

Universidad Central "Marta Abreu" de Las Villas
Facultad de Matemática, Física y Computación
Carrera Ingeniería Informática



TRABAJO DE DIPLOMA

Título:

Aplicación de métodos de análisis multicriterio en la evaluación de software

Autor:

Hector Rivera Herrada

Tutor:

Msc. Yoan Pacheco Cárdenas

Santa Clara, 2016

Declaración Jurada



El que suscribe, **Hector Rivera Herrada**, hace constar que el presente trabajo de diploma fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería Informática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicado sin autorización de la Universidad.

Hector Rivera Herrada

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del Tutor

Firma del Jefe del Laboratorio

Dedicatoria

Dedico esta tesis a mis amigos quienes fueron un gran apoyo emocional durante el tiempo en que escribía esta tesis.

A mi madre quien me apoyó todo el tiempo.

A María quien me apoyo y alentó para continuar, cuando parecía que me iba a rendir.

A todos los que me apoyaron para escribir y concluir esta tesis.

Para ellos es esta dedicatoria de tesis, pues es a ellos a quienes se las debo por su apoyo incondicional.

Agradecimientos

A mi familia fuente de apoyo constante e incondicional en toda mi vida y más aún en mis duros años de carrera profesional y en especial quiero expresar mi más grande agradecimiento a mi madre que sin su ayuda hubiera sido imposible para mí culminar mi profesión.

Pensamiento

¡Digamos que nuestra Revolución aprecia, más que a nadie, al héroe anónimo! ¡Al hombre humilde, al combatiente modesto que cumple con su deber por un problema de conciencia, sin importarle jamás ni siquiera si le reconocen sus méritos! ¡Ese es el modelo del comunista!

Discurso pronunciado por el compañero Fidel Castro Ruz
en la clausura del Primer Congreso del Partido Comunista de Cuba

Resumen

La siguiente investigación se desarrolla en la Universidad Central “Marta Abreu de las Villas” (UCLV) de Santa Clara, Villa Clara, con el objetivo de crear un sistema informático que brinde ayuda y soporte en el proceso de evaluación de software, integrando técnicas de análisis multicriterio, conduciendo a problemas estructurados que permiten obtener decisiones más informadas y mejores.

Las técnicas encontradas en la bibliografía se han integrado en una aplicación web que expone las anteriores como servicios web, permitiendo a los usuarios no solo interactuar con la aplicación a través de la interfaz creada para el consumo de los mismos sino también permitiendo a programadores y desarrolladores crear sus propias interfaces para la interacción con el mismo.

En el presente documento de investigación quedan descritos, en forma de capítulos, el proceso de análisis, diseño e implementación del sistema propuesto, utilizando el Lenguaje Unificado de Modelado (UML). En la implementación del sistema se utilizó MongoDB como sistema gestor de Bases de Datos y JavaScript como lenguaje de programación.

Abstract

The following research is conducted at the Central University "Marta Abreu de las Villas" (UCLV) Santa Clara, Villa Clara, with the aim of creating a computer system that provides help and support in the process of evaluating software, integrating techniques multi-criteria analysis, leading to structured problems that allow more informed and better decisions.

The techniques found in the literature have been integrated into a Web application that exposes the above as Web services, allowing users to not only interact with the application through the interface created for their consumption but also allowing programmers and developers create their own interfaces for interaction with it.

In this research paper are described in chapters, the process of analysis, design and implementation of the proposed system, using the Unified Modeling Language (UML). MongoDB was used in the implementation of the system as system manager Databases and JavaScript programming language.

Contenido

INTRODUCCIÓN.....	1
CAPÍTULO 1. ESTUDIO DEL MARCO TEÓRICO EN LOS SISTEMAS DE EVALUACIÓN DE SOFTWARE	4
1.1 OBJETIVOS ESTRATÉGICOS DEL SOFTWARE EN LAS ORGANIZACIONES E IMPORTANCIA DEL PROCESO DE EVALUACIÓN DE SOFTWARE.....	4
1.2 OBJETO DE ESTUDIO.....	4
1.2.1 PROCESO DE EVALUACIÓN DE SOFTWARE	4
1.2.2 PROCESOS OBJETO DE AUTOMATIZACIÓN.....	7
1.3 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN.....	7
1.4 FUNDAMENTACIÓN DE LOS OBJETIVOS.....	8
1.5 TENDENCIAS Y TECNOLOGÍAS ACTUALES	8
1.5.1 MCDA.....	¡ERROR! MARCADOR NO DEFINIDO.
1.6 SERVICIO WEB REST.....	9
1.7 NODEJS.....	10
1.8 EXPRESS	11
1.9 ANGULARJS.....	11
1.9.1 LA FILOSOFÍA DE ANGULARJS.....	11
1.9.2 LOS OBJETIVOS DE DISEÑO	12
1.10 MONGODB	12
1.11 HTML5.....	13
1.12 TWITTER BOOTSTRAP	13

<i>CAPÍTULO 2. MODELO DEL NEGOCIO Y REQUISITOS.....</i>	<i>14</i>
<i>2.1 MODELO DEL NEGOCIO ACTUAL.....</i>	<i>14</i>
<i>2.2 ACTORES DEL NEGOCIO.....</i>	<i>14</i>
<i>2.3 DIAGRAMA DE CASOS DE USO DEL NEGOCIO.</i>	<i>15</i>
<i>2.4 DESCRIPCIÓN DE LOS CASOS DE USO DEL NEGOCIO.....</i>	<i>16</i>
<i>2.5 ACTORES DEL SISTEMA A AUTOMATIZAR.....</i>	<i>17</i>
<i>2.6 DEFINICIÓN DE LOS REQUISITOS FUNCIONALES.</i>	<i>17</i>
<i>2.7 DEFINICIÓN DE LOS REQUISITOS NO FUNCIONALES.....</i>	<i>18</i>
<i>2.8 PAQUETES Y SUS RELACIONES.....</i>	<i>19</i>
<i>2.9 DIAGRAMA DE CASOS DE USO DEL SISTEMA.</i>	<i>20</i>
<i>2.10 DESCRIPCIÓN DE LOS CASOS DE USO DEL SISTEMA.....</i>	<i>21</i>
<i>2.11 CONSIDERACIONES FINALES.</i>	<i>22</i>
<i>CAPÍTULO 3. DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN</i>	<i>23</i>
<i>3.1 ARQUITECTURA DEL SISTEMA.....</i>	<i>23</i>
<i>3.1.1 MODELO CLIENTE-SERVIDOR.....</i>	<i>23</i>
<i>3.1.1.2 SERVIDOR WEB:</i>	<i>24</i>
<i>3.1.1.4 SERVIDOR MULTIMEDIA:..... ¡ERROR! MARCADOR NO DEFINIDO.</i>	
<i>3.2 DIAGRAMA DE SECUENCIA.</i>	<i>24</i>
<i>3.3 TRATAMIENTO DE ERRORES.</i>	<i>26</i>
<i>3.4 MODELO DE COMPONENTES.....</i>	<i>27</i>
<i>3.5 DIAGRAMA DE DESPLIEGUE.....</i>	<i>28</i>
<i>3.6 CONCLUSIONES PARCIALES.....</i>	<i>28</i>
<i>CAPÍTULO 4. PRUEBAS DEL SISTEMA Y ANÁLISIS DE FACTIBILIDAD.</i>	<i>29</i>
<i>4.1 ESTIMACIÓN BASADA EN EL MÉTODO DE LOS PUNTOS DE CASOS DE USOS. 29</i>	
<i>4.1.1 CÁLCULO DE PUNTOS DE CASOS DE USO SIN AJUSTAR</i>	<i>29</i>
<i>4.1.2 CÁLCULO DE PUNTOS DE CASOS DE USO AJUSTADOS.....</i>	<i>31</i>
<i>4.1.3 ESFUERZO HORAS-HOMBRE (E).</i>	<i>34</i>
<i>4.1.4 ESTIMACIÓN DEL ESFUERZO DEL PROYECTO.....</i>	<i>34</i>

<i>4.1.5 CÁLCULO DEL ESFUERZO TOTAL</i>	<i>35</i>
<i>4.1.6 CÁLCULO DEL TIEMPO DE DESARROLLO.</i>	<i>36</i>
<i>4.1.7 CÁLCULO DEL COSTO</i>	<i>36</i>
<i>4.2 CASOS DE PRUEBAS.</i>	<i>36</i>
<i>4.2.1 PRUEBAS DE CAJA NEGRA.</i>	<i>37</i>
<i>4.2.2 PRUEBAS DE CAJA NEGRA EN CASOS DE USO DEL SISTEMA.</i>	<i>37</i>
<i>4.3 CONCLUSIONES PARCIALES.....</i>	<i>41</i>
<i>CONCLUSIONES GENERALES.</i>	<i>42</i>

Lista de figuras

<i>Figura 1 Diagrama de Casos de Uso del Negocio</i>	<i>16</i>
<i>Figura 2 Diagrama de paquetes</i>	<i>20</i>
<i>Figura 3 Diagrama de Casos de Usos del Sistema.....</i>	<i>21</i>
<i>Figura 4 Arquitectura Cliente - Servidor</i>	<i>23</i>
<i>Figura 5 Diagrama de Secuencia del Caso de Uso Insertar Proyecto.....</i>	<i>25</i>
<i>Figura 6 Diagrama de secuencia del Caso de Uso Gestionar Objetivo General</i>	<i>26</i>
<i>Figura 7 Mensaje de Error 1.....</i>	<i>26</i>
<i>Figura 8 Mensaje de Error 2.....</i>	<i>26</i>
<i>Figura 9 Modelo de Componentes.....</i>	<i>27</i>
<i>Figura 10 Diagrama de Despliegue</i>	<i>28</i>

Lista de tablas

<i>Tabla 1 Descripción de los Actores del Negocio</i>	<i>15</i>
<i>Tabla 2 Descripción de los Casos de Usos del Negocio.....</i>	<i>16</i>
<i>Tabla 3 Actor del Sistema a Automatizar.....</i>	<i>17</i>
<i>Tabla 4 Requisitos funcionales.....</i>	<i>18</i>
<i>Tabla 5 Descripción de los Casos de Uso del Sistema</i>	<i>21</i>
<i>Tabla 6 Factor de Peso según los actores</i>	<i>30</i>
<i>Tabla 7 Factor de Peso de los Caso de Usos.....</i>	<i>30</i>
<i>Tabla 8 Factores de Complejidad Técnica</i>	<i>32</i>
<i>Tabla 9 Escala para la evaluación de los Factores de Complejidad Técnica</i>	<i>32</i>
<i>Tabla 10 Factor de Ambiente.....</i>	<i>33</i>
<i>Tabla 11 Distribución del Esfuerzo.....</i>	<i>35</i>
<i>Tabla 12 Distribución del Esfuerzo en el Proyecto.....</i>	<i>35</i>
<i>Tabla 13 Secciones a probar en el caso de uso "Gestionar Proyectos".....</i>	<i>38</i>
<i>Tabla 14 Escenarios a probar en el caso de uso "Gestionar Proyectos"</i>	<i>39</i>
<i>Tabla 15 Secciones a probar en el caso de uso "Gestionar Árbol de Valor"</i>	<i>40</i>
<i>Tabla 16 Escenarios a probar en el caso de uso "Gestionar Árbol de Valor".....</i>	<i>41</i>

Introducción

La evaluación de calidad en los productos o servicios de cualquier índole se ha hecho algo cotidiano y cada vez más importante puesto que se convirtió en un factor diferenciador cuando se está optando por adquirir un servicio o un bien. El concepto de calidad tiene diferentes definiciones, pero una ampliamente aceptada es la establecida por la ISO-9000 la cual define la calidad como el “grado en el que un conjunto de características inherentes cumple con los requisitos”.

Las organizaciones empresariales muchas veces se encuentran en la disyuntiva de desarrollar soluciones informáticas propias para gestionar sus procesos de negocio o utilizar soluciones desarrolladas por terceros para ello. El desarrollo de software es un proceso complejo y riesgoso, menos del 32% de los proyectos informáticos que comienzan terminan siendo exitosos, por lo que en ocasiones es deseable reutilizar soluciones ya existentes. Por otra parte, si la empresa no cuenta con suficientes recursos para comprar los programas que necesita entonces no tendrá otra alternativa que utilizar soluciones basadas en software libre y de código abierto (Pacheco, 2016).

En cualquier caso, para decidir si un programa desarrollado por un tercero cumple con los requisitos que los procesos de negocio de la empresa necesitan se debe realizar un proceso de evaluación de la calidad y determinar así, el grado de satisfacción a los requerimientos y necesidades del usuario.

El análisis multicriterio (del inglés, Multi-Criteria Decision Analysis) provee los instrumentos necesarios para apoyar la toma de decisiones durante la realización de este proceso (Keller, 2009).

Los métodos multicriterio se fundamentan a partir del proceso decisorio que requiere de la aplicación de métodos de comparación para apoyar al tomador decisiones de tal manera que sus decisiones sean plenamente consistente con algún marco de racionalidad adoptado (Pacheco, 2009).

Un proceso de decisión implica, necesariamente, la comparación entre las alternativas sobre las que se puede optar frente a cierta disyuntiva presente, en primer lugar se hace

necesario separar un problema de decisión en los elementos que lo componen, para la posterior comparación entre ellos, de esta manera la toma de decisión implica el hecho de comparar elementos que se traduce en la necesidad de realizar mediciones que permitan aplicar los criterios de comparación para establecer preferencias entre ellos, es decir, una jerarquía (Pacheco, 2009).

Problema de Investigación

Este trabajo investiga cómo integrar técnicas de análisis multicriterio en el proceso de evaluación de software.

Objetivo General

Aplicar técnicas de análisis multicriterio para estructurar el problema (objetivo, criterios, alternativas) conduciendo a decisiones más informadas y mejores en el proceso de evaluación de software.

Objetivos Específicos

- Identificar los métodos MCDA existentes en la literatura científica, que pueden ser utilizados para reducir el conjunto inicial de medidas de software.
- Integrar los métodos seleccionados a una aplicación informática que exponga los mismos como servicios Web.
- Validar la aplicación propuesta a través de diferentes pruebas de cajas negras.

Preguntas de Investigación

¿Cómo utilizar métodos de análisis multicriterio en la evaluación de la calidad del software?

¿Qué patrones de diseño pueden ser útiles en el desarrollo de una aplicación cliente servidor?

Estructura del documento

El presente trabajo estará estructurado según este orden: introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía y anexos.

En el Capítulo 1 se describen los principales conceptos relacionados con la evaluación de software, el análisis multicriterio y las tecnologías que se usaran para la implementación de la aplicación.

En el capítulo 2, se describen el modelo del negocio, modelación los principales diagramas y los requisitos del sistema tanto funcionales como no funcionales del sistema a implementar.

En el capítulo 3 se realiza una descripción de las propuestas de solución del sistema a implementar, la arquitectura del mismo y los diagramas que contribuyen a dicha implementación.

En el capítulo 4 se lleva a cabo la estimación preliminar del desarrollo del sistema así como las pruebas pertinentes para comprobar el correcto funcionamiento del software.

Capítulo 1. ESTUDIO DEL MARCO TEÓRICO EN LOS SISTEMAS DE EVALUACIÓN DE SOFTWARE

En este capítulo se abordan de manera general los conceptos necesarios para comprender la problemática expuesta en este trabajo de diploma, se exponen las tecnologías a usar en la implementación del nuevo sistema y las características de las mismas.

1.1 Objetivos estratégicos del software en las organizaciones e importancia del proceso de evaluación de software

El desarrollo de las tecnologías del software, hardware, redes de comunicaciones y otros medios están permitiendo transmitir y recibir cada vez más un mayor volumen de información, lo que está posibilitando que las empresas tengan un mayor soporte a sus procesos de negocios, logrando una mayor eficiencia. El objetivo fundamental de estas tecnologías es permitir a la empresa lograr mayor captación del mercado, mejores márgenes de rentabilidad, menores costos operativos al interior de la organización, con nuevas estructuras organizacionales que sean flexibles, más planas y que se adapten al cambio. Es por esto que el software en la empresa se ha convertido en un producto; es decir, debe ser rentable para la organización. Si las empresas invierten en él debe ser considerado como un proyecto de desarrollo o de adquisición de software, lo cual es muy importante dado que va a apoyar los procesos del negocio y permitir crear valor para la empresa. El desarrollo de software requiere personal, compra de equipos, las bases de datos, la participación de los usuarios, una definición de las nuevas formas de ejecución de las actividades y, como cualquier otro producto que hoy día se desarrolla en una organización, debe tener calidad. Las empresas necesitan reducir al mínimo los costos de mantenimiento.

1.2 Objeto de estudio

1.2.1 Proceso de evaluación de software

El proceso de evaluación de software es aquel proceso que se realiza con el objetivo de determinar el grado en que una serie de características o métricas es inherente al software.

1.2.1.1 Normas ISO para evaluar software

Con la creación del estándar ISO/IEC 25000 SQuaRE (Software Product Quality Requirements and Evaluation) se organizó, se enriqueció y se unificó las series que cubren dos procesos principales: especificación de requisitos de calidad del software y evaluación de la calidad del software, soportada por el proceso de medición de calidad del software. Este estándar actualmente vigente está dividido de la siguiente forma:

1. 25000:2005: Guide to SQuaRE.
2. 25001:2007: Planning and management.
3. 25010:2011: Quality model and guide.
4. 25012:2008: Data Quality model.
5. 25020:2007: Measurement reference model and guide.
6. TR 25021:2007: Quality Measure elements.
7. 25030:2007: Quality Requirements.
8. FDIS 25045: Evaluation module for recoverability
9. 25051:2006: Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing.
10. DTR 25060: Common Industry Format (CIF) for Usability -- General Framework for Usability-related Information
11. 25062:2006: Common Industry Format (CIF) for usability test report

1.2.1.2 Análisis crítico del proceso de evaluación

Con la norma ISO 25000 se cuenta con un sistema para elaborar los requisitos de la calidad del software y el proceso de evaluación de la calidad de software, sin embargo, el proceso de evaluación de la calidad de software solo evalúa aquellas características que son inherentes a la calidad del software. En la actualidad las empresas necesitan evaluar además de las características relacionadas con la calidad del software, aquellas inherentes al dominio del problema así también como aquellas asignadas, hacer comparaciones entre

diferentes alternativas, etc., con el objetivo de seleccionar el mejor el software que se adecue a las necesidades que presentan sus procesos de negocios.

Numerosas empresas ofrecen este servicio en Internet, la forma más común de evaluar los software radica en: plantillas en diferentes formatos (*.xls, *.doc, *.pdf), las cuales los clientes deben descargar, rellenar con los datos necesarios y deben enviar a la empresa para ser evaluada. También existen empresas privadas que ofrecen diferentes servicios que van desde la selección de la herramienta hasta la implementación y puesta en funcionamiento de la misma, en muchos casos se desconoce cómo se realiza el proceso de selección y evaluación.

A continuación, vamos a presentar las deficiencias encontradas en el proceso de evaluación de software.

1. Existe un desconocimiento acerca de cómo estas empresas realizan este proceso, por lo que solo se puede “confiar” en los resultados emitidos y en las opiniones de personas y empresas que esta recomienda.
2. Los clientes son atraídos por la variedad de servicios que ofrecen estas empresas, los cuales solo una parte son gratis, los mejores son servicios de pago.
3. En algunos casos el trato es impersonal. El contacto se realiza de manera digital (vía e-mail, formularios, etc.) y la respuesta suele venir por la misma vía.
4. Por otro lado existen empresas que prefieren un trato más personal con el cliente. Se reúnen directamente cliente donde discuten los servicios que este va a adquirir. Los servicios que ofrecen estas empresas van desde la definición de los criterios y las alternativas, hasta la presentación de los resultados, la ayuda a la hora de seleccionar el software, la implementación del sistema en la empresa, además de ofrecer ayuda y soporte técnico con el software seleccionado.
5. Existen algunos sitios como The Software Sustainability Institute que ofrecen una guía basada en su experiencia con los criterios a evaluar, así como un ejemplo de cómo dar con la solución utilizando el algoritmo MAVT, aunque dejan todo el proceso en manos del cliente.

1.2.2 Procesos objeto de automatización

Después de encontrar las deficiencias señaladas anteriormente se decidió informatizar el proceso de evaluación de software. Donde un usuario podrá definir cada uno de los criterios a evaluar y las alternativas, escogerá diferentes algoritmos y podrá hacer una comparación de los resultados.

1.3 Sistemas automatizados existentes vinculados al campo de acción

Los sistemas informatizados vinculados al campo de acción son los siguientes:

- **EvaluandoERP:** Un sitio web que ofrece los servicios de evaluación de software. Donde deberá seguir una serie de pasos que permitan conocer los requerimientos y necesidad específicos de la compañía a evaluar. Posteriormente, una vez ingresados todos los datos solicitados, recibirá como resultado un resumen vía mail de los productos que posean una mayor adaptabilidad a los requerimientos ingresados.
- **Business Analysis a Guide to Better Software Evaluation:** ofrece una guía acerca de cómo realizar este proceso, la forma de organización, etc. Es una guía muy completa, con ejemplos que sirven muy bien para los que se quieran iniciar.
- **Business Matters:** ofrece una guía para evaluar y seleccionar software con un enfoque más económico. Describe 10 pasos que se deben seguir para obtener la mejor herramienta que satisfaga sus necesidades.
- **SoftSelect:** un sitio web que ofrece servicios como comparaciones de software, costo del software. Estos servicios son de pago, presentan en el sitio un código de ética acerca de su trabajo, pero no dan mucha información acerca de las herramientas que utilizan.
- **Milenium:** sitio web donde ofrecen servicios de ayuda a las empresas para seleccionar los mejores sistemas de información de acuerdo a los requerimientos específicos de la organización. No sólo los asisten en la selección y adquisición del software, sino que además se comprometen con sus clientes con el fin de ayudarlos en las tareas de implementación, conversión de datos y entrenamiento, entre otras actividades

propias de la implementación de sistemas de información. Cuentan con un staff profesional para ofrecerles las mejores soluciones.

1.4 Fundamentación de los objetivos

Las herramientas encontradas en el mercado no explican cómo realizan el proceso de evaluación de software. Además la mayoría no brinda sus servicios gratuitamente. No se encontró alguna herramienta de código abierto y muy pocas herramientas permiten comparar resultados entre las distintas alternativas. En algunos casos se hacen referencias a las matrices de decisiones y cómo usarlas, dejando que sea el cliente quien empíricamente se encargue de puntuar las alternativas y de calcular el resultado. Es por ello que nuestro trabajo tiene como objetivo general el desarrollo de una aplicación Web para aumentar la eficiencia del proceso de evaluación de software mediante la aplicación de técnicas de análisis multicriterio.

1.5 Tendencias y Tecnologías actuales

1.5.1 Análisis Multicriterio (MCDA)

Tiene como objetivo principal “ayudar a los decisores a organizar y sintetizar la información asociada al problema de decisión, de tal forma que se minimicen las percepciones negativas sobre la decisión tomada y se contemplen debidamente todos los criterios pertinentes para tomarla” (STEWART, 2001).

Para ello utiliza métodos de análisis Multicriterio para obtener las mejores alternativas según los criterios evaluados. Entre los posibles métodos se encuentran Multi-attribute Value Theory (MAVT), Analytic Hierarchy Process (AHP), Multi Attribute Utility Theory (MAUT), entre otros.

1.5.1.1 AHP.

Tiene como base las preferencias del decisor, lo que lo clasifica como un método multicriterio del tipo subjetivo. Es un proceso de tres etapas, en la primera se modela el problema como una jerarquía de los objetivos, criterios y alternativas. Posteriormente se realiza una comparación por pares de elementos pertenecientes a la misma categoría, esta comparación tiene como objetivo definir el nivel de importancia de los elementos para los

decisores, lo que se establece mediante una escala de comparación de preferencias, la escala que generalmente se utiliza es la escala de Saaty (Saaty, 1996). Como última etapa del método AHP, se analizan y sintetizan los resultados obtenidos.

1.5.1.2 MAVT

La intención de MAVT es construir un medio de asociar un número real con cada alternativa, con el fin de producir una orden de preferencia sobre las alternativas consistentes con los juicios de valor que toma la decisión. Para ello, MAVT asume que en cada problema de decisión U existe una función de valor real que representa las preferencias de quien toma las decisiones. Esta función U se utiliza para transformar los atributos de cada política alternativa en un solo valor. La alternativa con el mayor valor se señala como el mejor.

1.6 Servicio Web REST

REST por sus siglas en inglés (Representational State Transfer) es una técnica de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web. Esta técnica se usa para cualquier interfaz web simple que utilice XML y HTTP sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicio web SOAP (LEONARD RICHARDSON, 2007).

Con REST la web ha disfrutado de una escalabilidad con resultado de una serie de diseños claves como:

- Protocolo cliente/servidor sin estado, cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.
- Operaciones bien definidas aplicadas a todos los recursos de información, HTTP en sí define un conjunto pequeño de operaciones.
 - POST: Actualiza un recurso sobre el servidor. El recurso es contenido en el cuerpo del pedido de POST. Post es análogo a una declaración de actualización de SQL.

- GET: Recupera un recurso del servidor. El recurso es especificado con una URL para delinear el pedido de los parámetros de solicitud. GET es análogo a una declaración seleccionar de SQL.
 - PUT: Añade un recurso sobre el servidor. El recurso es contenido en el cuerpo del pedido de POST. El lanzamiento es análogo a una declaración de Insertar de SQL
 - DELETE: Elimina un recurso sobre el servidor. El recurso es especificado en el URL solamente. El Eliminar es análogo a una declaración de eliminar en SQL.
- Sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccional únicamente a través de su URI.
 - Uso de hipermedias, tanto para la información de la aplicación como para las transiciones de estado de la aplicación, la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

1.7 NodeJS

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web (2011). Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla (Handy, 2011).

Node.js es similar en su propósito a Twisted o Tornado de Python, Perl Object Environment de Perl, React de PHP, libevent o libev de C, EventMachine de Ruby, vibe.d de D y de Java existe Apache MINA, Netty, Akka, Vert.x, Grizzly o Xsocket. Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor.

Node.js implementa algunas especificaciones de CommonJS. Node.js incluye un entorno REPL para depuración interactiva.

1.8 Express

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles (Haviv, 2014).

1.9 AngularJS

AngularJS es un framework de JavaScript de código abierto (Green, 2013), mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página (Freeman, 2014). Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles (Lerner, 2013).

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript (Green, 2013). Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticas o dinámicas (Seshadri, 2014).

1.9.1 La filosofía de AngularJS

AngularJS está construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de software, mientras que la programación imperativa es excelente para expresar la lógica de negocio (Sandeep, 2014). Este framework adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un data-binding bidireccional que permite la sincronización automática de modelos y vistas (Seshadri, 2014). Como resultado, AngularJS pone menos énfasis en la manipulación del DOM y mejora la testeabilidad y el rendimiento (Lerner, 2013).

1.9.2 Los objetivos de diseño

- Disociar la manipulación del DOM de la lógica de la aplicación. Esto mejora la capacidad de prueba del código.
- Considerar a las pruebas de la aplicación como iguales en importancia a la escritura de la aplicación. La dificultad de las pruebas se ve reducida dramáticamente por la forma en que el código está estructurado.
- Disociar el lado del cliente de una aplicación del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados.
- Guiar a los desarrolladores a través de todo el camino de la construcción de una aplicación: desde el diseño de la interfaz de usuario, a través de la escritura de la lógica del negocio, hasta las pruebas.

Angular sigue el patrón MVC (Lerner, 2013) de ingeniería de software y alienta la articulación flexible entre la presentación, datos y componentes lógicos. Con el uso de la inyección de dependencias (Freeman, 2014), Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el backend se reduce, lo que conlleva a aplicaciones web mucho más ligeras (Seshadri, 2014).

1.10 MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto (Chodorow, 2013).

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida (Chodorow, 2013).

El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software 10gen. Ahora MongoDB es una base de datos lista para la producción de uso y con muchas características (features). Esta base de datos es altamente utilizada en las industrias y MTV

Network, Craigslist y Foursquare son algunas de las empresas que utilizan esta base de datos (Chodorow, 2013).

1.11 HTML5

Es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 es un término de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y las nuevas capacidades de JavaScript. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML (Franganillo, 2010). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

El desarrollo de este lenguaje de marcado es regulado por el Consorcio W3C.

1.12 Twitter Bootstrap

Twitter Bootstrap es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales (Cochran, 2012).

Capítulo 2. MODELO DEL NEGOCIO Y REQUISITOS

En el presente capítulo se abordarán temas relacionados con la implementación del software a desarrollar, se mencionarán los requisitos funcionales y no funcionales del mismo, se presentarán los principales diagramas del sistema y se presentarán los cálculos correspondientes a la estimación del proyecto.

2.1 Modelo del negocio actual

Las empresas actuales desean buscar la mejor solución a un problema planteado, no solo tomando como criterio de elección los beneficios que se obtendrán, sino considerando otros criterios como por ejemplo el volumen de ventas o el riesgo.

La evaluación de software supone la capacidad de identificar, medir y valorar, todos los costos y beneficios involucrados, cuestión que en la práctica no siempre es posible. Así, los beneficios y costos que se deben identificar, medir y valorar, son aquellos que resulten relevantes desde el punto de vista del inversionista que desea llevar a cabo el proyecto (Pacheco).

En los sistemas actuales encontrados en la bibliografía que se encargan de automatizar este proceso se encontraron diferentes problemas. El desconocimiento acerca de los métodos o algoritmos que utilizan las diferentes empresas para evaluar los software o herramientas hacen que el cliente se pregunte si realmente estás obteniendo la mejor respuesta.

2.2 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados (Wendy Boggs, 2002). En la **¡Error! No se encuentra el origen de la referencia.** se muestra una descripción del actor del negocio.

Actor del negocio	Descripción
Cliente	Persona encargada de crear proyectos, añadir objetivo general, criterios y alternativas al árbol de valor, evaluar el árbol obtenido utilizando los algoritmos implementados y hacer un análisis de los resultados mostrados

Tabla 1 Descripción de los Actores del Negocio

2.3 Diagrama de casos de uso del negocio.

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores (Pressman, 2010).

Para tener una visión general del proceso de negocio de la organización, se construyó el diagrama de casos de uso del negocio, en el que aparece el proceso de negocio como un caso de uso, relacionado con el actor del negocio. Este diagrama permite mostrar los límites y el entorno de la organización bajo estudio.

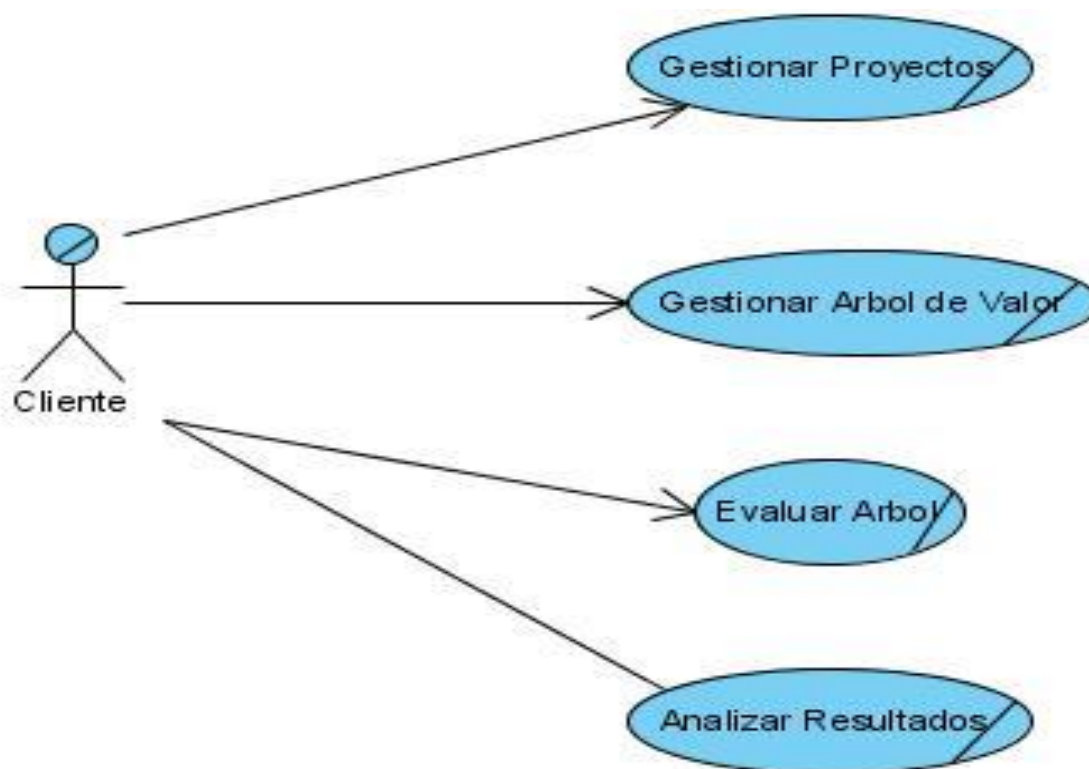


Figura 1 Diagrama de Casos de Uso del Negocio

2.4 Descripción de los Casos de Uso del Negocio

A continuación en la Tabla 2 se muestra una breve descripción de los casos de del negocio mostrado en la Figura 1.

Caso de uso del negocio	Descripción
Gestionar Proyectos	El cliente inicia el proceso cuando tiene un nuevo proyecto que evaluar.
Gestionar el Árbol de Valor	El cliente inicia el proceso después de crear un nuevo proyecto sobre el que construye el árbol de valor.
Evaluar Árbol de Valor	El cliente inicia el proceso cuando termina de definir el árbol de valor, selecciona los algoritmos y evalúa el árbol de valor.
Analizar los resultados	El cliente inicia el proceso después de haber evaluado el árbol de valor.

Tabla 2 Descripción de los Casos de Usos del Negocio

2.5 Actores del sistema a automatizar

Un actor es una agrupación uniforme de personas, sistemas o máquinas que interactúan con el sistema que estamos construyendo de la misma forma. Los actores son externos al sistema que vamos a desarrollar. Por lo tanto, al identificar actores estamos empezando a delimitar el sistema, y a definir su alcance (Ceria, 2002). Teniendo en cuenta lo anterior podemos plantear que el actor del sistema a automatizar sería el cliente como se muestra en la *Tabla 3*

Actor del Sistema	Descripción
Cliente	Persona o entidad encargada de chequear y gestionar toda la información referente a los Proyectos y su árbol de valor asociado. Interactúa con todos los Casos de Usos del Negocio.

Tabla 3 Actor del Sistema a Automatizar

2.6 Definición de los requisitos funcionales

Según (Sommerville, 2002) los requisitos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares y de cómo se comportará en situaciones particulares.

A continuación en la *Tabla 4* se muestran la definición de los mismos.

RF1	Gestionar Proyectos
Descripción	El sistema debe poder añadir, eliminar, actualizar y consultar los diferentes proyectos con que cuenta el cliente.
Precedencia	CUN_ Gestionar_Proyectos.
Seguimiento	CUS_Gestionar_Proyectos.
RF2	Gestionar el árbol de valor
Descripción	El sistema debe poder crear o actualizar el árbol de valor asociado a cada proyecto.

Precedencia	CUN_ Gestionar_el_árbol_de_valor.
Seguimiento	CUS_Gestionar_el_árbol_de_valor.
RF3	Evaluar árbol de valor
Descripción	El sistema debe poder evaluar cada árbol de valor usando los algoritmos implementados
Precedencia	CUN_ Evaluar_árbol_de_valor
Seguimiento	CUS_ Evaluar_árbol_de_valor
RF4	Analizar los resultados
Descripción	El sistema debe poder permitir analizar los resultados mostrados.
Precedencia	CUN_ Analizar_los_resultados
Seguimiento	CUS_ Analizar_los_resultados
RF5	Gestión de Usuarios
Descripción	El sistema debe poder agregar, eliminar, actualizar y consultar los Usuarios.
Precedencia	-
Seguimiento	CUS_Gestión_de_Usuarios
RF6	Autentificarse
Descripción	El sistema debe solicitar y validar las credenciales al Usuario para poder acceder a la gestión de la información del sistema.
Precedencia	-
Seguimiento	CUS_ Autentificarse

Tabla 4 Requisitos funcionales

2.7 Definición de los requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, como restricciones del entorno o de implementación, rendimiento, entre otros (Sommerville, 2002).

1. Usabilidad

RNF1	Exploración
Descripción	Se incluirán índices que faciliten la exploración y señalamientos que indiquen al usuario en dónde se encuentra en todo momento.

2. Rendimiento

RNF1	Tiempo de respuesta
Descripción	El tiempo de respuesta no debe exceder los cinco segundos ante las solicitudes del usuario.

3. Restricciones de diseño e implementación

RNF1	Sistema de Gestión de Bases de Datos
Descripción	Se utilizará como gestor de base de datos MongoDB
RNF2	Plataforma Web
Descripción	Se utilizará la plataforma NodeJS junto al framework Express para crear el servidor web
RNF3	Herramienta CASE
Descripción	Se utilizará la herramienta Visual Paradigma

4. Ayuda y documentación

RNF1	Documentación y Ayudas
Descripción	Se debe incluir manuales de uso del sistema.

2.8 Paquetes y sus Relaciones

Los paquetes son (Wendy Boggs, 2002) usados para agrupar clases con propósitos comunes.

Para un desarrollo organizado del sistema este se divide dos paquetes: el cliente y el servidor. El paquete Cliente está compuesto por varios los paquetes que forman parte del framework AngularJS (AngularJS, AngularJS Resource, AngularJS Routing) y los paquetes Proyectos y Árboles, que agrupan las funcionalidades de la aplicación. El paquete Servidor NodeJS está compuesto por el paquete Express (framework que permite crear los servicios web) y los paquetes que agrupan las API's de la aplicación, es decir, en cada paquete API se encuentran los diferentes servicios web para interactuar con el cliente. A continuación, se presenta la Figura 1 donde se representa gráficamente lo explicado anteriormente.

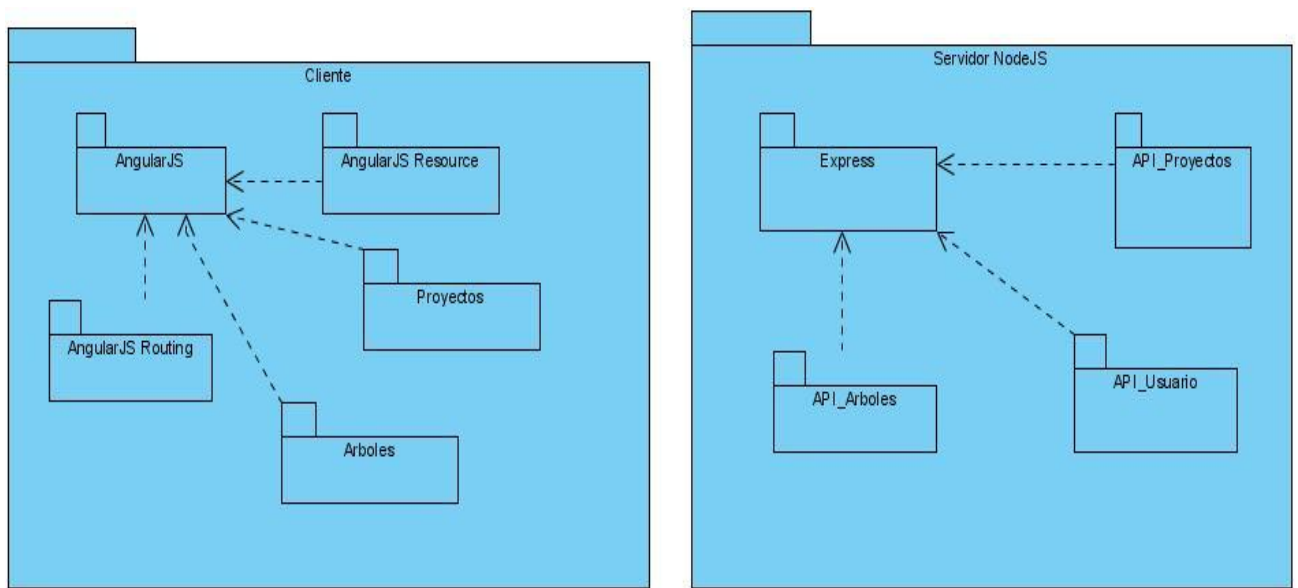


Figura 2 Diagrama de paquetes

2.9 Diagrama de Casos de Uso del Sistema

Los casos de uso del sistema muestran la relación existente entre los actores del sistema y las acciones que se realizan en el mismo, representadas mediante casos de uso (Larman, 1999).

Una vez conocidos los requisitos funcionales y no funcionales del sistema, así como los actores del mismo, se procede a representar los casos de uso del sistema asociados a los actores, los cuales son los encargados de interactuar con el sistema como se muestra en la Figura 3.

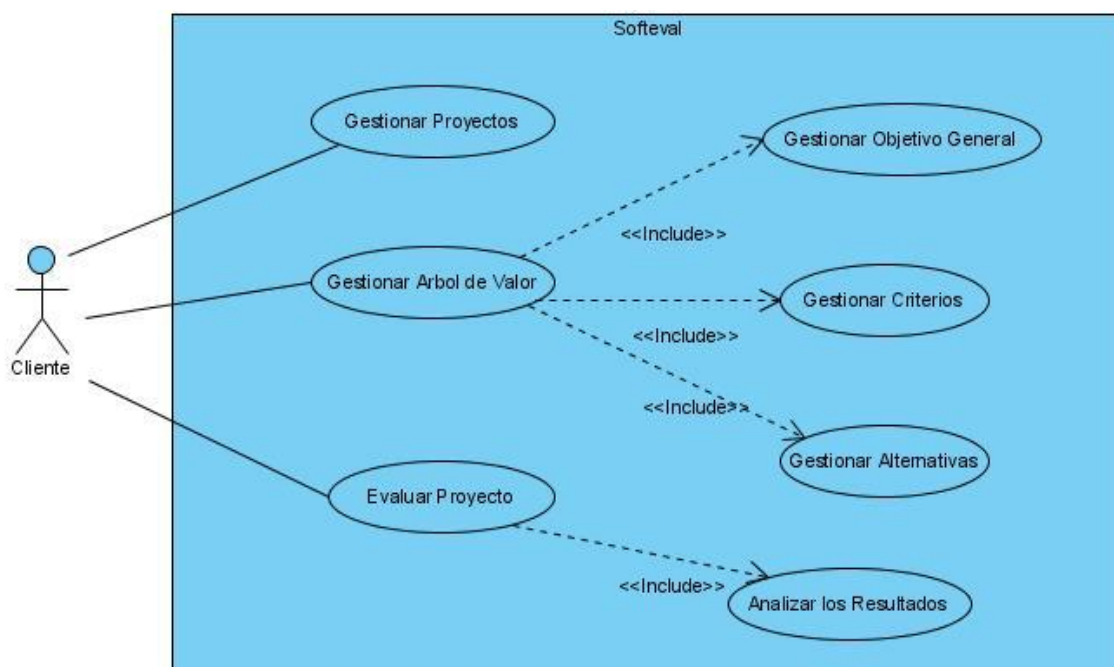


Figura 3 Diagrama de Casos de Usos del Sistema

2.10 Descripción de los Casos de Uso del Sistema

El actor Cliente, en la ventana principal, selecciona las operaciones correspondientes, descritas en la Tabla 5.

Caso de Uso del sistema	Descripción
CUS_ Autenticarse	Se autentica un usuario para acceder a la información del sistema.
CUS_ Gestionar_Proyectos	El cliente gestiona los datos de los proyectos.
CUS_ Gestionar_Árbol_de_Valor	El cliente gestiona los datos del árbol de valor.
CUS_ Gestionar_Objetoivo_General	El cliente gestiona los datos del objetivo general del árbol de valor.
CUS_ Gestionar_Criterios	El cliente gestiona los datos de los criterios del árbol de valor.
CUS_ Gestionar_Alternativas	El cliente gestiona los datos de las alternativas del árbol de valor.
CUS_ Evaluar_Proyecto	El cliente analiza los datos recogidos de la evaluación del proyecto.

Tabla 5 Descripción de los Casos de Uso del Sistema

2.11 Consideraciones finales

En este capítulo se realizó un estudio del modelo del negocio donde, quedan definidas las especificaciones de los casos de uso así como los requisitos funcionales y no funcionales. Con el modelo del negocio y la descripción de los procesos del negocio quedan plasmadas las bases para que se cree una propuesta computacional.

Capítulo 3. DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

En el presente capítulo se tratan los temas relacionados con la arquitectura del sistema a implementar. Se desarrollan los diferentes diagramas que permiten modelar, construir y documentar los elementos esenciales del sistema. Además, se presentan las posibles acciones que se pueden realizar en el sistema a partir de los casos de uso diseñados.

3.1 Arquitectura del Sistema

3.1.1 Modelo Cliente-Servidor

Una aplicación Web sigue un modelo cliente-servidor cuando posee un componente que hospede a la aplicación y a todos sus componentes (servidor Web) y un componente cliente, en nuestro caso un navegador Web. Una vez que tenemos estos dos componentes necesitamos un sistema de archivos en donde se van a almacenar todos los recursos para la generación de contenido. El sistema de archivos puede almacenar una gran cantidad de archivos multimedia (fotos, videos, audio, etc.), archivos XML y HTML, necesarios para el funcionamiento de la aplicación (Hernández, 1997) como se muestra en la Figura 4.

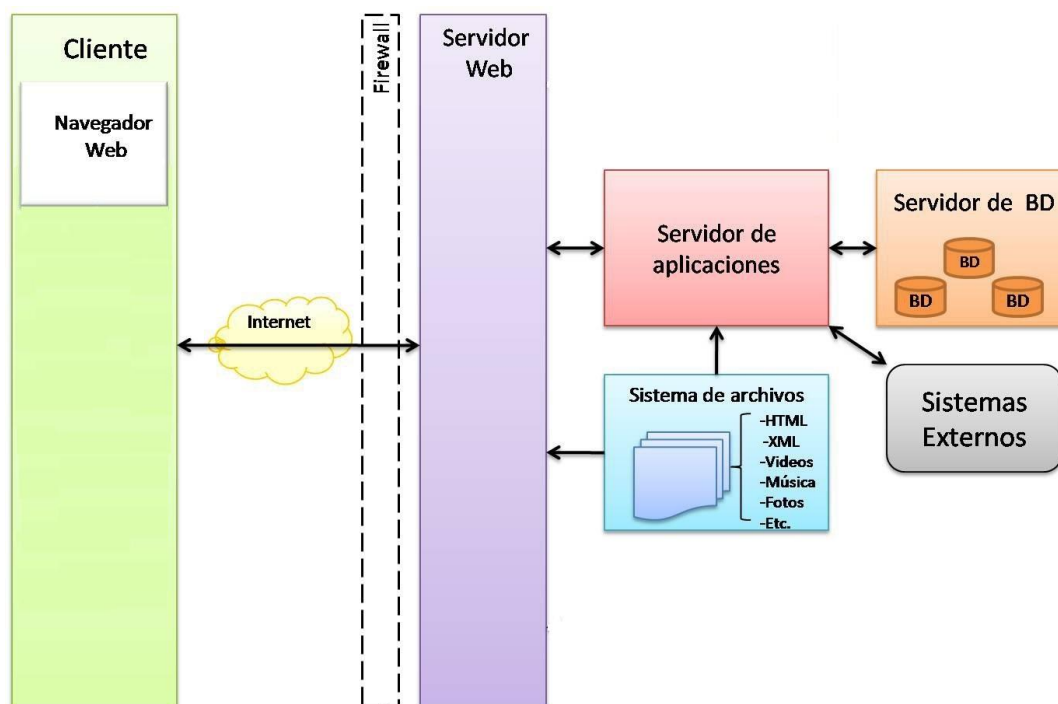


Figura 4 Arquitectura Cliente - Servidor

3.1.1.2 Servidor Web: es el encargado de recibir todas las peticiones de la aplicación, y es el componente que se encargará de atender la petición y enviarla en formato HTML por medio de los demás componentes. Como se puede apreciar, es el componente que mantiene una comunicación con la mayoría de los componentes.

3.1.1.2 Servidor de Aplicaciones: es el encargado de hospedar a la aplicación y de proporcionar lo necesario para que la aplicación pueda funcionar de forma correcta. Este servidor es el encargado de transformar la petición proveniente del servidor Web, y transformarla en un recurso (archivo HTML) utilizando todas las reglas de negocio establecidas.

3.1.1.3 Servidor de Base de Datos: es el encargado de proporcionar la persistencia de los datos de la aplicación por medio de un Sistema Gestor de Base de Datos (SGBD). Este servidor se encargará de almacenar cualquier dato que la aplicación necesite.

La arquitectura modelo cliente-servidor permite sentar las bases para que cualquier aplicación Web pueda funcionar correctamente. La arquitectura queda abierta a la agregación de nuevos componentes que sean necesarios para satisfacer las necesidades de la aplicación que se desea desarrollar (Hernández, 1997).

3.2 Diagrama de secuencia.

Los diagramas de secuencia son usados para modelar la interacción entre los objetos de un sistema. Estos muestran los objetos que intervienen en el escenario mediante líneas verticales discontinuas y los mensajes pasados entre estos con flechas horizontales (Larman, 1999).

A través de los diagramas de secuencia se proporciona una forma de mirar un escenario basada en el tiempo por lo que éste resulta una herramienta de gran utilidad a la hora de mostrar cómo se realizan los casos de uso y cómo se comporta la interacción dada entre los determinados objetos o entidades del software.

A continuación se muestran y explican los diagramas de secuencia para los casos de uso siguientes: Agregar Proyecto y Gestionar Objetivo General.

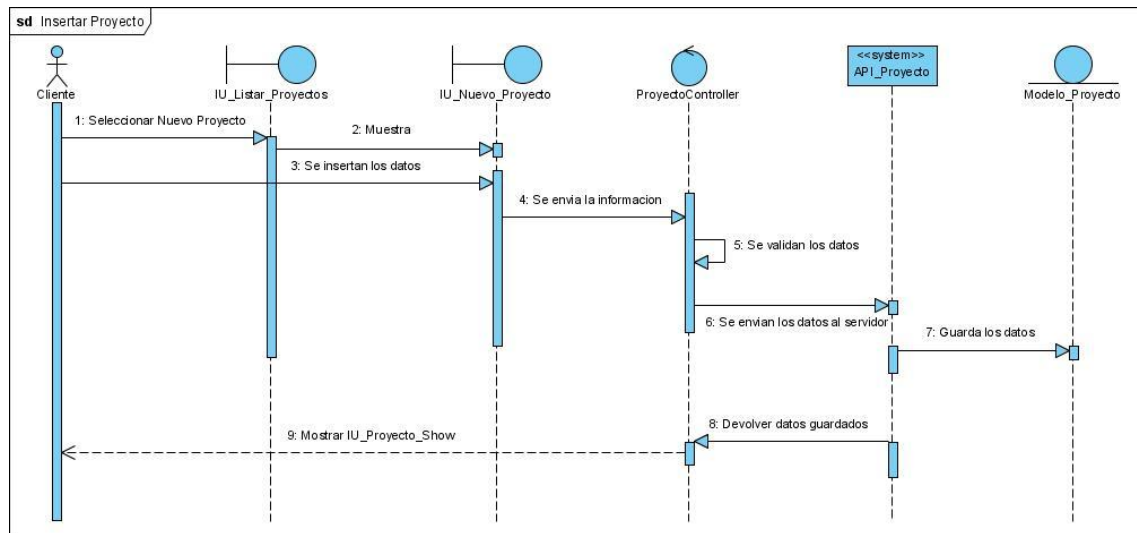


Figura 5 Diagrama de Secuencia del Caso de Uso Insertar Proyecto

La Figura 5 se muestra los pasos lógicos a seguir para que el sistema inserte un nuevo proyecto en la base de datos. Para ejecutar esta opción el Cliente elige el enlace Nuevo Proyecto y seguidamente el sistema le muestra un formulario con los datos a llenar.

Una vez insertados los datos el sistema los envía a la clase controladora ProyectoController, la cual se encarga de validar la información recibida. Posteriormente la información validada se envía al servicio web API_Proyecto, el cual es el encargado de insertar la información en la base de datos. Seguidamente el sistema re direcciona hacia la vista Proyecto_Show donde puede ver los datos recién insertados.

En la Figura 6 se expone el diagrama de secuencia Gestionar Objetivo General, que muestra un procedimiento análogo al anterior, solo que en este caso el Cliente modifica el objetivo general de un proyecto.

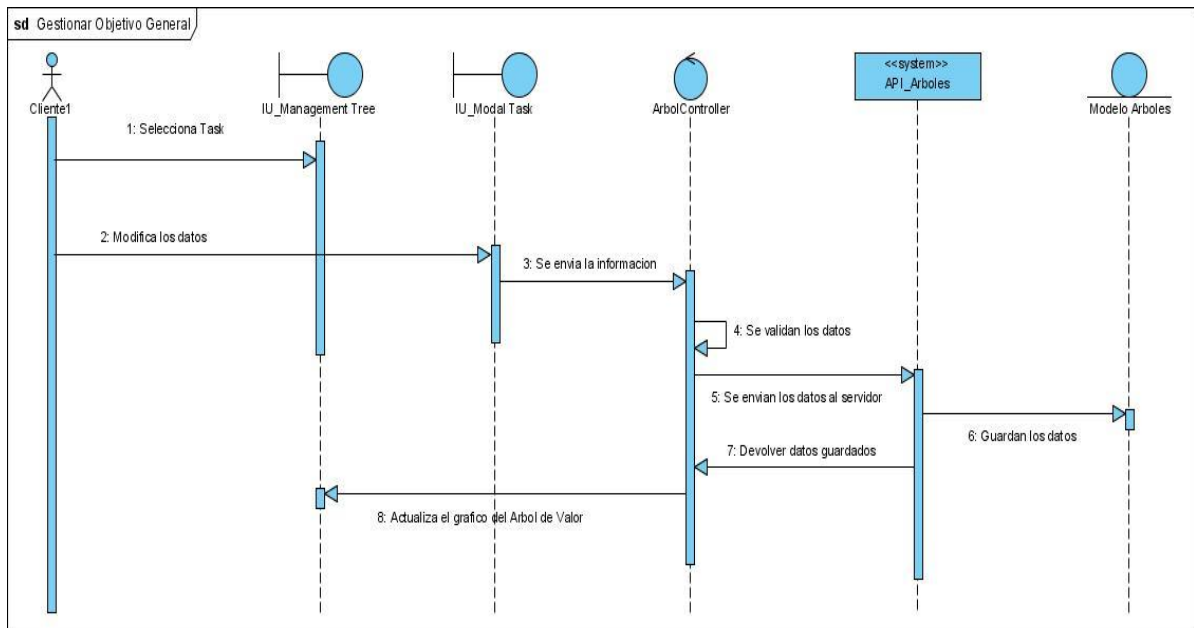


Figura 6 Diagrama de secuencia del Caso de Uso Gestionar Objetivo General

3.3 Tratamiento de errores

Cuando no se han añadido alternativas o criterios el cliente puede encontrarse con mensajes de errores como el de la Figura 7 y el de la Figura 8.

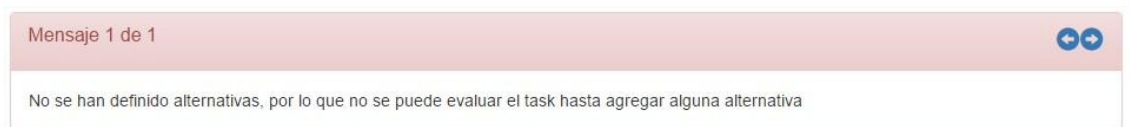


Figura 7 Mensaje de Error 1

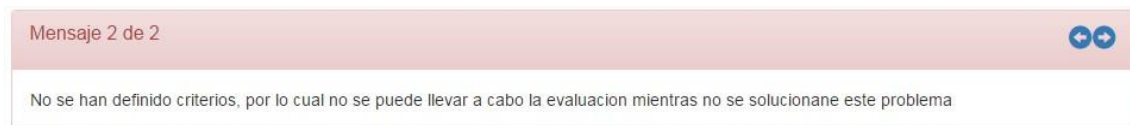


Figura 8 Mensaje de Error 2

El sistema valida todos los campos necesarios en los que el cliente introduce datos impidiendo que pasen datos corruptos o códigos maliciosos.

3.4 Modelo de componentes

Los diagramas de componentes muestran las dependencias del compilador y del “runtime” entre los componentes del software, un ejemplo que se puede citar de lo mencionado anteriormente serían los archivos del código fuente y los DLL (Larman, 1999). Mediante los diagramas de componentes, podemos representar un sistema dividido en componentes y mostrar las dependencias entre estos. Dentro de los componentes físicos están incluidos archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables o paquetes. Los diagramas de componentes pueden ser usados para modelar y documentar cualquier arquitectura de sistema, estos nos muestran qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema (Pressman, 2010). A continuación se muestra el diagrama de componentes del sistema en la Figura 9.

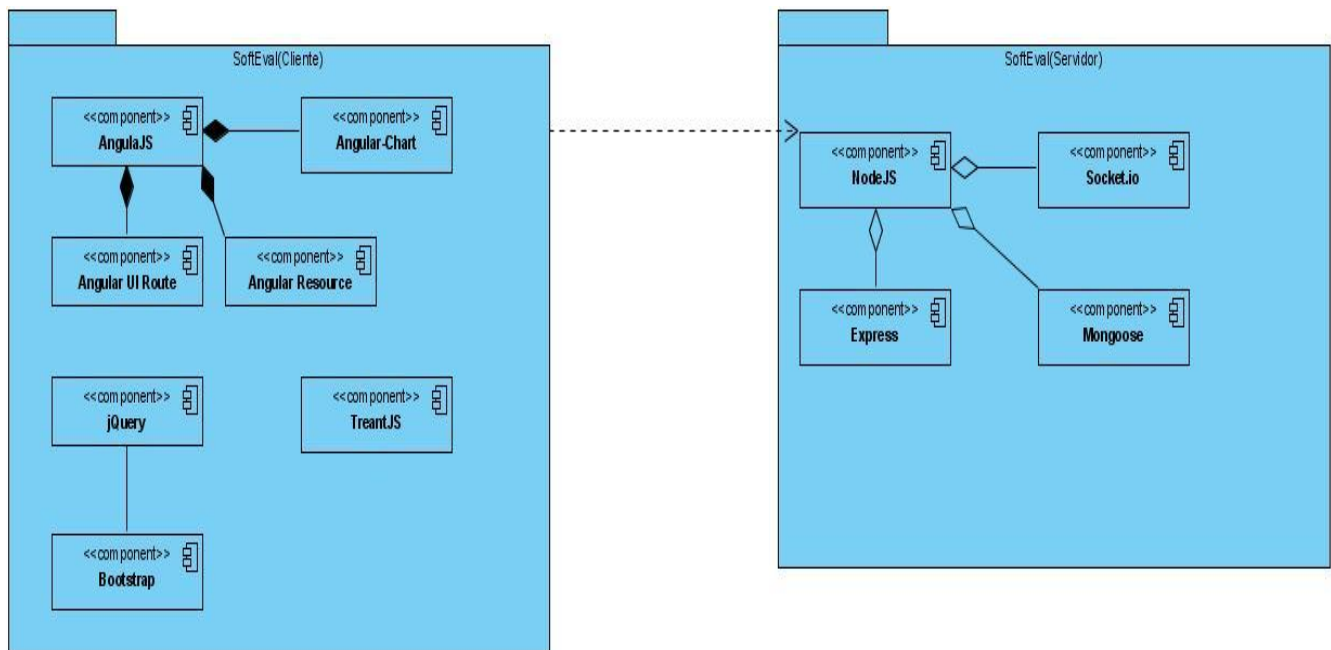


Figura 9 Modelo de Componentes

3.5 Diagrama de despliegue.

Los diagramas de despliegue muestran a los nodos procesadores la distribución de los procesos y de los componentes (Larman, 1999). El diagrama de despliegue representa la topología del hardware sobre el que se ejecuta el sistema. Generalmente se utilizan para modelar sistemas empotrados, sistemas Cliente-Servidor y sistemas completamente distribuidos (Pressman, 2010). A continuación se muestra el diagrama de despliegue en la Figura 10.

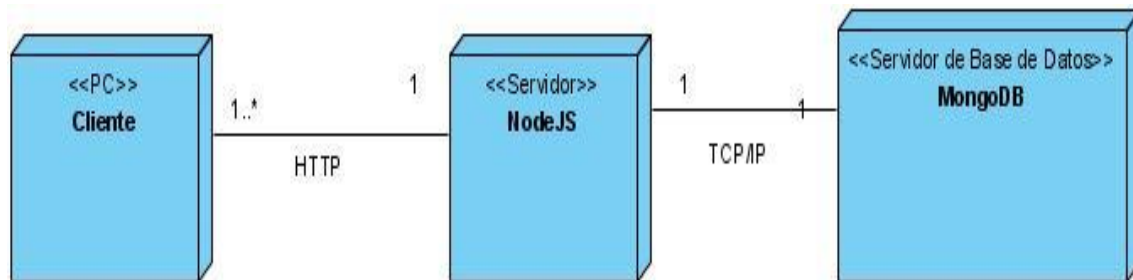


Figura 10 Diagrama de Despliegue

3.6 Conclusiones Parciales

En este capítulo se obtuvo el modelado del sistema, partiendo de la arquitectura del sistema hasta llegar al despliegue, no sin antes tocar el tratamiento de errores tenido en cuenta para la implementación de este sistema, en dicho modelado se encuentra implícita la descripción de la propuesta de solución de acuerdo a las necesidades del cliente.

Capítulo 4. PRUEBAS DEL SISTEMA Y ANÁLISIS DE FACTIBILIDAD

4.1 Estimación basada en el método de los Puntos de Casos de Usos.

Puntos de caso de uso es un método de estimación de esfuerzo para proyectos de software, a partir de sus casos de uso. Fue desarrollado por Gustav Karner en 1993, basándose en el método de punto de función, y supervisado por Ivar Jacobson. Ha sido analizado posteriormente en otros estudios, como la tesis de Kirsten Ribu en 2001.

El método utiliza los actores y casos de uso relevados para calcular el esfuerzo que significará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan factores de entorno y de complejidad técnica para ajustar el resultado (VALERO OREA).

4.1.1 Cálculo de Puntos de Casos de Uso sin ajustar

Ecuación

$$UUCP=UAW+UUCW$$

$$UUCP=3+20=23$$

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar

4.1.1.1 Factor de peso de los actores sin ajustar (UAW)

Consiste en la evaluación de la complejidad de los actores con los que tendrá que interactuar el sistema. Este puntaje se calcula determinando si cada actor es una persona u otro sistema, a la forma en la que este interactúa con el caso de uso, y la cantidad de actores de cada tipo. Los criterios para el cálculo se encuentran en la Tabla 6.

Tipo de Actor	Descripción	Factor de Peso	Número de Actores
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (<u>Application Programming Interface</u>)	1	0
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	1

Tabla 6 Factor de Peso según los actores

Entonces:

$$UAW = \sum (\text{Actor}_i * \text{Factor de Peso}_i)$$

$$UAW = 1 \times 0 + 2 \times 0 + 3 \times 1 = 3$$

4.1.1.2 Factor de peso de los casos de uso sin ajustar (UUCW)

Este punto funciona muy similar al anterior, pero para determinar el nivel de complejidad se puede realizar mediante dos métodos: basado en transacciones o basado en clases de análisis. Los criterios para el calcula se encuentran en la Tabla 7.

Tipo de Caso de Uso	Descripción	Factor de Peso	Número de Casos de Uso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	2
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	1
Complejo	El Caso de Uso contiene más de 8 transacciones	15	0

Tabla 7 Factor de Peso de los Caso de Usos

Entonces:

$$UUCW = \sum (\text{Caso de Uso } i * \text{Factor de Peso}_i)$$

$$UUCW = 5 \times 2 + 10 \times 1 + 15 \times 0 = 20$$

4.1.2 Cálculo de Puntos de Casos de Uso ajustados

Para esto se utilizan las siglas UCP y se obtiene al multiplicar el UUCP el TCF y el EF quedando la operación de la siguiente forma:

$$UCP = UUCP * TCF * EF$$

$$UCP = 23 * 0.99 * 0.815 = 18.55755$$

Donde

UCP: Puntos de Casos de Uso ajustado

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de Ambiente

4.1.2.1 Factor de complejidad técnica (TCF)

Este se compone de 13 puntos que evalúan la complejidad de los módulos del sistema que se desarrolla, cada uno de estos factores tienen un peso definido con los cuales se obtendrá puntos ponderados por cada uno de ellos, según la valoración que se le asigne. Para una mejor comprensión, en la Tabla 8 se mostrarán los criterios para el cálculo:

Factor	Descripción	Peso	Valor Asignado
T1	Sistema distribuido	2	1
T2	Objetivos de performance o tiempo de respuesta	1	5
T3	Eficiencia del usuario final	1	5
T4	Procesamiento interno complejo	1	3

T5	El código debe ser reutilizable	1	5
T6	Facilidad de instalación	0.5	4
T7	Facilidad de uso	0.5	4
T8	Portabilidad	2	5
T9	Facilidad de cambio	1	5
T10	Concurrencia	1	2
T11	Incluye objetivos especiales de seguridad	1	0
T12	Provee acceso directo a terceras partes	1	0
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	0

Tabla 8 Factores de Complejidad Técnica

Cada uno de estos puntos se debe evaluar según la que aparece en la Tabla 9:

Descripción	Valor
Irrelevante	De 0 a 2.
Medio	De 3 a 4.
Esencial	5

Tabla 9 Escala para la evaluación de los Factores de Complejidad Técnica

Entonces se tiene la siguiente ecuación

$$TCF = 0.6 + 0.01 * \sum (\text{Peso}_i * \text{valor asignado})$$

$$TCF = 0.6 + 0.01 \times (2*1 + 1*5 + 1*5 + 1*3 + 1*5 + 0.5*4 + 0.5*4 + 2*5 + 1*5 + 1*2)$$

$$TCF = 0.6 + 0.01 \times (39) = 0.99$$

4.1.2.2 Factor de ambiente (EF)

Los factores sobre los cuales se realiza la evaluación son 8 puntos, que están relacionados con las habilidades y experiencia del grupo de personas involucradas con el desarrollo del proyecto. Estos factores se muestran en la Tabla 10:

Factor	Descripción	Peso	Valor Asignado
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3
E2	Experiencia en la aplicación	0.5	2
E3	Experiencia en orientación a objetos	1	3
E4	Capacidad del analista líder	0.5	0
E5	Motivación	1	5
E6	Estabilidad de los requerimientos	2	4
E7	Personal part-time	-1	0
E8	Dificultad del lenguaje de programación	-1	3

Tabla 10 Factor de Ambiente

Para los factores E1 al E4, un valor asignado de 0 significa sin experiencia, 3 experiencia media y 5 amplia experiencia (experto).

Para el factor E5, 0 significa sin motivación para el proyecto, 3 motivación media y 5 alta motivación.

Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.

Para el factor E7, 0 significa que no hay personal part-time (es decir todos son full-time), 3 significa mitad y mitad, y 5 significa que todo el personal es part-time (nadie es full-time).

Para el factor E8, 0 significa que el lenguaje de programación es fácil de usar, 3 medio y 5 que el lenguaje es extremadamente difícil.

El Factor de ambiente se calcula mediante la siguiente ecuación:

$$EF = 1.4 - 0.03 \times \sum (\text{Peso } i \times \text{Valor asignado})$$

$$EF = 1.4 - 0.03 * (1.5*3+0.5*2+1*3+0.5*0+1*5+2*4-1*0-1*3) = 0.815$$

4.1.3 Esfuerzo horas-hombre (E).

Este cálculo se realiza con el fin de tener una aproximación del esfuerzo, pensando solo en el desarrollo según las funcionalidades de los casos de uso.

$$E = UCP * CF$$

$$E = 18.55755 * 20$$

$$E = 371.151$$

Donde

E: Esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: Factor de conversión (20 horas-hombre por defecto)

Para el cálculo del Factor de conversión se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.

Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

- Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un punto de Caso de Uso toma 28 horas-hombre.
- Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

4.1.4 Estimación del esfuerzo del proyecto.

La [Tabla 11](#) detalla la distribución en porcentaje, para el esfuerzo total del desarrollo del proyecto:

Actividad	Porcentaje
Análisis	10.00%
Diseño	20.00%
Programación	40.00%
Pruebas	15.00%
Sobrecarga(otras actividades)	15.00%

Tabla 11 Distribución del Esfuerzo

Con esta distribución y tomando como entrada la estimación de tiempo calculada a partir de los Puntos de Casos de Uso la cual pertenece a la Actividad de Programación , se pueden calcular las demás estimaciones para obtener la duración total del proyecto como se muestra en la Tabla 12.

Actividad	Porcentaje	Horas / hombre
Análisis	10.00%	92.788
Diseño	20.00%	185.576
Programación	40.00%	371.151
Pruebas	15.00%	139.182
Sobrecarga(otras actividades)	15.00%	139.182
Total	100%	927.879

Tabla 12 Distribución del Esfuerzo en el Proyecto

4.1.5 Cálculo del esfuerzo total.

$$E_{total} = \sum \text{actividades}$$

$$E_{total} = 927.879 \text{ horas/hombres}$$

Donde:

E_{Total}: esfuerzo total

4.1.6 Cálculo del tiempo de desarrollo.

$$TDesarrollo = E_{Total} / CH_{Total} / CH_{Trabajo}$$

$$TDesarrollo = 927.879 / 1 / 6$$

$$TDesarrollo = 154.6 \text{ días}$$

Donde:

TDesarrollo: tiempo de desarrollo total en horas

CHTotal: cantidad total de hombres

CHTrabajo: cantidad de horas de trabajo diario

4.1.7 Cálculo del costo

$$\text{CostoTotal} = E_{Total} * CH_{Total} * TH$$

$$\text{CostoTotal} = 927.879 * 1 * 1,031$$

$$\text{CostoTotal} = \$956.643249$$

Donde:

TH: tarifa horaria (suponga \$1.031) (VALERO OREA).

4.2 Casos de Pruebas

Las pruebas de software son investigaciones técnicas, las cuales tienen como objetivo proporcionar información objetiva acerca de la calidad del producto a la parte interesada, o sea, al usuario interesado en el mismo. Podemos denominar las pruebas como una actividad más en el proceso de control de calidad del software, siendo estas un conjunto de actividades dentro del desarrollo del mismo. En dependencia del tipo de prueba que se realice, estas actividades se podrán desarrollar en cualquier momento del proceso de desarrollo. Los elementos que condicionarán las pruebas a realizar, deben ser seleccionadas y utilizadas de la manera más eficiente posible según el contexto del proyecto (Sommerville, 2002).

4.2.1 Pruebas de Caja Negra

Las pruebas de caja negra no necesitan acceder al código fuente. Los casos de prueba se derivan de la especificación del programa. En estas pruebas, el sistema es una “caja negra”, cuyo funcionamiento sólo se puede determinar estudiando las entradas y salidas relacionadas. Las pruebas de caja negra también son denominadas “pruebas funcionales”, ya que el probador sólo se interesa en la funcionalidad y no en la implementación del software (Sommerville, 2002).

El probador del software, introduce las estradas en los componentes del sistema, luego examina las salidas correspondientes, si estas no son como se previeron, entonces la prueba ha detectado un problema con el software. El primer paso, el cual es un problema clave para el probador, es seleccionar las entradas que tienen un alta probabilidad de presentar comportamiento anómalo, con una correcta selección de estas, existe una alta probabilidad de que la prueba detecte correctamente los problemas en el software (Sommerville, 2002)

4.2.2 Pruebas de caja negra en casos de uso del sistema

4.2.2.1 Descripción General

Planteando como objetivo, la construcción de un software con la calidad requerida, se implementan una serie de pruebas para la corrección de posibles errores que pueda presentar el mismo. En el presente epígrafe, se realizan las pruebas de caja negra a los casos de uso del sistema.

4.2.2.2 Condiciones de ejecución

1. La aplicación debe correr sobre el entorno de programación NodeJS.
2. La aplicación debe tener acceso a una base de datos en MongoDB que se usará como repositorio.
3. Deberá existir un usuario logueado dentro de la aplicación.

4.2.2.3 Secciones a probar en el caso de uso “Gestionar Proyectos”

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Adicionar Proyecto	EC 1.1: El cliente desea adicionar un nuevo Proyecto	La aplicación muestra una ventana donde el cliente debe ingresar los datos	1-El cliente selecciona en el menú Project la opción <u>New Project</u> 2-El sistema muestra una ventana donde el cliente debe ingresar los datos 3-El cliente introduce los datos y hace clic en el botón <u>Create</u> . 4-El sistema redirige hacia la vista <u>Show Project</u>
SC 2: Modificar Proyecto	EC 2.1: El cliente desea modificar un Proyecto existente	La aplicación muestra una ventana con los datos que debe modificar el cliente	1-El cliente presiona el botón <u>Edit</u> sobre un Proyecto de los que se muestran en la Vista 2-El sistema muestra una ventana con los datos que debe modificar el cliente 3-El cliente modifica los datos y hace clic en el botón <u>Update</u> . 4-El sistema redirecciona hacia la vista <u>Show Project</u>

Tabla 13 Secciones a probar en el caso de uso "Gestionar Proyectos"

4.2.2.4 Escenarios a probar en el caso de uso “Gestionar Proyectos”

Id del escenario	Escenario	Variable 1 <i>Name</i>	Variable 2 <i>Description</i>	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	El cliente desea agregar un nuevo Proyecto	V	N/A	El sistema recibe los datos del cliente y los guarda en la base de datos	Prueba superada con éxito
		I	N/A	El sistema deshabilita el botón Create impidiendo que se envíen los datos al servidor.	Prueba superada con éxito
EC 2.1	El cliente desea modificar un Proyecto existente	V	N/A	El sistema recibe los datos del cliente y los guarda en la base de datos	Prueba superada con éxito
		I	N/A	El sistema deshabilita el botón Update impidiendo que se envíen los datos al servidor.	Prueba superada con éxito

Tabla 14 Escenarios a probar en el caso de uso "Gestionar Proyectos"

4.2.2.5 Secciones a probar en el caso de uso “Gestionar Árbol de Valor”

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Gestionar Objetivo General	EC 1.1: El cliente desea gestionar el objetivo general	La aplicación muestra una ventana de confirmación para eliminar el proyecto	1-El cliente presiona el botón <u>Management Tree</u> sobre un Proyecto de los que se muestran en la Vista 2-El cliente presiona el link <u>Task</u> en la ventana de <u>Management Tree</u> 3-El cliente modifica el objetivo general. 4-El cliente presiona el botón <u>Close</u> para terminar

Tabla 15 Secciones a probar en el caso de uso "Gestionar Árbol de Valor"

4.2.2.6 Escenarios a probar en el caso de uso “Gestionar Árbol de Valor”

Id del escenario	Escenario	Variable 1 <i>Task</i>	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	El cliente desea gestionar el objetivo general	V	El sistema recibe los datos del cliente y los guarda en la base de datos	Prueba superada con éxito
		I	El sistema añade un mensaje de alerta a la ventana e impide que se	Prueba superada con éxito

			pueda evaluar el árbol.	
--	--	--	-------------------------	--

Tabla 16 Escenarios a probar en el caso de uso "Gestionar Árbol de Valor"

4.3 Conclusiones Parciales

Con la realización de las pruebas al software, se logra detectar los posibles errores que pueda presentar el mismo a la hora de importar, exportar, clonar y eliminar las colecciones, garantizando así un mejor funcionamiento del mismo.

Conclusiones Generales

Como consecuencia de la presente investigación se logra:

- La identificación de métodos MCDA existentes en la literatura científica, que pueden ser utilizados para reducir el conjunto inicial de medidas de software.
- Se lograron identificar los métodos MAVT y AHP, además de que se logró integrar el primero en una aplicación informática que expone el mismo como un servicio Web.
- Se validó la aplicación propuesta a través de la realización de diferentes pruebas de cajas negras.

Recomendaciones

Para próximos trabajos se recomienda lo siguiente:

- Integrar el método AHP encontrado en la bibliografía
- Investigar acerca de nuevos métodos que puedan ser utilizados en el proceso de evaluación de.
- También se recomienda la refactorización del código fuente de la aplicación y su posterior emigración del framework AngularJS de la versión 1.X.X a la versión 2.X.X.

Referencias bibliográficas

- .
Sistemas de gestión de la calidad. Conceptos y vocabulario. Norma Internacional ISO 9000, 2000.
Software Evaluation Guide [Online]. Available: www.software.ac.uk.
 2011. Available: <http://www.readwriteweb.com/hack/2011/01/wait-whats-nodejs-good-for-aga.php>.
 CERIA, S. 2002. *Casos de Uso - Un Método Práctico para Explorar Requerimientos*.
 COCHRAN, D. 2012. *Twitter Bootstrap Web Development How-To*.
 CHODOROW, K. 2013. *MongoDB The Definitive guide*.
 FRANGANILLO, J. 2010. HTML5: el nuevo estándar básico del web.
 FREEMAN, A. 2014. *Pro AngularJS*.
 GREEN, B. 2013. *AngularJS*.
 HANDY, A. 2011. Node.js pushes JavaScript to the server-side.
 HAVIV, A. Q. 2014. *MEAN Web Development*.
 HERNÁNDEZ, E. 1997. Cliente/Servidor.
 KELLER, L. R. 2009. *Multiple-Objective Decision Analysis Involving Multiple Stakeholders*.
 LARMAN, C. 1999. UML y Patrones, Introducción al análisis y diseño orientado a objetos.
 LEONARD RICHARDSON, S. R. 2007. RESTful Web Service.
 LERNER, A. 2013. *ng-book The complete book on AngularJS*.
 PACHECO, J. F. 2009. Manual metodológico de evaluación multicriterio para programas y proyectos.
 PACHECO, Y. 2016. SELECCIÓN DE MEDIDAS DOMINANTES EN LA EVALUACIÓN DE SISTEMAS INFORMÁTICOS
 PRESSMAN, R. S. 2010. *Software Engineering (Seven Edition)*.
 SAATY, T. L. 1996. The Analytic Network Process.
 SANDEEP, P. 2014. *AngularJS Novice to Ninja*.
 SESHADRI, S. 2014. *AngularJS UP & Running*.
 SOMMERVILLE, I. 2002. *Ingeniería de Software*.
 STEWART, V. B. A. T. J. 2001. *Multiple Criteria Decision Analysis: An Integrated Approach*.
 VALERO OREA, S. ESTIMACIÓN DE PROYECTOS DE SOFTWARE CON PUNTOS DE CASOS DE USO.
 WENDY BOGGS, M. B. 2002. *Mastering UML with Rational Rose*