Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta para informatizar la reutilización de requisitos en el ERP.

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Aibett Cabrera Vega

Tutor: Tahirí Rivero Álvarez

Co-Tutor: Dairo Reyes Rodríguez

Ciudad de la Habana, Cuba

Declaración de Autoría

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

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

Autor:

_____

Aibett Cabrera Vega

Tutores:

_____ _____

Tahirí Rivero Álvarez Dairo Reyes Rodríguez

DATOS DE CONTACTO

Tutor: Ing. Tahirí Rivero Álvarez

Ingeniero en Ciencias Informáticas, Instructor, desempeña el rol de Analista principal del proyecto ERP-Cuba en el Centro de Informatización de Gestión de Entidades (CEIGE), Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: trivero@uci.cu

Co-Tutor: Ing. Dairo Reyes Rodríguez

Ingeniero en Ciencias Informáticas, Instructor, desempeña el rol de Especialista en replicación de datos del proyecto Sistema de informatización de la gestión de las fiscalías (SIGEF) en el Centro de Gobierno Electrónico (CEGEL) Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: dreyes@uci.cu

"Mi trabajo es cantar todo lo bello, encender el entusiasmo por todo lo noble, admirar y hacer admirar todo lo grande."

José Martí

AGRADECIMIENTOS

Agradezco profundamente a mis padres por su contante apoyo especialmente a mi mamá, a mi esposo por su ayuda y consideración especialmente en los momentos difíciles.

A Dairo por aportar su granito de arena para llegar hasta donde estoy hoy.

A todos los que fraternalmente me ayudaron….

Les entrego un mundo de gratitud.

DEDICATORIA

A los seres que más quiero en el mundo:

A mi mamá, corazón por el cual vivo y ocupa todo mi cariño, quien justifica cada obra que realizo, cada tarea que emprendo y cada acción que concluyo, es la luz y guía de sacrificio, dedicación, sufrimiento, consagración, sencillez y honestidad, de la cual siempre estaré orgullosa.

Mi esposo, espíritu de paz, quien me hace sentir necesaria y en cada momento del día vivo agradecida por su existencia.

A la Revolución Cubana, principal baluarte y ejemplo de justicia y humanidad.

RESUMEN

En la ingeniería de requisitos el crecimiento de la capacidad tecnológica ha estado siempre unido con el incremento de la complejidad, y ello a su vez ha propiciado el nacimiento de nuevas técnicas para hacer frente a dicha complejidad. La reutilización de los requisitos han aparecido en la ingeniería de requisitos como una técnica cuyo objetivo es el de poder crear diferentes variantes software a partir de uno ya existente. En el presente trabajo se aborda la realización de una herramienta la cual informatizará la reutilización de los requisitos en el proyecto ERP Cuba.

Se realizó un estudio de las herramientas existente para la reutilización de requisitos llegándose a la conclusión de que dichas herramientas no suplen las necesidades de los analistas del proyecto ERP Cuba. Se estudiaron además, herramientas y lenguaje para el desarrollo del sistema, así como consideraciones para la arquitectura del mismo.

En el trabajo se explicaron las principales características del sistema propuesto y se muestran los artefactos que posibilitan la implementación de la herramienta. Finalmente, se aplicaron métricas de diseño y se realizaron pruebas de funcionalidad obteniéndose resultados satisfactorios. Para el desarrollo de la herramienta se utilizó el marco de trabajo Symfony 1.4.11, como lenguaje de programación del lado del servidor PHP 5, como gestor de base de datos PostgreSQL versión 8.4. La construcción de la solución se realizó basándose en la arquitectura Cliente/Servidor y en Tres Capas. Como resultado se obtuvo una herramienta que facilite conformar nuevos productos a partir de requisitos reutilizables.

PALABRAS CLAVES: Herramienta, implementación, reutilización de requisitos.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLA

ÍNDICE DE ECUACIONES

INTRODUCCIÓN

Hoy en día el desarrollo tecnológico da pasos agigantados a nivel mundial, debido a la utilización de las tecnologías en todas las esferas de la sociedad, para ello existe como principal objetivo elevar el nivel económico de muchas empresas u organizaciones, y para lograrlo han puesto en práctica el desarrollo de soluciones informáticas que ayuden a mantener un control total de todos los recursos que se manejan, obteniendo como resultado las llamadas aplicaciones ERP (Enterprise Resource Planning) o de Planificación de Recursos Empresariales que permiten centralizar la gestión integral de las entidades empresariales y presupuestadas de un país, la optimización de los procesos empresariales, el acceso a toda la información de forma confiable, precisa y oportuna (integridad de datos) y la posibilidad de compartir información entre todos los componentes de la organización.

En estos últimos tiempos la dirección del país ha comprendido la necesidad de la independencia tecnológica y a su vez la importancia que tiene el desarrollo de sistemas informáticos propios y para ello se está llevando a cabo una estrategia para lograr mayor eficiencia económica y ahorro de recursos.

Cuba, en aras de lograr un fortalecimiento de la gestión de entidades y la informatización de la sociedad se encuentra inmersa en el desarrollo de un producto integral de planificación de recursos empresariales, como solución a la necesidad de informatizar los procesos de gestión de las entidades presupuestadas y empresariales a escala nacional utilizando plataformas confiables y eficientes.(1) Este producto integral está siendo desarrollado en la Universidad de las Ciencias Informáticas (UCI) desde el 2008, específicamente por el Centro de Informatización de la Gestión de Entidades (CEIGE). El producto actual cuenta con 15 subsistemas y su integración dan lugar al producto Cedrux.

Para el desarrollo de este producto se conformó un equipo multidisciplinario dentro del cual se encuentran los analistas del sistema, que tienen como tarea enfrentarse a una de las etapas más críticas en el proceso de desarrollo del software: el entendimiento del negocio y análisis del sistema. Esta etapa tiene la labor de guiar el desarrollo del software, de ahí que se considere fundamental para la construcción de un producto informático, además tiene gran impacto en el progreso de las demás, y se hace necesario dirigir toda la atención, recursos y esfuerzos a la misma. En este período el equipo de desarrollo se enfrentó a disímiles situaciones, lo cual provocó la existencia de los siguientes problemas durante esta etapa:

- Se omitieron funcionalidades en la etapa de análisis que luego surgieron en la etapa de implementación.
- La gran cantidad de subsistemas a desarrollar de forma simultánea, provocó que los errores en los requisitos se repitieran en las todas las soluciones.
- No se identificaron requisitos comunes, ni reutilizables por lo que se implementaron los mismos

requisitos en varios subsistemas.

Con el análisis de lo antes expuesto se plantea como **problema a resolver** que la forma actual de desarrollar los requisitos en el ERP-Cuba, está dificultando la reutilización de los mismos.

Para la resolución del problema planteado se asume como **objeto de estudio** la ingeniería de requisitos y como **campo de acción** la reutilización de requisitos en sistemas integrales de gestión.

Se traza como **objetivo general** de la investigación desarrollar una herramienta que permita gestionar información referente a los requisitos en el ERP-Cuba de manera que facilite la reutilización de los mismos.

Además se proponen los siguientes **objetivos específicos**:

- Realizar un marco teórico relativo a los sistemas de reutilización de requisitos.
- Implementar una herramienta para el desarrollo de la reutilización de requisitos.
- Realizar pruebas de funcionalidad a la herramienta para comprobar su correcto funcionamiento.
- Validar el correcto funcionamiento de la herramienta.

Para el cumplimiento de los objetivos propuestos se realizaron una serie de **tareas**, dentro de las cuales se encuentran:

- Estudio del estado del arte de la reutilización de requisitos.
- Estudio de los sistemas que existen para la reutilización de requisitos, tanto propietarios como libres, para analizar sus características, ventajas e inconvenientes y tomar una posición al respecto.
- Identificar los requisitos de la versión 1.0 de la herramienta.
- Describir los requisitos.
- Validar los requisitos.
- Realizar el análisis del sistema.
- Realizar el diseño del sistema.
- Implementar los requisitos a partir del análisis y el diseño antes desarrollado.
- Aplicar métricas de diseño para comprobar el correcto diseño de la herramienta.
- Realizar pruebas de caja negra a la herramienta para comprobar su correcto funcionamiento.

Se plantea como **idea a defender**: Si se desarrolla una herramienta que permita gestionar información referente a los requisitos en el ERP-Cuba contribuirá a la reutilización de los mismos.

Se ha determinado además que se utilizarán las estrategias y métodos de investigación siguientes:

El **método histórico lógico** se utilizará para realizar una revisión histórica del desarrollo de la Ingeniería de Requisitos y para determinar si actualmente están desarrollados sistemas informáticos similares, tomando una posición al respecto.

El **método hipotético deductivo** se utilizará para a partir del planteamiento de la hipótesis de la investigación y siguiendo reglas de deducción poder llegar a un nuevo conocimiento.

El **método de observación** es un método empírico que se utilizará para realizar una evaluación de la situación problemática en cuestión.

En la presente tesis el contenido está estructurado en tres capítulos, los cuales se describen brevemente a continuación:

CAPÍTULO 1. Fundamentación Teórica: Se describen los aspectos y conceptos asociados al dominio del problema de la investigación, para su mejor entendimiento. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta.

CAPÍTULO 2. Propuesta de Solución: Se exponen los requisitos a cumplir por la herramienta incluyendo los artefactos generados durante el diseño de la misma.

CAPÍTULO 3. Implementación y Prueba: Se exponen los artefactos generados durante la implementación conteniendo también las métricas y las pruebas utilizadas para la validación de la solución.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN

En este capítulo se presenta un estudio sobre el estado actual de la reutilización de requisitos de software. Se definen los principales conceptos relacionados con la ingeniería de requisitos, además de analizar algunas herramientas que actualmente permiten la reutilización de requisitos. Se selecciona las herramientas y tecnologías para el desarrollo de la aplicación. Finalmente, se definen algunos aspectos de la arquitectura sobre la que se basará el desarrollo de la herramienta que propone el presente trabajo de diploma.

1.2. MARCO CONCEPTUAL

En el mundo, se han desarrollado aplicaciones relacionadas con la Reutilización de Requisitos, cuyo estudio ha servido para tener en cuenta las ventajas y defectos que tienen. A menudo surgen dudas sobre cuál utilizar para este tipo de actividad, debido a que frecuentemente aparecen nuevas herramientas con nuevas mejoras. Es por esto, que surge la necesidad de hacer un estudio de las herramientas a utilizar, buscando obtener la información más actual sobre ellas.

El Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers IEEE por sus siglas en inglés) ha definido el concepto de requisito como.

- *Una condición o capacidad que necesita el usuario para resolver algún problema o alcanzar un objetivo.*
- *Condición o capacidad que debe cumplir o poseer un sistema o componente del sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto*. (IEEE, 1998)

La Real Academia Española (RAE) lo define como:

- *Circunstancia o condición necesaria para algo*. (*RAE*, 2009)

Según Craig Larman los requisitos se definen como *la cualidad necesaria de un sistema, una declaración que identifique una capacidad, una característica, o un factor de calidad de un sistema que proporcione valor y utilidad al cliente o usuario. Los requisitos son las descripciones de los servicios y las restricciones que se generan durante el proceso de ingeniería de requisitos.* (Larman, 1999)

Estas definiciones enmarcan el concepto de lo que algunos autores han referido acerca de la reutilización.

Según Johannes Sametinger *"la reutilización de software se puede definir como el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo"* (Sametinger, 1997)

Para Jag Sodhi y Prince Sodhi *"la reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes"* (Sodhi & Sodhi, 1999)

Por otra parte, Ian Sommerville ha plasmado que "*la reutilización es un enfoque de desarrollo [de software] que trata de maximizar el uso recurrente de componentes de software existentes"* (Sommerville, 1995).

A partir de lo antes mencionado se puede concluir que la reutilización de requisitos es el proceso de conformar nuevos productos de software a partir de requisitos identificados e implementados en una solución anterior, siendo adaptados a las necesidades del producto a desarrollarse Por lo que se puede definir como punto de partida para la solución del problema planteado en el diseño teórico.

1.3. LA INGENIERÍA DE REQUISITOS PARA LA REUTILIZACIÓN

Donald Reifer ha plasmado que *"la ingeniería de requisitos es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un coste reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario."* (Reifer, 1994)

Por la anterior razón, este epígrafe tiene como objetivo estudiar las actividades para la reutilización, las técnicas empleadas para reutilizar además de los patrones de reutilización.

1.3.1. Actividades de la reutilización

Cuando la reutilización está presente en un proceso de desarrollo software, este debe integrar todas las actividades necesarias para producir y reutilizar componentes de software. Así se distinguen dos actividades principales dentro de la reutilización: el desarrollo para la reutilización y el desarrollo con reutilización.

1.3.1.1. Desarrollo para la reutilización

El desarrollo para la reutilización consiste en la realización de un software que cumple con un conjunto de restricciones sobre su reutilización y su calidad. A la hora de desarrollar componentes reutilizables es fundamental centrarse en criterios de calidad, en la disminución de los costes de producción de los componentes.

Cuando se va a crear un componente reutilizable debe analizarse la variabilidad de los requisitos que satisface dicho componente, de forma que se construya como un componente genérico que pueda ser especializado en el momento de su reutilización para ajustarse a unos requisitos específicos.

1.3.1.2. Desarrollo con reutilización

Las técnicas utilizadas en el proceso de desarrollo con reutilización dependen en gran medida de los componentes que se hayan preparado en el proceso de desarrollo para la reutilización.

El desarrollo con reutilización consiste en la generación de nuevos productos software integrando elementos existentes, de forma directa o pasando por un proceso de adaptación. (2)

1.3.2. Técnicas de Reutilización

Esta es una de las temáticas que debería ser trabajada con más profundidad por la importancia que tiene para el desarrollo de la industria del software.

Aplicar la reutilización optimiza el tiempo de trabajo. En ocasiones se invierte mucho tiempo elaborando sistemas y componentes que ya han sido implementados anteriormente, y este, se podría reducir con solo utilizarlas y adaptarlas al nuevo sistema. Se podría incluso reducir el costo del producto y sus esfuerzos, logrando que la satisfacción de los clientes sea mucho mayor.

1.3.2.1. Marcos de Trabajo

Muchas de las personas que se involucran en el desarrollo de software principalmente los desarrolladores de aplicaciones web, desde sus inicios se enfrentan a disímiles conceptos de los cuales deben tener conocimiento a la hora de desarrollar sus soluciones, unos de estos conceptos es el de marco de trabajo.

Los marcos de trabajo son colecciones de patrones de diseño, interfaces, clases y código fuente real, que conforman un sistema que proporciona cierto apoyo al programador. Lo mejor es la práctica de uso y explotación de los marcos existentes y subsistemas en lugar de crear algo desde cero, propenso a introducir errores y que no cuente con respaldo mundial como lo tienen mucho de los principales marcos de trabajo. (3)

1.3.2.2. Línea de productos de software

Las Líneas de Producto son una estrategia ampliamente utilizada en numerosas ramas de la industria que consiste en la producción de múltiples productos utilizando una infraestructura común. Este enfoque se está introduciendo también en la ingeniería del software, en la medida en que ésta va siendo cada vez más una actividad industrial que necesita mejorar su eficiencia y optimizar sus costes.

Concepto de Línea de productos de software

Una línea de productos software, también llamada familia de aplicaciones en la literatura, es un conjunto de sistemas de software de una misma área de negocio y que tienen ciertas funcionalidades en común. (4)

Ventajas de las Líneas de Productos de Software

Las líneas de productos de software presentan ventajas en cuanto a:

- Reducción del esfuerzo de desarrollo dedicado a cada producto
- Reducción del tiempo de entrada en el mercado de cada producto

- Mantienen el tiempo de resolución de las peticiones del cliente constante, y no proporcional al número de productos.
- Permiten producir más funcionalidades en la misma cantidad de tiempo.
- Permiten producir más funcionalidades con la misma cantidad de dinero. (5)

1.3.2.3. Reutilización de productos COTS.

Los productos de software COTS (por sus siglas en inglés Commercial Off The Shelf) son productos que se aplican a un sistema software que puede utilizarse sin cambios por su comprador. Incluye muchas características y funciones para que sea potencialmente reutilizable en diferentes aplicaciones y entornos.

En la actualidad una de la formas de reutilizar este tipo de productos es mediante los sistemas grandes cuando tienen definidas interfaces API´S[1] que permiten programar el acceso a las funciones de dichos sistemas. Debido a la funcionalidad que estos productos COTS ofrecen, es posible reducir costes y tiempos de entrega en varios órdenes de magnitud comparados con el desarrollo de nuevo software. (6) (7)

Por lo que se puede resumir que los beneficios de reutilización de productos COTS son significativos debido a que estos sistemas ofrecen mucha más funcionalidad a la persona que los reutiliza, se ahorra mucho tiempo si se reutiliza un sistema existente, los tiempos de desarrollo del producto en construcción se reducen drásticamente.

1.3.2.4. Patrones de Diseño

Christopher Alexander a plasmado que *"cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de forma tal que esa solución puede ser usada un millón de veces, sin hacerlo de la misma manera dos veces."*(C. Alexander, 1979)

Es evidente que a lo largo de multitud de diseños de aplicaciones hay problemas que se repiten o que son similares, es decir, que responden a un cierto patrón. Por lo antes escrito se puede resumir que los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos.

[1] Interfaz de Programación (API por sus siglas en inglés de *Application Programming Interface*.) En otras palabras, son los métodos que el desarrollador de cualquier aplicación ofrece a otros desarrolladores para que puedan interactuar con su aplicación.

1.3.3. Patrones de Reutilización

1.3.3.1. Patrones de análisis:

Los patrones de análisis son hoy en día una de las mejores herramientas para asegurar el mantenimiento de una organización de software. Estos patrones parten de un objetivo de negocio hasta una solución técnica que desde la perspectiva del software no son otra cosa que un análisis que le indica al arquitecto de software cómo se debe enfocar una solución tecnológica. Estos patrones son un conjunto de clases y relaciones entre ellas, que tienen algún sentido en el contexto de una aplicación. Representan una estructura que puede ser válida para otras aplicaciones.

Características de los patrones de análisis

- Ayudan a construir la experiencia colectiva de Ingeniería de Software.
- Son una abstracción de "problema – solución".
- Se ocupan de problemas recurrentes.
- Identifican y especifican abstracciones de niveles más altos que componentes o clases individuales.
- Proporcionan vocabulario y entendimiento común.(8)

1.3.3.2. Patrones de Creación:

Los patrones de creación con los relacionados con la creación de instancias de clase. Estos se pueden dividir en los patrones de la creación de clases y los patrones de objetos de creación, los mismos facilitan y simplifican la creación de objetos. También permiten crear objetos sin definir la clase concreta, sólo la interfaz que debe implementar incluyendo reutilizar otros objetos en vez de crear nuevos debido a restricciones o eficiencia. Existen varios tipos de patrones de creación. A continuación se muestra una lista de ellos.

- **Abstract Factory:** Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.
- **Builder:** Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

- **Factory Method:** Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.

- **Prototype:** Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.

- **Singleton:** Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. (9)

La herramienta que se elabore tendrá como misión facilitar la reutilización de los requisitos en el ERP-Cuba ya que el propósito de la dicha reutilización es identificar descripciones de sistemas que puedan ser reutilizadas (en su totalidad o en el parte) con un mínimo de modificaciones, de manera que se reduzca el esfuerzo total de desarrollo. Esta tesis se centra en la reutilización de requisitos, para mejorar aspectos como la reutilización temprana o en el nivel de análisis, planes de gestión del proyecto, planes de gestión de calidad y pruebas de validación.

Respecto a esto Favaro afirma que “un requisito bien formulado, cuantificado, y reutilizable (...) es tan valioso como un módulo de software reutilizable”. Además Robertson y Robertson postulan que comenzar con un conjunto de requisito que han sido especificados para otros proyectos o dominios sirve para mejorar la precisión de la especificación de requisitos y para reducir el tiempo para elaborar esta especificación. (9)

Como afirman Robertson y Robertson, “cualquier diagrama (de contexto, de clases, se uso) o especificación puede ser utilizado para hacer los requisitos visibles. De manera que cualquiera de estas fuentes puede ser reutilizable.”(10)

1.4. HERRAMIENTAS PARA LA REUTILIZACIÓN

CAPTAINFEATURE: fue desarrollada en la Universidad de Ciencias Aplicadas de Kaiserslautern en Alemania, 2002. Es una herramienta para el modelado con un configurador integrado para especializar la función de crear diagramas y fue desarrollada para el análisis en el dominio de las familias del software del sistema. (11)

FEATUREIDE: Alemania desarrolló el FeatureIDE en 2005. *FeatureIDE* es un IDE [2]basado en Eclipse que soporta todas las fases del desarrollo de software orientado al desarrollo de los niveles: análisis del dominio, la aplicación de dominio, el análisis de requisitos, y la generación de software. (112

[2] **Integrated Development Environment (Entorno de Desarrollo Integrado)**

ODYSSEY: Se trata de un entorno de desarrollo de software cuyo objetivo es construir una infraestructura de reutilización basado en modelos de dominio y el desarrollo basado en componentes. Fue desarrollado por la Universidad Federal de Río de Janeiro, Brasil, en 2002. (13)

PURE::VARIANTS: Implementado por la compañía Pure-Systems en Alemania en 2003, Pure::Variants tiene tres tipos de licencia, una libre y dos de pago: una profesional y una empresarial. El análisis de esta herramienta se desarrolló con la versión que posee licencia libre. Es la herramienta para delinear y gestionar de forma eficiente todas las partes de productos de software y sus componentes, restricciones y condiciones de uso. (14)

REQUILINE: Se desarrolló en Alemania, en el año 2005. Es una herramienta de ingeniería de requisitos que fue creada con el fin de apoyar la gestión eficiente de las líneas de productos, permite modelarlas con sus características y requisitos, además de obtener las configuraciones del producto a partir del modelo especificado. (15)

XFEATURE: Fue desarrollada en 2005 por la P&P Software Company con la colaboración de Swiss Federal Institute of Technology. Es una herramienta para el modelado de las familias de productos y se presenta como un plugin para la plataforma Eclipse. (16)

1.3.1 Planificación de las Funcionalidades

Tabla 1 Funcionalidades que cada herramienta soporta.

Herramientas	Funcionalidades															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CaptainFeature	X	X	X	X										X	X	
FeaturelDE	X	X	X	X										X	X	
Odyssey	X	X	X					X		X	X					X
Pure::Variants	X	X	X	X				X	X				X	X	X	
RequiLine	X	X	X	X		X	X	X	X	X	X		X	X	X	X
XFeature	X	X	X	X			X	X	X					X	X	X

A continuación se describe cada una de las funcionalidades y aparecen en el orden que se van mencionando en la Tabla 1:

El grupo de modelado representa el ámbito de dominio definido en la fase de planificación. Sus funcionalidades son:

1. **La representación de dominio:** Representa el alcance definido. Esta representación puede ser a través de tablas, modelos, árboles y otros. Este modelo intenta formalizar las variaciones en el dominio.

2. **Variabilidad:** Representa las características de variabilidad que puede tener. Los tipos posibles son opcionales – puede o no estar presente en el producto; alternativos - de un grupo de características de una sola función será en el producto, y / o - de un grupo de características al menos una característica estará en el producto.

3. **Características obligatorias:** representan las características que siempre estarán en los productos.

4. **Normas de composición:** crear restricciones en el ámbito de representación y sobre las características. Pueden ser de exclusión mutua y la dependencia, las expresiones regulares, o la inteligencia artificial, entre otros.

5. **Identificación del grupo de características:** clasifica las características de acuerdo al tipo de información que representan. Pueden ser la capacidad, la tecnología de dominio, las técnicas de aplicación y el ambiente de la operación.

6. **Tipos de relación:** ofrece diferentes tipos de relaciones entre las características.

7. **Atributos de funciones:** permite la inclusión de información específica para cada función. También puede representar un punto de variación, si el número de variantes ya que es demasiado grande, proporcionando un modelo de características más conciso.

Las funciones del grupo de validación no dependen de los grupos anteriores, pero que proporcionan una mayor comprensión del dominio. Ellos son:

8. **Documentación de dominio:** proporciona documentación sobre el dominio, que consiste en: descripción, contexto, las dependencias entre los sistemas en el dominio, entre otros. Cada herramienta proporciona un conjunto diferente de campos para la documentación.

9. **Documentación de características:** proporciona documentación para todas las características del dominio, con información como: descripción, justificación, prioridades, etc. Cada herramienta proporciona un conjunto diferente de campos para la documentación.

10. **Gestión Requisitos:** proporciona soporte para la inclusión de requisitos y / o casos de uso de la herramienta.

11. **Relación entre las características y requisitos:** relaciona las características existentes de un dominio a los requisitos (funcionales o no funcionales) y / o casos de uso definidos. A través de esta relación es posible mantener la trazabilidad entre los artefactos producidos en el dominio.

12. **Diccionario:** identifica y define las palabras de uso común en el dominio y en su mayoría se encuentran en pre-análisis.

13. **Informes:** genera informes sobre la información disponible en el dominio. Los informes se pueden representar, por ejemplo, el número de combinaciones posibles, la frecuencia con que aparecen en las características del producto, la documentación de los artefactos.

14. **Verificación de consistencia:** verifica si el dominio generado sigue las reglas de composición creada.

Funcionalidades extra.

Además de las funcionalidades descritas en los grupos anteriores, que son intrínsecas al análisis de dominio, otras dos funcionalidades que hacen referencia a un paso después del dominio, es decir, la obtención de productos, que se incluyeron. Ellos son:

15. **Derivados del producto:** identifica las características que pertenecen a un producto de acuerdo con las características definidas para el dominio.

16. **Documentación del producto:** proporciona documentación para cada producto con información como la descripción del producto y la versión de dominio. (17)(18)

A partir del estudio del análisis realizado a las herramientas se observó que proporcionan casi todas las necesidades básicas exigibles a una herramienta de reutilización de requisitos para que sea incorporada por las entidades. Sin embargo, dicho estudio revela que ninguna de estas herramientas cubren las necesidades identificadas para dar soporte automatizado a la reutilización de requisitos en el ERP-Cuba, puesto que presentan limitaciones en cuanto a licencia por lo que no cumple con el principio de una independencia tecnológica y además presentan problemas de usabilidad y baja facilidad de uso. Por lo que se decidió realizar una aplicación para dar soporte automatizado a la reutilización de requisitos en el proyecto ERP-Cuba utilizando como guía el modelo de desarrollo orientado a componentes seleccionado por el CEIGE.

Se desarrollará una **Aplicación Web**, que es una aplicación que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una Intranet mediante un navegador y sus ventajas más significativas son:

Compatibilidad Multiplataforma: una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows, Linux, Mac, etc.

Actualización: las aplicaciones web siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones y realizar tareas de instalación.

Acceso inmediato y desde cualquier lugar: las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas y configuradas. Además pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.

Menos requisitos de hardware: este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM[3] en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor en donde reside la aplicación.

Seguridad en los datos: los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco. (19)

1.5. MODELO DE DESARROLLO DE SOFTWARE.

Para el desarrollo de esta herramienta se utilizará un modelo de desarrollo orientado a componentes, seleccionado por el Centro para la Informatización de Gestión de Entidades (CEIGE). Este es un modelo de desarrollo orientado a las necesidades y artefactos generados durante el proceso de desarrollo de cualquier producto en el centro CEIGE. Es una combinación de diferentes metodologías de las cuales se ha tomado lo que sería más conveniente para llevar a término un proyecto. Este modelo de desarrollo permite la generación de artefactos de vital importancia en el análisis y el diseño como son:

- Modelo de proceso de negocio.
- Descripción de procesos de negocio.
- Modelo conceptual.
- Prototipo de interfaz de usuario.
- Especificación de requisitos.
- Modelo de datos.

[3] *Random-access memory (Memoria de Acceso Aleatorio)*

- Diagrama de clases.
- Descripción del diseño de clases.
- Diagrama de componente.
- Diagrama de despliegue.
- Casos de prueba.

Características del modelo

- 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.
- Es un método muy estructurado que funciona bien con gente de poca experiencia.
- 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. (20)

1.6. HERRAMIENTA Y LENGUAJE PARA EL MODELADO

1.6.1. Herramientas CASE

Las Herramientas de Ingeniería de Software Asistida por Computadoras (Computer Aided Software Engineering, CASE por sus siglas en inglés) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Dentro de sus objetivos está realizar el diseño del proyecto, cálculo de costo, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación y detección de errores. (21)

1.6.1.1. Visual Paradigm 6.4

Visual Paradigm para Lenguaje Unificado de Modelado (UML: Unified Modeling Language, por sus siglas en inglés) 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. Este software ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite crear diagramas de clases, generar código desde diagramas y documentación. (22)(23)(24)

Dentro de sus características fundamentales están:

- Multiplataforma.
- Interoperabilidad.

- Modelamiento de los Requisitos.
- Editor de Detalles de Casos de Uso.
- Modelado de Procesos de Negocio.
- Modelamiento de Bases de Datos.

1.6.2. Lenguaje de modelado UML

El UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. Fue diseñado para ser un lenguaje de modelado de propósito general, que ofrece gran flexibilidad y expresividad y puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes. Igualmente es empleado para modelar aplicaciones de muy diversos dominios de aplicación. (25)

1.7. HERRAMIENTAS Y TECNOLOGÍAS

Teniendo en cuenta que el presente trabajo de diploma está centrado en el desarrollo de una herramienta para la reutilización de requisitos, y para ello se adopta la arquitectura definida por el proyecto ERP Cuba, la misma está basada en desarrollo por componentes. Estas tecnologías y herramientas serán descritas en este epígrafe.

1.7.1. LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una computadora, este consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural. (26)

1.7.1.1. Lenguaje del lado del servidor

Estos lenguajes de programación en la tecnología cliente-servidor[4] son clasificados así, ya que se ejecutan del lado del servidor y los usuarios sólo obtienen el beneficio del procesamiento de la información. (27)

Lenguaje PHP 5

[4] La arquitectura Cliente/Servidor tiene como objetivo optimizar el uso tanto del hardware como del software, a través de la separación de funciones: el cliente, quien inicia una determinada petición y el servidor, dedicado a responder dichas peticiones.

PHP (acrónimo recursivo de PHP: Hypertext Pre-processor) es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. Su interpretación y ejecución se realiza en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, enriquecida con código PHP, el servidor interpretará las instrucciones mezcladas en el cuerpo de la página y las sustituirá con el resultado de la ejecución antes de enviar el resultado a la computadora del cliente.

Además es posible utilizarlo para generar archivos PDF o JPG. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle lo cual permite la creación de Aplicaciones web muy robustas.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, GNU/Linux, Windows y Mac OS X. (28)

1.7.1.2. Lenguajes del lado del cliente

Un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Es necesario tener en cuenta que en dependencia del navegador algunas páginas no se verán bien si el ordenador cliente no tiene instalados los plugin adecuados. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad. (29)

Lenguaje Javascript

JavaScript es un lenguaje de scripts desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML[5]. Sus características más importantes son:

- JavaScript es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.
- JavaScript es un lenguaje orientado a eventos. Cuando un usuario pincha sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante JavaScript se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos.

[5] HyperText Markup Language (Lenguaje de Marcado de Hipertexto)

- JavaScript es un lenguaje orientado a objetos. El modelo de objetos de JavaScript está reducido y simplificado, pero incluye los elementos necesarios para que los scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

- Javascript es soportado por la mayoría de los navegadores como Internet Explorer, Google Chrome, Opera, Mozilla Firefox, entre otros. (30)

Lenguaje XHTML

XHTML, acrónimo en inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. En su versión 1.0, XHTML es solamente la versión XML (eXtensible Markup Language) de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium (WWWC) de lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas. La versión 1.1 es similar, pero parte a la especificación en módulos.

Las principales ventajas del XHTML sobre el HTML son:

- Se pueden incorporar elementos de distintos espacios de nombres XML [6]

- Como es XML se pueden utilizar fácilmente herramientas creadas para procesamiento de documentos XML genéricos.(31)

1.7.2. Apache 2.2.6

Apache, es un servidor de protocolo para la transferencia de hipertextos (Hypertext Transfer Protocol, HTTP por sus siglas en inglés) de software libre para plataformas Unix, Windows, y Macintosh, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Las características que lo definen son las siguientes:

- Multiplataforma

- Es un servidor de web conforme al protocolo HTTP/1.1

- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.

[6] Extensible Markup Language (lenguaje extensible de marcado).

- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor. (32) (33)

1.7.3. Base de Datos PostgreSQL 8.4

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional basado en el proyecto POSTGRES, de la universidad de Berkeley. Es una derivación libre (OpenSource)

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Características de PostgreSQL

A continuación se enumeran las principales características de este gestor de bases de datos:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, entre otros. También permite la creación de tipos propios.
- Incorpora una estructura de datos como arreglos.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas y orientadas a operaciones con redes.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos. (34)(35)

1.7.4. Mozilla Firefox 4.0.1

Mozilla Firefox 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 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.
- 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. (36)(37)

1.7.5. NetBeans IDE 6.9

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso. (38)

1.7.6. Marco de trabajo

Los marcos de trabajos (framework) son 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. (39)

Dentro de las tecnologías definidas por el CEIGE se encuentra el marco de trabajo Sauxe el cual tiene como desventaja que al utilizarlo se debe hacer uso de todas las demás tecnologías que tiene incluida,

dígase librería Extjs 2.2, lo que para la herramienta a desarrollar no era de utilidad ya que no trae la funcionalidad de generar gráficos y es por eso que se decidió utilizar el marco de trabajo Symfony en su versión 1.4.11 y la librería Extjs en su versión 3.0, que da la facilidad de generar los gráficos necesarios para la aplicación a desarrollar.

1.7.6.1. Symfony Framework 1.4.11

Symfony es un marco de trabajo para construir aplicaciones web con PHP. En otras palabras, Symfony es un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web y su licencia es de tipo software libre.

Symfony emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista. (40)

Características de Symfony

Symfony fue diseñado para ajustarse a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Doctrine, permiten cambiar con facilidad de Sistema Gestor de Base de Datos (SGBD) en cualquier fase del proyecto.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.
- Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones Web que para pequeños proyectos.
- Código fácil de leer que incluye comentarios y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo.(41)

1.7.6.2. ExtJS Framework 3.0

Es un marco de trabajo JavaScript del lado del cliente para el desarrollo de aplicaciones Web. Tiene un sistema dual de licencia: Comercial y Código Abierto. Este marco de trabajo puede correr en cualquier

plataforma que pueda procesar POST y en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso del DOM. Los datos son obtenidos mediante mensajes AJAX a través de XML y/o JSON. (42)(43)

Dispone de un conjunto de componentes para incluir dentro de una aplicación web, como:

- Cuadros y áreas de texto.
- Campos para fechas.
- Campos numéricos.
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.

1.7.6.3. Doctrine Framework 0.9

Es un potente y completo sistema ORM (Object Relational Mapper, por sus siglas en inglés) para PHP 5 que incorpora una capa de abstracción a base de datos. Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (44)(45)

1.8. TIPOS DE ARQUITECTURAS A UTILIZAR

Toda arquitectura de software debe describir diversos aspectos del desarrollo de una aplicación. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial

de una misma arquitectura y es deseable que exista cierto solapamiento[7] entre ellos. A continuación de describen las arquitecturas que serán utilizadas para el desarrollo de la herramienta. (46)

1.8.1. Arquitectura cliente/servidor

La arquitectura cliente-servidor (C/S)[8] consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta.

Características

En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Al receptor de la solicitud enviada por el cliente se conoce como servidor. Sus características son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales. (47)

1.8.2. Arquitectura en tres capas

Es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos.

Capas y niveles

Capa de presentación: Esta capa es la que se muestra públicamente, presentándole el sistema al usuario, le comunica la información y captura los datos relacionados con el mismo en un mínimo de proceso, esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica (para el usuario generalmente se presenta como formularios).

[7] Solapamiento: Cubrir una cosa a otra en su totalidad o en parte.

[8] Cliente-Servidor

Capa de negocio: Aquí es donde, se reciben las peticiones del usuario y se envían las respuestas tras el procesamiento de las mismas. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

Capa de datos: Es donde residen los métodos de acceso a datos con el código necesario para acceder a los mismos. Utiliza o se vale de drivers[9] de acceso a datos para comunicarse con el gestor de base de datos que realizan todo el almacenamiento de datos. Esta capa recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (48)

1.9. CONCLUSIONES DEL CAPÍTULO I.

En este capítulo se exponen conceptos necesarios sobre reutilización, las herramientas que se han desarrollado para la reutilización en el mundo de la industria del software y aspectos en general de la ingeniería de requisitos para la reutilización, con el objetivo de obtener un mayor dominio sobre estos temas y poder tomar decisiones importantes para la realización de la herramienta propuesta, tales como la elección de los lenguajes de programación, en este caso PHP 5 del lado del servidor y Java Script en el lado del cliente con la librería ExtJS 3.0, la base de datos a usar es PostgreSQL 8.4, como herramienta CASE Visual Paradigm para UML 6.4 y el navegador web Mozilla Firefox 4.0.11 además del Apache 2.2.6 como servidor de aplicaciones..

[9] Controlador de dispositivo, llamado normalmente controlador (en inglés, device driver) es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz -posiblemente estandarizada- para usarlo.

CAPÍTULO 2 PROPUESTA DE SOLUCIÓN

2.1. INTRODUCCIÓN.

El siguiente capítulo aborda sobre los resultados obtenidos durante el proceso de desarrollo de la solución, así como algunos de los artefactos generados por el mismo. Primeramente son descritos los requisitos funcionales de la solución, mostrando una propuesta de lo que se desea lograr. Posteriormente se realiza una descripción del diseño elaborado por el autor para alcanzar las metas trazadas, además de la estrategia a seguir durante el proceso de implementación.

2.2. PROCESO DE INGENIERÍA DE REQUISITOS

Las siguientes definiciones enmarcan el objetivo de lo que algunos autores han referido acerca de la ingeniería de requisitos

Según Roger Pressman la ingeniería de requisitos se puede definir como *"un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software."* (Pressman, 2005)

Para Thayer y Dorfman "*la ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional."* (Thayer, et al., 1997)

A partir de las citas anteriores se puede resumir que la ingeniería de requisitos es un proceso que se centra en la búsqueda de las necesidades del cliente. Es por esto que el presente epígrafe tiene como objetivo mostrar cómo se realizó el proceso de desarrollo de los requisitos de la solución al problema existente.

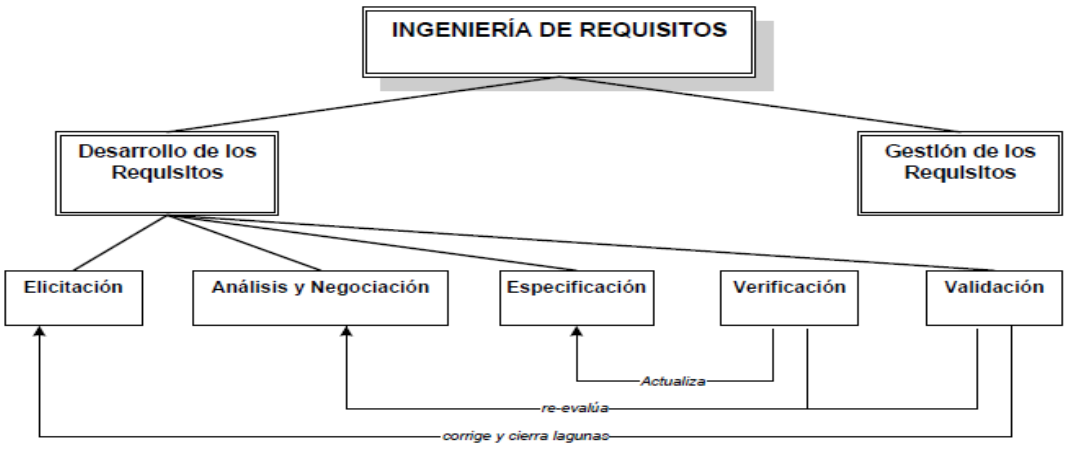

Ilustración 1 Dominio y sub-procesos de la IR

El proceso de **DESARROLLO DE LOS REQUISITOS** del sistema actual se desarrolló de la siguiente forma.

Elicitación: Se realizó a partir del estado del arte realizado acerca de los sistemas de reutilización de requisitos, en el cual se identificaron una serie de requisitos candidatos.

Análisis y negociación: Se presentaron los requisitos candidatos a la analista principal del proyecto ERP-Cuba y mediante un taller donde se utilizó la *tormenta de idea*, se definieron los requisitos que contendría la herramienta en su primera versión.

La tormenta de idea es una de las técnicas utilizadas en la Ingeniería de Requisitos (IR) para la captura de requisitos y se decidió utilizar porque es un método que se implementa en un grupo para ayudar al surgimiento de nuevas ideas y dar solución a un problema, esta técnica es muy popular, ya que todos en un grupo pueden dar sus ideas y dejar una en particular o unirlas para crear una solución más compleja.

Otras técnicas existentes para la captura de requisitos son: reuniones informales, entrevistas (con o sin cuestionario), mera observación, estudio de documentación existente, JAD (Joint Application Development/ Desarrollo conjunto de aplicaciones), pero estas no se utilizan en el desarrollo de la tesis.

Especificación: Se describieron los requisitos usando las plantillas del proyecto ERP- Cuba.

Verificación y validación: En este caso se hizo por la analista principal del proyecto ERP-Cuba, ya que este es un producto interno del centro y se realizó utilizando prototipos no funcionales y realizando revisiones técnicas a las especificaciones de requisitos.

2.3. PROCEDIMIENTO PARA IDENTIFICAR UN REQUISITO REUTILIZABLE

Un requisito es una condición necesaria que debe cumplir un sistema, estos pueden ser reutilizables con el objetivo de reducir los costos, el tiempo y sobre todo obtener como resultado soluciones con mejor calidad.

Siguiendo la línea de lo antes planteado se definió el siguiente procedimiento para identificar aquellos requisitos que pueden ser reutilizables en futuros desarrollos de software, partiendo como base, los requisitos del producto CEDRUX.

En los requisitos identificados para el desarrollo de la solución propuesta se trata información sobre:

- Parámetros
- Relación
- Transacción
- Restricción

Dicho procedimiento se apoyará en la métrica que se fundamenta a continuación:

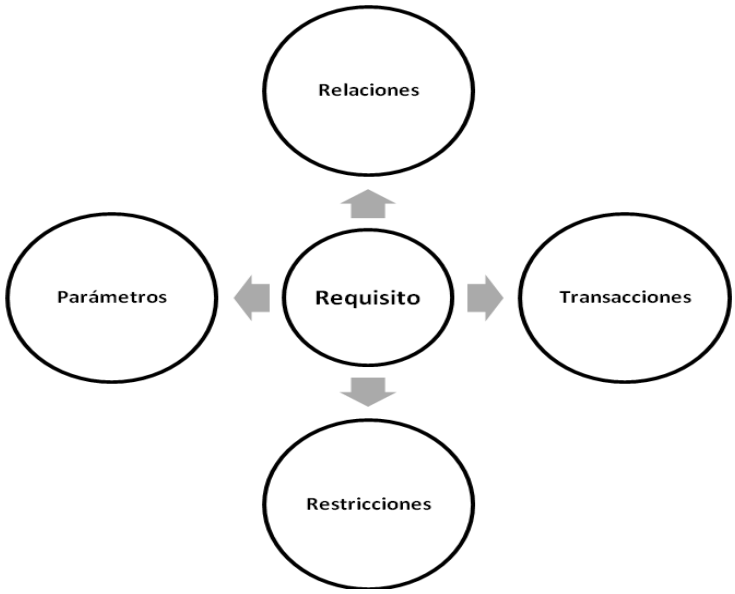

Ilustración 2 Composición de un Requisito

En la ilustración 2 se ilustra la composición de un requisito. A continuación se muestra cómo se determina si un requisito es reutilizable.

2.3.1. Parámetros

El 25% que representa los parámetros de un requisito que incluyen *Nombre*, *Proceso de Negocio, Componente, Versión, Agrupación, Objetivo, Referencia* y la *Complejidad* se definen que serán reutilizables, debido a que son altamente parametrizados, es decir que pueden ser ajustados a las necesidades del producto que lo reutilice.

2.3.2. Relación

Existen 4 tipos de relaciones entre requisito que son:

- **De padre a hijo:** No necesariamente hay que reutilizar al padre a no ser que sea el hijo susceptible a tener traza por no ser reutilizable.
- **Bidireccional:** Si se reutiliza A se debe considerar la posibilidad de reutilizar B.
- **Requiere:** A requiere de B por lo que es obligatoria la reutilización de los dos requisitos.
- **Exclusiva:** Si está A no tiene que estar B.

Por lo antes se definió que una relación va ser reutilizable cuando exista una relación de **Requiere** debido a la obligatoria inserción de todos los requisitos que tengan este tipo de relación.

2.3.3. Transacción

- ¿Cuántas transacciones existen en el requisito?

- ¿Cuántas son reutilizables?

Partiendo de lo anterior se propone las siguientes fórmulas a seguir:

$$\frac{Tr}{Tnr}$$

Ecuación 1 Ecuación para calcular la parte que representa las transacciones reutilizables del total.

Tr: Cantidad total de transacciones reutilizables

Tnr: Cantidad total de transacciones existentes en el requisito.

$$\frac{Ttr}{0.25}$$

Ecuación 2 Ecuación para calcular la parte que representan las transacciones reutilizables en un requisito.

Ttr: Total de transacciones reutilizables en el requisito.

2.3.4. Restricción

- ¿Cuántas restricciones existen en el requisito?
- ¿Cuántas son reutilizables?

Partiendo de lo anterior se propone las siguientes fórmulas a seguir:

$$\frac{Rr}{Rnr}$$

Ecuación 3 Ecuación para calcular la parte que representa las transacciones reutilizables del total.

Rr: Cantidad total de restricciones reutilizables

Rnr: Cantidad total de restricciones existentes en el requisito.

$$\frac{Trr}{0.25}$$

Ecuación 4 Ecuación para calcular la parte que representan las restricciones reutilizables en un requisito.

Trr: Total de restricciones reutilizables en el requisito.

A partir de todo lo antes descrito se plantea la siguiente métrica de reutilización:

Tabla 2 Métrica para determinar si un requisito es reutilizable.

Dimensión	Indicador	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpreta-ción del valor obtenido	Unidad de medida
Reutilización	Índice de reutilización	¿Cómo identificar los requisitos reutilizables en un sistema?	Se identificarán todos aquellos requisitos que puedan ser reutilizados en el sistema y todos aquellos requisitos candidatos a reutilizarse por otro sistema.	$Ir = \frac{A}{C}$ A= Cantidad de elementos que componen a un requisito y se desean reutilizar. C=Cantidad de componentes que posee un requisito. Ir: Índice de reutilización.	**0 <= Ir <= 1** A mayor cercanía al 1 resultará más reutilizable.	_Reutilizable. (0.60 - 1 Ir). _No reutilizable. (0 – 0.59 Ir).

A partir de lo antes planteado se propone una métrica para calcular el porciento de uso del requisito lo que apoyará a determinar si un requisito es potencialmente reutilizado como se fundamenta a continuación:

Tabla 3 Métrica para calcular si un requisito es potencialmente reutilizado.

Dimensión	Indicador	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpreta-ción del valor obtenido	Unidad de medida

Reutilización	Índice de uso	Cuán usados son los requisitos reutilizables por otros productos.	Se identificarán todos aquellos productos que se desarrollen a partir de requisitos reutilizados identificados con anterioridad.	$Is = \frac{A-1}{C-1}$ A= Cantidad de veces que se reutiliza un requisito. C=Cantidad de producto que reutilizan el requisito. Is: Índice de uso.	**0 <= Is <= 1** A mayor cercanía al 1 resultará más potencialmente reutilizable.	___Potencialmente reutilizado. (1 Is). _Altamente reutilizado. (0.81-0.99 Is). _Medianamente reutilizado. (0.41-0.80 Is). _Bajamente reutilizado. (0 – 0.40 Is).

2.4. PROPUESTA DEL SISTEMA

El sistema propuesto debe manejar los requisitos reutilizables de un proyecto de manera eficiente, permitiendo a un usuario crear un proyecto a partir de requisitos reutilizables identificados anteriormente por el sistema. Debe generar un reporte con la información manejada, ya que esta puede servir para tomar decisiones importantes en los proyectos a desarrollar en el futuro. El sistema presentará al usuario una interfaz amigable y fácil de usar. Debe ser una herramienta en línea de forma que ayude a centralizar la reutilización de los requisitos en el ERP-Cuba.

2.5. MODELO CONCEPTUAL

El modelo conceptual no es más que un modelo visual de un sistema que ilustra las interconexiones de los componentes del modelo y tiene como objetivo facilitar el entendimiento de los conceptos del dominio. A continuación de muestra el modelo conceptual de la herramienta ReuReq. (Ver ilustración 4)

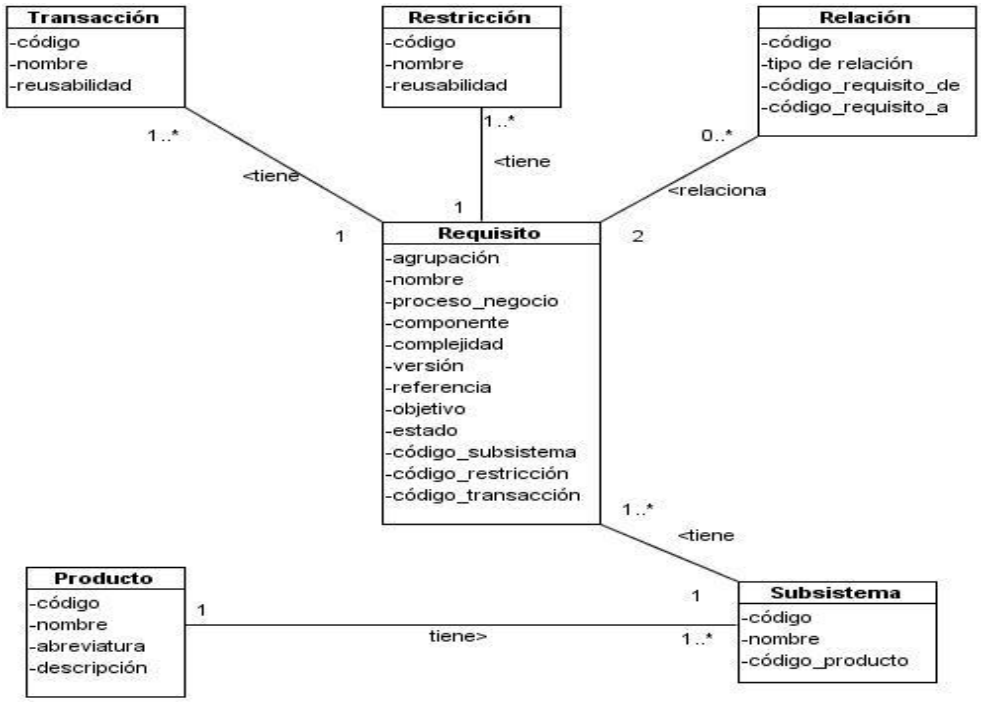

Ilustración 4 Modelo Conceptual

A continuación de muestran los conceptos más significativos del modelo conceptual. (Ver ilustración 4)

- **Requisitos:** Aspectos fundamentales que debe cumplir un software.
- **Subsistemas:** Descomposición del producto en varias áreas distintas para su mejor desarrollo.
- **Productos:** Proyectos que serán conformados con los requisitos que genere el sistema.
- **Relaciones:** Relaciones que existen entre dos o más requisitos.
- **Restricciones:** Restricciones o validaciones que poseen los requisitos.
- **Transacciones:** Flujo básico de un requisito.

2.6. LISTADO DE REQUISITOS

Un requisito funcional es una característica requerida del sistema que expresa una capacidad de acción del mismo (una funcionalidad) generalmente expresada en una declaración en forma verbal. Durante la captura de requisitos del sistema a implementar, se identificó un total de 35 requisitos como se muestra a continuación:

2.6.1. Requisitos Funcionales (RF)

RF1. Gestionar Requisitos.

RF1.1. Adicionar un nuevo requisito.

RF1.2. Modificar un requisito existente.

RF1.3. Eliminar un requisito.

RF1.4. Buscar Requisitos.

RF1.5. Listar Requisitos.

RF1.6. Búsqueda Avanzada de Requisitos

RF2. Gestionar Producto.

RF2.1. Adicionar un nuevo producto.

RF2.2. Modificar un producto existente.

RF2.3. Eliminar un producto.

RF3. Gestionar Subsistema.

RF3.1. Adicionar un nuevo subsistema.

RF3.2. Modificar un subsistema existente.

RF3.3. Eliminar un subsistema.

RF3.4. Listar Subsistema

RF4. Gestionar Restricciones.

RF4.1. Adicionar una nueva restricción.

RF4.2. Modificar una restricción existente.

RF4.3. Eliminar una restricción.

RF5. Gestionar Transacciones.

RF5.1. Adicionar una nueva transacción.

RF5.2. Modificar una transacción existente.

RF5.3. Eliminar una transacción.

RF6. Gestionar Relaciones.

RF6.1. Adicionar una nueva relación.

RF6.2. Eliminar una relación.

RF6.3. Listar Relaciones.

RF7. Conciliar Catálogo.

RF7.1. Adicionar requisito reutilizable a catálogo específico.

RF7.2. Listar requisitos reutilizables.

RF7.3. Listar requisitos específicos.

RF8. Asignar Restricciones.

RF8.1. Adicionar restricción reutilizable a catálogo específico.

RF.8.2 Listar restricciones específicas.

RF.8.3 Listar restricciones reutilizables.

RF9. Asignar Transacciones.

RF9.1. Adicionar transacción reutilizable a catálogo específico.

RF9.2. Listar transacciones específicas.

RF.9.3 Listar transacciones reutilizables.

RF13. Generar Reporte Producto.

RF15. Ver estabilidad de requisitos de un producto.

RF16. Ver variabilidad.

RF18. Ver detalles.

A continuación se muestra la tabla con la descripción textual del requisito Adicionar Requisito (Ver tabla 4). Las restantes descripciones ver en: (Anexo 2)

Tabla 4 Descripción textual del requisito: Adicionar Requisito.

Precondiciones	Debe existir al menos un subsistema insertado.
Flujo de eventos	
Flujo básico	
1	Seleccionar la opción Adicionar Requisito.
2	Introducir datos del requisito.
3	Dar clic en el botón Aceptar.
4	El sistema confirma que se adicionó correctamente el requisito.
Pos-condiciones	
1	Se adiciona satisfactoriamente el requisito.
Flujos alternativos	
Flujo alternativo 1.a Información errónea	
1	El sistema no permitirá que el usuario introduzca datos de números en los campos de texto, ni texto en campos que requieran números.
2	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 1.b Información incompleta.	
1 El sistema señala los datos vacíos y permite corregirlos.	
2 El usuario corrige los datos.	
3 Volver al paso 2 del flujo básico.	

Pos-condiciones	
1	N/A

Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.

Pos-condiciones	
1	No se adiciona el Requisito.

Validaciones	
1	N/A

Relaciones	**Requisitos Incluidos**	
	Extensiones	N/A.
Conceptos		**Visibles en la interfaz:** nombre proceso de negocio versión reusabilidad objetivo complejidad agrupación referencia componente
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Prototipo elemental de interfaz gráfica de usuario

Ilustración 3 Interfaz de usuario para el requisito funcional Adicionar Requisito.

2.6.2. Requisitos no Funcionales (RNF)

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, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

El presente trabajo forma parte de un proceso productivo iniciado por el CEIGE y los resultados que se obtengan formarán parte del marco de trabajo Sauxe. Por tanto los requisitos no funcionales con los que debe cumplir la aplicación a desarrollar fueron establecidos por el centro al inicio del proceso de desarrollo, a continuación se describen los más importantes. (48)

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra maniobras no autorizadas o que puedan afectar la integridad de los datos. La vigilancia al sistema así como la salva de la información, se realizará de forma centralizada por el administrador, además el sistema debe mostrar opción de advertencia antes de borrar cualquier elemento o información que pueda existir.

Software

Para el cliente:

- Navegador Mozilla Firefox 2.2 o superior.
- Sistema operativo Windows XP o GNU/Linux.

Para el servidor:

- Sistema operativo GNU/Linux en cualquiera de sus distribuciones.
- Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.
- Un servidor de base de datos PostgreSQL 8.3 o superior.

Hardware

Para el servidor:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Al menos 50Gb de espacio libre en disco duro.
- Tarjeta de red.

Para el cliente:

- Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- Tarjeta de red.

Confiabilidad

El subsistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error. Estará disponible todo el tiempo, permitiendo el trabajo a los usuarios y las acciones de mantenimiento. Este debe ser estable, fiable y la velocidad de respuesta debe ser rápida durante la utilización del mismo. La información almacenada debe ser confiable en cuanto a su veracidad e integridad desde su recopilación.

Portabilidad

El subsistema será multiplataforma (Linux-Windows) lo que permitirá ejecutarse sobre diferentes sistemas operativos sin importar sus versiones, y sin necesidad de modificar su código fuente.

2.7. MODELO DE DISEÑO.

La fase de diseño expande y detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones básicas. El propósito del diseño es especificar una solución que trabaje fácilmente en el código fuente y construir una arquitectura simple y fácilmente extensible.

2.7.1. Diagrama de Clases del diseño

El Diagrama de Clases del Diseño (DCD) es donde se muestran las relaciones entre las clases del modelo, las controladoras y las vistas, así como sus atributos y operaciones y las relaciones entre las mismas. A continuación de muestran el diagramas de clases del diseño de la herramienta ReuReq. (Ver ilustración 4)

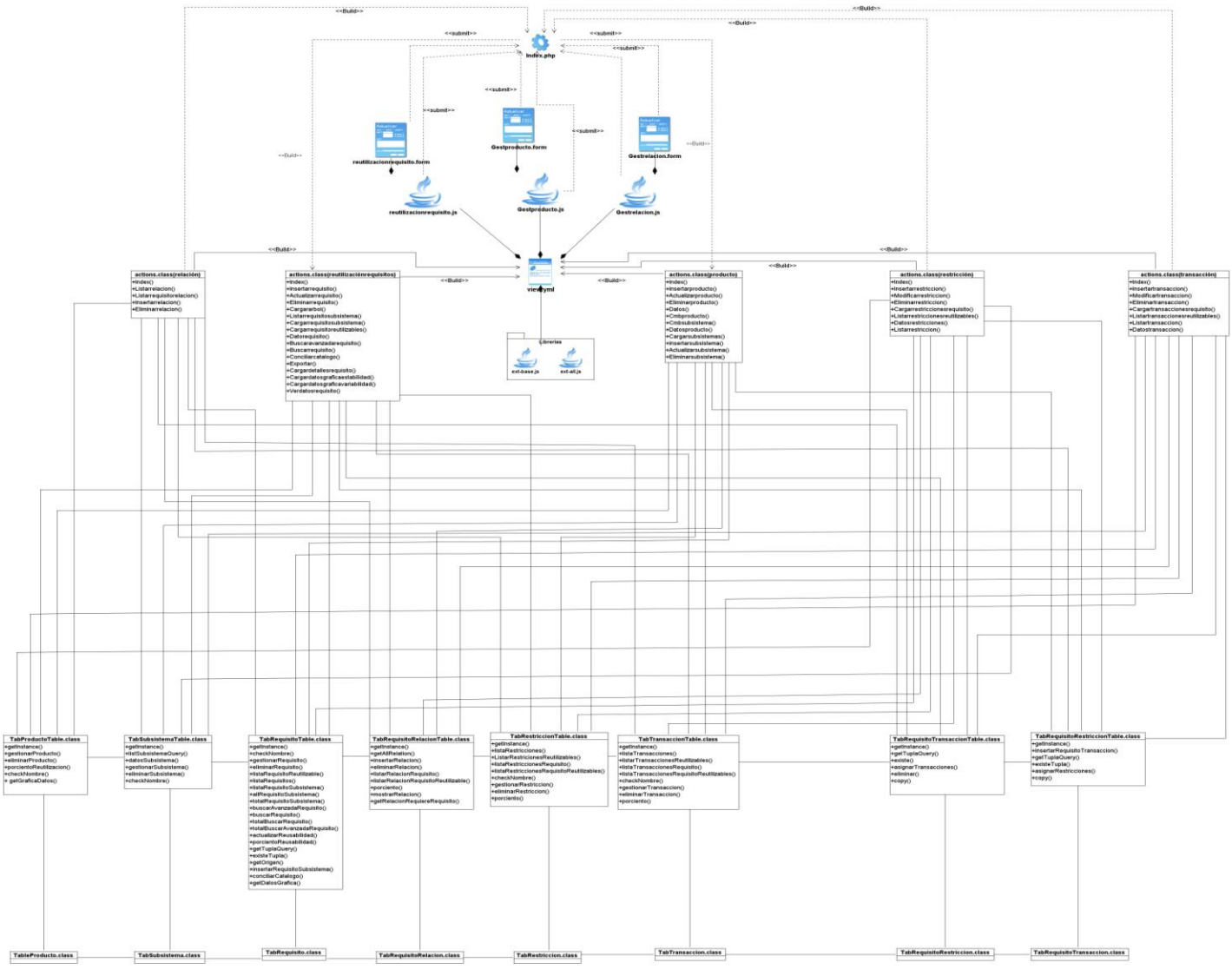

Ilustración 4 Diagrama de Clases del Diseño

- **Ext-base.js:** Librería que genera el marco de trabajo Ext y que es utilizada para la realización de la herramienta.
- **Ext-all.js:** Librería que genera el marco de trabajo Ext y que es utilizada para la realización de la herramienta.
- **reutilizacionrequisitos.js:** Clase donde se encuentra el código correspondiente a la interfaz que se utiliza para Gestionar Requisitos.
- **gestproducto.js:** Clase donde se encuentra el código correspondiente a la interfaz que se utiliza para la Gestionar Producto.
- **View.yml:** Archivo donde se encuentran todas las direcciones de la clase principal a utilizar.

- **Action.Class (reutilizacionrequisitos):** Clase controladora donde se encuentran todos los métodos referentes a la gestión de los requisitos.
- **Action.Class (producto):** Clase controladora donde se encuentran todos los métodos referentes a la gestión de los productos.
- **Action.Class (relación):** Clase controladora donde se encuentran todos los métodos referentes a la gestión de las relaciones.
- **Action.Class (restricción):** Clase controladora donde se encuentran todos los métodos referentes a la gestión de las restricciones.
- **Action.Class (transacción):** Clase controladora donde se encuentran todos los métodos referentes a la gestión de las transacciones.
- **TabSubsistemaTable:** Clase donde se implementa la lógica del negocio referente a los subsistemas.
- **TabRelacionTable:** Clase donde se implementa la lógica del negocio referente a las relaciones.
- **TabRestriccionTable:** Clase donde se implementa la lógica del negocio referente a las restricciones.
- **TabTransaccionTable:** Clase donde se implementa la lógica del negocio referente a las transacciones.
- **TabRequisitosTable:** Clase donde se implementa la lógica del negocio referente a los requisitos.
- **TabProductoTable:** Clase donde se implementa la lógica del negocio referente a los productos.
- **TabSubsistema:** Clases donde se realizan las consultas a la base de datos referente a los subsistemas.
- **TabRelacion:** Clases donde se realizan las consultas a la base de datos referente a las relaciones.
- **TabRestriccion:** Clases donde se realizan las consultas a la base de datos referente a las restricciones.
- **TabTransaccion:** Clases donde se realizan las consultas a la base de datos referente a las transacciones.
- **TabRequisitos:** Clases donde se realizan las consultas a la base de datos referente a los requisitos.

- **TabProducto:** Clases donde se realizan las consultas a la base de datos referente a los productos.
- **TabRequisitoRestriccionTable:** Clase donde se implementa la lógica del negocio referente a los requisitos en conjunto con las restricciones.
- **TabRequisitoRestriccion:** Clases donde se realizan las consultas a la base de datos referente a las restricciones y los requisitos.
- **TabRequisitoTransaccionTable:** Clase donde se implementa la lógica del negocio referente a los requisitos en conjunto con las transacciones.
- **TabRequisitoTransaccion:** Clases donde se realizan las consultas a la base de datos referente a las transacciones y los requisitos.
- **reutilizacionrequisitos.form:** Formulario que envía la interfaz reutilización de requisitos.
- **Gestproducto.form:** Formulario que envía la interfaz gestionar producto.
- **Gestrelacion.form:** Formulario que envía la interfaz gestionar relaciones.

2.7.2. Patrones utilizados

Patrón modelo- vista- controlador

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 interfaz 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.

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (49)

2.7.3. Modelo de Datos

El modelo de datos actualmente cuenta con un total de 8 tablas cada una con diferentes funciones como se explica a continuación (Ver ilustración 5):

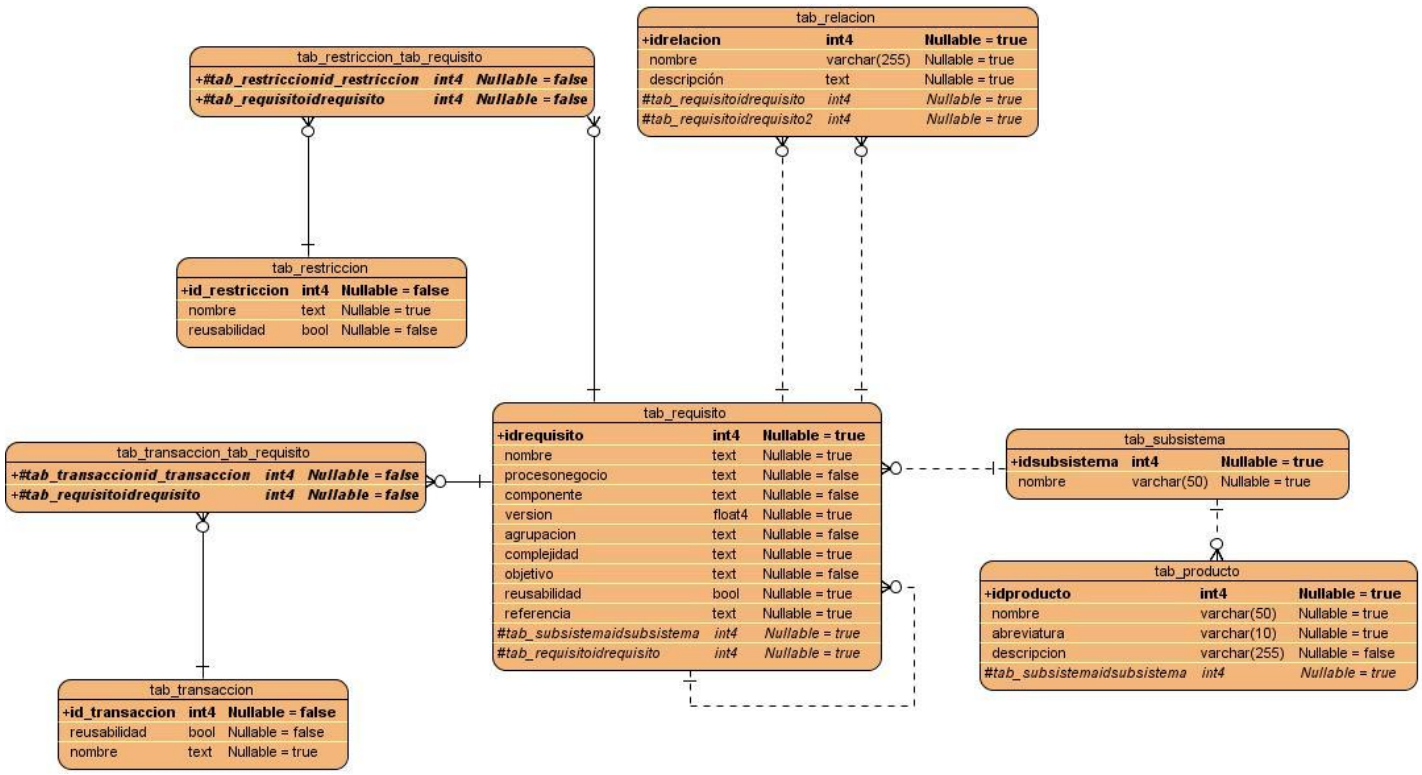

Ilustración 5 Diagrama de base de datos

- **tab_subsistemas:** Tabla que guarda la información relacionada con los requisitos que tiene cada subsistema de un producto.
- **tab_productos:** Tabla que guarda la información relacionada con los atributos que lo componen.
- **tab_requisitos:** Tabla que guarda la información relacionada con los requisitos existentes de cada producto.
- **tab_relacion:** Tabla que guarda la información relacionada con las relaciones entre requisitos
- **tab_transaccion:** Tabla que guarda la información relacionada con las transacciones.
- **tab_requisitos_tab_transacciones:** Tabla que se genera después de una cardinalidad de mucho a muchos, por lo que se guardarías los identificadores de los requisitos y las transacciones a los que estos pertenecen.
- **tab_restrincion**: Tabla que guarda la información relacionada con las restricciones.
- **tab_requisitos_tab_restricciones:** Tabla que se genera después de una cardinalidad de mucho a muchos, por lo que se guardarías los identificadores de los requisitos y las restricciones a los que estos pertenecen.

2.8. CONCLUSIONES PARCIALES

En el presente capítulo se desarrolló un procedimiento que permitirá identificar a los requisitos reutilizables dentro de la solución propuesta, además de realizó la modelación del sistema, efectuándose los diagramas de clases del diseño, así como el diseño de la base de datos. Además, se describieron detalladamente los requisitos identificados. Este capítulo es de suma importancia para el desarrollo del sistema, los artefactos generados constituyen la entrada para el flujo de trabajo de implementación, debido a que brinda una vista de la arquitectura del sistema, y describe detalladamente las acciones a tomar en el siguiente flujo de trabajo.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA

3.1. INTRODUCCIÓN

El trabajo en este capítulo parte de los resultados obtenidos en el capítulo anterior y se procederá a describir el modelo de implementación con el objetivo de definir la estructura y organización de la aplicación, además de implementar las clases encontradas durante el diseño representado en los diagramas de componentes.

3.2. MODELO DE IMPLEMENTACIÓN

El modelo de implementación describe cómo los elementos del modelo de diseño, se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (50)

3.2.1. Diagramas de Componentes.

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Son usados para estructurar los elementos en los sistemas de software. Ellos examinan y controlan las dependencias entre módulos o sus interfaces. Un componente representa una parte modular, desplegable y reutilizables de un sistema. (51)(Ver ilustración 6)

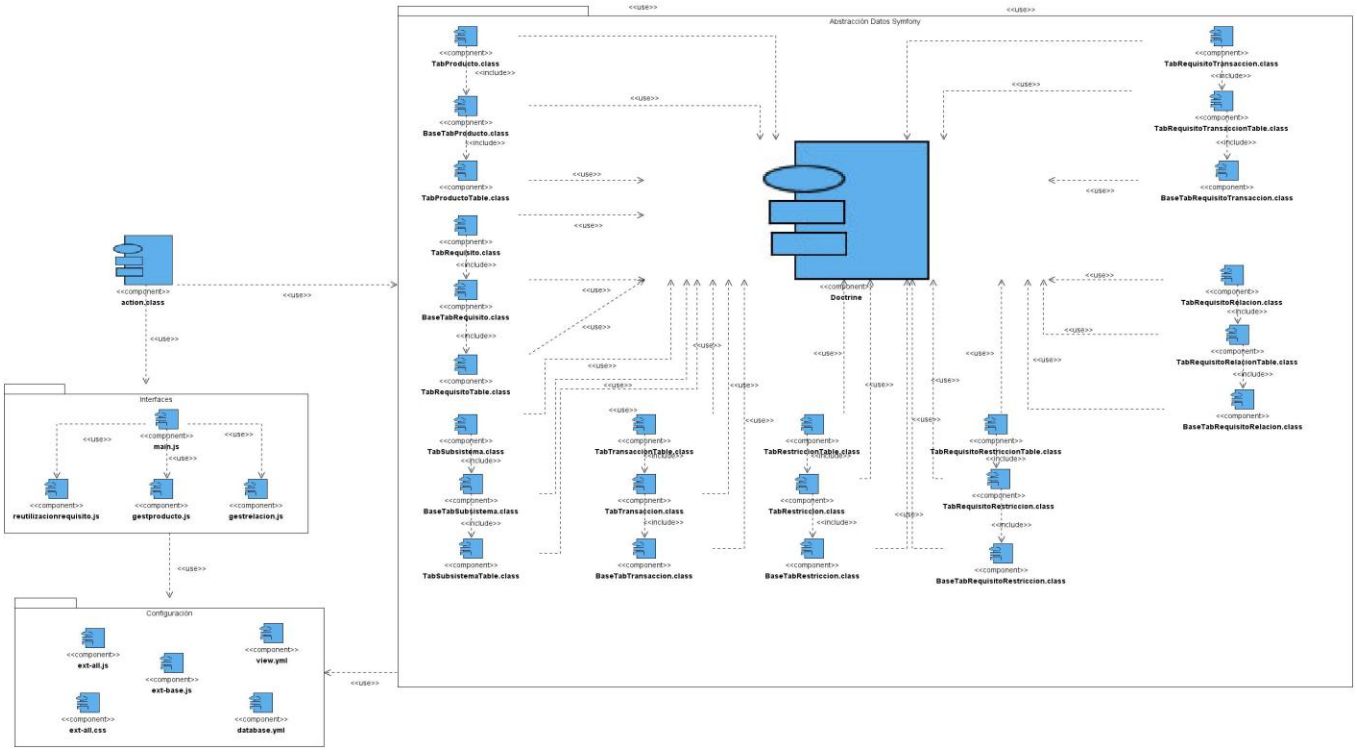

Ilustración 6 Diagrama de Componentes

- **Componente Abstracción Datos Symfony**: Contiene todas las clases entidades y todas están hacen uso del marco de trabajo Doctrine.
- **Componente Action.Class:** Agrupa todas las clases controladoras de la herramienta ReuReq.
- **Componente Interfaces:** Reúne todas las clases interfaces de la herramienta ReuReq.
- **Componente Configuración:** Concentra toda la configuración de la librería ExtJS además de la configuración de la base de datos y de la página principal de la herramienta ReuReq.

3.2.2. Estándares de codificación

Los estándares de codificación son reglamentos de la programación que están enfocadas a la estructura y apariencia física del programa con el objetivo de facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación comprende todos los aspectos de la generación de código. Los programadores deben implementar un estándar de forma prudente que tienda siempre a lo práctico. La legibilidad del código fuente influye directamente en el entendimiento que puedan tener los desarrolladores del equipo de trabajo, todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades. El mejor método para lograr que un grupo de desarrolladores mantenga un código de calidad es establecer un estándar de codificación sobre el cual se realizarán revisiones rutinarias. (52)

Los Estándares utilizados en la codificación fueron los siguientes:

- **Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.
- **Notación camelCasing:** Es parecido al PascalCasing con la excepción que la letra inicial del identificador no debe estar en mayúscula.

Nomenclatura de las clases

Los nombres de las clases comenzarán con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing.

El nombre de las clases estará dado según la función que cumplan, es decir se le agregará al nombre original un prefijo o sufijo que describa el tipo de clase que es la misma. A continuación se listan los prefijos y sufijos determinados para cada tipo de clase.

- **Clases controladoras:** A las clases controladoras se le adiciona el sufijo “Actions”, ejemplo: productoActions.
- **Clases entidades:** A las clases pertenecientes al modelo de negocio se le adiciona el sufijo “Table”, ejemplo: TabRequisitoRelacionTable.

Nomenclatura de las funciones

Los nombres a emplear para las funciones se escribirán con minúscula, en caso de que sea un nombre compuesto se empleará la notación camelCasing.

En el caso particular de las funciones de las clases controladoras se les adiciona el prefijo “execute”, ejemplo: executeInsertarproducto.

3.3. MÉTRICAS DE DISEÑO

Las métricas del software permiten medir de forma cuantitativa la calidad de los atributos internos del producto, esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema. Son varios los puntos de vista relacionados con la calidad del software. (53) El objetivo de este epígrafe es desarrollar una evaluación del diseño propuesto para la herramienta Reutilización de Requisitos (ReuReq). Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente.

Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software, inclinando sus objetivos a mejorar la comprensión de la calidad del producto, estimar efectividad del proceso y mejorar la calidad del trabajo. Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad (54):

- **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 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 costes 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, modulo, clase, conjunto de clases, etc.) diseñado.

Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad son las siguientes:

- **Tamaño operacional de clase (TOC):** Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 5 Atributos de calidad evaluados por la métrica 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.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 6 Criterios de evaluación para la métrica 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
	Baja	>2*Promedio

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

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 7 Atributos de calidad evaluados por la métrica 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.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 8 Criterios de evaluación para la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
	Baja	<=Promedio

Complejidad de mantenimiento	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

- **Resultados obtenidos de la aplicación de la métrica TOC**

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 75% de las clases empleadas en el sistema posee 6 operaciones o menos lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización).

A continuación se muestran los resultados obtenidos.

Tabla 9 Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
accion.class(requisito)	17	Alta	Alta	Baja
accion.class(producto)	12	Alta	Alta	Baja
accion.class(restricción)	8	Media	Media	Media

accion.class(transacción)	8	Media	Media	Media
accion.class(relación)	5	Media	Media	Media
TabProductoTable	6	Media	Media	Media
TabRequisitosTable	21	Alta	Alta	Baja
TabSubsistemaTable	6	Media	Media	Media
TabSubsistema	0	Baja	Baja	Alta
TabProducto	0	Baja	Baja	Alta
TabRequisitos	0	Baja	Baja	Alta
TabRequisitoRelacionTable	9	Media	Media	Media
TabRequisitoRelacion	0	Baja	Baja	Alta
TabTransaccionTable	9	Media	Media	Media
TabTransaccion	0	Baja	Baja	Alta
TabRestriccionTable	9	Media	Media	Media
TabRestriccion	0	Baja	Baja	Alta
TabRequisitoRestriccionTable	6	Media	Media	Media
TabRequisitoRestriccion	0	Baja	Baja	Alta
TabRequisitoTransaccionTable	6	Media	Media	Media
TabRequisitoTransaccion	0	Baja	Baja	Alta

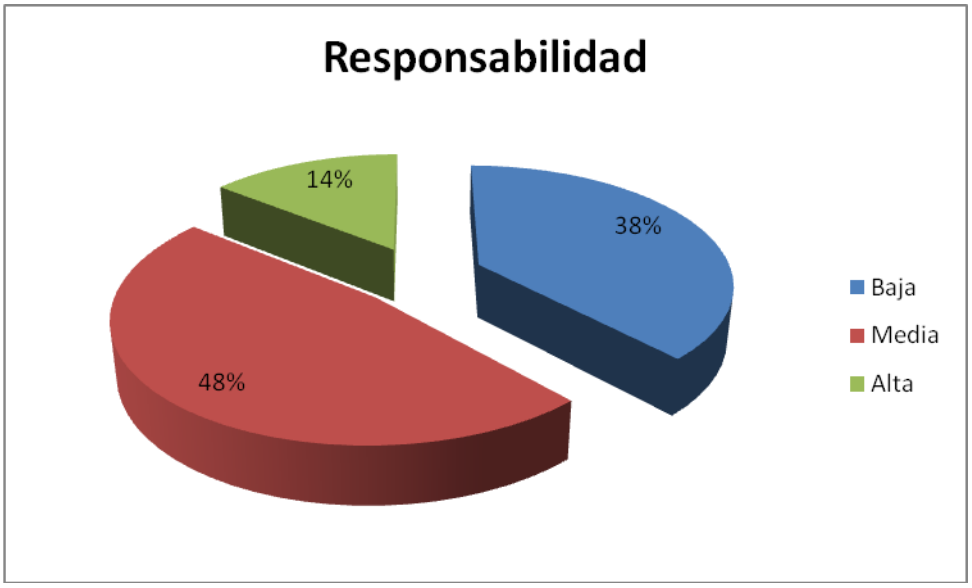

Ilustración 7 Resultados de la evaluación de la métrica TOC para el atributo responsabilidad.

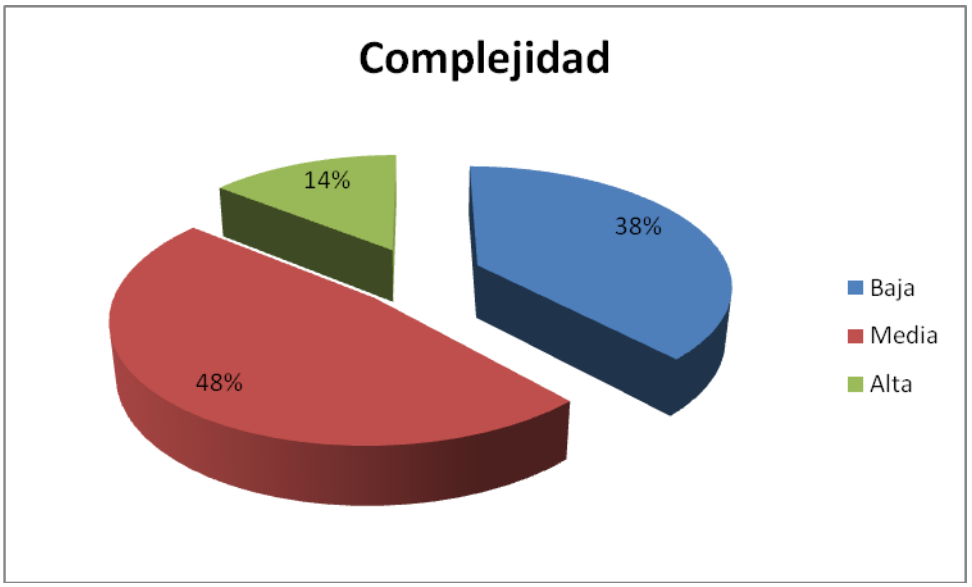

Ilustración 8 Resultados de la evaluación de la métrica TOC para el atributo complejidad.

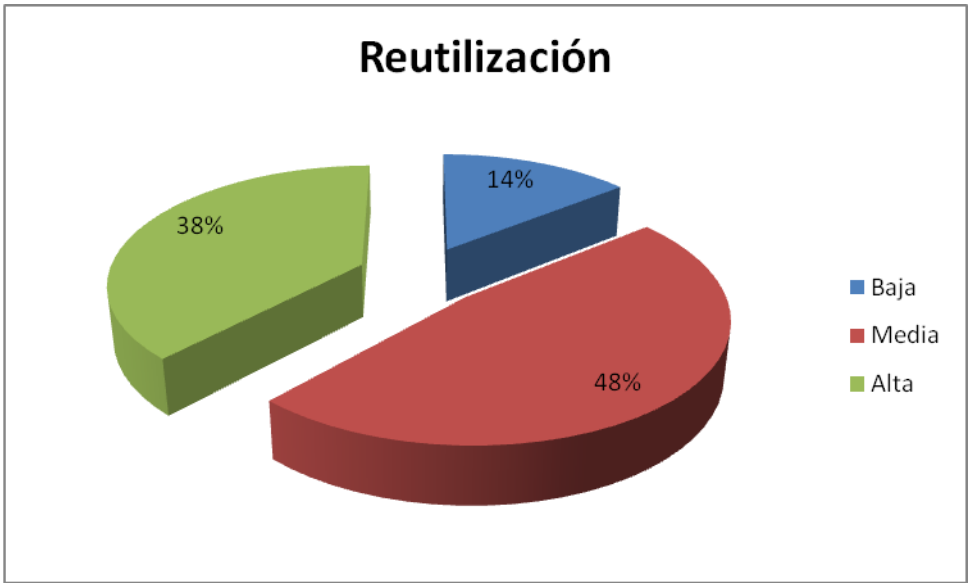

Ilustración 9 Resultados de la evaluación de la métrica TOC para el atributo reutilización.

Resultados obtenidos de la aplicación de la métrica RC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 9.52% de las clases empleadas posee más de 2 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización).

A continuación se muestran los resultados obtenidos.

Tabla 10 Instrumento de evaluación de la métrica RC.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
accion.class(requisito)	3	Alto	Alta	Baja	Alta
accion.class(producto)	2	Medio	Media	Media	Media
accion.class(restricción)	2	Medio	Media	Media	Media
accion.class(transacción)	2	Medio	Media	Media	Media
accion.class(relación)	2	Medio	Media	Media	Media

TabProductoTable	1	Bajo	Baja	Alta	Media
TabRequisitoTable	5	Alto	Alta	Baja	Alta
TabSubsistemaTable	1	Bajo	Baja	Alta	Baja
TabSubsistema	0	Ninguno	Baja	Alta	Baja
TabProducto	0	Ninguno	Baja	Alta	Baja
TabRequisito	0	Ninguno	Baja	Alta	Baja
TabRequisitoRelacionTable	1	Bajo	Baja	Alta	Baja
TabRequisitoRelacion	0	Ninguno	Baja	Alta	Baja
TabTransaccionTable	2	Medio	Media	Media	Media
TabTransaccion	0	Ninguno	Baja	Alta	Baja
TabRestriccionTable	2	Medio	Media	Media	Media
TabRestriccion	0	Ninguno	Baja	Alta	Baja
TabRequisitoRestriccionTable	1	Bajo	Baja	Alta	Baja
TabRequisitoRestriccion	0	Ninguno	Baja	Alta	Baja
TabRequisitoTransaccionTable	1	Bajo	Baja	Alta	Baja
TabRequisitoTransaccion	0	Ninguno	Baja	Alta	Baja

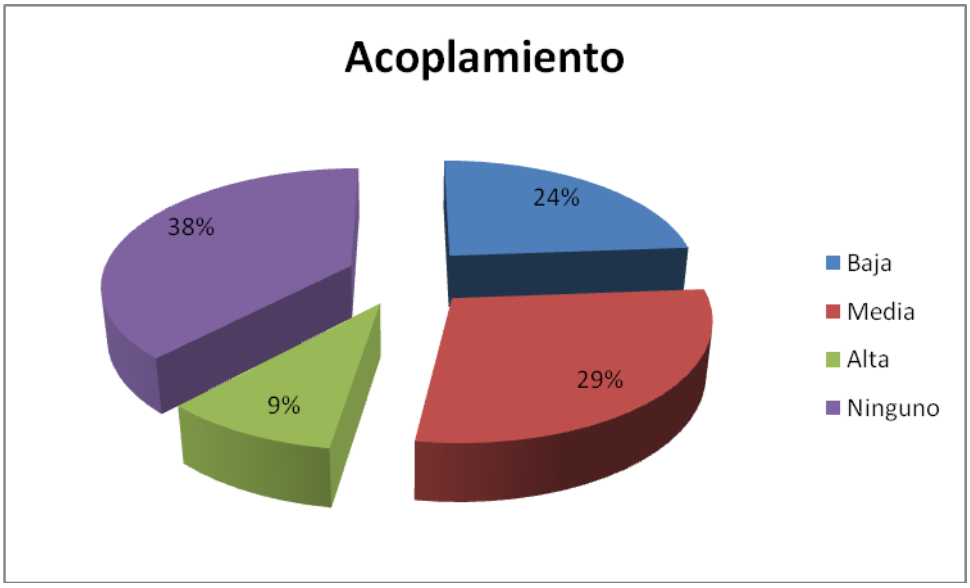

Ilustración 10 Resultados de la evaluación de la métrica RC para el atributo acoplamiento.

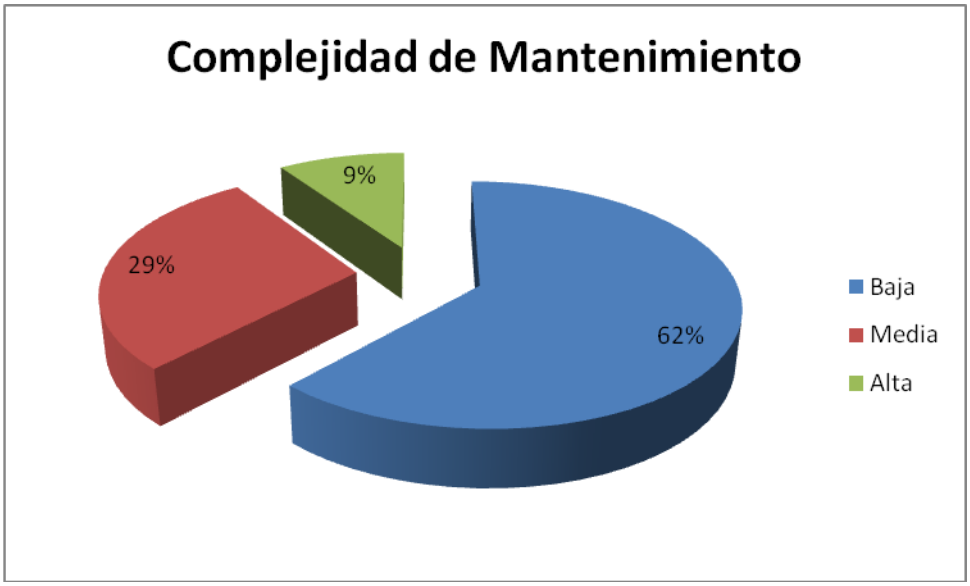

Ilustración 11 Resultados de la evaluación de la métrica RC para el atributo complejidad de mantenimiento.

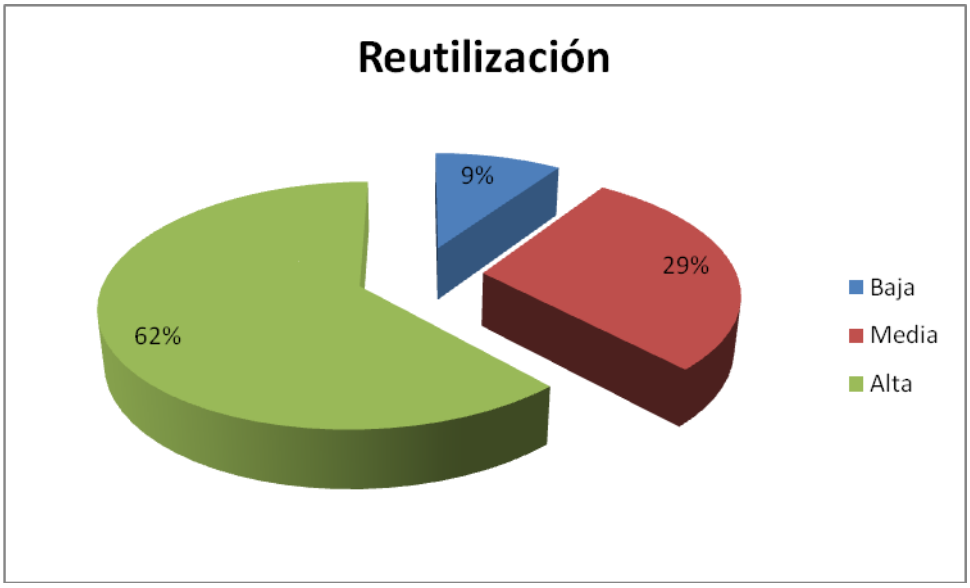

Ilustración 12 Resultados de la evaluación de la métrica RC para el atributo reutilización.

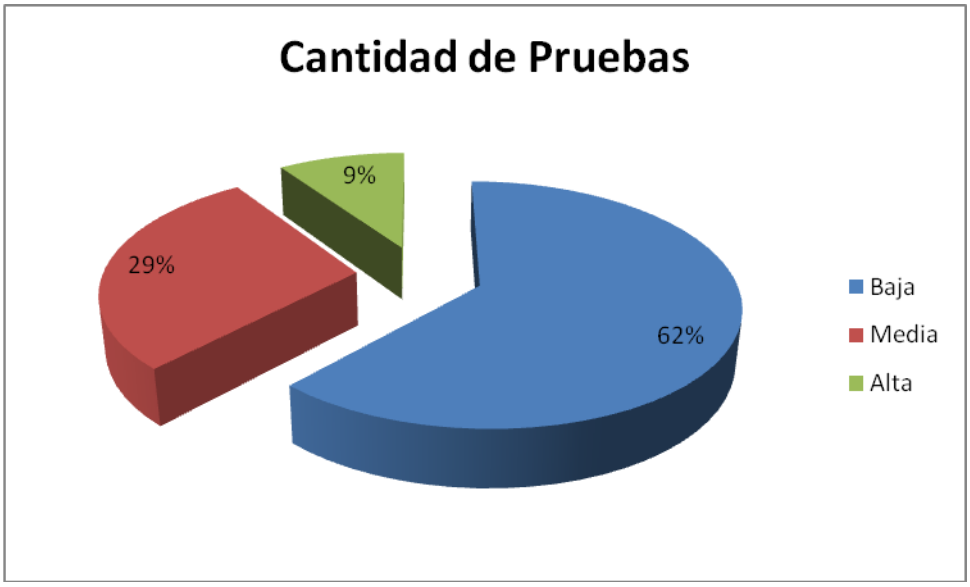

Ilustración 13 Resultados de la evaluación de la métrica RC para el atributo cantidad de pruebas.

Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Dicha matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guión simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

A continuación se muestran los resultados obtenidos.

Tabla 11 Resultados de la evaluación de la relación atributo/métrica.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de Implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de pruebas	(-)	1	1

Tabla 12 Rango de valores para la evaluación de la relación atributo/métrica.

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

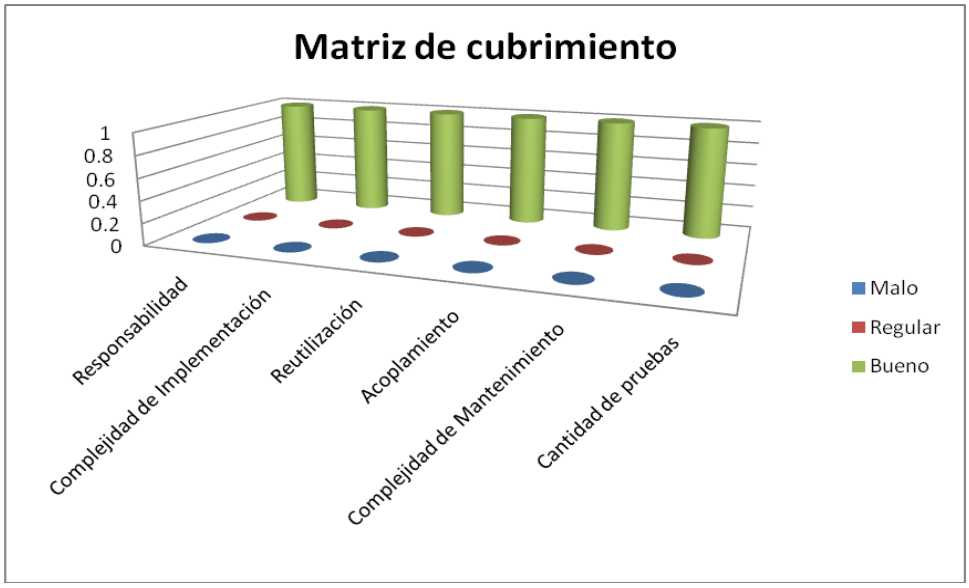

Ilustración 14 Resultados obtenidos de la evaluación de los atributos de calidad.

3.4. PRUEBAS DE SOFTWARE

En este epígrafe se le realizarán las pruebas de software a la aplicación, para lograr verificar y revelar la calidad del producto. Estas pruebas son utilizadas para identificar posibles fallos de las funcionalidades que se integran dentro de las diferentes fases del ciclo del desarrollo de dicha aplicación y para poder determinar el nivel de calidad y comprobar el grado de cumplimiento respecto de las especificaciones iniciales de la solución propuesta.

3.4.1. Pruebas de caja negra

Para el desarrollo de la prueba de caja negra se utilizará el requisito Adicionar Producto que se encuentra dentro de Gestionar Producto. El objetivo que persigue dicho requisito es adicionar los productos con sus respectivos datos de manera que estos puedan persistir en la base de datos dando la posibilidad de modificarlos o eliminarlos con posterioridad.

Condiciones de ejecución

- Debe seleccionar la opción Gestionar Producto.
- Debe seleccionar la opción Adicionar Producto.

A continuación de muestra una tabla con la descripción del requisito de prueba:

Tabla 13 Descripción del requisito Adicionar Producto.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar producto	Se realizará la adición de un producto.	EP 1.1: Adicionar producto introduciendo	En este escenario se debe seleccionar la raíz

		datos válidos	del árbol de Productos. Presionar el botón Adicionar Producto. Deberá introducir los datos correspondientes a la denominación, abreviatura y la descripción, siendo esta última opcional, el sistema muestra un mensaje de error en caso de la entrada de datos incorrectos. Debe presionar el botón Aceptar. El sistema muestra un mensaje: 'El producto fue insertado satisfactoriamente.'. Presionar el botón Aceptar.
		EP 1.2: Adicionar producto introduciendo datos inválidos.	En este escenario se debe seleccionar el árbol de productos. Presionar el botón Adicionar producto. Luego se introducen datos inválidos y se presiona el botón aceptar. El sistema no permite adicionar datos

		inválidos y muestra un mensaje de error: ‘Los datos introducidos son incorrectos.’. Se presiona el botón Aceptar. Se procede a introducir los datos válidos.
	EP 1.3: Adicionar producto dejando campos vacíos.	En este escenario se debe seleccionar el árbol de producto. Presionar el botón Adicionar Luego se presiona el botón aceptar dejando cualquiera de los campos obligatorios vacío. El sistema muestra un mensaje para que llene los campos vacíos: ‘No se pueden dejar campos vacíos.’. Presionar el botón Aceptar. Se procede a introducir los datos válidos.
	EP 1.4: Adicionar producto cuando ya existe en la Base de datos.	En este escenario se debe seleccionar el árbol de productos. Presionar el botón

		Adicionar. Luego se introducen los datos del producto donde en el campo Denominación se introducirá un nombre que ya existe en la Base de datos. El sistema muestra un mensaje de información: 'Ya existe un producto con ese nombre. ' . Se presiona el botón Aceptar.
	EP 1.5: Cancelar.	En este escenario se debe seleccionar el árbol de producto. Presionar el botón Adicionar productos. Se introducen los datos del producto. Se presiona el botón Cancelar. Se cierra la ventana sin realizar ninguna operación.

En la siguiente tabla se muestra la descripción de las variables utilizadas al Adicionar Producto.

Tabla 14 Variables utilizadas al Adicionar Producto.

No.	Nombre de campo	Tipo	Válido	Inválido
1	Nombre	Campo Texto	En este campo de	Vacío

			texto puede escribirse solamente letras.	
2	Abreviatura	Campo Texto	En este campo de texto puede escribirse solamente letras.	Vacío
3	Descripción	Campo Texto		NA

En la siguiente tabla se muestran los juegos de datos a probar y los resultados obtenidos para estos.

Tabla 15 Juegos de datos a probar y los resultados obtenidos.

Id del escenario	Escenario	Variable 1 (Nombre)	Variable 2 (abreviatura)	Variable 3 (Descripción)	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar producto introduciendo datos válidos.	V(CEDRUX)	V(CE)	V(El producto deberá tener un subsistema)	El sistema registra el producto en la Base de datos. El sistema muestra un mensaje de información de que fue insertado correctamente el producto: 'Producto insertado correctamente.'	"Producto insertado correctamente."
EP 1.2	Adicionar producto introduciendo datos inválidos.	I(h$%GT%&ĝ)	I(YXR5As7·$)	N/A	El sistema muestra un mensaje de error: 'Los datos introducidos son incorrectos.'	"El valor en este campo es inválido" "No se pueden dejar campos obligatorios vacíos."

EP 1.3	Adicionar producto dejando campos vacíos	Vacío	Vacío	Vacío	El sistema muestra un mensaje para que llene los campos vacíos: 'No se pueden dejar campos vacíos.'.	"No se pueden dejar campos obligatorios vacíos."
EP 1.4	Adicionar producto cuando ya existe en la Base de datos.	N/A	N/A	N/A	El sistema muestra un mensaje de información: "Ya existe un producto con ese nombre.".	"Ya existe un producto con ese nombre."
EP 1.5	Cancelar	N/A	N/A	N/A	Se cierra la ventana sin realizar ninguna operación.	Se cierra la ventana sin realizar ninguna operación.

De esta forma se le aplicaron las pruebas de caja negra a todas las funcionalidades del sistema. A continuación se resumen la cantidad de escenarios de prueba utilizados para cada funcionalidad, así como el número de no conformidades para los casos donde fueron encontradas estas. Los valores porcentuales que representan estos resultados se muestran en la ilustración 15 a continuación (Resultados obtenidos por escenarios al aplicar las pruebas de caja negra).

Funcionalidades:

1. Adicionar Producto: 5 escenarios.
2. Modificar Producto: 6 escenarios.
3. Eliminar Producto: 1 escenario.
4. Adicionar Subsistema: 4 escenarios.
5. Modificar Subsistema: 5 escenarios.
6. Eliminar Subsistema: 1 escenario.
7. Adicionar Requisito: 5 escenarios.
8. Modificar Requisito: 6 escenarios.
9. Eliminar Requisito: 1 escenario.
10. Buscar Requisito: 3 escenarios.
11. Búsqueda Avanzada Requisito: 3 escenarios.
12. Listar Requisito: 1 escenario.
13. Listar Requisito Reutilizables: 1 escenario.

14. Listar Requisito Específicos: 1 escenario.

15. Listar Subsistema: 1 escenario.

16. Adicionar Requisito Reutilizables a catálogo específico: 1 escenario.

17. Generar Reporte Producto: 1 escenario.

18. Adicionar Restricción: 5 escenarios.

19. Modificar Restricción: 6 escenarios.

20. Eliminar Restricción: 1 escenario.

21. Adicionar Transacción: 5 escenarios.

22. Modificar Transacción: 6 escenarios.

23. Eliminar Transacción: 1 escenario.

24. Adicionar Relación: 2 escenarios.

25. Eliminar Relación: 1 escenario.

26. Listar Relaciones: 1 escenario

27. Listar Restricciones Reutilizables: 1 escenario.

28. Listar Restricciones Específicas: 1 escenario.

29. Listar Transacciones Reutilizables: 1 escenario.

30. Listar Transacciones Específicas: 1 escenario.

31. Adicionar Restricciones Reutilizables a catálogo específico: 1 escenario.

32. Adicionar Transacciones Reutilizables a catálogo específico: 1 escenario.

33. Ver variabilidad: 1 escenario.

34. Ver estabilidad: 1 escenario.

35. Ver detalles: 1 escenario.

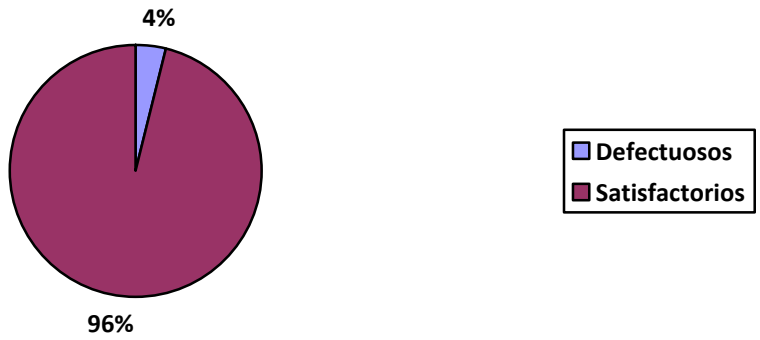

Ilustración 15 Resultados obtenidos por escenarios al aplicar las pruebas de caja negra

Como evidencian los resultados solo el 4 % de los escenarios de pruebas presentan no conformidades en una primera iteración. Las mismas fueron cubiertas para dejar en un 100% de funcionamiento el sistema en una segunda iteración.

3.5. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Para realizar la validación del sistema primeramente se aplicaron métricas de diseño y los resultados de dichas métricas aplicadas permitieron concluir que por lo general las clases no están cargadas en responsabilidad, existe bajo acoplamiento entre ellas así como un alto nivel de reutilización. Indican además que el diseño no es complejo, son bajas la complejidad de mantenimiento y la complejidad en las pruebas. Resultados que confirman la calidad del diseño. También se aplicaron pruebas de caja negra donde se aplicaron los casos de pruebas a la herramienta y se obtuvo en la primera iteración un 4% de no conformidades las cuales fueron corregidas y presentadas al cliente con un 100% de aceptación.

Por último el cliente emitió una carta de aceptación de las funcionalidades que posee la herramienta en su primera versión. (Ver Anexo1).

3.6. CONCLUSIONES PARCIALES

En este capítulo se crearon los diagramas correspondientes a esta fase en el desarrollo del software, se aplicaron métricas de diseño obteniendo la matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto evaluando el diseño de la aplicación de positivo. También se realizó un análisis de los resultados de las pruebas de caja negra aplicadas al sistema a través de los diferentes casos de pruebas que cubren las funcionalidades permitiendo encontrar errores que afectan el funcionamiento de este. Los errores encontrados con estas pruebas fueron corregidos lo que permitió que actualmente el sistema esté al 100% de funcionamiento.

CONCLUSIONES GENERALES

Se concluye al término de la presente investigación que se logró el desarrollo de una herramienta que permite gestionar información referente a los requisitos, facilitando la reutilización de los mismos y ayudando de esta forma a conformar nuevos productos de software con requisitos reutilizables en el ERP-Cuba, dándole esta forma cumplimiento al objetivo general.

Alcanzándose adicionalmente los siguientes resultados:

- Se realizó un estudio sobre los principales sistemas de reutilización de requisitos el cual permitió crear las bases del conocimiento expuesto en el documento.
- Se aplicaron las pruebas de funcionalidades al sistema obteniéndose resultados positivos.
- Mediante varias técnicas se validó el correcto funcionamiento de la herramienta.

RECOMENDACIONES

Teniendo como base los resultados de este trabajo y la experiencia adquirida durante el desarrollo del mismo, se recomienda:

- Continuar con la investigación para garantizar nuevas mejoras en futuras versiones del sistema.
- Agregar funcionalidades, como graficar toda la información que se gestiona.
- Reprogramarse sobre Sauxe, integrarse a SeReq y al mismo marco de trabajo para brindar un valor agregado a la solución.
- Integrar la herramienta a un repositorio de activos de la UCI en el campo de los requisitos.
- Hacer un uso efectivo del diseño del sistema mediante el uso de patrones para posteriores versiones logrando de esta forma un diseño de alto nivel.

BIBLIOGRAFÍA REFERENCIADA

(IEEE, 1998): IEEE. 1998. IEEE Guide to Software Requirements Specifications. 1998.

(Larman, 1999): Larman, Craig. *Applying Uml and Patterns.* México: s.n., 1999. ISBN 9780130255594

(Sametinger, 1997): Sametinger, Johannes. *Software Engineering with Reusable Components.* Berlin : s.n., 1997. ISBN 3540626956.

(Sodhi & Sodhi, 1999): Jag Sodhi, Prince Sodhi. *Domain analysis and design processes.* McGraw-Hill (New York): s.n., 1999. ISBN 0070579237

(Sommerville, 1995): Sommerville, Ian. Ingeniería de Software: s.n., 1995 5ª Edición.

(Reifer, 1994): Reifer, Donald. 1994. Encyclopedia of Software Engineering. [ed.] J.J Marciniack. s.l.: Wiley, 1994. pp. 1043-1054.

(C. Alexander, 1979): Alexander, Christopher. *"The Timeless Way of Building",* s.n., 1979

(Pressman, 2005): Pressman, Roger S. 2005. Ingeniería del Software un enfoque practico. La Habana: Félix Varela, 2005. pp. 171-187.

(Thayer, et al., 1997): Thayer, R.H. and Dorfman, M. 1997. Software Requirements Engineering. 2da. s.l.: IEEE Compute Society Press, 1997.

BIBLIOGRAFÍA CONSULTADA

1. **Dr.C José Carlos del Toro Ríos, Ing Henry Raúl Gonzalez Brito.** *Documento Visión del ERP 1.0.* 2008.

2. **Francisco José García Peñalvo, José Manuel Marqués Corral, Jesús Manuel Maudes Raedo.** *Análisis y Diseño Orientado al Objeto para Reutilización.* Burgos. España : s.n. 2007

3. **MSDN.** Microsoft. [En línea] 2011. [Citado el: 6 de Junio de 2011.] http://msdn.microsoft.com/en-us/library/aa659223%28AX.10%29.aspx.

4. **Ramírez, Jose Ramón Salazar.** Lenguajes y Ciencias de la Computación. [En línea] 2009. [Citado el: 22 de noviembre de 2010.] http://caosd.lcc.uma.es/spl/hydra/documents/PFC_JRSalazar.pdf

5. **Torío, Julio Mellado.** Estrategias de prueba de líneas de producto de sistema de tiempo real especificados con diagramas de estados jerárquicos. *Tesis Doctoral .* UNIVERSIDAD POLITÉCNICA DE MADRID : s.n., 2004.

6. **Claudia P. Ayala, Pere Botella, Xavier Franch.** *Construcción de una Taxonomía de componentes COTS orientados a la Gestión de Requisitos.* Barcelona. España : s.n. 2008

7. **EPS, Informática.** Reutilización. *Conferencia.*

8. **Ruiz, Alvaro Ernesto Carmona.** De los patrones de análisis y de integración a los componentes de negocio. *Conferecia.* Bogotá, Colombia : s.n., 2005.

9. **Mühlrad, Daniel.** *Patrones de diseño.* 2008.

10. **Ros, Joaquín Nicolás.** Una propuesta de gestión integrada de modelos y requisitos en líneas de productos de software. *Tesis Doctoral .* UNIVERSIDAD DE MURCIA, Facultada de Informática : s.n., 2009.

11. **sourceforge.** Captain Feature. [En línea] 2011. [Citado el: 17 de Junio de 2011.] http://sourceforge.net/projects/captainfeature/.

12. **Science, Facultad of Computer.** WorkGroup. *FeatureIDE.* [En línea] [Citado el: 17 de Junio de 2011.] http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/.

13. **Laboratory, Software Engineering.** Software Reuse Team. *Odyssey-Share.* [En línea] [Citado el: 17 de Junio de 2011.] http://reuse.cos.ufrj.br/site/en/index.php?option=com_content&task=view&id=21&Itemid=24.

14. **pure.systems.** pure.systems. *pure::Variants.* [En línea] [Citado el: 17 de Junio de 2011.] http://www.pure-systems.com/pure_variants.49.0.html.

15. **RequiLine.** The RequiLine project is complete. [En línea] 2005. [Citado el: 17 de Junio de 2011.] http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/.

16. **XFeature.** XFeature -- Feature Modelling Tool . [En línea] 2004. [Citado el: 17 de Junio de 2011.] http://www.pnp-software.com/XFeature/

17. Liana Barachisio Lisboa, Vinicius Cardoso García , Daniel Lucrédio, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira , Renata Pontín de Mattos Fortes. *A systematic review of domain analysis tools.* Brasil : s.n., 2010

18. Ramírez, Jose Ramón Salazar. Lenguajes y Ciencias de la Computación. [En línea] 2009. [Citado el: 22 de noviembre de 2010.] http://caosd.lcc.uma.es/spl/hydra/documents/PFC_JRSalazar.pdf

19. Solcre. Ventajas de las Aplicaciones Web . [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf

20. Producción, Equipo de. Modelo de Desarrollo orientado a componentes del proyecto ERP-CUBA.

21. Fundamentos de la Ingeniería de Software. Herramientas Case. *Conferencia.*

22. Slideshare. *Visual Paradigm For Uml.* [En línea] [Citado el: 10 de 11 de 2010.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

23. Gonzalo Génova, José M. Fuentes, Juan Llorens. *Evaluación de herramientas CASE para UML.* Madrid. España : s.n. 2005

24. Marzo, Josep Villalta. Criterios de selección de una herramienta CASE - UML. [En línea] 2004. [Citado el: 10 de 11 de 2010.] http://www.vico.org/aRecursosPrivats/UML_TRAD/talleres/mapas/UMLTRAD_101A/LinkedDocuments/SeleccionCASE_vvc.pdf.

25. James Rumbaugh, Ivar Jacobson, Grady Booch. *El lenguaje de modelado. Manual de referencia.*

26. programación, Lenguajes de. Lenguajes de programación. [En línea] 2009. [Citado el: 23 de Noviembre de 2010.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

27. TORRES, Ariel. Qué podemos hacer con PHP?, 2007. Disponible en: http://www.articuloweb.com/category.php?cat_id=3 (10/11/2010).

28. Torre, Aníbal de la. adelat. *Lenguajes del lado servidor o cliente.* [En línea] 2006. [Citado el: 23 de Noviembre de 2010.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

29. Pérez, Javier Eguíluz. *Introducción a JavaScript.*

30. W3C. W3C. *Guía Breve de XHTML.* [En línea] 2008. [Citado el: 16 de Junio de 2011.] http://www.w3c.es/divulgacion/guiasbreves/XHTML.

31. El servidor de web Apache: Introducción práctica. Disponible en: http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/index.html (10/11/2010).

32. QuerSystem Informática. Tecnología, 2008. [Disponible en: http://www.quersystem.com/index.php?option=com_content&view=article&id=20&Itemid=3]

33. Pecos, Daniel. PostGreSQL vs. MySQL. [En línea] [Citado el: 15 de 11 de 2010.] http://danielpecos.com/docs/mysql_postgres/index.html.

34. **¿Qué es Postgres?** .p http://www.postgres-ql.com.ar/html/informacion.php?opcion=queespostgres (10/11/2010).

35. **H8Red.** H8Red . *Ventajas de usar Mozilla Firefox.* [En línea] 1 de 2 de 2006. [Citado el: 10 de 11 de 2010.] http://h8red.cl/2006/ventajas-de-usar-mozilla-firefox/.

36. **Teen, Mafer en Movida.** teens. *Cuáles son las ventajas de Mozilla Firefox frente a Internet Explorer.* [En línea] 24 de 7 de 2008. [Citado el: 10 de 11 de 2010.] http://www.teens.com.pe/2008/07/24/cuales-son-las-ventajas-de-mozilla-firefox-frente-a-internet-explorer/.

37. **Netbeans**. [En línea] [Citado el: 17 de Junio de 2011.] http://netbeans.org.

38. **¿Qué son los FrameWorks?** *The SOA agenda.* [En línea] [Citado el: 16 de Junio de 2011.] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.

39. **Symfony**. *Open-Source PHP Web marco de trabajo .* [En línea] [Citado el: 15 de 11 de 2010.] http://www.symfony-project.org/.

40. **EXTJS**. [En línea] http://www.extjs.com/

41. **David Tavárez**. Comparación de marco de trabajo s en Javascript. [En línea] http://www.maestrosdelweb.com/editorial/comparacion- marco de trabajo s-javascript/

42. **Guardado, Iván.** Utilizando Doctrine como ORM en PHP. *Web.Ontuts.* [En línea] 6 de Julio de 2010. [Citado el: 15 de Junio de 2011.] http://web.ontuts.com/tutoriales/utilizando-doctrine-como-orm-en-php/.

43. **PHP Object Persistence Libraries and More**. *Doctrine.* [En línea] [Citado el: 14 de Junio de 2011.] http://www.doctrine-project.org/.

44. **González, Mario.** Arquitectura de software. *sophia.* [En línea] 2008. [Citado el: 17 de Junio de 2011.] http://sophia.javeriana.edu.co/~cbustaca/Arquitectura%20Software/Clases/Ensayos_2008/Presentacion_P10

45. **Server, Microsoft Windows.** Introducción a Redes. *Arquitectura Cliente/Servidor.* [En línea] 2009. [Citado el: 16 de Junio de 2011.] http://www.juansa.net/Admin2003/cliser.htm

46. **slideshare**. slideshare. Arquitectura en tres capas. [En línea] 2009. [Citado el: 17 de Junio de 2011.] http://www.slideshare.net/Decimo/arquitectura-3-capas.

47. **Martín, S. López**, E. Boletín Nº8 Rama de Estudiantes de IEEE-UNED, 2007. [Disponible en: http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin/boletin_8_Octubre_2007.pdf].

48. **Gómez Baryolo, Oiner and Mena, Grette Leydi**. 2008. Arquitectura de seguridad del software ERP Cuba. 2008. Especificaciones de la Arquitectura del proyecto ERP Cuba.

49. **Sergio Martín, Eugenio López, José Antonio Cámara, Germán Carro, Guillermo F. Lafuente, Maria M. García, Gloria Murillo,Serafín Doblado.** DIEEC - Departamento de Ingenieria Eléctrica, Electrónica y de Control (ieee). DIEEC - Departamento de Ingenieria Eléctrica, Electrónica y de Control (ieee). [En línea] 2007. [Citado el: 15 de Abril de 2011.] http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin/boletin_8_Octubre_2007.pdf.

50. **Brites, Walter Fernando.** eumed. *Los modelos de implementación.* [En línea] 2010. [Citado el: 24 de Enero de 2011.] http://www.eumed.net/libros/2010e/811/modelos%20de%20implementacion.htm.

51. **dsi.** *Diagramas de Componentes y Despliegue.* [En línea] [Citado el: 4 de Febrero de 2011.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

52. **chubut.** *Estándares de codificación de sistemas.* [En línea] 6 de Octubre de 2006. [Citado el: 1 de Junio de 2011.] http://www.chubut.gov.ar/informatica/docs/EstandaresCodificacion.pdf.

53. **DesarrolloWeb.com** . *Métricas de Software.* [En línea] [Citado el: 16 de Mayo de 2011.] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html

54. **Pressman, Roger S.** *Ingeniería del Software - Un Enfoque Práctico*. s.l.: McGraw-Hill Companies, 2002. 8448132149.

ANEXO

Anexo 1: Carta de aceptación del cliente.

UCI Universidad de las Ciencias Informáticas | Centro de informatización de la gestión de entidades

Aval

Mediante el presente la Ing. Tahirí Rivero Alvarez, analista principal del ERP-Cuba certifica que la herramienta "ReuReq" desarrollada por la tesista Aibett Cabrera Vega cumple con las funcionalidades previstas para la primera versión de la misma.

Sin otro asunto.

Ing. Tahirí Rivero Alvarez
Analista principal del ERP.
UCI

Ilustración 16 Carta de aceptación del cliente.

GLOSARIO DE TÉRMINOS

Bits: Dígito binario.

Componentes: adj. y com. Que forma parte de alguna cosa o de su composición m. Pieza o elemento de un aparato o electrodoméstico.

Dominio: Una colección de aplicaciones (software) actuales y futuras que comparten un conjunto de características comunes.

Métrica: Del latín *metrĭcus*, la métrica hace referencia a aquello perteneciente o relativa al metro. Este puede ser, a su vez, una unidad de longitud del Sistema Internacional o la medida del verso.

Parametrizados: Es la organización de elementos (grupos y bloques) que permiten la modificación.

Parámetros: Hace referencia a los atributos o información que se puede añadir a los requisitos en el sistema.

Patrón: Se denomina patrón, a aquellos temas o problemáticas que se reiteran en el tiempo. Un patrón de diseño, por ejemplo, abstrae los elementos comunes y recurrentes que ocurren en los ámbitos de diseño, presentando un modelo que ofrece soluciones.

Plugin: Un plugin es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Relaciones: Relación es una correspondencia o conexión entre algo o alguien con otra cosa u otra persona. De esta forma, la noción de relación se utiliza en diversas ciencias para explicar todo tipo de fenómenos.

Requisito: Circunstancia o condición necesaria para algo.

Restricciones: El término restricciones o restricción puede utilizarse en diferentes ámbitos, sin embargo en la mayoría de estos implicará lo mismo: una limitación o una reducción ya sea natural o impuesta, según corresponda.

Reutilización: Reutilizar es la acción de volver a utilizar los bienes o productos. La utilidad puede venir para el usuario mediante una acción de mejora o restauración, o sin modificar el producto si es útil para un nuevo usuario.

Scripts: Un script es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Son ejecutados por un intérprete de línea de comandos. Usualmente son archivos de texto.

Transacciones: Flujo básico que se realizar al describir textualmente un requisito.

GLOSARIO DE SIGLAS

AJAX: JavaScript Asíncrono y XML (Asynchronous JavaScript and XML por sus siglas en inglés).

API: Interfaz de Programación (Application Programming Interface por sus siglas en inglés).

CASE: Herramientas de Ingeniería de Software Asistida por Computadoras (Computer Aided Software Engineering por sus siglas en inglés).

CEIGE: Centro de Informatización de la Gestión de Entidades.

COTS: Productos de software (Commercial off the Shelf por sus siglas en inglés).

ERP: (Enterprise Resource Planning por sus siglas en inglés) o de Planificación de Recursos Empresariales.

HTML: Lenguaje de Marcado de Hipertexto (HyperText Markup Language por sus siglas en inglés).

HTTP: Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol por sus siglas en inglés).

HTTPS: Protocolo de Transferencia de Hipertexto Seguro (Hypertext Transfer Protocol Secure por sus siglas en inglés).

IDE: Entorno de Desarrollo Integrado (Integrated Development Environment por sus siglas en inglés).

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers IEEE por sus siglas en inglés).

IP: Protocolo de Internet (Internet Protocol por sus siglas en inglés).

IR: Ingeniería de Requisitos.

JAD: Desarrollo conjunto de aplicaciones (Joint Application Development por sus siglas en inglés).

JSON: JavaScript Object Notation por sus siglas en inglés.

MAC: Control de Acceso al Medio (Media Access Control por sus siglas en inglés).

MVC: Modelo-Vista-Controlador (Model-View-Controller por sus siglas en inglés).

ORM: Object Relational Mapper por sus siglas en inglés.

PHP: Pre Procesador de Hipertexto (Hypertext Pre-processor por sus siglas en inglés).

Post: Power-on self-test por sus siglas en inglés.

RAE: Real Academia Española.

RAM Memoria de Acceso Aleatorio: (Random-access memory por sus siglas en inglés).

SQL: Lenguaje de consulta estructurado (Structured Query Language por sus siglas en inglés).

SSL: Protocolo de capa de conexión segura (Secure Sockets Layer por sus siglas en inglés).

UML: Lenguaje Unificado de Modelado (Unified Modeling Language por sus siglas en inglés).

XHTML: Lenguaje Extensible de Marcado de Hipertexto (eXtensible Hypertext Markup Language por sus siglas en inglés).

XML: Lenguaje Extensible de Marcado (Extensible Markup Language por sus siglas en inglés).