creadas manualmente por el desarrollador, lo que provoca pérdidas de tiempo en el proceso de desarrollo.
- No posee interfaz visual.
- Incluye cálculo de métricas como una herramienta alternativa, pero no como funcionalidad principal. (6)

1.5.4 Cocomo(COnstructive COst Model):

Es un modelo matemático de base empírica utilizado para estimación de costes de software. Incluye tres submodelos, cada uno ofrece un nivel de detalle y aproximación, cada vez mayor.

Principales Ventajas:

- Pertenece a la categoría de modelos de subestimaciones basados en estimaciones matemáticas.
- Está orientado a la magnitud del producto final.
- Mide el "tamaño" del proyecto, en líneas de código principalmente.

Desventajas:

- No está implementado en ningún lenguaje específico.
- Se miden los costes del producto, de acuerdo a su tamaño y otras características, pero no la productividad.
- La medición por líneas de código no es válida para la programación Orientada a Objetos. (7)

1.5.5 A nivel nacional.

La investigación realizada arrojó como resultados que no existen en Cuba, ni en la UCI, softwares que realicen el proceso de cálculo de métricas que puedan ser integrados al marco de trabajo Sauxe, por lo que el cálculo de métricas en el proyecto ERP se realiza manualmente lo que hace el trabajo más engorroso e ineficiente.

A partir del estudio realizado de los sistemas existentes, se concluyó que, Process Dashboard es una aplicación de escritorio que solo puede ser ejecutada por una persona a la vez. No brinda un espacio de uso colectivo. No permite la definición de nuevas métricas. PHP_ CodeSniffer no precisa si la aplicación

funciona bien, solo si el código PHP está correcto. Los mensajes de error no aparecen claramente y existe muy poca documentación oficial. En PHPUnit las pruebas deben ser creadas manualmente por el desarrollador, lo que provoca pérdidas de tiempo en el proceso de desarrollo, no posee interfaz visual e incluye el cálculo de métricas como una herramienta alternativa, pero no como funcionalidad principal. Cocomo(COnstructive COst Model) no está implementado en ningún lenguaje específico. En él se miden los costos del producto, de acuerdo a su tamaño y otras características, pero no la productividad y su medición por líneas de código no es válida para la programación Orientada a Objetos. Teniendo en cuenta las desventajas antes mencionadas se decidió realizar una herramienta capaz de realizar el cálculo y reporte de métricas para evaluar la calidad de un software en el proyecto ERP- Cuba, dicho componente contará con una interfaz gráfica para el intercambio con el usuario que se incluye dentro del marco de trabajo Sauxe y además tendrá su módulo correspondiente en la base de datos de éste para realizar las salvas y conexiones pertinentes.

1.6 Herramientas y tecnologías a utilizar para el desarrollo de la herramienta.

Los sistemas de gestión representan un eslabón fundamental en el control de la información en el mundo de hoy. El creciente uso de la Internet en los diferentes ámbitos de la vida diaria, ha hecho que un gran número de las herramientas y tecnologías utilizadas giren en torno a elementos como las aplicaciones Web y los sistemas gestores de bases de datos. A continuación se muestran diferentes herramientas y tecnologías a utilizar, así como el modelo de desarrollo escogido para el desarrollo de la aplicación.

1.6.1 Tecnología a utilizar en la capa de presentación.

ExtJS

Para el desarrollo de las vistas se utilizará ExtJS. Es una librería Javascript para la creación de aplicaciones enriquecidas del lado del cliente. Sus características principales son: Gran desempeño, componentes de interfaz de usuario personalizables, con buen diseño y documentación. Es un framework que permite hacer interfaces bastante poderosas y amigables para sistemas sobre internet.

Extjs soporta:

- Internet Explorer 6+
- FireFox 1.5+ (PC, Mac)

- Safari 3+(PC,Mac)
- Opera 9+ (PC, Mac)

Algunas de las cualidades de Extjs:

- Alto rendimiento.
- Interfaces personalizables.
- Muy buena arquitectura que permite extender los componentes gracias a su buen modelado.
- API (interfaz de programación de aplicaciones) intuitiva.
- Licencia comercial y open source (código abierto) gratis. (8)

Tiene incluidos la mayoría de los controles de los formularios Web incluyendo tablas para mostrar datos y elementos semejantes a la programación de escritorio como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería de componentes incluye componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX. Presenta el uso de JavaScript con una programación orientada a objetos.

1.6.2 Tecnología a utilizar en la capa de negocio.

PHP

Es un lenguaje script para el desarrollo de páginas Web dinámicas del lado del servidor. Es Open Source (código abierto), interpretado y de alto nivel, especialmente pensado para desarrollos Web.

Ventajas:

- Lenguaje robusto y estable.
- Código sencillo de aprender.
- Acompañado por una excelente biblioteca de funciones.
- Capacidad de ejecución en la mayoría de los sistemas operativos.
- Lenguaje multiplataforma.

- Capacidad de conexión con la mayoría de los manejadores de base de datos como MySQL, MSSQL, Oracle, Informix, PostgreSQL y otros muchos, con un excelente soporte.
- Utilizado como módulo de Apache, lo hace extremadamente veloz.
- Está completamente escrito en C, por lo que se ejecuta rápidamente utilizando poca memoria.
- Soporta en cierta medida la programación orientada a objeto.
- Realiza de forma automática el análisis léxico para recoger las variables que se pasan en la dirección, lo cual libra al usuario de tener que separar las variables y sus valores. (9)

Entre otras ventajas, PHP es un lenguaje gratuito y multiplataforma que permite al cliente interactuar con la aplicación de forma rápida, segura y eficiente. Cuenta con una amplia librería de funciones que permiten realizar cualquier tipo de operaciones, como procesamiento de formularios, trabajo con archivos y carpetas, entre otras que le brindan al desarrollador la posibilidad de realizar un producto a la altura de las necesidades del usuario.

1.6.3 Marco de Trabajo.

Los Frameworks (marco de trabajo) ayudan en el desarrollo de software, proporcionan una estructura definida la cual ayuda a crear aplicaciones con mayor rapidez. Ayuda a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la aplicación. Son desarrollados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura a sus proyectos. Se utiliza la Programación Orientada a Objetos (POO), permitiendo la reutilización de nuestro código. Un Framework es un conjunto de componentes (por ejemplo clases en java, descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. (10)

Comenzar desde cero, no es la mejor opción a la hora de desarrollar un software, por lo que el uso de un framework ó marco de trabajo, siempre será una alternativa viable para cualquier solución. En el desarrollo de software un framework no es más que una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los

diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. El creciente uso de la programación web y del software libre ha traído consigo que actualmente exista una gran variedad de frameworks orientados a diferentes lenguajes.

1.6.4 Marcos de Trabajo a utilizar.

Doctrine Framework

Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado. Entre muchas otras cosas tienes la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.

Por otro lado, como la librería es bastante grande ésta tiene un método para ser 'compilada' al pasar a producción. (11)

Ventajas:

- Accede a la base de datos utilizando la programación orientada a objetos a través del patrón Active Record.
- Tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente (lenguaje de consulta DQL).
- Fácil integración a los principales frameworks de desarrollo utilizados actualmente.

Zend Framework 1.6.1.

Zend Framework se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP, te brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source (código abierto) y trabaja con PHP 5.

Características:

- Trabaja con MVC (Model View Controller).

- Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo), etc.
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- Completa documentación y tests de alta calidad.
- Un buscador compatible con Lucene.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron. (12)

Zend_Ext Framework.

Es un framework de código abierto, diseñado para PHP 5 o superior. Se deriva de Zend Framework por lo que cumple con todas sus características. Posee un controlador vertical para el control de las acciones realizadas por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor y se le agregó el IoC para la integración entre los módulos o componentes. Tiene incorporado el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos y el ExtJS Framework para el desarrollo de las vistas.

Ventajas:

- Comunicación entre sesiones (Global Concept).
- Abstracción de las operaciones transaccionales (Transaction Manager).
- Notificaciones declarativas a eventos, software y personas (Trazas).
- Emulación de eventos (Trazas).
- Evaluador de rendimiento (Trazas).
- Extensión de los dominios de tipo (Validador).
- Basada en componentes.
- Principios de integración distribuidos.

- Conectores y configuraciones mediante interfaces bien definidas.
- Integración por IoC y con posibilidad de emulación.

UCID Framework.

Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJs Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación y el multilenguaje.

1.6.5 Fundamentación del modelo de desarrollo a utilizar.

Un modelo de desarrollo establece el orden en el que se harán las cosas en el proyecto. Para realizar la herramienta el proceso de desarrollo estará guiado por el modelo de desarrollo definido por la Subdirección Técnica del CEIGE el que se basa en los principios y buenas prácticas de las metodologías ágiles como Extreme Programming (XP), SCRUM, Crystal Methodologies, entre otras.

Características:

- Se utilizan solamente los artefactos necesarios para documentar el producto.
- Se basa en la reutilización de componentes.
- Se desarrollan partes pequeñas y se ensambla después el producto.
- Los flujos se integran a través de la arquitectura de software y de negocio. Todos los flujos de desarrollo de cada fase se integran siempre detrás de la arquitectura de software y de negocio.
- Existen áreas dedicadas a tareas específicas y especializadas en temas específicos.
- Se hacen pruebas continuas sobre los componentes y/o productos y los cambios se hacen a tiempo. Antes de subir un componente en al repositorio se hacen pruebas unitarias y cuando se va a utilizar como parte de otro producto se hacen pruebas de integración, al igual que antes de liberar el producto también. Todo esto demuestra que se está probando en todo el proceso de desarrollo.
- Las áreas de proceso están especializadas, en temas de apoyo a la producción que es el elemento fundamental de la Subdirección y llevan unido al proceso productivo procesos tales como Investigación, Formación, Gestión del capital humano y calidad.
- Es un método muy estructurado que funciona bien con gente de poca experiencia.
- Reduce los riesgos ya que:

- Provee visibilidad sobre el progreso a través de sus nuevas versiones.
- Provee retroalimentación a través de la funcionalidad mostrada.
- Permite atacar los mayores riesgos desde el inicio.

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo y según esto ajusta su comportamiento.
- Preocupación por el aprendizaje de los desarrolladores. (13)

1.6.6 Herramienta a utilizar en el desarrollo de la capa de presentación.

NetBeans IDE 6.7.

NetBeans IDE es una aplicación de código abierto ("open source") diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. NetBeans IDE dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y sus funcionalidades son ampliables mediante la instalación de packs. (14)

1.6.7 Herramienta a utilizar en el desarrollo de la capa de negocio.

Zend Studio.

Zend Studio es el único entorno de desarrollo integrado (IDE) disponible para desarrolladores profesionales que ofrecen las capacidades necesarias para desarrollar aplicaciones de negocio. Características como la refactorización, la generación de código, asistente de código y análisis semántico se combinan para permitir el desarrollo rápido de aplicaciones. (15)

Incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Zend Studio nos permite agilizar el desarrollo web y permite simplificar proyectos complejos.

Características:

- Excelente completamiento de código, coloreado en la sintaxis del código.

- Administración avanzada de proyectos, múltiples lenguajes, incorpora el Framework de Zend, PHP Documentor, manual de PHP. Integración con subversión, los navegadores, integración avanzada con FTP.
- Soporte para Web Services, PHP4, PHP5 y SQL. (16)

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. (17)

1.6.8 Gestores de bases de datos.

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System) 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 manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Un SGBD debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Las características de un Sistema Gestor de Base de Datos SGBD son:

Abstracción de la información: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias. (18)

1.6.9 Servidor Web

Un servidor Web es un programa que implementa el protocolo HTTP (Hypertext Transfer Protocol). Se encarga de atender las peticiones HTTP llevadas a cabo por un cliente y responder con el contenido que el cliente solicita. Este protocolo HTTP está diseñado para transferir lo que se conoce como hipertextos, páginas Web o páginas HTML. Entre los servidores Web más usados a nivel mundial están Wamp Server, el Apache y el Internet Information Server.

1.6.10 Herramienta a utilizar en la capa de acceso a dato.

PostgreSQL

Es el sistema gestor de base de datos a utilizar dado que ofrece muchas ventajas respecto a otros sistemas de bases de datos. PostgreSQL es multiplataforma, extensible y está diseñado para ambientes de alto volumen. Además de su estabilidad y confiabilidad, cuenta con herramientas gráficas de diseño y administración de bases de datos.

PostgreSQL es el servidor de bases de datos de código abierto más utilizado por aquellos programadores que realizan aplicaciones cliente/servidor, complejas o críticas. Es además la alternativa más cercana a MySQL cuando se precisa de operaciones avanzadas como transacciones, procedimientos almacenados, vistas, o cuando se precisa de una base de datos que soporte gran cantidad de información. Este servidor es muy utilizado actualmente y es una opción económica a SQL Server, pues su costo es menor y las prestaciones son similares. El mismo se puede utilizar sobre cualquier sistema operativo, característica que lo pone por encima de SQL Server y a la par con MySQL. Algunas de sus principales características son:

1. Es libre.
2. Alta concurrencia.
3. Amplia variedad de tipos nativos.
4. Integridad transaccional.
5. Herencia de tablas.
6. Replicación que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica.

7. Interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby.
8. Procedimientos almacenados. (19)

1.6.11 Servidor Web a utilizar.

Wamp Server

Debido a las características que presenta el componente que se desea construir y a la factibilidad del servidor Wamp Server, el mismo ha sido el seleccionado para llevar a cabo el desarrollo de la aplicación.
Wamp Server es una de las alternativas gratuitas más usadas para la gestión de servidores de prueba locales y resulta una completa alternativa al conocido AppServ también de uso libre.
Algunas de las características de Wamp Server son:

- Incluye PhpMyAdmin.
- Servidor Web con soporte para Windows.
- Se pueden Instalar todos los CMS disponibles (Alfresco, Joomla!, Drupal, Redmine, Mediawiki, Trac, WordPress, Roller, DokuWiki, Subversion, SugarCRM, Flux, TangoCMS, Dot Clear, bbpress, etc).
- Incluye Apache, MySQL, PHP.
- Fácil instalación y Administración.
- Bajo consumo de Memoria.
- Actualizaciones continuas. (20)

1.6.12 Lenguaje de modelado y estándar gráfico a utilizar.

UML

El lenguaje para modelamiento unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Fue originalmente concebido por la Corporación Rational Software y tres de los más prominentes métodologistas en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh, e Ivar Jacobson ("The Three Amigos"). El lenguaje ha ganado un significante soporte de la industria de varias

organizaciones vía el consorcio de socios de UML y ha sido presentado al Object Management Group (OMG) y aprobado por éste como un estándar. (21)

BPMN

Es un conjunto de tecnologías y estándares para el diseño, ejecución, administración y supervisión de procesos de negocio. Business Process Modeling Notation (Notación de Modelo de Procesos del Negocio) es un estándar gráfico para crear diagramas de flujo, que sean fácilmente comprensibles por todos los accionistas de la empresa como los analistas de negocios, desarrolladores técnicos y gerentes de empresas.

El estándar BPMN es gestionado por la OMG, la misma organización que administra UML. Los elementos y las reglas para los diagramas de BPMN son muy similares a los diagramas de actividad UML, proporcionando una transición natural, fácil de usar. (22)

1.6.13 Herramienta a utilizar para el modelado.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Entre sus principales características se encuentran:

1. Es multiplataforma.
2. Soporte de UML versión 2.1.
3. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
4. Modelado colaborativo con CVS y Subversion (nueva característica).
5. Interoperabilidad con modelos UML2 (meta modelos UML 2.x para plataforma Eclipse) a través de XMI.
6. Ingeniería de ida y vuelta.
7. Ingeniería inversa - Código a modelo, código a diagrama.

8. Generación de código - Modelo a código, diagrama a código.
9. Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
10. Diagramas EJB - Visualización de sistemas EJB.
11. Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
12. Diagramas de flujo de datos.
13. Soporte ORM - Generación de objetos Java desde la base de datos.
14. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
15. Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
16. Generador de informes para generación de documentación.
17. Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
18. Importación y exportación de ficheros XMI. (23)

1.6.14 Herramienta a utilizar para el control de versiones.

TortoiseSVN

Es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos y quién hizo el cambio.

Las principales características de TortoiseSVN son:

- **Integración con el shell de Windows**: TortoiseSVN se integra perfectamente en el shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya conoce.
- **Iconos sobreimpresionados:** El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.
- **Fácil acceso a los comandos de Subversion**: Todos los comandos de Subversion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí. (24)

1.6.15 Navegador Web a utilizar.

Mozilla Firefox 3.6

Es un navegador de Internet libre y de código abierto. Es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva, marcadores dinámicos y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. Además se pueden añadir funciones a través de complementos desarrollados por terceros.

Las características de Mozilla Firefox 3.6 son las siguientes:

- Multiplataforma.
- Cuenta con una protección antiphishing, antimalware e integración con el antivirus.
- La navegación por pestañas.
- Bloqueador de ventanas emergentes.
- Múltiples Extensiones.
- Incluye un buscador integrado en la interfaz que hace búsquedas en Google.
- Posee gestor de descargas.
- Utiliza el sistema SSL para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS. (25)

1.7 Conclusiones.

En este capítulo se realizó un estudio de los diferentes sistemas nacionales e internacionales que pueden ser utilizados para realizar cálculo de métricas en el proceso de evaluación de la calidad de un producto de software. Se sentó el basamento teórico de las herramientas y tecnologías a utilizar, propuestas por el equipo de Arquitectura del Centro de Soluciones de Gestión; teniendo como principales recursos las tecnologías libres, elemento fundamental en las nuevas concepciones de informatización en nuestro país.

CAPÍTULO 2: DESCRIPCION, ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN.

2.1 Introducción.

El presente capítulo identifica y fundamenta las características de la solución propuesta, especificando aspectos como el flujo actual del proceso a informatizar. Se listan los requisitos funcionales y no funcionales que debe tener el sistema propuesto, permitiendo tener una concepción general del mismo, además de describirlo en términos de escenarios y procesos. También se realiza la representación del sistema a través de sus diagramas de clases del diseño.

2.2 Propuesta del sistema.

La herramienta calculador de métricas del proyecto ERP-Cuba, es diseñada con el objetivo de evaluar mediante la Web los atributos de calidad en los productos de software que se desarrollan en el mismo, permitiendo que un usuario pueda tener acceso a la herramienta mediante cualquier ordenador. Además se tiene en cuenta que el cálculo de métricas se realiza de forma manual, lo cual hace el trabajo más engorroso e ineficiente, provocando pérdida de tiempo al entregar los productos finales.

La informatización del proceso de cálculo de métricas en el proyecto ERP busca cubrir las necesidades de los desarrolladores y el personal de calidad en cuanto a la evaluación de la calidad para todos los componentes que se desarrollen con el marco de trabajo que utiliza el sistema Cedrux.

Para dar solución a la problemática actual, se propone elaborar una herramienta que tendrá como objetivo principal evaluar la calidad de software mediante el cálculo de métricas. La misma permitirá al usuario configurar los nomencladores de atributo, métrica y escala, necesarios para el funcionamiento de la aplicación. De igual forma el usuario será capaz de utilizar los nomencladores para gestionar configuraciones mediante las cuales se realizará el proceso de evaluación de un producto y se generará un reporte con el resultado de la evaluación.
La arquitectura del sistema responderá al MVC, el cual estructura el sistema de una forma lógica.

2.3 Modelo Conceptual.

Siguiendo lo planteado por el modelo de desarrollo se decidió realizar un modelo conceptual para mayor entendimiento del sistema que se desea realizar. Un modelo conceptual es aquel que agrupa y define los principales conceptos de un determinado sistema y sus relaciones. Su objetivo es señalar los elementos más significativos para un mayor entendimiento de la problemática descrita. En otras palabras, describe la forma en que piensan las personas que dan solución a un problema.

En la siguiente figura se muestra el diagrama conceptual correspondiente a este sistema, como parte de su modelo conceptual, en él se representan los principales conceptos que describen la solución de la herramienta Métricas, así como sus relaciones y objetivos fundamentales. El estudio y análisis de este diagrama puede ser muy útil para entender el funcionamiento del sistema descrito, partiendo de las principales funcionalidades que dispone y el objetivo de cada una de ellas, hasta llegar a los detalles más básicos de su implementación.

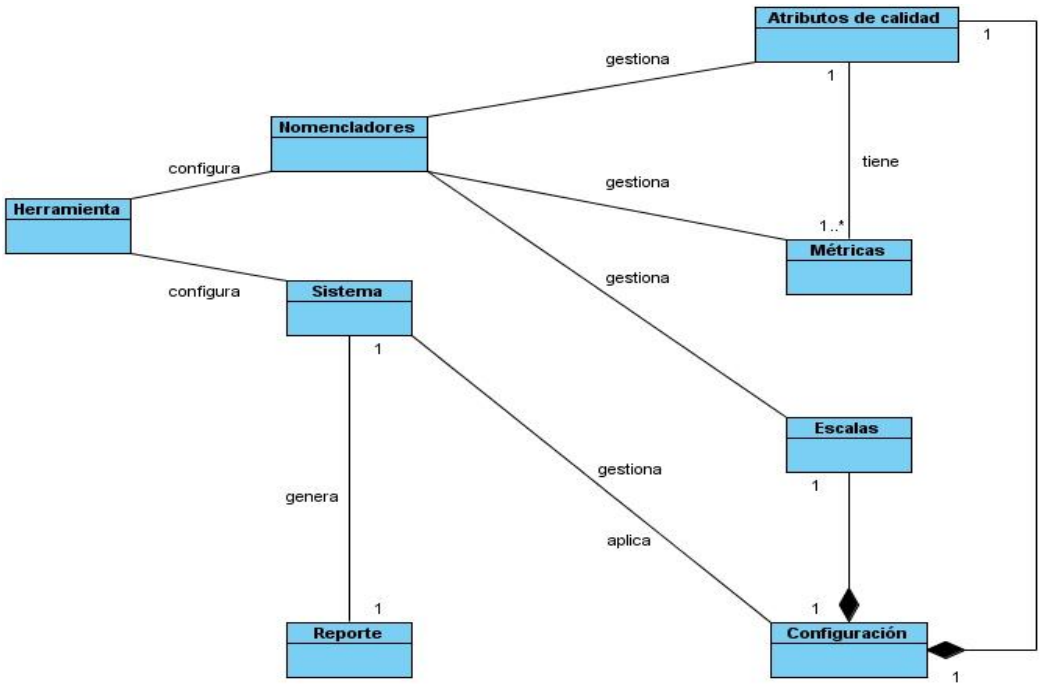


Figura 1: Modelo de dominio de la solución.

2.3.1 Conceptos principales del dominio.

- **Herramienta:** Programa utilizado para el desarrollo de software y el mantenimiento de sistemas.
- **Sistema:** Conjunto de elementos con relaciones de interacción e interdependencia que le confieren entidad propia al formar un todo unificado.
- **Nomencladores:** Es un objeto que se crea con el objetivo de ser utilizado en futuras operaciones.
- **Atributo de calidad:** Aquella característica diferencial que posea el producto como rango distintivo de otro producto similar y cuyo proceso de elaboración y condiciones finales de calidad, cumplan las normas establecidas en el protocolo correspondiente.
- **Métricas:** Es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. (26)
- **Escalas:** Rango numérico que se establece con el fin de compararlo con el resultado de una operación.
- **Configuración:** Adaptar una aplicación software o un elemento hardware al resto de los elementos del entorno y a las necesidades específicas del usuario. (27)
- **Reporte:** El reporte es aquel documento que se utilizará cuando se quiera informar o dar noticia acerca de una determinada cuestión. (28)

2.4 Descripción de los procesos.

Un proceso es un conjunto de actividades que se realizan desde un estado inicial hasta un estado final con el fin de conocer no solo estos estados, sino también las interacciones experimentadas por el sistema mientras está en comunicación con el usuario.

- **Proceso Crear Atributo:** En este proceso el usuario tiene la posibilidad de adicionar un atributo de calidad. Para adicionar, el usuario debe introducir la denominación del atributo y la fórmula con que se evaluará. El sistema validará los datos y si están correctos los adiciona. El proceso termina cuando el sistema muestra un mensaje diciéndole al usuario que el atributo fue adicionado.
- **Proceso Modificar Atributo:** Es el proceso donde el usuario puede cambiar los datos registrados en un atributo de calidad. Para que este inicie el sistema debe contar con atributos ya adicionados. El usuario escoge uno y luego comienza el proceso de modificación. En este los datos de la

denominación y la fórmula se modifican y finaliza cuando el sistema muestra un mensaje diciendo que el atributo fue modificado.

- **Proceso Eliminar Atributo:** Este proceso le permite al usuario eliminar un atributo de calidad existente en el sistema. Se inicia cuando el usuario escoge un atributo para eliminarlo. El sistema pide la confirmación de que realmente se desea eliminar ese atributo. Si el usuario confirma, entonces el sistema elimina el atributo y le muestra un mensaje al usuario diciéndole que el mismo fue eliminado.
- **Proceso Crear Métrica:** En este proceso el usuario tiene la posibilidad de definir cuáles métricas utilizará el sistema para analizar la calidad de un software. En el proceso de creación, el usuario tendrá la oportunidad de definir la denominación, la fórmula y el tipo de la métrica que será utilizada en la evaluación. El sistema validará los datos y si están correctos los adiciona. El proceso termina cuando el sistema muestra un mensaje diciéndole al usuario que la métrica fue adicionada.
- **Proceso Modificar Métrica:** Es el proceso donde el usuario puede cambiar los datos registrados en una métrica. Para que este inicie el sistema debe contar con métricas ya adicionadas. El usuario escoge una y luego comienza el proceso de modificación. En este los datos de la denominación, la fórmula y el tipo se modifican y finaliza cuando el sistema muestra un mensaje diciendo que la métrica fue modificada.
- **Proceso Eliminar Métrica:** Este proceso le permite al usuario eliminar una métrica existente en el sistema. Se inicia cuando el usuario escoge una métrica para eliminarla. El sistema pide la confirmación de que realmente se desea eliminar esa métrica. Si el usuario confirma, entonces el sistema la elimina y le muestra un mensaje al usuario diciéndole que la misma fue eliminada.
- **Proceso Crear Escala:** En este proceso el usuario tiene la posibilidad de definir cuáles escalas utilizará el sistema para analizar la calidad de un software. En el proceso de creación, el usuario tendrá la oportunidad de definir la denominación, el valor de bien, el valor de regular y el de mal de la escala que será utilizada en la evaluación. El sistema validará los datos y si están correctos los adiciona. El proceso termina cuando el sistema muestra un mensaje diciéndole al usuario que la métrica fue adicionada.
- **Proceso Modificar Escala:** Es el proceso donde el usuario puede cambiar los datos registrados en una escala. Para que este inicie el sistema debe contar con escalas ya adicionadas. El usuario

escoge una y luego comienza el proceso de modificación. En éste, los datos de la denominación, bueno, regular, y malo, se modifican, y finaliza cuando el sistema muestra un mensaje diciendo que la escala fue modificada.

- **Proceso Eliminar Escala:** Este proceso le permite al usuario eliminar una escala existente en el sistema. Se inicia cuando el usuario escoge una escala para eliminarla. El sistema pide la confirmación de que realmente se desea eliminar esa escala. Si el usuario confirma, entonces el sistema la elimina y le muestra un mensaje al usuario diciéndole que la misma fue eliminada.
- **Proceso Crear Configuración:** En este proceso el usuario tiene la posibilidad de adicionar una nueva configuración al sistema. Este proceso consiste en crear una configuración que contará con una denominación, un atributo y una escala que hayan sido definidos anteriormente por el usuario. Para crearla el usuario debe introducir los datos anteriores y luego el sistema deberá validarlos. Finalmente el sistema mostrará un mensaje diciendo que la configuración fue adicionada.
- **Proceso Modificar Configuración:** Es el proceso donde el usuario puede cambiar los datos registrados en una configuración. Para que este inicie el sistema debe contar con configuraciones ya adicionadas. El usuario escoge una y luego comienza el proceso de modificación. En este los datos de la denominación, del atributo y de la escala se modifican y finaliza cuando el sistema muestra un mensaje diciendo que la configuración fue modificada.
- **Proceso Eliminar Configuración:** Este proceso le permite al usuario eliminar una configuración existente en el sistema. Se inicia cuando el usuario escoge una configuración para eliminarla. El sistema pide la confirmación de que realmente se desea eliminarla. Si el usuario confirma, entonces el sistema la elimina y le muestra un mensaje al usuario diciéndole que la misma fue eliminada.
- **Proceso Generar Reporte:** Es el encargado de realizar el proceso de evaluación utilizando la configuración que considere más efectiva para evaluar un proyecto. El mismo da la posibilidad al usuario de introducir los valores de las variables de la fórmula del atributo seleccionado, para luego proceder a ejecutar la evaluación. El sistema internamente calcula las funciones y compara el resultado final con los valores asociados a la escala. Finalmente el sistema muestra el resultado de la evaluación.

2.5 Vista de procesos.

Los procesos describen un conjunto de actividades requeridas para desarrollar un sistema. A continuación se muestra la vista de procesos donde se describen de forma gráfica las actividades que son desarrolladas entre el usuario y el sistema.

2.5.1 Proceso de creación de un atributo de calidad.

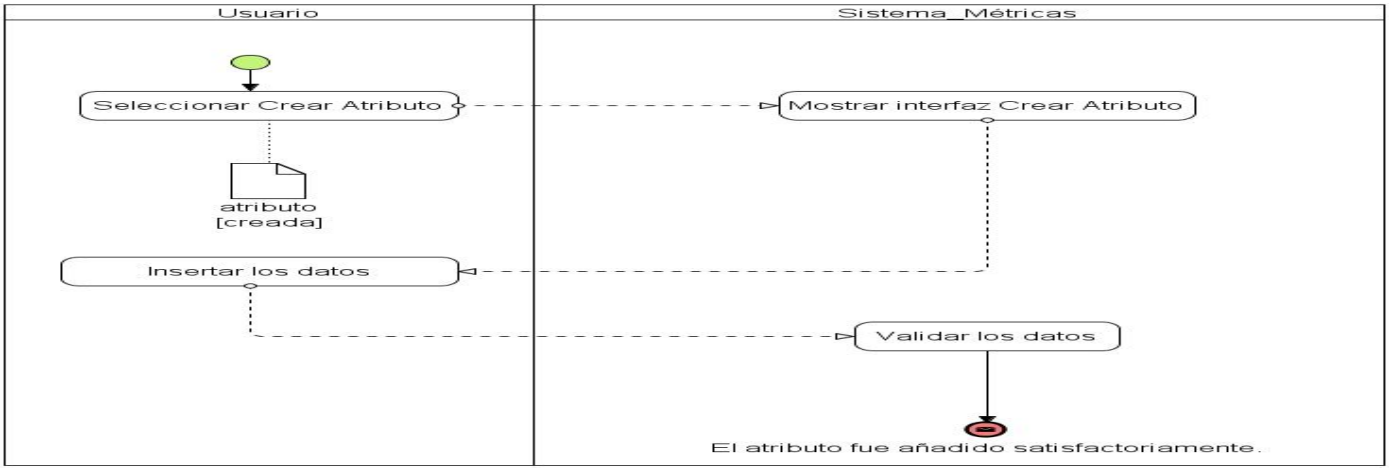


Figura 2.1: Vista de proceso para crear un atributo de calidad.

2.5.2 Proceso de modificación de un atributo de calidad.

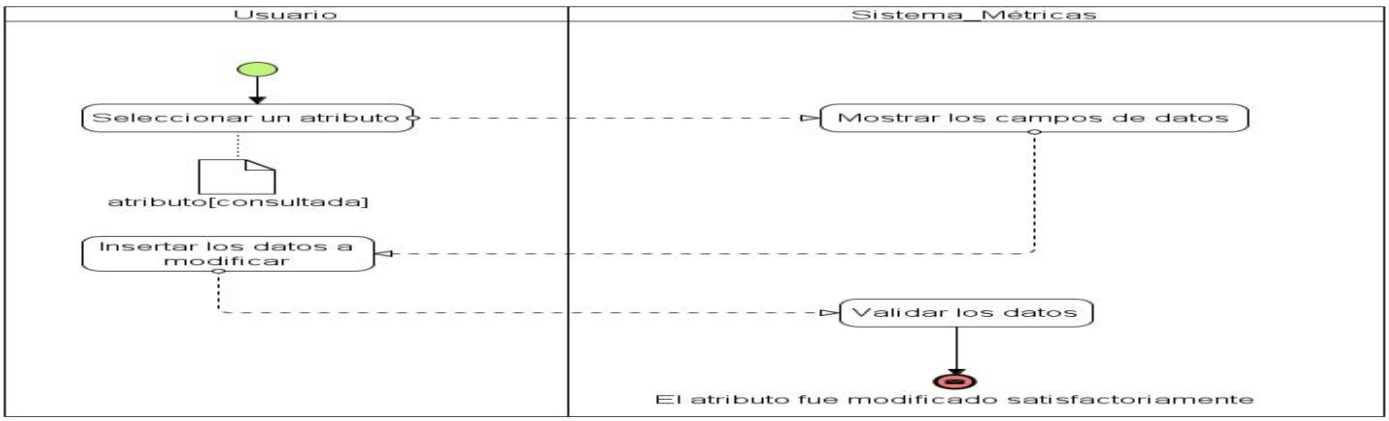


Figura 2.2: Vista de proceso para modificar un atributo de calidad.

2.5.3 Proceso de eliminación de un atributo de calidad.

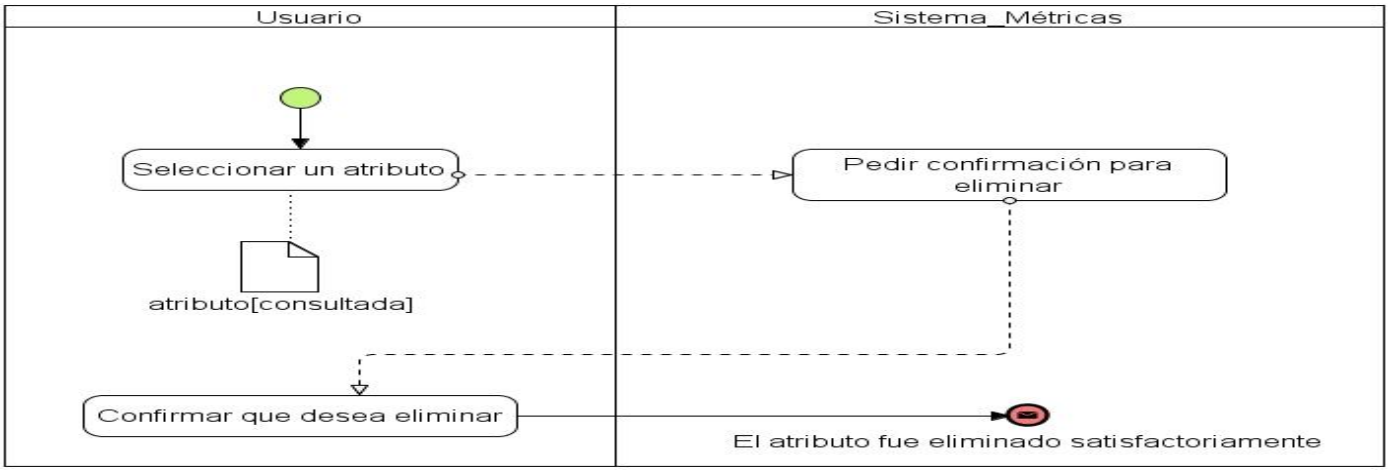


Figura 2.3: Vista de proceso para eliminar un atributo de calidad.

2.6 Vista lógica.

La vista lógica brinda un panorama de la arquitectura desde varios enfoques como puede ser del negocio, de la aplicación, de la información, etc. Estos diagramas representan la lógica de los procesos que implementa el sistema, así como las acciones realizadas por el usuario.

Proceso Crear Atributo.

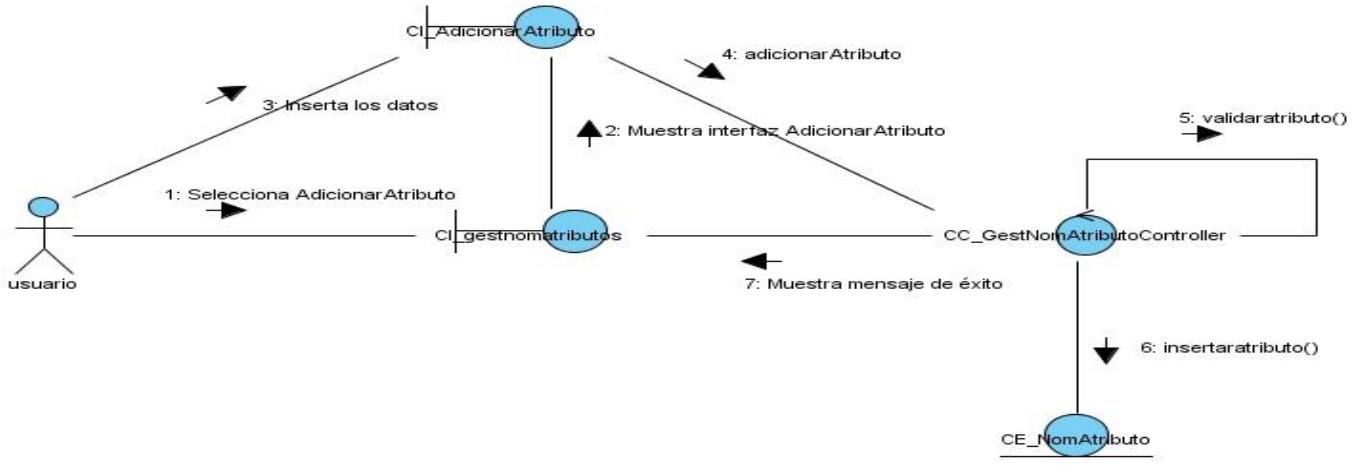


Figura 3.1: Vista lógica del proceso de creación de un atributo.

Proceso Modificar Atributo.

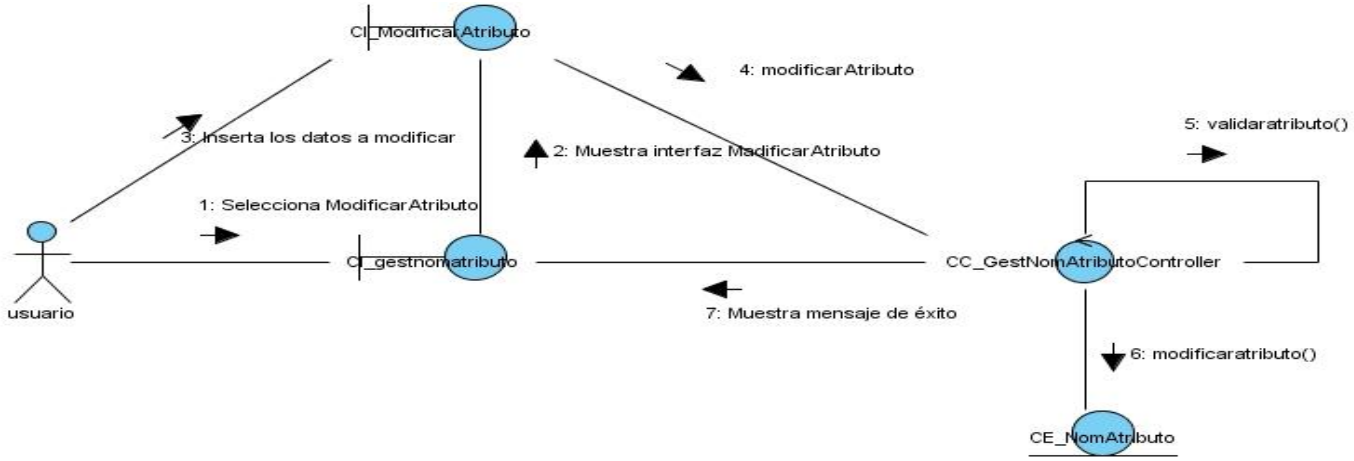


Figura 3.2: Vista lógica del proceso de modificación de un atributo.

Proceso Eliminar Atributo.

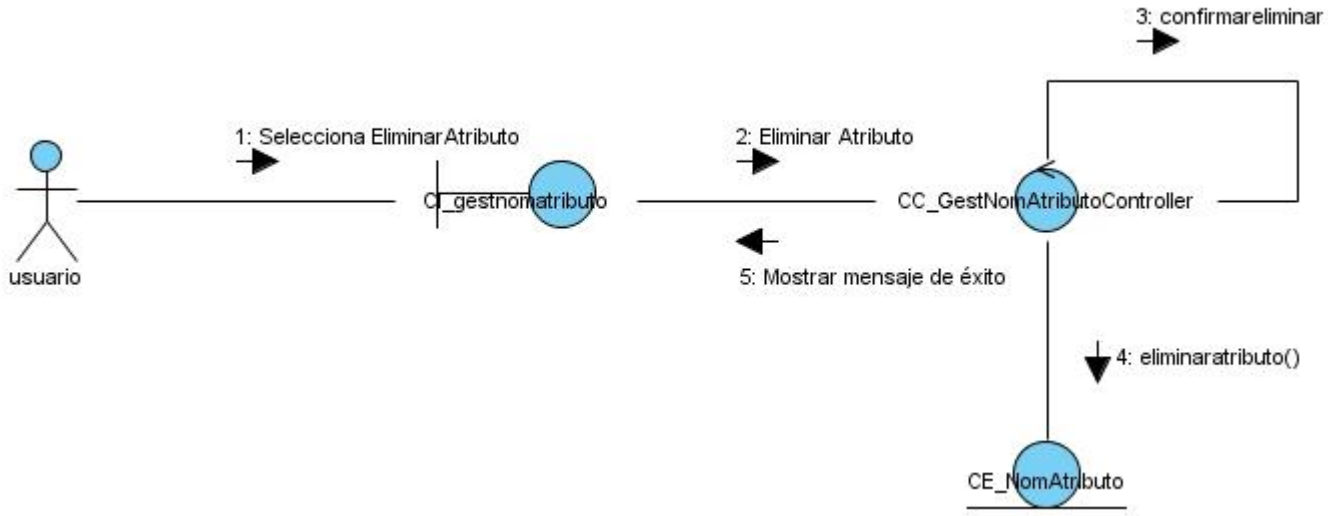


Figura 3.3: Vista lógica del proceso de eliminación de un atributo.

2.7 Requerimientos funcionales de la herramienta propuesta.

- RF 1: Gestionar Atributos de calidad.

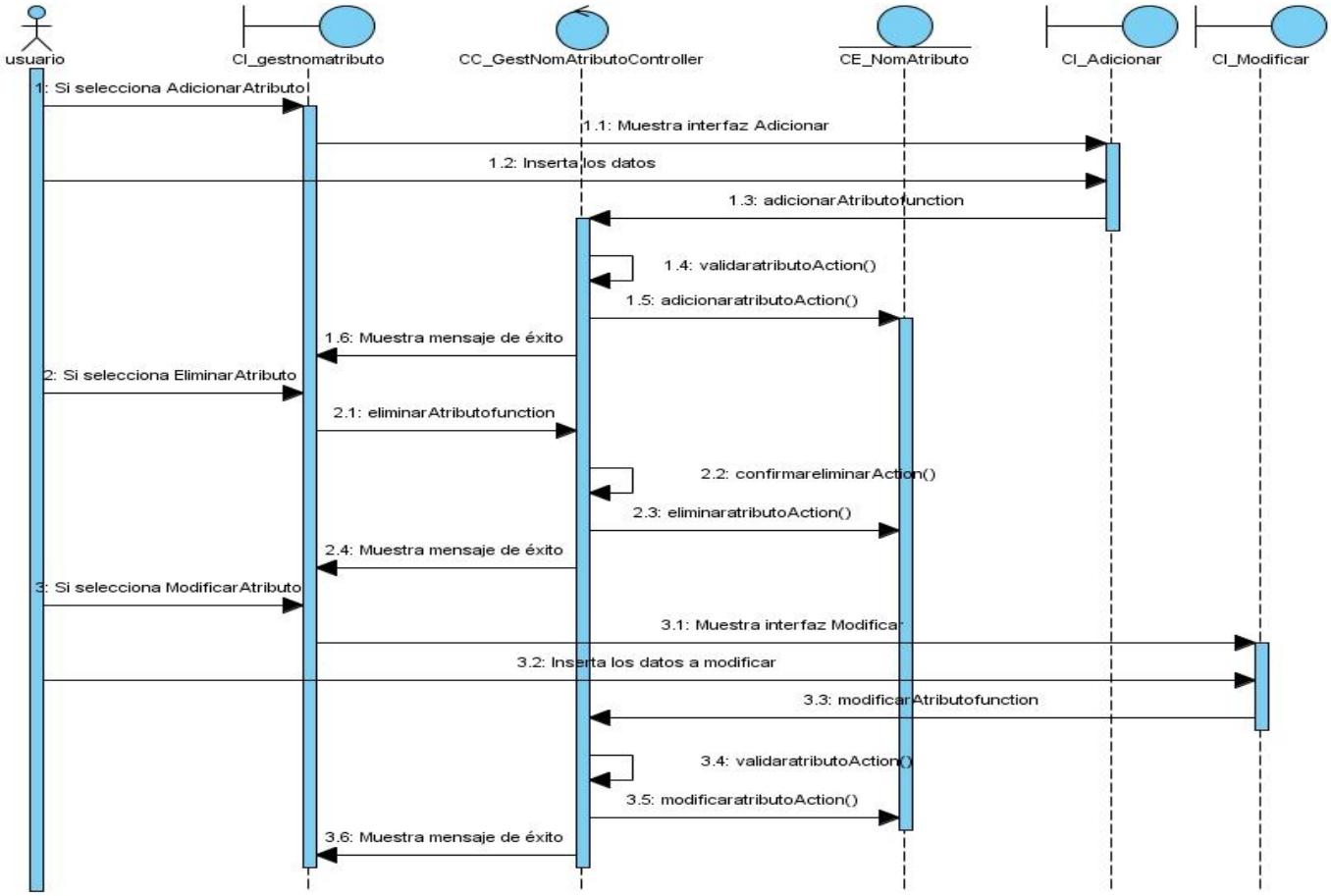
- RF 1.1: Adicionar Atributo de calidad.
- RF 1.2: Modificar Atributo de calidad.
- RF 1.3 Eliminar Atributo de calidad.

- RF 2: Gestionar Métricas.
  - RF 2.1: Adicionar Métricas.
  - RF 2.2: Modificar Métricas.
  - RF 2.3 Eliminar Métricas.
- RF 3: Gestionar Escala.
  - RF 3.1: Adicionar Escala.
  - RF 3.2: Modificar Escala.
  - RF 3.3: Eliminar Escala.
- RF 4: Gestionar Configuración.
  - RF 4.1: Adicionar Configuración.
  - RF 4.2: Modificar Configuración.
  - RF 4.3: Eliminar Configuración.
  - RF 4.4: Generar Reporte.

Durante el proceso de desarrollo se definen un conjunto de requisitos que especifican comportamientos particulares de la herramienta. El modelo de desarrollo propone agruparlos por escenarios para su mejor comprensión. A continuación se realiza una breve descripción de los escenarios que cubren los elementos fundamentales del negocio.

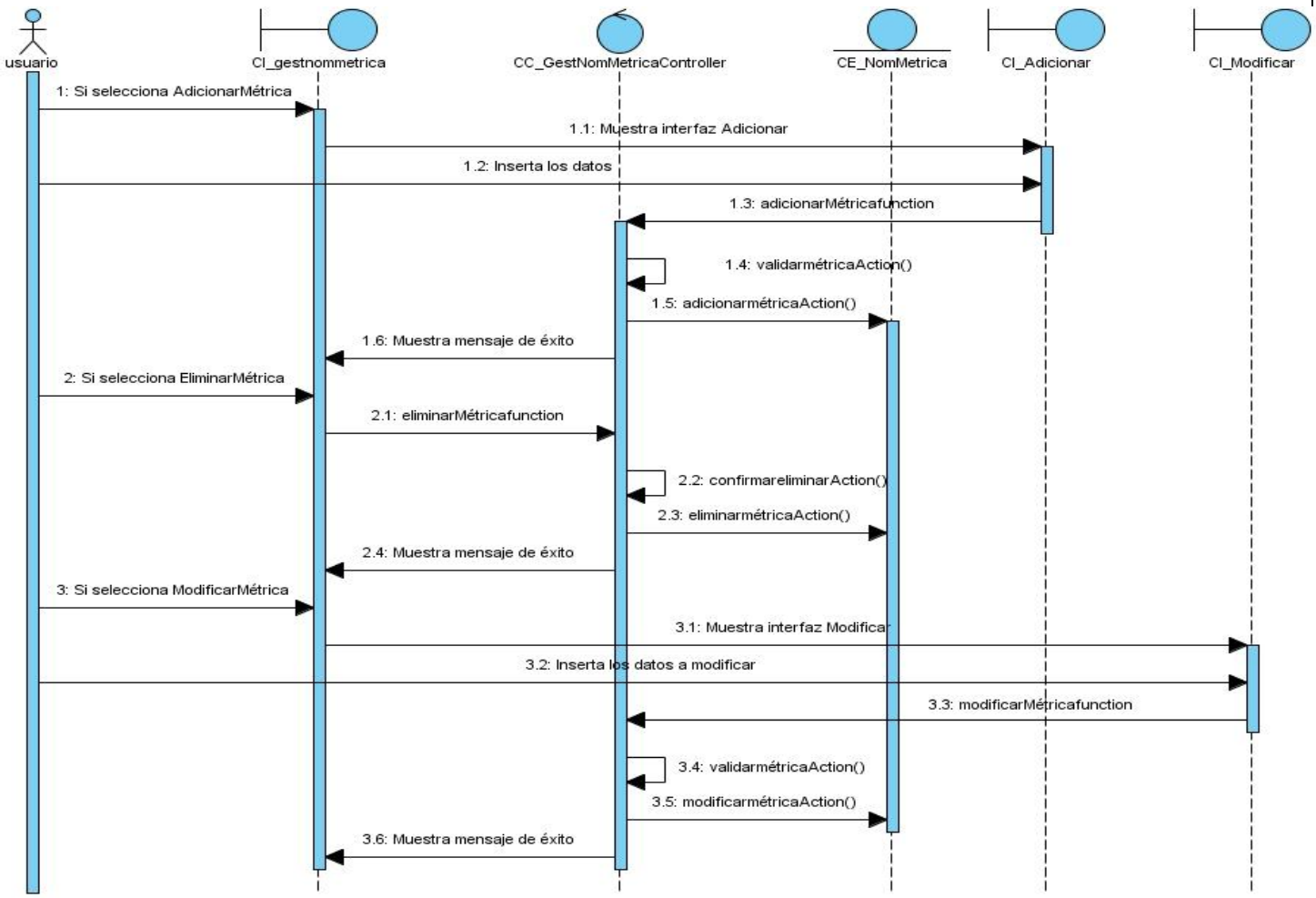
2.8 Descripción de los escenarios de la solución.

Tabla 1: Descripción de los escenarios Gestionar Atributos de Calidad, Gestionar Métricas, Gestionar Escalas y Gestionar Sistema.

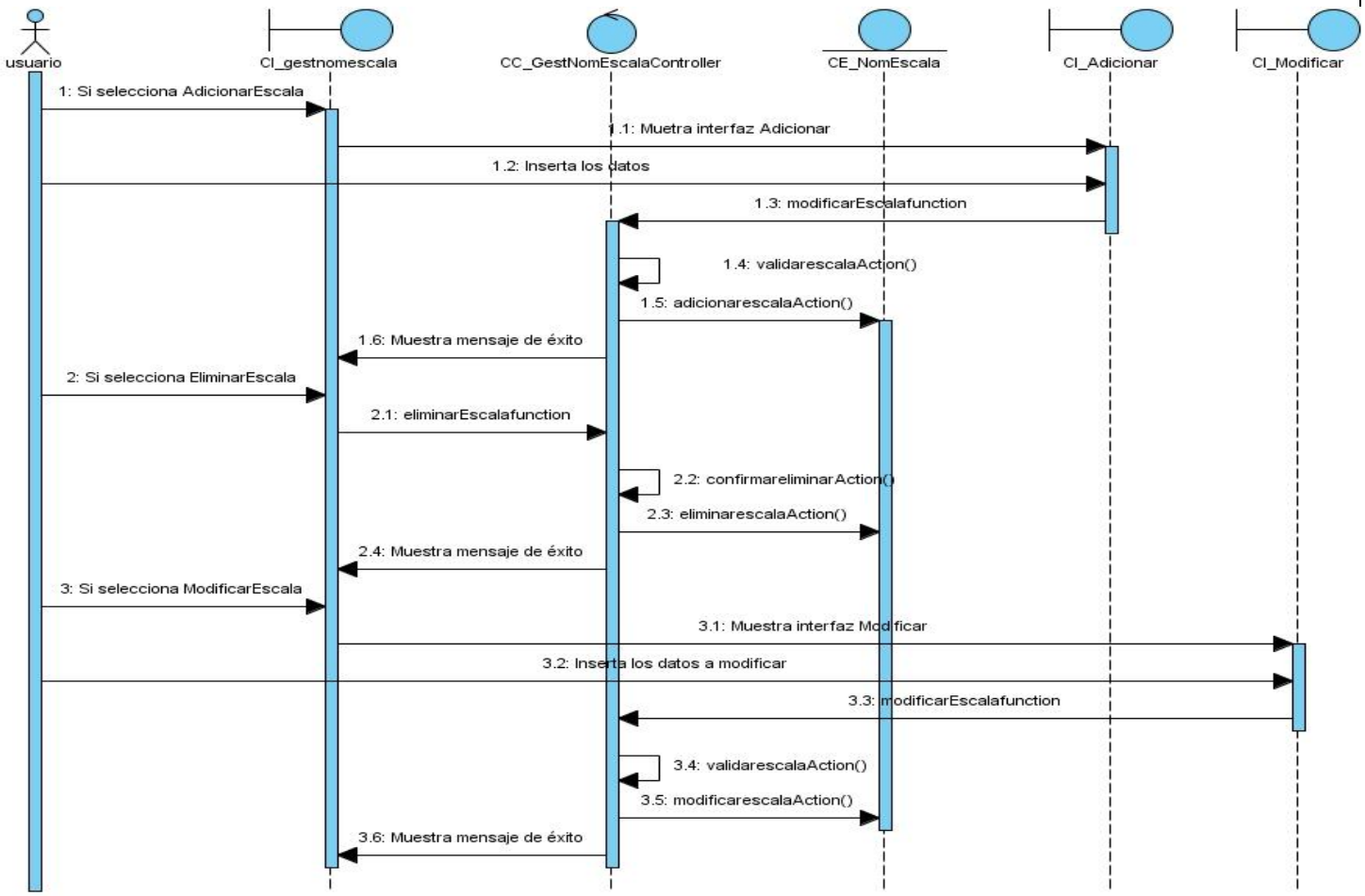
Escenario	**Requisitos Asociados**	**Descripción**
➢ Gestionar Atributos de Calidad.	❖ RF 1: Gestionar Atributos de calidad. ➢ RF 1.1: Adicionar Atributo de calidad. ➢ RF 1.2: Modificar Atributo de calidad. ➢ RF 1.3: Eliminar Atributo de calidad.	Es el escenario que permite la gestión de los atributos de calidad que utilizará el sistema.
Diagrama de secuencia.		
		
➢ Gestionar	❖ RF 2: Gestionar Métricas.	En este escenario el usuario define las

Métricas.	➢ RF 2.1: Adicionar Métricas. ➢ RF 2.2: Modificar Métricas. ➢ RF 2.3 Eliminar Métricas.	métricas que van a ser utilizadas durante el proceso de evaluación. Permite la gestión de las métricas que utilizará el sistema.

Diagrama de secuencia.

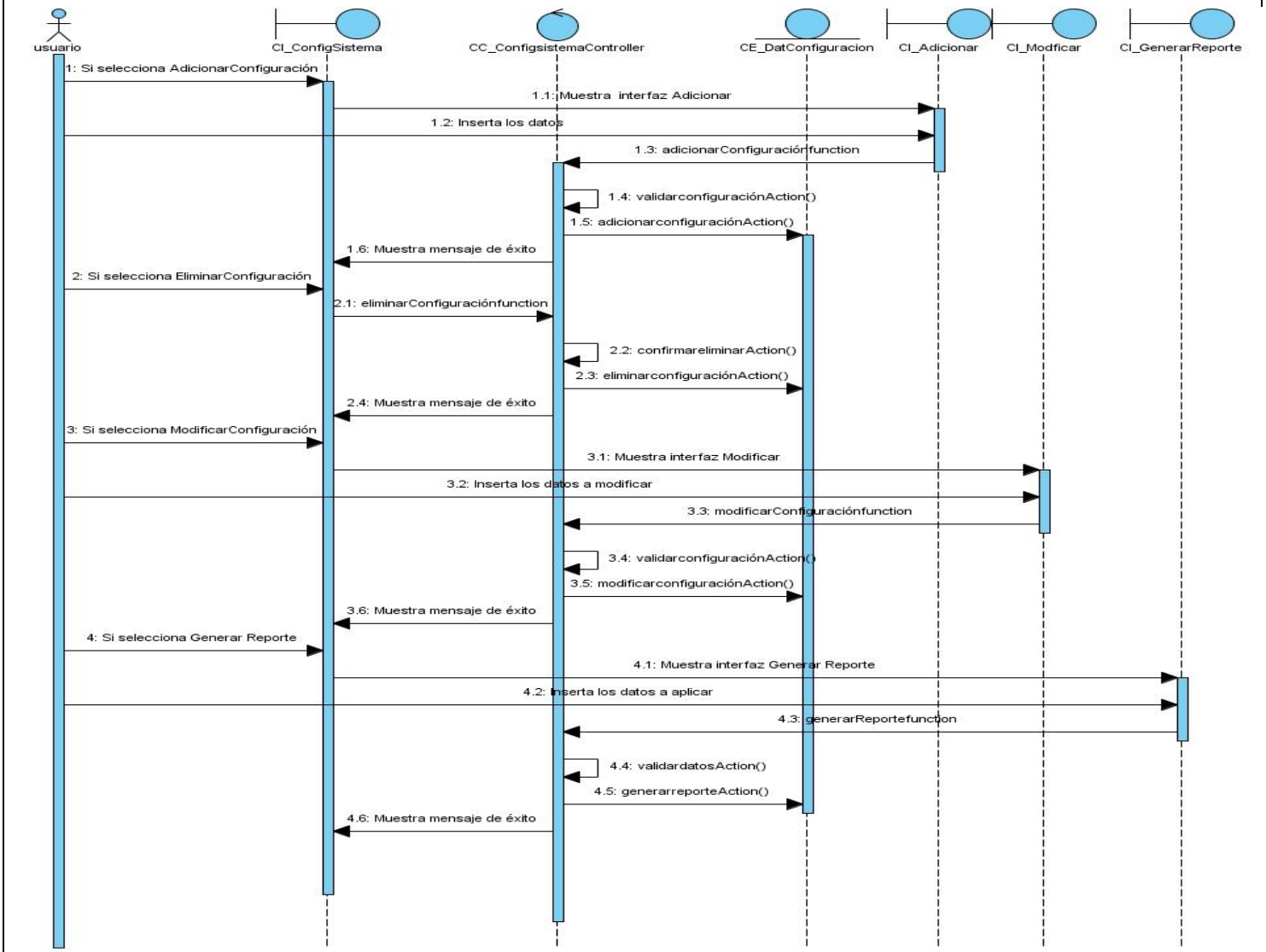


➢ Gestionar	❖ RF 3: Gestionar Escala.	Es el escenario que

Escalas.	➢ RF 3.1: Adicionar Escala. ➢ RF 3.2: Modificar Escala. ➢ RF 3.3: Eliminar Escala.	permite la gestión de las escalas que utilizará el sistema.
Diagrama de secuencia.		
		
➢ Gestionar sistema.	❖ RF 4: Gestionar Configuración. ➢ RF 4.1: Adicionar Configuración. ➢ RF 4.2: Modificar Configuración.	Es el escenario que permite gestionar las configuraciones así como aplicar las mismas a un

	➢ RF 4.3: Eliminar Configuración. ➢ RF 4.4: Generar Reporte.	producto para evaluar la calidad del mismo y generar reportes de las evaluaciones realizadas.

Diagrama de secuencia.



2.9 Patrones utilizados en el diseño de la solución.

MVC

El Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la interface de usuario y el código es el que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

El **modelo** es el componente encargado del acceso a datos.

La **vista** transforma el modelo en una página Web que permite al usuario interactuar con ella.

El **controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones, en tanto el modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes. (29)

Singleton

El Patrón de Diseño Singleton (Instancia Única) se utiliza para garantizar que una clase sólo tenga una única instancia y para facilitar un punto de acceso global a la misma.

Se utiliza cuando se necesita:

- que sea un único objeto el que coordine acciones a lo largo de todo el sistema
- que el objeto posea estado.
- variables globales en el sistema.

La propia clase es única responsable de crear la única instancia (ocultando al constructor) y de facilitar el acceso global a la instancia. (30)

En la solución que se propone se utiliza el patrón Singleton en todas las clases dentro del paquete Models, donde los objetos que se crean mediante el trabajo con la herramienta son instanciados por las clases controladoras (ConfigsistemaController.php, GestnomatributosController.php, GestnommetricasController.php, GestnomescalaController.php).

Controlador

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. (31)

Bajo Acoplamiento

Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuánto software podemos extraer de un modo independiente y reutilizarlo en otro proyecto? Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia**.** (32)

2.10 Diagramas de clases del diseño (Diagramas de clases Web).

En el diagrama de clases del diseño con estereotipos web se representa de forma inteligente cómo está estructurada una aplicación que funciona mediante el navegador y cuáles son sus relaciones. Además los escenarios son realizados por las clases y los objetos del diseño. Esto se representa por colaboraciones en el modelo de diseño y denota una realización de escenario-diseño, esto ligado a algunas operaciones,

atributos y asociaciones sobre una clase específica es manipulado por los diagramas de clases que muestran sus clases participantes y sus relaciones.

2.10.1 Diagrama de clases del diseño: Escenario: Gestionar Atributos de Calidad.

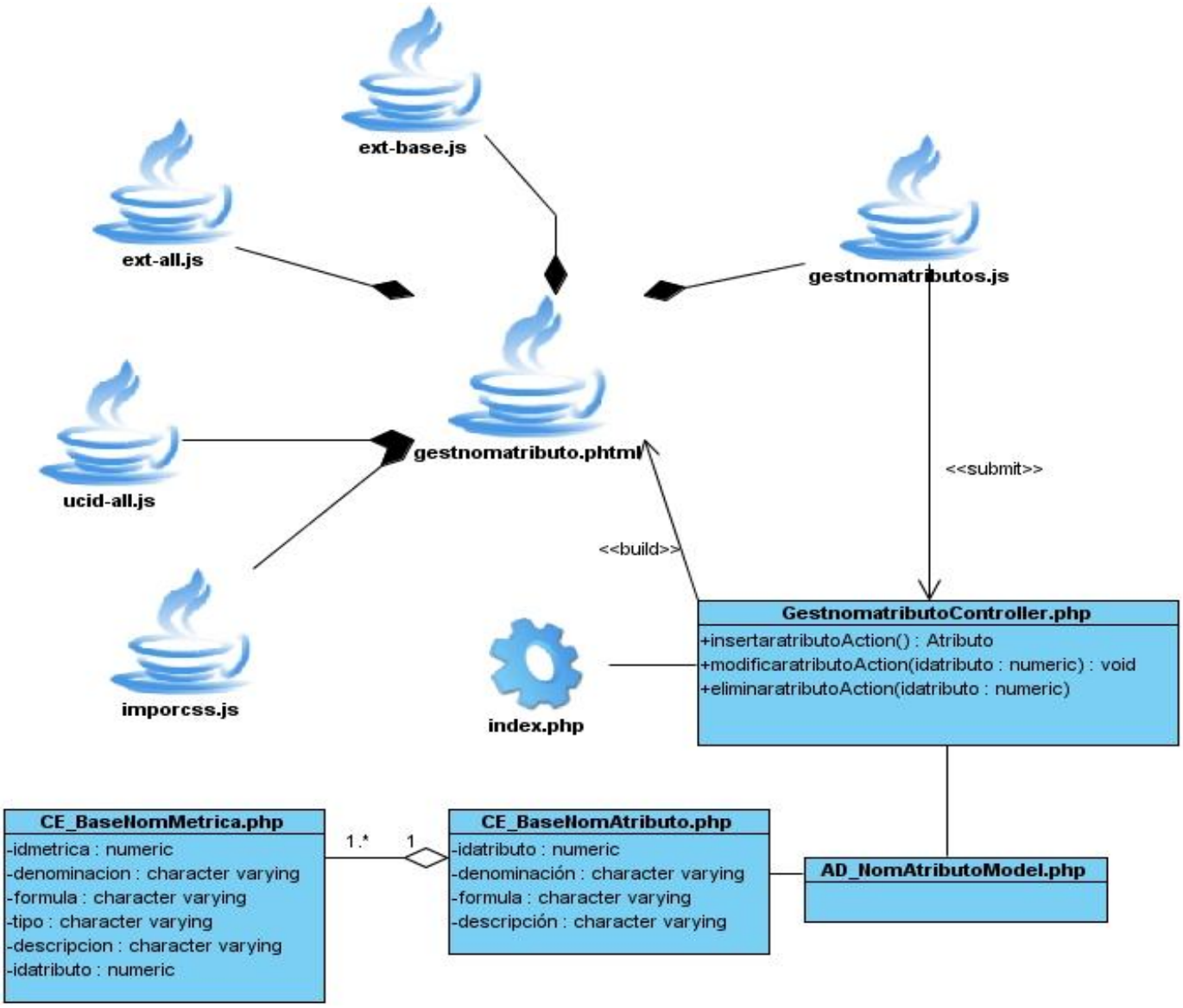


Figura 4.1: Diagrama de clases del diseño: Escenario: Gestionar Atributos de calidad.

2.10.2 Diagrama de clases del diseño: Escenario: Gestionar Métricas.

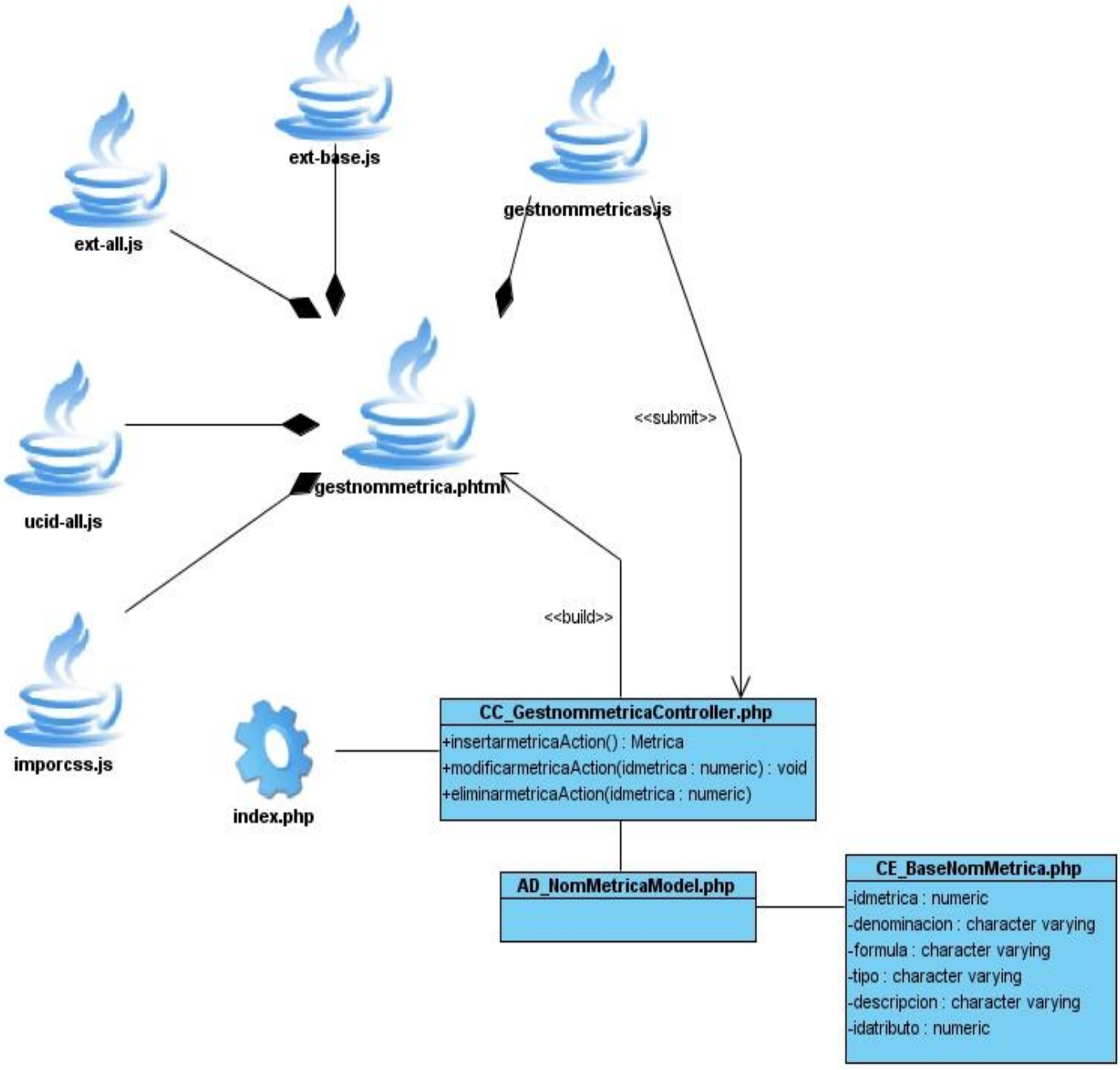


Figura 4.2: Diagrama de clases del diseño: Escenario: Gestionar Métricas.

2.10.3 Diagrama de clases del diseño: Escenario: Gestionar Escalas.

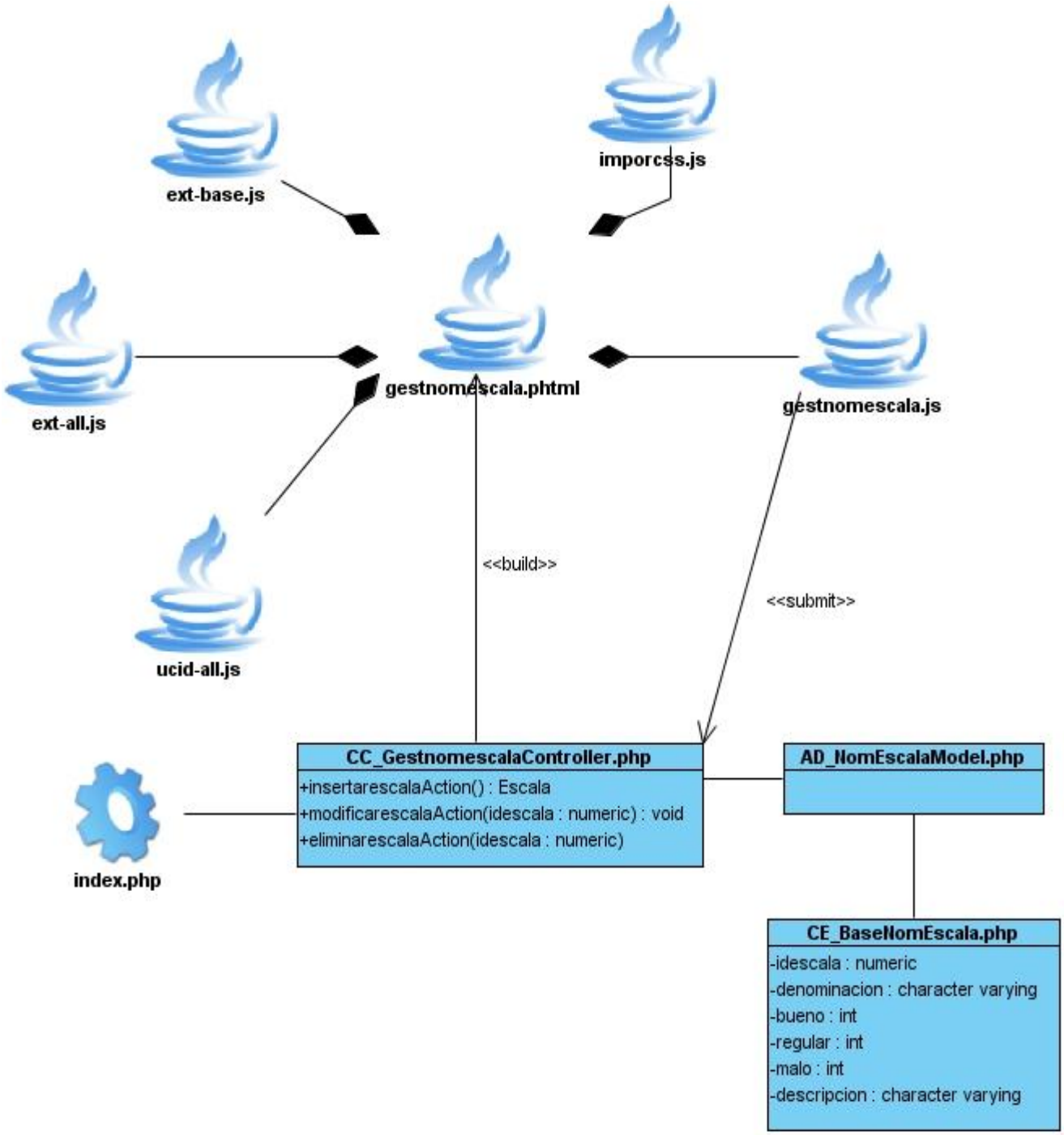


Figura 4.3: Diagrama de clases del diseño: Escenario: Gestionar Escalas.

2.10.4 Diagrama de clases del diseño: Escenario: Gestionar Sistema.

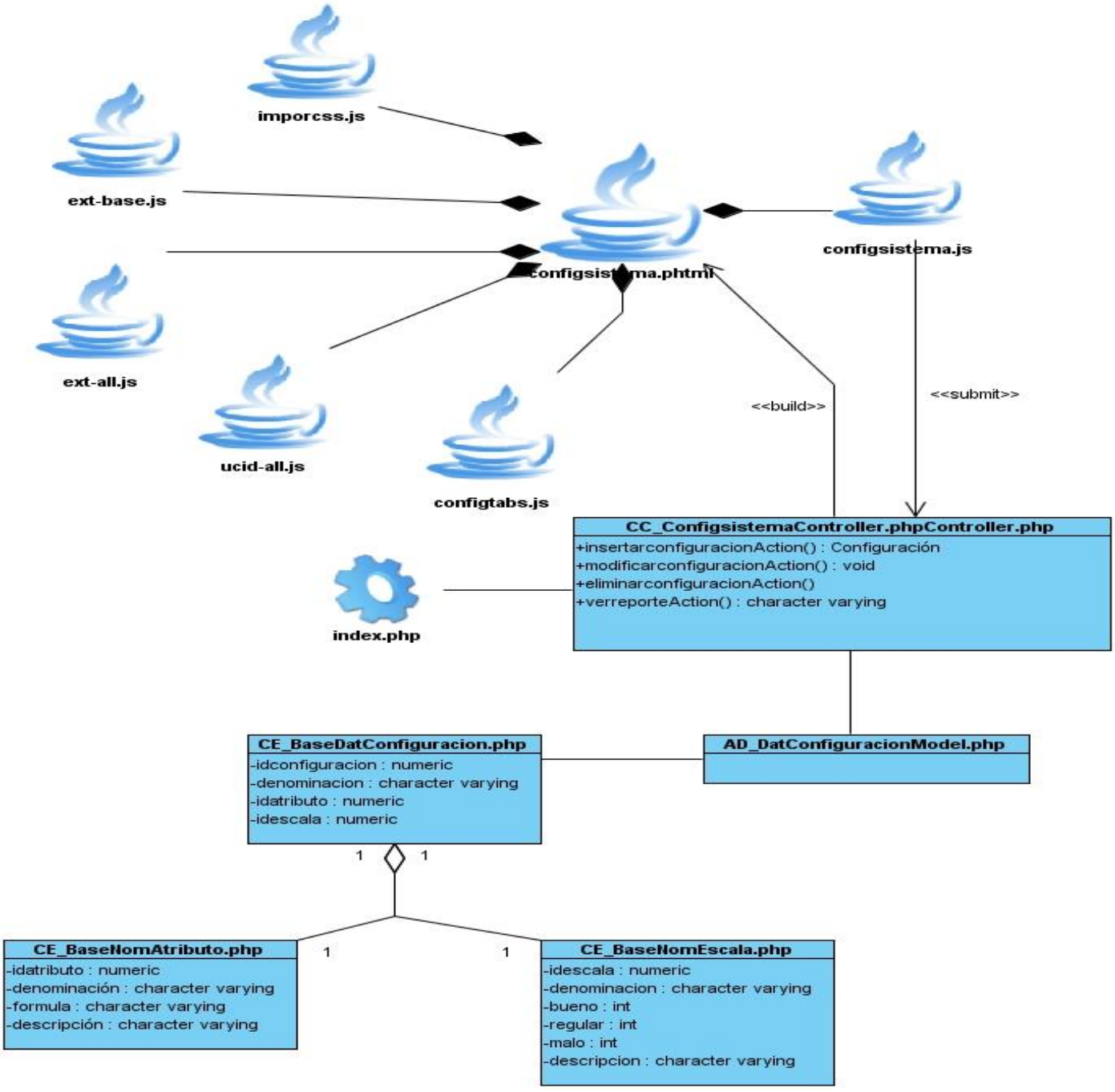


Figura 4.4: Diagrama de clases del diseño: Escenario: Gestionar Sistema.

2.11 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Existen múltiples categorías para clasificar a los requerimientos no funcionales, a continuación se muestran los requerimientos no funcionales que el sistema necesita:

Apariencia o interfaz externa.

- El sistema cuenta con una interfaz amigable permitiendo el usuario el reconocimiento visual de las funcionalidades a través de elementos que las identifican.

Usabilidad.

- La información que brinda el sistema puede ser consultada por cualquier persona, que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

- La aplicación podrá ser administrado solo por las personas que conozcan bien el negocio que el mismo implementa.

Rendimiento.

- Disponibilidad constante de trabajo en red contra el servidor.
- Rápido acceso a la información en tiempos relativamente cortos.
- El sistema debe requerir un consumo mínimo de recursos.

Soporte.

- Este producto de software podrá ser usado en diferentes plataformas, por ejemplo: Windows, Linux, no tiene ningún tipo de requisitos especial en cuanto a soporte, solo depende del contrato que establezca con el cliente.

Portabilidad.

- El sistema es independiente de plataforma, podrá ser usado bajo los sistemas operativos Windows y Linux.

Seguridad.

- El sistema solo puede ser administrado por la persona que cuenta con los permisos necesarios para hacerlo.

- La información se valida tanto en el cliente como en el servidor.

Confiabilidad.

- El sistema debe estar disponible en todo momento permitiendo el trabajo de los usuarios y las acciones de mantenimiento, previendo también los siguientes posibles factores:
  - Frecuencia y severidad de los fallos.
  - Protección contra fallos.
  - Recuperación.
  - Predicción de fallos.
  - Tiempo medio entre fallos.

Requerimientos de Software.

- Se debe disponer de un Sistema Operativo Windows 98 o Superior, Mac OS, Linux, entre otros, y un servidor de base de datos con PostgresSQL.

Requerimientos de Hardware

- Se requiere un procesador Pentium 3 o superior, memoria RAM de 512 MB o superior.

El servidor debe tener las siguientes características: capacidad de disco duro superior a 80.0 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

2.12 Conclusiones.

Con el desarrollo de este capítulo se obtiene la propuesta de solución del sistema que se desea desarrollar. Además, se describen los escenarios arquitectónicos definidos con sus requisitos asociados. Se identifican los procesos que serán objeto de automatización. Se obtienen los diagramas de clases del diseño y se describen los patrones utilizados en la realización de estos diagramas. Finalmente se obtuvieron los modelos mediante los que se representa la base de datos que se necesita para guardar y manejar la información con que el componente trabaja.

CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN DE LA SOLUCIÓN.

3.1 Introducción.

Este capítulo tiene como objetivo el desarrollo de la implementación de la propuesta de solución. Esta se realiza construyendo la codificación, es simplemente la programación del sistema, siguiendo el modelo de diseño. El propósito de este capítulo es describir cómo los elementos presentes en el diseño se implementan en términos de componentes y cómo el sistema se organiza de acuerdo a los nodos específicos en el modelo de despliegue, utilizando para ello los diagramas de componentes y el diagrama de despliegue. Además se valida mediante casos de prueba y métricas, el sistema que será entregado.

3.2 Diagrama de componentes.

Los diagramas de componentes estructuran el modelo de implementación en términos de subsistemas de implementación y muestran las relaciones entre los elementos de implementación. Estos modelan la vista estática del sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes. Los elementos de modelado en un diagrama de componentes son componentes y paquetes.

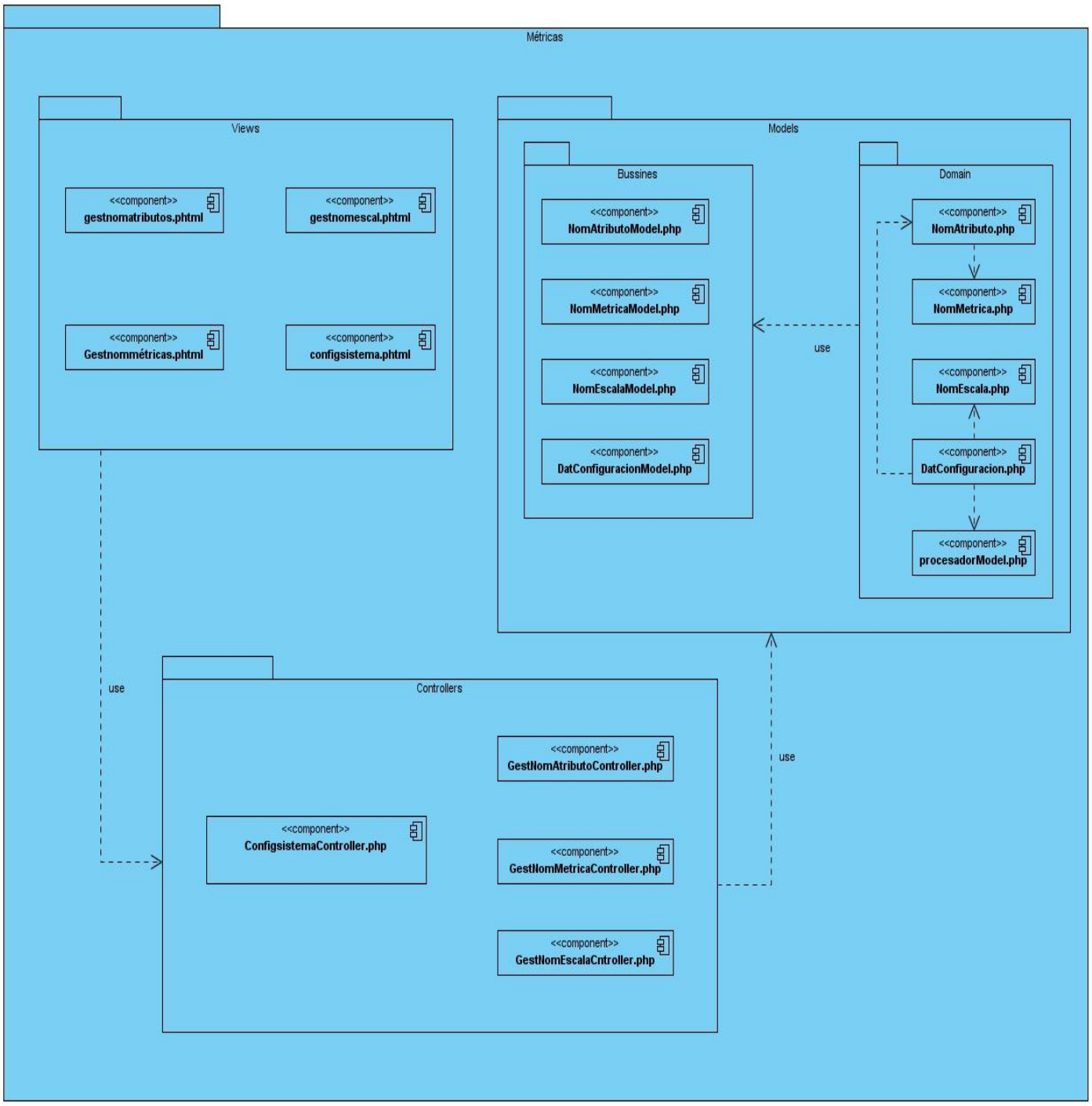


Figura 5: Diagrama de componentes.

3.3 Diagrama de despliegue.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). En general los nodos pueden estar formados por unidades de computación de algún tipo, impresoras, PC, servidores, cámaras, etc.

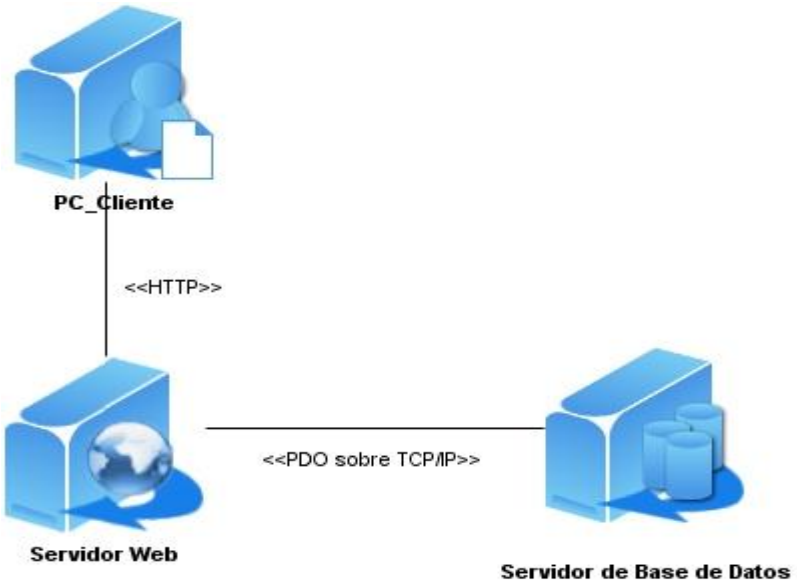


Figura 6: Diagrama de despliegue.

En este caso particularmente se cuenta con una PC_ Cliente conectada al servidor web Wamp a través del protocolo http, definido para conexiones mediante navegadores web. El diagrama contará con un servidor de base de datos que se ejecuta sobre PostgreSQL, este es accedido por el servidor web utilizando PDO y mediante el protocolo TCP/IP. Este protocolo permite una comunicación fiable entre dos aplicaciones. De esta forma, las aplicaciones que lo utilicen no tienen que preocuparse de la integridad de la información: dan por hecho que todo lo que reciben es correcto.

3.4 Prueba.

El desarrollo de sistemas de software implica una serie de actividades de producción en la que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde

el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta, así como en posteriores pasos de diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad. Las pruebas del software son un elemento crítico para la garantía de la calidad del mismo y representan una revisión final de las especificaciones, del diseño y de la codificación. Estas no son más que la ejecución de un sistema bajo ciertas condiciones con la intención de descubrir errores.

La fase de pruebas añade valor al producto que se maneja: todos los programas tienen errores y la fase de pruebas los descubre; ese es el valor que añade. El objetivo específico de la fase de pruebas es encontrar la mayor cantidad de errores existentes en el producto. (33)

3.4.1 Pruebas de Caja Negra.

Durante la validación, el componente fue sometido a pruebas de caja negra por parte del grupo de calidad del proyecto ERP- Cuba donde se evaluaron las funcionalidades utilizando casos de prueba. Para la revisión se emplearon un total de 3 iteraciones en las que se detectaron 14 no conformidades en general, de ellas, las 14 fueron significativas, 5 de validación y 9 de interfaz.

Después de terminadas las iteraciones el componente fue liberado por el grupo de calidad del proyecto ERP obteniéndose un resultado satisfactorio con 0 No Conformidades. (Ver Anexo II).

3.5 Validación de la solución.

En diseño y desarrollo, la validación está relacionada con el proceso de reexaminación de un producto para determinar la conformidad con las necesidades del usuario. La validación es realizada normalmente sobre el producto final bajo condiciones operacionales definidas. La valoración de la solución es el paso final en el proceso de evaluación del software, donde una solución puede ser valorada de diversas maneras, debido a que existen distintas tipos de valoraciones como son:

- Valoración cualitativa.
- Valoración indirecta.

- Valoración directa.

Una valoración se realiza mediante una métrica para asignar uno de los valores de una escala (el mismo puede ser número o categoría) al atributo de la entidad (sistema, subsistemas o componentes).

Valoración cualitativa: Es una evaluación sistemática del grado o capacidad de una entidad para satisfacer necesidades o requerimientos específicos. Además se emplean categorías, como algunos de los atributos más importantes de una entidad, ejemplo: el lenguaje de desarrollo del programa (C, C ++, C #, PHP, JAVA).

Valoración indirecta: Es la valoración de un atributo derivada del valor de uno o más atributos diferentes. Es la valoración externa de un atributo de un sistema, ejemplo: el tiempo de respuesta a la información alimentada por el usuario, es una valoración indirecta de los atributos del software, debido a que esta medida se verá influenciada por los atributos externos del sistema, así como los propios internos.

Valoración directa: Es una valoración del producto, de forma indirecta o directa. Ejemplo: El número de líneas de código, las valoraciones de la complejidad, el número de fallas encontradas durante el proceso y el índice de señales o alertas, son todas las valoraciones internas propias del producto en sí.

3.6 Métricas de software.

Un aspecto importante a tener en cuenta en la fase de evaluación de la calidad del diseño ha sido la creación de métricas, teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software.

Atributos de calidad que se abarcan:

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.

Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño del Sistema Calculador de Métricas y su relación con los atributos de calidad definidos en este trabajo son las siguientes:

3.6.1 Tamaño operacional de clase (TOC).

El TOC está dado por el número de métodos asignados a una clase.

Tabla 2.1: Tamaño operacional de clase (TOC)

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 2.2: Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio

	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 2.3: Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestnomatributosController	8	Media	Media	Media
GestnommetricasController	7	Media	Media	Media
GestnomescalaController	7	Media	Media	Media
ConfigsistemaController	12	Alta	Alta	Baja
NomAtributoModel	4	Baja	Baja	Alta
NomMetricaModel	4	Baja	Baja	Alta
NomEscalaModel	4	Baja	Baja	Alta
procesadorModel	22	Alta	Alta	Baja
DatConfiguracionModel	4	Baja	Baja	Alta
NomAtributo	7	Media	Media	Media
NomMetrica	6	Media	Media	Media
NomEscala	6	Media	Media	Media
DatConfiguracion	8	Media	Media	Media
BaseNomAtributo	1	Baja	Baja	Alta
BaseNomMetrica	1	Baja	Baja	Alta
BaseNomEscala	1	Baja	Baja	Alta
BaseDatConfiguracion	1	Baja	Baja	Alta
gestnomatributos	7	Media	Media	Media
gestnommetricas	7	Media	Media	Media
gestnomescala	7	Media	Media	Media
configsistema	8	Media	Media	Media

Tabla 2.4: Resultado del promedio de procediminetos.

Total de clases	**21**
Promedio de procedimientos	**6,285714286**

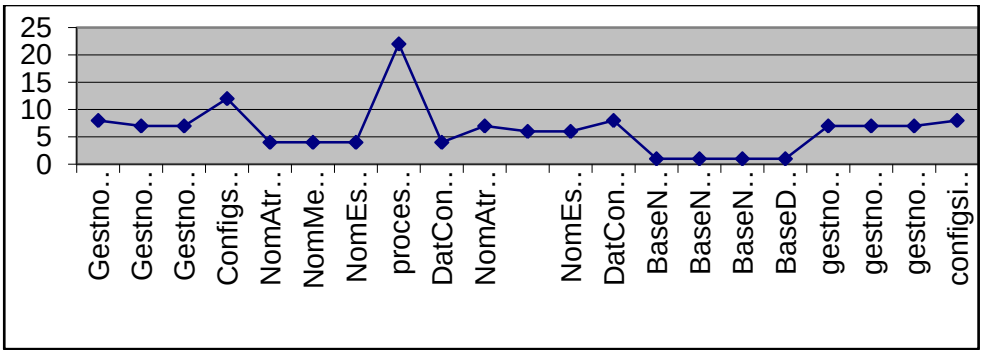


Figura 7.1: Gráfica que representa los resultados obtenidos después de aplicar la métrica TOC (relación de la cantidad de procedimientos por clases).

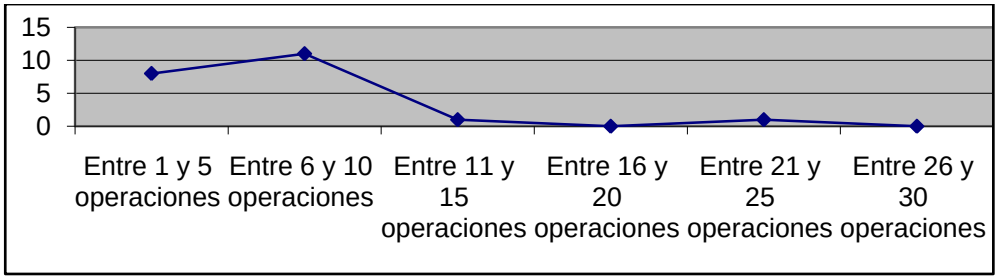


Figura 7.2: Gráfica que representa la cantidad de clases por intervalos de procedimientos definidos.

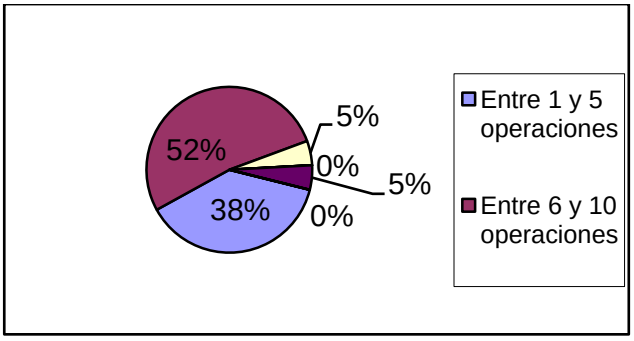


Figura 7.3: Gráfica que representa el % que representa la cantidad de clases por intervalos de procedimientos.

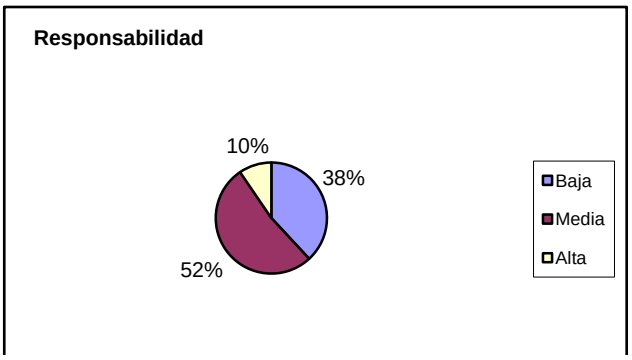


Figura 7.4: Gráfica que representa el % de clases por categorías del atributo Responsabilidad obtenidos en la aplicación de la métrica TOC.

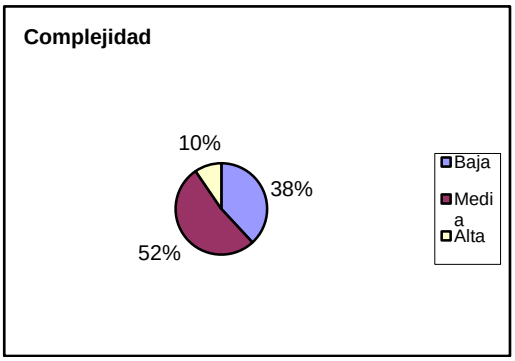


Figura 7.5: Gráfica que representa el % de clases por categorías del atributo Complejidad de implementación obtenidos en la aplicación de la métrica TOC.

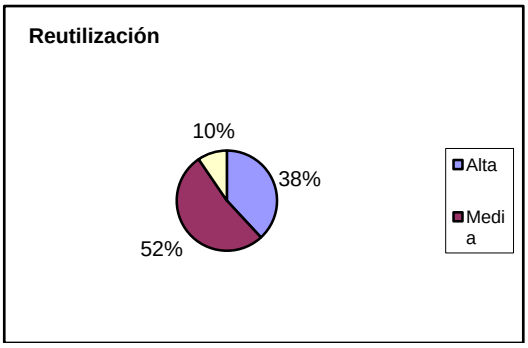


Figura 7.6: Gráfica que representa el % de clases por categorías del atributo Reutilización de implementación obtenidos en la aplicación de la métrica TOC.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del Sistema Calculador de Métricas tiene una calidad

aceptable teniendo en cuenta que el 90 % de las clases incluidas en este sistema posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 90% de las clases poseen evaluaciones positivas en los atributos de calidad Responsabilidad, Complejidad de Implementación y Reutilización.

3.6.2 Relaciones entre clases (RC).

Las RC están dado por el número de relaciones de uso de una clase con otra.

Tabla 3.1: Tabla Relaciones entre clases (RC).

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 3.2: Rango de valores para los criterios de evaluación de la métrica Relaciones entre clases (RC).

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Tabla 3.3: Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
GestnomatributoController	2	Medio	Media	Media	Media
GestnommétricaController	1	Bajo	Baja	Alta	Baja
GestnomescalaController	1	Bajo	Baja	Alta	Baja
GestconfigsistemaController	4	Alto	Alta	Baja	Alta
NomAtributoModel	1	Bajo	Baja	Alta	Baja
NomMétricaModel	1	Bajo	Baja	Alta	Baja
NomEscalaModel	1	Bajo	Baja	Alta	Baja
procesadorModel	0	Ninguno	Baja	Alta	Baja
DatConfiguraciónModel	1	Bajo	Baja	Alta	Baja
NomAtributo	1	Bajo	Baja	Alta	Baja
NomMétrica	1	Bajo	Baja	Alta	Baja
NomEscala	1	Bajo	Baja	Alta	Baja
DatConfiguración	1	Bajo	Baja	Alta	Baja
BaseNomAtributo	0	Ninguno	Baja	Alta	Baja
BaseNomMétrica	0	Ninguno	Baja	Alta	Baja
BaseNomEscala	0	Ninguno	Baja	Alta	Baja
BaseDatConfiguración	0	Ninguno	Baja	Alta	Baja
gestnomatributos	1	Bajo	Baja	Alta	Baja
gestnommétricas	1	Bajo	Baja	Alta	Baja
gestnomescala	1	Bajo	Baja	Alta	Baja
configsistema	1	Bajo	Baja	Alta	Baja

Tabla 3.4: Tabla de promedio de asociaciones de uso.

Total de clases	**21**
Promedio de asociaciones de uso	**0,952380952**

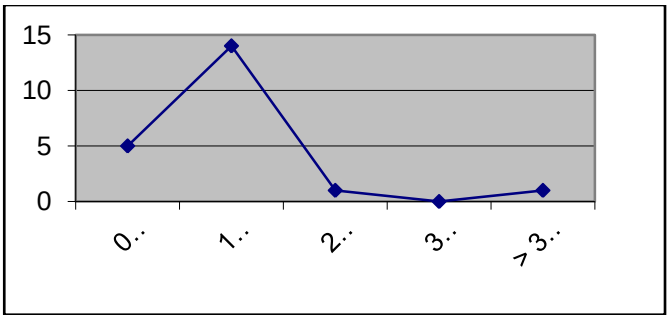


Figura 8.1: Gráfica que representa los resultados obtenidos después de aplicar la métrica RC (relación de la cantidad de dependencias por clases).

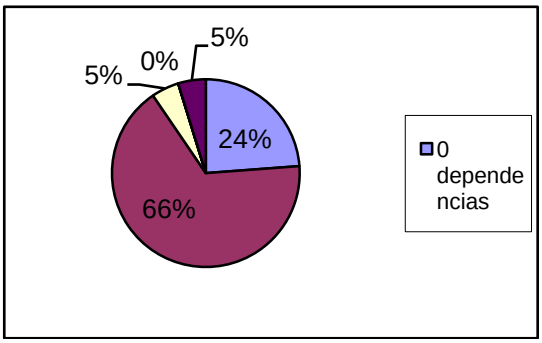


Figura 8.2: Gráfica que representa el % de la cantidad de clases por intervalos de dependencia.

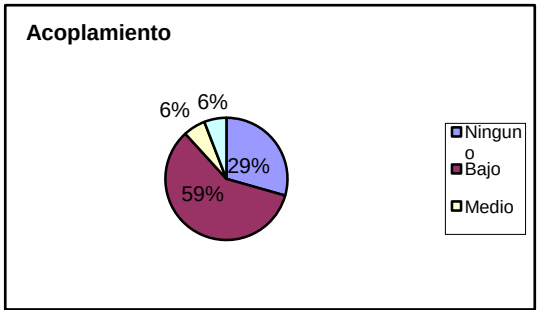


Figura 8.3: Gráfica que representa el % de clases por categorías del atributo Acoplamiento obtenidos en la aplicación de la métrica RC.

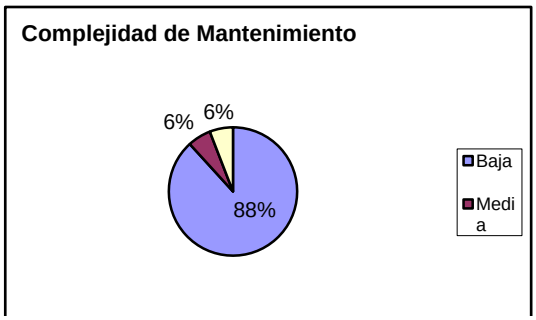


Figura 8.4: Gráfica que representa el % de clases por categorías del atributo Complejidad de Mantenimiento obtenidos en la aplicación de la métrica RC.

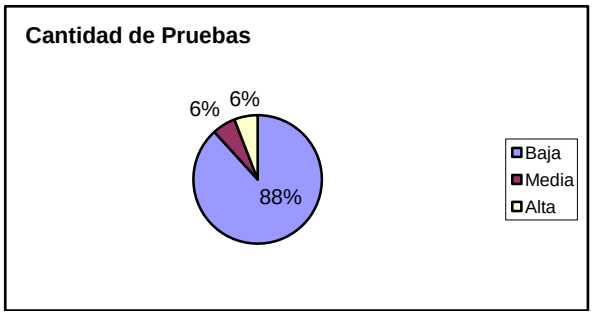


Figura 8.5: Gráfica que representa el % de clases por categorías del atributo Cantidad de Pruebas obtenidos en la aplicación de la métrica RC.

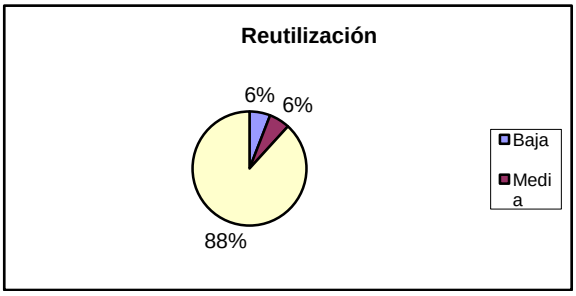


Figura 8.6: Gráfica que representa el % de clases por categorías del atributo Reutilización obtenidos en la aplicación de la métrica RC.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del Sistema Calculador de Métricas tiene una calidad aceptable teniendo en cuenta que el 95 % de las clases incluidas en este subsistema posee menos de 3 dependencias de otras clases. Además el 94% de las clases posee índices aceptables en cuanto a

Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 94 % de las clases.

3.6.3 Matriz de cubrimiento o matriz de inferencia de indicadores de calidad.

La Matriz de cubrimiento o matriz de inferencia permite conocer si el resultado obtenido de la relación atributo/métricas para cada componente es positivo o negativo. Llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos tendrá un valor de 1, si son negativos de 0 y si no existe relación alguna se tomará como nula -1. Una vez completado los datos de dicha relación se realiza un cálculo donde se promedia la sumatoria de los valores obtenidos de un atributo por cada métrica evaluada, y la división de dicha sumatoria por la cantidad de métricas evaluadas (solo se promedian las que arrojan un resultado, las nulas no). Este valor es el que va a tener el atributo dentro de una tabla que medirá si los atributos fueron buenos, regulares o malos. Se lograron los siguientes resultados:

Tabla 4.1: Resultados evaluados de la relación Atributos/Métricas por cada componente que integran la solución.

Atributos/Métricas	TOC	RC	Promedio
Responsabilidad	1	-1	1
Complejidad de implementación	1	-1	1
Reutilización	1	1	1
Complejidad del mantenimiento	-1	1	1
Cantidad de pruebas	-1	1	1
Acoplamiento	-1	1	1

Tabla 4.2: Rango de valores para la evaluación técnica de los atributos de calidad evaluados por cada métrica.

Catrgoría	Rango de valores
Malo	<= 0.4
Regular	>0.4 y <=0.7
Bueno	>0.7

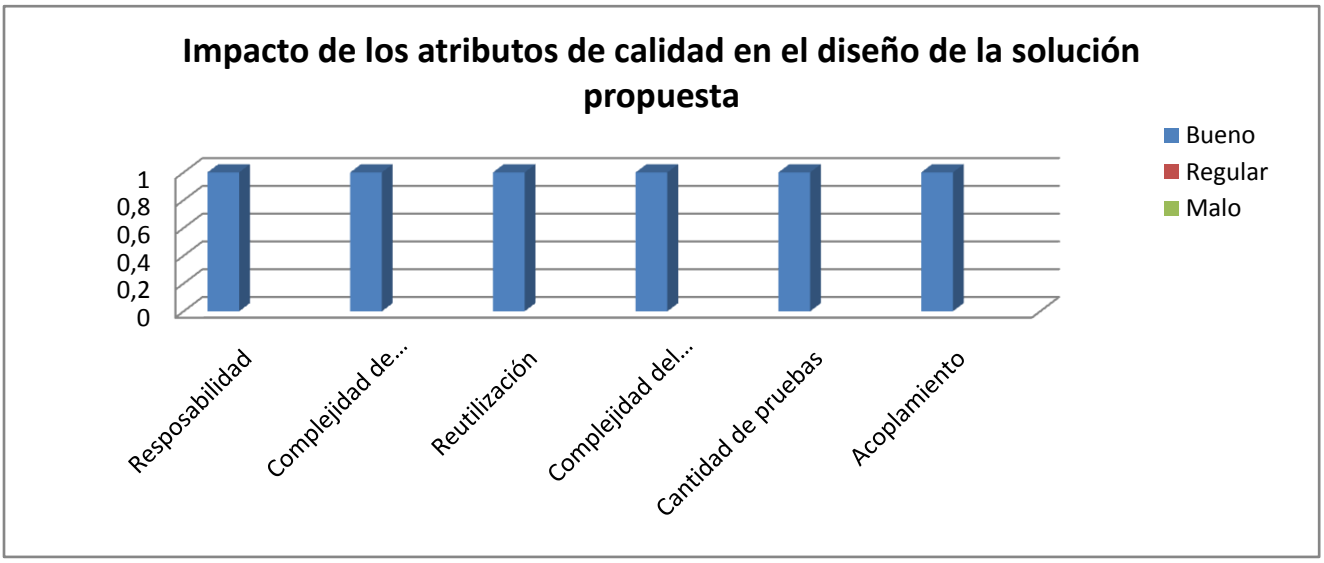


Figura 9: Gráfica de los resultados obtenidos de los atributos de calidad evaluados en las métricas.

3.7 Conclusiones.

Con el desarrollo de este capítulo se obtiene la implementación del componente que fue propuesto en el capítulo anterior. Se modelan los componentes con que contará la herramienta y se agrupan en paquetes mediante el diagrama de componentes. También se muestra mediante el diagrama de despliegue cómo quedará distribuida la herramienta una vez que sea desplegada.

Durante el desarrollo de este capítulo se realizaron pruebas al componente obtenido por parte del grupo de calidad de proyecto ERP obteniéndose buenos resultados.

Después de aplicadas las métricas y analizados los resultados de las mismas, se determinó que el diseño de la solución propuesta posee una calidad aceptable.

Conclusiones Generales

A modo de conclusión se plantea que:

- Se analizaron las soluciones existentes detectándose deficiencias en las mismas.
- Se definió el modelo de desarrollo así como las herramientas y tecnologías a utilizar.
- Se generaron un conjunto de artefactos necesarios para documentar el análisis y el diseño de la solución.
- Se realizó el diagrama de componentes y de despliegue de la herramienta, además, se realizaron las pruebas de caja negra y se validó el diseño de la solución mediante la aplicación de métricas obteniéndose buenos resultados.
- Por todo lo antes mencionado se evidencia el cumplimiento de los objetivos propuestos los que dan solución al objetivo general del presente trabajo de diploma.

Recomendaciones

Al terminar este trabajo es necesario realizar algunas recomendaciones, las cuales podrán tenerse en cuenta en un futuro desarrollo e incremento de las funcionalidades del sistema.

- Continuar profundizando en el estudio del uso de las métricas, con el objetivo de añadir nuevas funcionalidades.
- Agregar nuevas funcionalidades en versiones posteriores de este sistema, como por ejemplo que la aplicación pueda brindar reportes gráficos donde se muestre el grado de avance que van obteniendo los componentes durante el desarrollo.
- Integrar la herramienta con el Reporteador utilizado en el proyecto ERP.

Referencias Bibliográficas

1. **Fenton, Norman E.** *Software Metrics.* 1991.

2. alarcos . [En línea] [Citado el: 12 de 11 de 2009.] http://alarcos.inf-cr.uclm.es/doc/Calidad/capitulo09.ppt.

3. www.ldc.usb.ve. [En línea] [Citado el: 15 de 11 de 2009.] http://www.ldc.usb.ve/~abianc/materias/ci4712/metricas.pdf.

4. processdash.com. [En línea] [Citado el: 15 de 11 de 2009.] http://www.processdash.com/functionality.

5. pear.php.net. [En línea] [Citado el: 17 de 11 de 2009.] http://pear.php.net/package/PHP_CodeSniffer.

6. www.phpunit.de. [En línea] [Citado el: 20 de 11 de 2009.] http://www.phpunit.de/.

7. sunset.usc.edu. [En línea] [Citado el: 12 de 12 de 2009.] http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html.

8. www.webmaster-mexico.com. [En línea] [Citado el: 20 de 12 de 2009.] http://www.webmaster-mexico.com/extjs-framework.

9. www.creargratisunapaginaweb.com. [En línea] [Citado el: 20 de 12 de 2009.] http://www.creargratisunapaginaweb.com/PHP/Ventajas-y-desventajas-del-Personal-Home-Page-4/.

10. www.lsi.us.es. [En línea] [Citado el: 20 de 12 de 2009.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

11. www.joangarnet.com. [En línea] [Citado el: 22 de 12 de 2009.] http://www.joangarnet.com/blog/?p=415.

12. techtastico.com. [En línea] [Citado el: 23 de 12 de 2009.] http://techtastico.com/post/zend-framework-una-introduccion/.

13. **Pérez, Ing. en Ciencias Informáticas Mileidy Magalys Sarduy.** *Propuesta de modelo de desarrollo.* La Habana : s.n., 2009.

14. www.netbeans-ide.softonic.com. [En línea] [Citado el: 11 de 01 de 2010.] http://netbeans-ide.softonic.com/.

15. www.zend.com. [En línea] [Citado el: 12 de 01 de 2010.] http://www.zend.com/products/studio/.

16. paratupagina.com. [En línea] [Citado el: 12 de 01 de 2010.] http://paratupagina.com/index.php?showtopic=124.

17. www.desarrolloweb.com. [En línea] [Citado el: 12 de 01 de 2010.] http://www.desarrolloweb.com/articulos/1178.php.

18. www.cavsi.com. [En línea] [Citado el: 28 de 01 de 2010.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

19. profesores.elo.utfsm.cl. [En línea] [Citado el: 02 de 02 de 2010.] http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/postgresql.ppt.

20. www.culturacion.com. [En línea] [Citado el: 09 de 02 de 2010.] http://culturacion.com/2009/12/montar-un-servidor-local-con-wamp-server/.

21. tecnologia.glosario.net. [En línea] [Citado el: 15 de 02 de 2010.] http://tecnologia.glosario.net/terminos-tecnicos-internet/uml-1655.html.

22. onjava.com. [En línea] [Citado el: 15 de 02 de 2010.] http://onjava.com/onjava/2005/07/20/businessprocessmodeling.html.

23. www.freedownloadmanager.org. [En línea] [Citado el: 23 de 02 de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

24. forja.rediris.es. [En línea] [Citado el: 24 de 02 de 2010.] https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf.

25. www.maestrosdelweb.com. [En línea] [Citado el: 24 de 02 de 2010.] http://www.maestrosdelweb.com/editorial/firefox/.

26. catarina.udlap.mx. [En línea] [Citado el: 20 de 03 de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo2.pdf.

27. www.mastermagazine.info. [En línea] [Citado el: 12 de 04 de 2010.] http://www.mastermagazine.info/termino/4404.php.

28. www.definicionabc.com. [En línea] [Citado el: 16 de 05 de 2010.] . [En línea] [Citado el: 16 de 05 de 2010.] http://www.definicionabc.com/comunicacion/reporte.php.

29. www.proactiva-calidad.com. [En línea] [Citado el: 20 de 05 de 2010.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

30. www.hachisvertas.net. [En línea] [Citado el: 20 de 05 de 2010.] http://www.hachisvertas.net/blog/01/2008/11/19/patrones-de-diseno-singleton.

31. uso-de-patrones-de-arquitectura-capitulo-4.ppt http://www.google.com.cu/url?sa=t&source=web&cd=6&ved=0CCcQFjAF&url=http%3A%2F%2Fjjegonzalezf.files.wordpress.com%2F2009%2F07%2Fuso-de-patrones-de-arquitectura-capitulo-4.ppt&rct=j&q=patrones+de+asignaci%F3n+de+responsabilidades+controlador+definicion&ei=.

32. jorgesaavedra.wordpress.com. [En línea] [Citado el: 25 de 05 de 2010.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

33. www.lab.dit.upm.es. [En línea] [Citado el: 25 de 05 de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm.

Bibliografía

1. **Fenton, Norman E.** *Software Metrics.* 1991.

2. alarcos . [En línea] [Citado el: 12 de 11 de 2009.] http://alarcos.inf-cr.uclm.es/doc/Calidad/capitulo09.ppt.

3. www.ldc.usb.ve. [En línea] [Citado el: 15 de 11 de 2009.] http://www.ldc.usb.ve/~abianc/materias/ci4712/metricas.pdf.

4. processdash.com. [En línea] [Citado el: 15 de 11 de 2009.] http://www.processdash.com/functionality.

5. pear.php.net. [En línea] [Citado el: 17 de 11 de 2009.] http://pear.php.net/package/PHP_CodeSniffer.

6. www.phpunit.de. [En línea] [Citado el: 20 de 11 de 2009.] http://www.phpunit.de/.

7. sunset.usc.edu. [En línea] [Citado el: 12 de 12 de 2009.] http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html.

8. www.webmaster-mexico.com. [En línea] [Citado el: 20 de 12 de 2009.] http://www.webmaster-mexico.com/extjs-framework.

9. www.creargratisunapaginaweb.com. [En línea] [Citado el: 20 de 12 de 2009.] http://www.creargratisunapaginaweb.com/PHP/Ventajas-y-desventajas-del-Personal-Home-Page-4/.

10. www.lsi.us.es. [En línea] [Citado el: 20 de 12 de 2009.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

11. www.joangarnet.com. [En línea] [Citado el: 22 de 12 de 2009.] http://www.joangarnet.com/blog/?p=415.

12. techtastico.com. [En línea] [Citado el: 23 de 12 de 2009.] http://techtastico.com/post/zend-framework-una-introduccion/.

13. **Pérez, Ing. en Ciencias Informáticas Mileidy Magalys Sarduy.** *Propuesta de modelo de desarrollo.* La Habana : s.n., 2009.

14. www.netbeans-ide.softonic.com. [En línea] [Citado el: 11 de 01 de 2010.] http://netbeans-ide.softonic.com/.

15. www.zend.com. [En línea] [Citado el: 12 de 01 de 2010.] http://www.zend.com/products/studio/.

16. paratupagina.com. [En línea] [Citado el: 12 de 01 de 2010.] http://paratupagina.com/index.php?showtopic=124.

17. www.desarrolloweb.com. [En línea] [Citado el: 12 de 01 de 2010.] http://www.desarrolloweb.com/articulos/1178.php.

18. www.cavsi.com. [En línea] [Citado el: 28 de 01 de 2010.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

19. profesores.elo.utfsm.cl. [En línea] [Citado el: 02 de 02 de 2010.] http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/postgresql.ppt.

20. www.culturacion.com. [En línea] [Citado el: 09 de 02 de 2010.] http://culturacion.com/2009/12/montar-un-servidor-local-con-wamp-server/.

21. tecnologia.glosario.net. [En línea] [Citado el: 15 de 02 de 2010.] http://tecnologia.glosario.net/terminos-tecnicos-internet/uml-1655.html.

22. onjava.com. [En línea] [Citado el: 15 de 02 de 2010.] http://onjava.com/onjava/2005/07/20/businessprocessmodeling.html.

23. www.freedownloadmanager.org. [En línea] [Citado el: 23 de 02 de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

24. forja.rediris.es. [En línea] [Citado el: 24 de 02 de 2010.] https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf.

25. www.maestrosdelweb.com. [En línea] [Citado el: 24 de 02 de 2010.] http://www.maestrosdelweb.com/editorial/firefox/.

26. catarina.udlap.mx. [En línea] [Citado el: 20 de 03 de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo2.pdf.

27. www.mastermagazine.info. [En línea] [Citado el: 12 de 04 de 2010.] http://www.mastermagazine.info/termino/4404.php.

28. www.definicionabc.com. [En línea] [Citado el: 16 de 05 de 2010.] . [En línea] [Citado el: 16 de 05 de 2010.] http://www.definicionabc.com/comunicacion/reporte.php.

29. www.proactiva-calidad.com. [En línea] [Citado el: 20 de 05 de 2010.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

30. www.hachisvertas.net. [En línea] [Citado el: 20 de 05 de 2010.] http://www.hachisvertas.net/blog/01/2008/11/19/patrones-de-diseno-singleton.

31. uso-de-patrones-de-arquitectura-capitulo-4.ppt

http://www.google.com.cu/url?sa=t&source=web&cd=6&ved=0CCcQFjAF&url=http%3A%2F%2Fjjegonzalezf.files.wordpress.com%2F2009%2F07%2Fuso-de-patrones-de-arquitectura-capitulo-4.ppt&rct=j&q=patrones+de+asignaci%F3n+de+responsabilidades+controlador+definicion&ei=.

32. jorgesaavedra.wordpress.com. [En línea] [Citado el: 25 de 05 de 2010.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

33. www.lab.dit.upm.es. [En línea] [Citado el: 25 de 05 de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm.

34. www.lsi.us.es. [En línea] [Citado el: 27 de 05 de 2010.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

35. jordisan.net. [En línea] [Citado el: 27 de 05 de 2010.] http://jordisan.net/blog/2006/que-es-un-framework.

36. www.guii.com.uy. [En línea] [Citado el: 30 de 05 de 2010.] http://www.guii.com.uy/soporte/5549-php-concepto.

37. www.dcc.uchile.cl. [En línea] [Citado el: 30 de 05 de 2010.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

38. www.visual-paradigm.com. [En línea] [Citado el: 30 de 05 de 2010.] http://www.visual-paradigm.com/product/vpuml/.

39. msdn.microsoft.com. [En línea] [Citado el: 01 de 06 de 2010.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

40. www.proactiva-calidad.com. [En línea] [Citado el: 01 de 06 de 2010.] http://www.proactiva-calidad.com/java/patrones/index.html.

41. www.ingenierosoftware.com. [En línea] [Citado el: 03 de 06 de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

42. java.sun.com. [En línea] [Citado el: 03 de 06 de 2010.] http://java.sun.com/blueprints/patterns/MVC.html.

43. www.tufuncion.com. [En línea] [Citado el: 06 de 06 de 2010.] http://www.tufuncion.com/zend-studio.

44. www.ingenierosoftware.com. [En línea] [Citado el: 07 de 06 de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

45. blog.carlosmayo.net. [En línea] [Citado el: 06 de 06 de 2010.] http://blog.carlosmayo.net/2008/04/patrones-de-diseo-singleton.html.

46. eisc.univalle.edu.co. [En línea] [Citado el: 08 de 06 de 2010.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/PATRONES_A9.pdf.

47. www.doctrine-project.org. [En línea] [Citado el: 08 de 06 de 2010.] http://www.doctrine-project.org/.

Anexos

Anexo I: Interfaz gráfica.

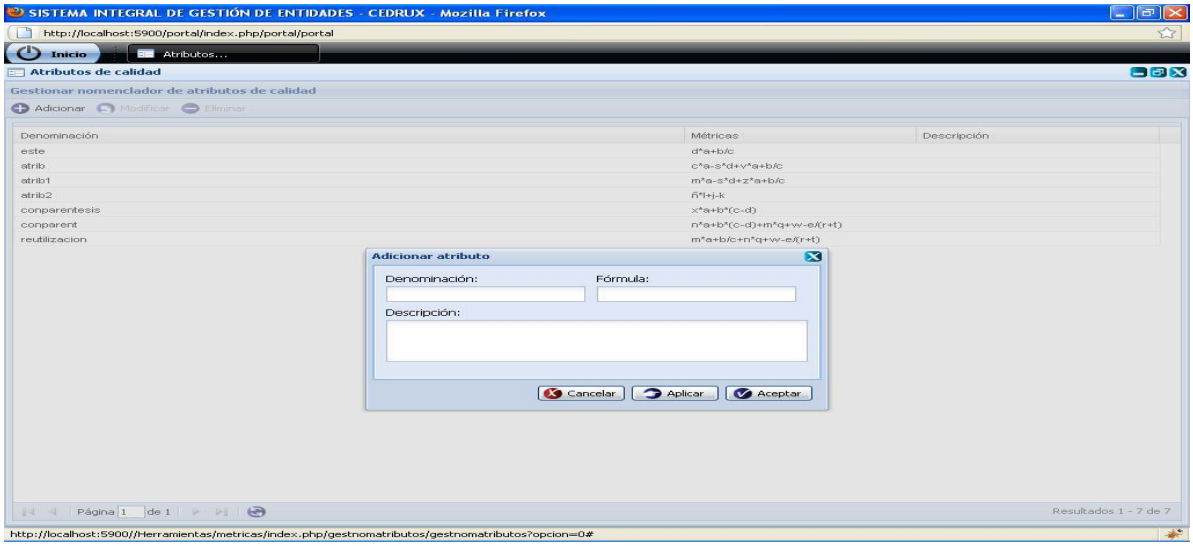


Figura 10: Interfaz Adicionar atributo de calidad.

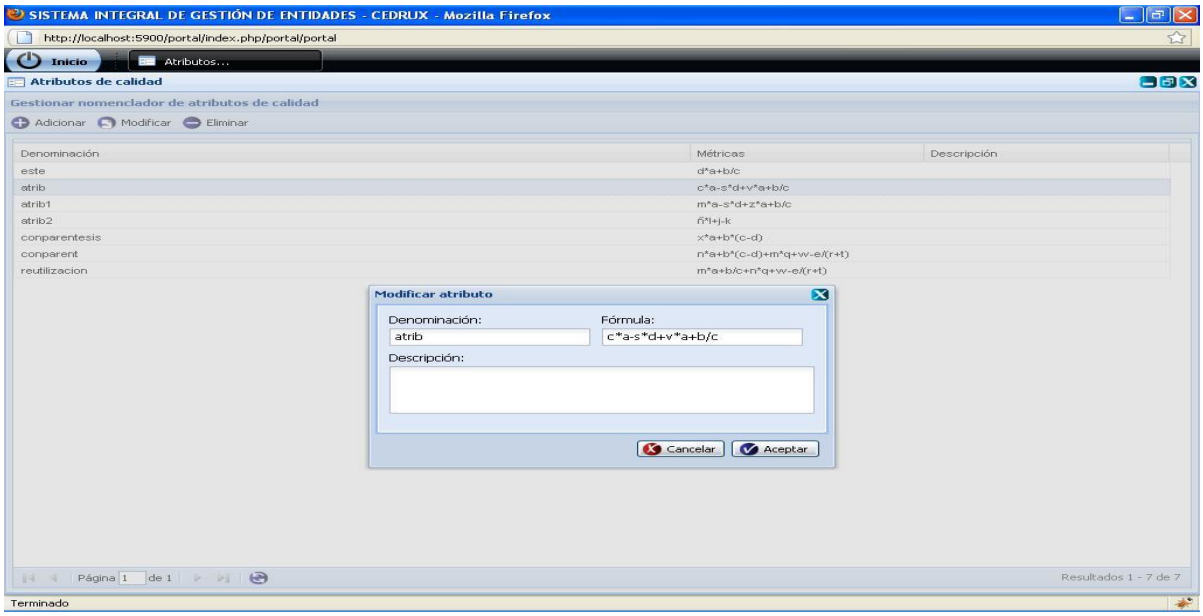


Figura 11: Interfaz Modificar atributo de calidad.

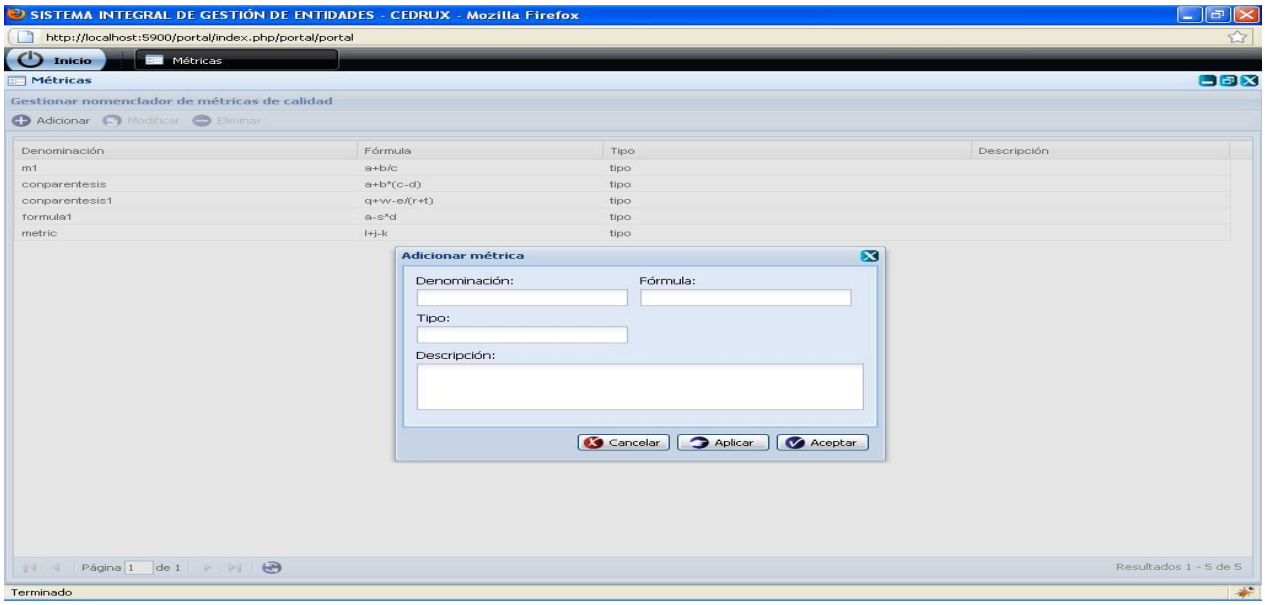


Figura 12: Interfaz Adicionar métricas.

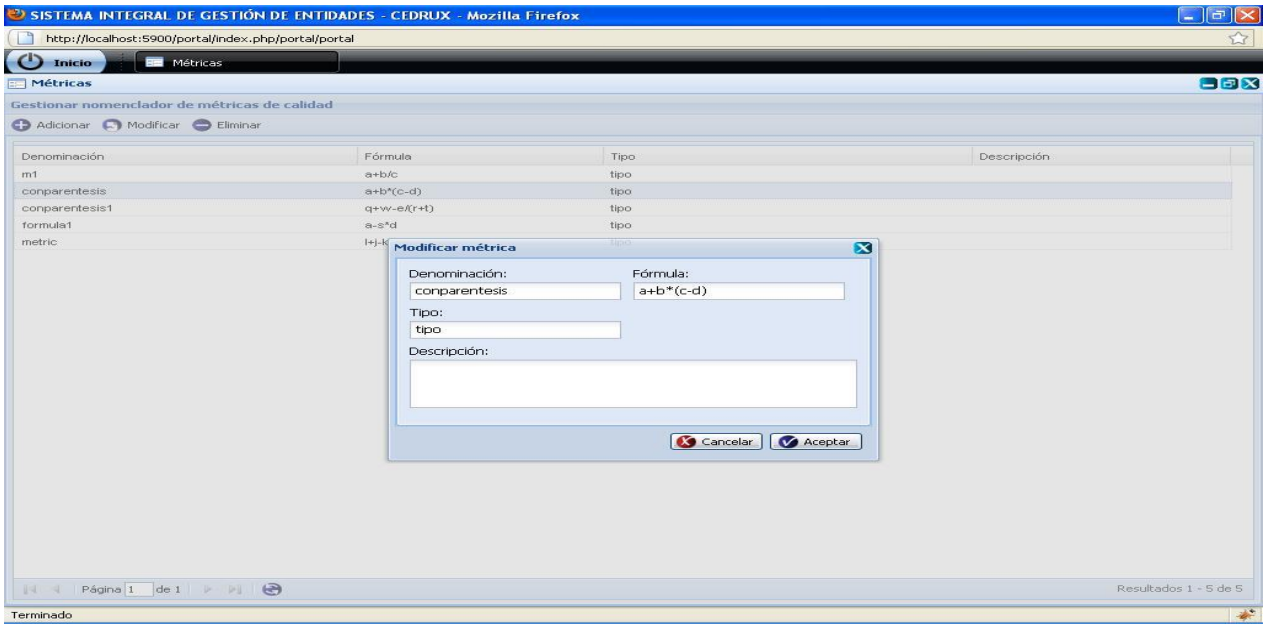


Figura 13: Interfaz Modificar métricas.

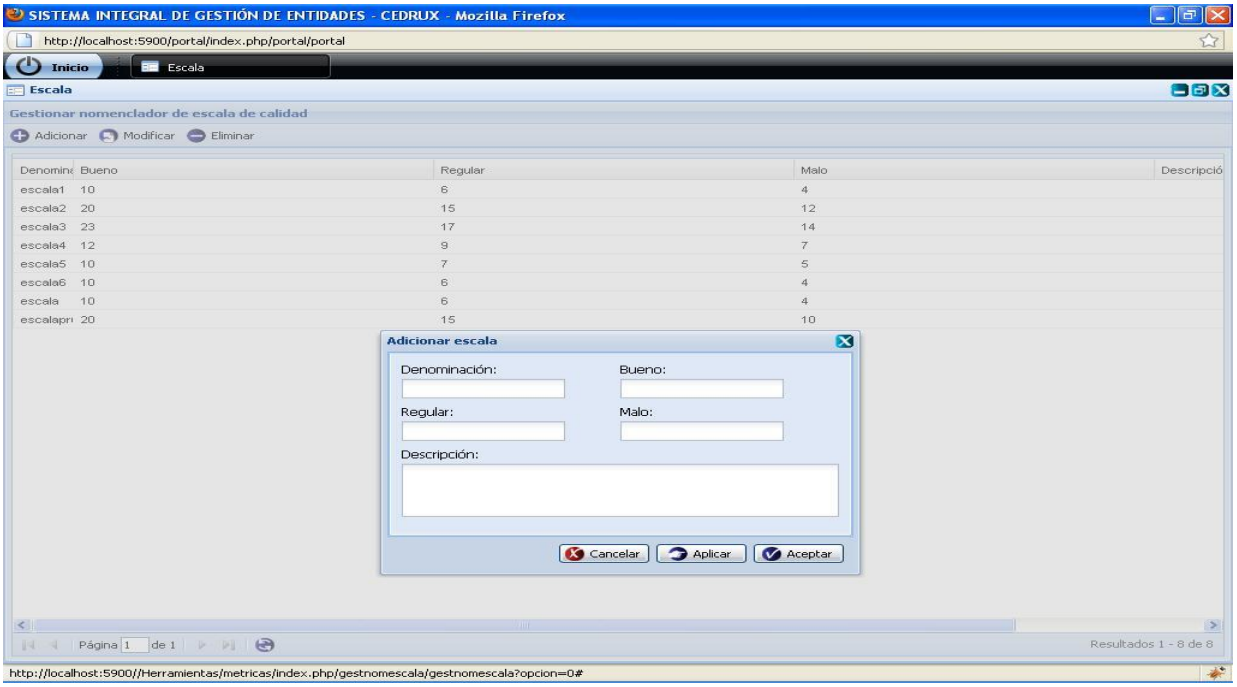


Figura 14: Interfaz Adicionar escala.

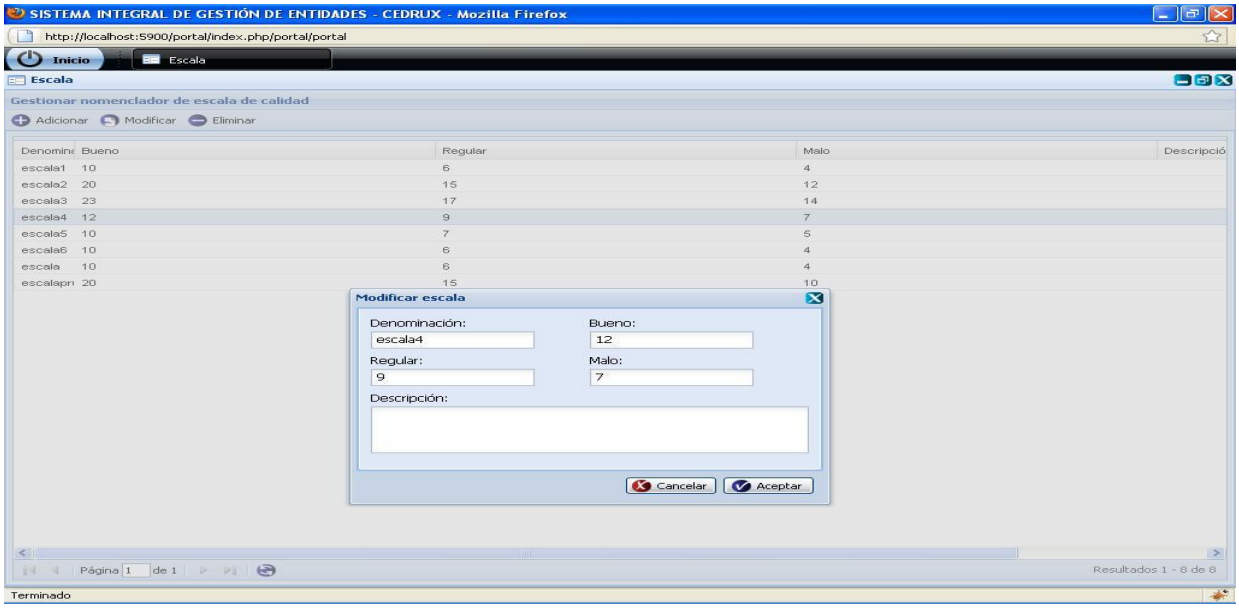


Figura 15: Interfaz Modificar escala.

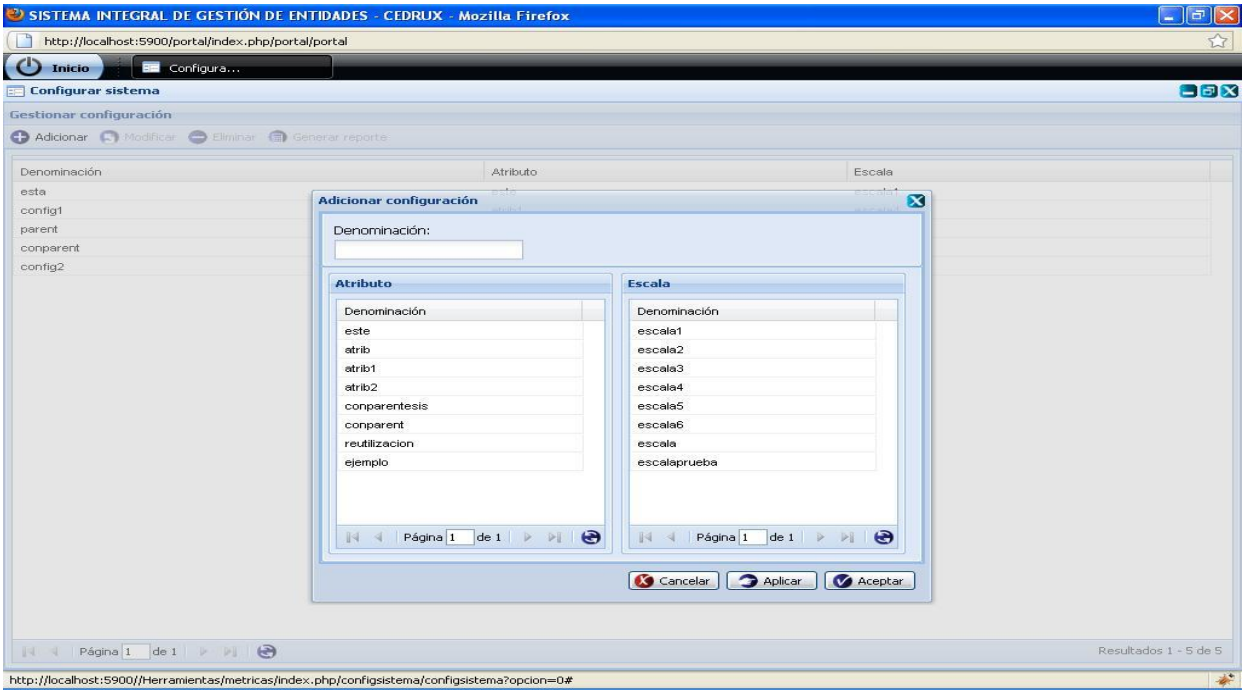


Figura 16: Interfaz Adicionar configuración.

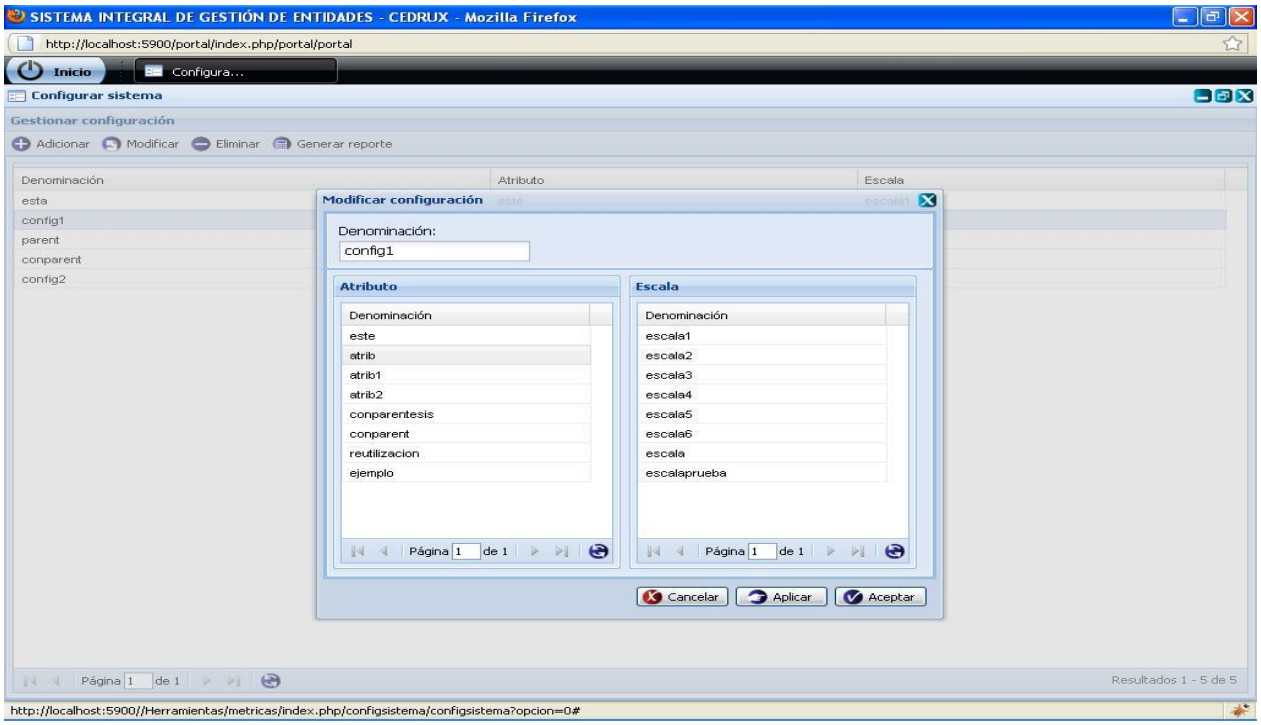


Figura 17: Interfaz Modificar configuración.

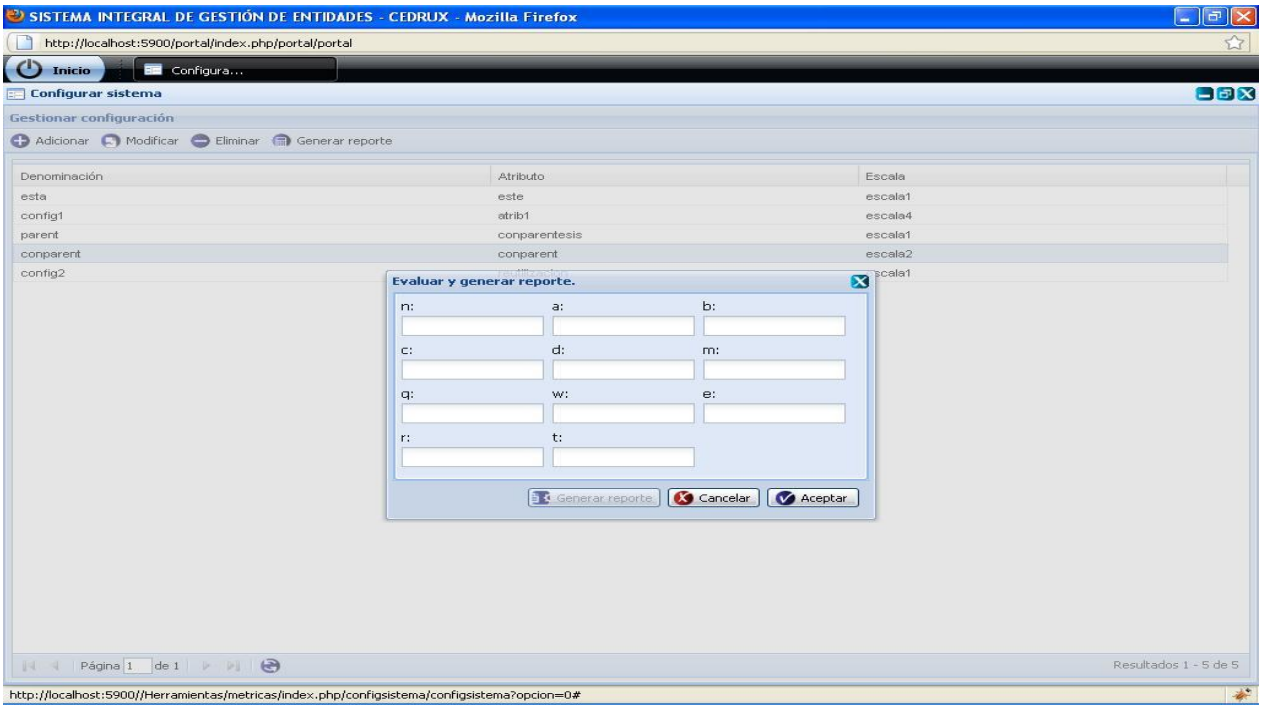


Figura 18: Interfaz Generar reporte.

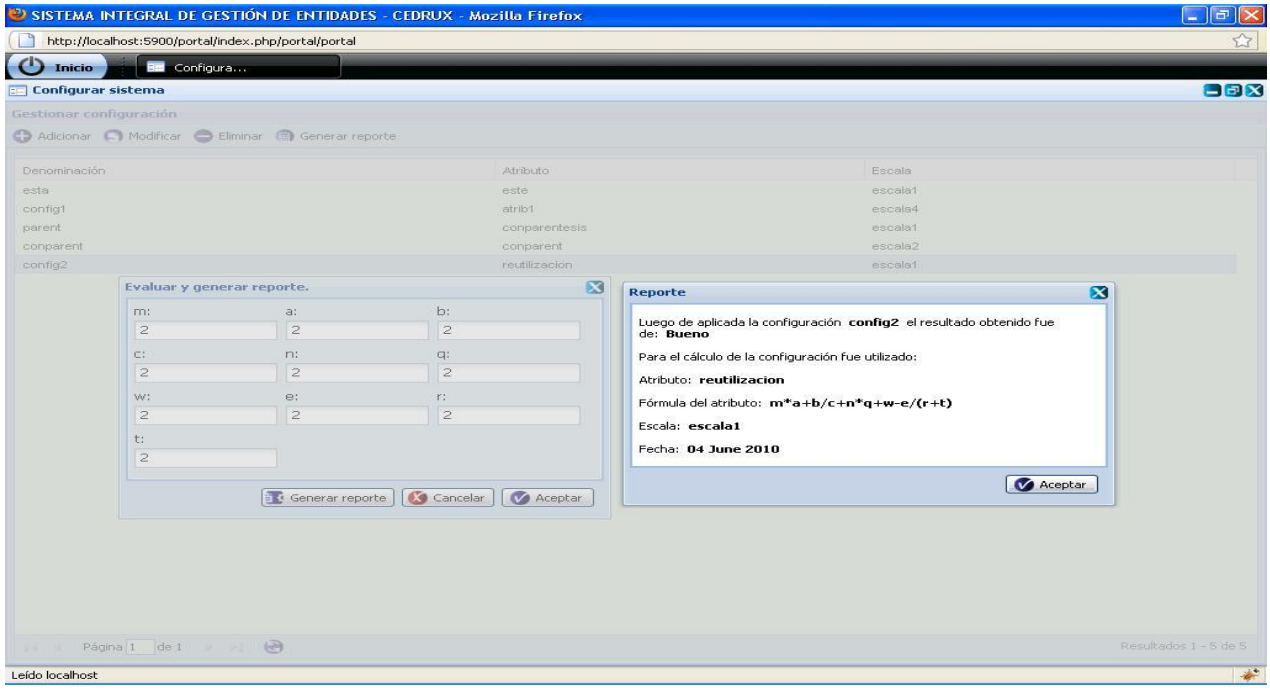


Figura 19: Reporte del resultado de la evaluación.

Anexo II: Acta de Liberación.

CENTRO PARA LA INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

Validación y Verificación de Requisitos (V&V)

ACTA DE LIBERACIÓN

Acta de Liberación, v 1.0,23/06/2010

Referencia: <<CIG-AF-CL-0001>>
Actualizado: 22/06/2010

Control del documento

Título:

Versión:

	Nombre	Cargo	
Elaborado por	Ing. Giselle Almeida González	Administrador de Calidad	
Revisado por	<<Rango. Nombre y Apellidos>>	<<Cargo>>	
Aprobado por	Ing. Adieren Acosta Zamora	**Firma**	
Cargo	Jefa del Dpto de Calidad	**Fecha**	23/06/2010

Reglas de confidencialidad

Clasificación: Uso Interno

Forma de distribución: <<PDF Digital>>

Este documento contiene información propietaria del **CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES y/o <<CLIENTE>>,** y es emitido confidencialmente para un propósito específico.

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimientos público su contenido, excepto para cumplir el propósito para el cual se ha generado.

Las reglas son aplicables a las 7 páginas de este documento.

Control de cambios

Versión	Lugar*	Tipo**	Fecha	Autor	Descripción
0.1			23/06/2010	Ing. Giselle Almeida González	Creación del documento.

* Sección del documento, Tabla, Figura.
** **A** Alta; **B** Baja; **M** Modificación

Índice de contenidos

Introducción

1. Objetivo

El objetivo de este documento es definir una plantilla para los documentos de contenido que se generen por la empresa.

2. Alcance

Está destinado a todos los miembros de la empresa que participen en el uso de la documentación afín de la empresa.

3. Definiciones y acrónimos

1. **Plantilla:** Documento de ejemplo que sustenta un formato y que describe los lineamientos para la elaboración de documentos similares.

4. Referencias

IEEE. 1991. *IEEE Standard Glossary of Software Engineering Terminology.* Spring 1991 Edition. 1991. IEEE Standard 610.12–1990.

Datos del producto

Emitida a favor de: Departamento de Tecnología

Responsable: Ing. Oiner Gómez Baryolo

Cargo: Líder de la línea.

1. Clasificado como:

- Aplicación Web.

2. Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Métricas	0 No Conformidad
…	…

3. Cantidad de iteraciones:

Para la revisión se emplearon un total de 3 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

Elementos revisados o probados y herramientas utilizadas

Elemento	Herramienta
Complejidad	Estándar de interfaz, Lista de Chequeo Interfaz
Ortografía y redacción	Revisión Técnica

1. Cantidad total de horas empleadas y rango de fechas:

Se emplearon un total de 6 horas efectivas de trabajo con la siguiente distribución: 2h (17/junio/2010) y 4h (22/junio/2010).

2. Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas se realizaron en un total de 3 turnos de trabajo, toda la actividad fue realizada por la Jefa de Pruebas.

Evaluado por:

1. Especialista principal Asignado:

Ing. Giselle Almeida González

2. Otro personal especializado participante:

Elemento	Herramienta
Ing. Iliannis Pupo Leyva	Especialista de Calidad
Alexey Talavera Calas	Desarrollador

_____ _____

Ing. Giselle Almeida González — Especialista del Laboratorio de Calidad

Ing. Oiner Gómez Baryolo — Responsable por el Equipo de Desarrollo

Glosario de Términos

Software: conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

Métricas: aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos para suministrar información relevante a tiempo

ERP: conjunto de sistemas de información gerencial que permite la integración de ciertas operaciones de una empresa, especialmente las que tienen que ver con la producción, la logística, el inventario, los envíos y la contabilidad.

PHP: lenguaje de programación de estilo clásico, con variables, sentencias condicionales, bucles, funciones.

Atributo de calidad: característica diferencial que posea el producto como rango distintivo de otro producto similar y cuyo proceso de elaboración y condiciones finales de calidad, cumplan las normas establecidas en el protocolo correspondiente.

MVC: (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

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

Framework: estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.