Universidad de las Ciencias Informáticas

Facultad 3

Título: Implementación del método TOPSIS en un sistema de ayuda en el proceso de Toma de Decisiones Multicriterio

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

Autor: Yanet Peña Táramo
Tutores: Ing. Lizandra Arza Pérez
Ing. Hermes Miguel Velázquez Domínguez

La Habana, Cuba 2013

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo al Macro proyecto de investigación: Modelo de integración docencia-producción-investigación de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los __ días del mes de ___ del año __.

_____

Yanet Peña Táramo

Autor

_____ _____

Lizandra Arza Pérez
Tutor

Hermes Miguel Velázquez Domínguez
Tutor

DATOS DE CONTACTO

Lizandra Arza Pérez: Graduada de Ingeniería Informática en la CUJAE, año 2002. Profesora Asistente. Directora para la gestión de proyectos.

Hermes Miguel Velázquez Domínguez: Graduado en la Universidad de las Ciencias Informáticas, 2008. Profesor Instructor.

AGRADECIMIENTOS

A mi mamá y mi familia en general.
A Ángel.
A mis tutores.
A todas mis amistades.
A todos los que confiaron en mí y a todos los que aportaron su granito de arena.

DEDICATORIA

A mi mamá por ser mi regalo más lindo.
A mi papá negro por darme su amor incondicional.
A mi papá.
A mis hermanos, por siempre tenerme presente.
A mi otra mamá, mi tía Libia.
A mis primos más queridos Mayte y Maikel.
A mi abuelita del alma, Gladys.

RESUMEN

Se entiende por Técnicas de Decisión Multicriterio al conjunto de herramientas y procedimientos utilizados en la resolución de problemas de decisión; en los que intervienen diferentes criterios. Existen varios métodos de toma de decisión multicriterio que procesan la información de manera diferente; estos tienen una formalización matemática que requiere de complejos cálculos, lo cual dificulta su aplicación de manera manual; uno de ellos es el método TOPSIS *(Technique for Order Preference by Similarity to Ideal Solution).*

El presente trabajo expone los resultados de la investigación desarrollada con el objetivo de obtener un sistema que implemente el método TOPSIS como parte de una *suite* de métodos de toma de decisiones que se desarrolla en la Universidad de las Ciencias Informáticas. Este sistema de ayuda a la toma de decisión multicriterio elimina la complejidad de la realización manual de los cálculos y evita errores en el proceso de aplicación. Además posibilita agilizar el proceso de toma de decisiones y puede ser utilizado en diferentes problemas, esto último es posible a partir de funcionalidades que permiten la configuración del problema de decisión para el cual se desea utilizar.

El método ha sido implementado siguiendo la metodología SXP, utilizando como sistema gestor de base de datos PostgresSQL, y como Entorno de Desarrollo Integrado el Eclipse. El módulo implementa varias funcionalidades que permiten la ejecución del algoritmo del método TOPSIS. Se realizaron pruebas de caja negra para validar las funcionalidades de la aplicación y se aplicaron las métricas de diseño Tamaño Operacional de la Clase (TOC) y Relación entre Clases (RC) para evaluar los requisitos de calidad por lo que se puede concluir que la herramienta desarrollada cumple con los requisitos definidos en la etapa de concepción del sistema.

PALABRAS CLAVE

Proceso de toma de decisión multicriterio, Sistema de Ayuda a la Decisión, Método TOPSIS, Metodología SXP.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En muchas situaciones, los seres humanos toman decisiones, esta es una práctica de la vida diaria. Las personas que tienen que asumir la responsabilidad de tomar decisiones difíciles cuyas consecuencias influirán en el proyecto o en la organización a la que pertenecen o dirigen, están sometidas a tensiones profesionales y emocionales.

En todas las organizaciones existen problemas de diversa naturaleza, sin embargo tienen un denominador común: la necesidad de elegir entre diferentes alternativas que han de evaluarse en base a criterios. En muchas ocasiones un proceso de toma de decisión está definido por la complejidad a la hora de elegir la mejor o las mejores alternativas, en función de varios criterios tanto de carácter cualitativo como cuantitativo. Entre los diversos métodos de la Toma de Decisión Multicriterio, existe un subgrupo que incluye aspectos de costes y beneficios, entre los que se encuentra el método TOPSIS, técnica para ordenar las preferencias mediante la similitud a la solución ideal. Este se basa en el concepto que es deseable que una alternativa determinada se ubique a la distancia más corta respecto a una solución ideal positiva y a la mayor distancia respecto a una solución ideal negativa (1).

La implementación manual de dicho método es muy engorrosa y muchas veces se vuelve casi impracticable, así como susceptible a errores durante el proceso de toma de decisión.

Se han realizado intentos para resolver o atenuar este problema con algunos sistemas informáticos. Estos presentan desventajas entre las que se encuentran ser sistemas propietarios y que solo pueden ser instalados en el sistema operativo Windows, además de estar implementados para una solución específica.

En la Universidad de las Ciencias Informáticas (UCI) se desea implementar una *suite* que implemente varios métodos de toma de decisión multicriterio, entre ellos el método TOPSIS. La investigación se desarrolla con el fin de añadir el método TOPSIS, como un módulo de esta *suite* de métodos de toma de decisión multicriterio.

Con los elementos mencionados anteriormente se define el **problema** de la presente investigación: *La complejidad de la confección manual del método TOPSIS, propicia que el proceso de aplicación de los mismos sea susceptible a errores y demora el proceso de toma de decisiones.*

El **objeto de estudio** está constituido por *Los Sistemas de soporte a la Toma de Decisiones Multicriterio* y el **objetivo general**: *Desarrollar un módulo para el sistema de soporte a la toma de decisiones multicriterio que implemente el método TOPSIS que elimine la complejidad de la confección manual del mismo, evite que el proceso de aplicación sea susceptible a errores y*

agilice el proceso de toma de decisiones. El **campo de acción** estará determinado por *los métodos de Toma de Decisiones Multicriterio.*

Los **objetivos específicos** de la investigación son:

1. Establecer el marco teórico referencial sobre los sistemas de soporte a la decisión, el método TOPSIS y las herramientas para el desarrollo de sistemas de ayuda a la decisión.
2. Modelar las funcionalidades del método TOPSIS como parte del sistema de ayuda a la toma de decisión multicriterio.
3. Diseñar e implementar las funcionalidades modeladas.
4. Validar la solución propuesta.

Para garantizar el cumplimiento de los objetivos propuestos se llevarán a cabo las siguientes **tareas de la investigación**:

1. Caracterizar los sistemas de ayuda a la toma de decisión multicriterio.
2. Analizar los métodos para la toma de decisión multicriterio.
3. Caracterizar el método de toma de decisión multicriterio TOPSIS.
4. Seleccionar y caracterizar la metodología, las herramientas y las tecnologías a utilizar para el desarrollo de la solución propuesta.
5. Definir las funcionalidades del módulo TOPSIS.
6. Elaborar el diseño de la solución.
7. Realizar el modelo de datos.
8. Elaborar el diagrama de componentes.
9. Implementar las funcionalidades definidas.
10. Validar el diseño mediante la aplicación de métricas.
11. Validar la solución mediante la aplicación de pruebas de caja negra.

Se plantea la siguiente **idea a defender** para la investigación: *Con el desarrollo de un módulo para el sistema de ayuda a la toma de decisión multicriterio que implemente el método TOPSIS, se eliminará la complejidad de la confección manual de los mismos, se evitará que el proceso de aplicación sea susceptible a errores y agilizará el proceso de toma de decisiones.*

Los **métodos y técnicas** empleados fueron los siguientes:

Métodos Teóricos

Permiten descubrir en el objeto de investigación las relaciones esenciales y las cualidades fundamentales, no detectables de manera senso-perceptual. Por ello se apoya básicamente en los procesos de abstracción, análisis, síntesis, inducción y deducción.

Análisis-Síntesis: Mediante este método se descompone un objeto, fenómeno o proceso en los principales elementos que lo integran para analizar, valorar y conocer sus particularidades. En este caso se utilizó para el análisis de la información y para llegar a conclusiones sobre el estudio realizado.

Histórico-Lógico: A través de este método se establece la necesaria correspondencia entre los elementos de los métodos lógico e histórico, proyectando el análisis de la evolución histórica de los fenómenos, con la proyección lógica de su comportamiento futuro. En esta investigación se le dio utilidad para establecer la evolución y tendencia de los sistemas de apoyo a la decisión.

Modelación: Este consiste en la representación ya sea material o teórica de los objetos, fenómenos, o particularidades de estos, lo que permite descomponerlos, abstraer determinadas cualidades, operar y experimentar con él.

Métodos Empíricos

Estos métodos posibilitan revelar las relaciones esenciales y las características fundamentales del objeto de estudio, accesibles a la detección de la percepción, a través de procedimientos prácticos con el objeto y diversos medios de estudio.

Observación: Es un método para reunir información visual sobre lo que ocurre, lo que el objeto de estudio hace o cómo se comporta. En este caso se usó para entender cómo funcionan los sistemas de ayuda a la decisión.

Análisis de documentos: Se estudiaron los documentos relacionados con la metodología y el método a implementar.

Técnicas de recopilación de información

La recopilación de datos se utiliza para verificar los métodos empleados en lo investigado, para llegar a la conclusión del suceso, teniendo las pruebas y una serie de pasos que se llevan a cabo para comprobar la hipótesis planteada.

Entrevistas: Es una técnica para obtener datos, que consiste en un diálogo entre dos personas. En este caso se empleó con el fin de obtener información referente al método y la metodología a utilizar.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los conceptos necesarios para entender el desarrollo de esta investigación, como son: proceso de toma de decisión, sistema de soporte a la decisión, método TOPSIS, entre otros.

Se presenta el estudio realizado para definir la metodología y las herramientas que se utilizan en el desarrollo de la aplicación.

1.1 Sistemas de Soporte a la Decisión

El proceso de tomar una decisión es una de las actividades que se realiza en el mundo de los negocios con mayor frecuencia. La misma se presenta en todos los niveles de la organización, ya sean asistentes, auxiliares, o directores generales de las empresas. En cualquier caso se tiene uno o varios objetivos a cumplir teniendo en cuenta un conjunto de restricciones. Los Sistemas de Ayuda para la toma de Decisiones (SAD) constituyen un tipo específico de sistemas de información orientados a la toma de decisiones organizacionales. Se los puede describir como aplicativos que tienen incorporadas las estrategias necesarias y la inteligencia apropiada para poder evaluar consignas complejas por su contexto o características. Muchas veces se trata de especulaciones a futuro o evaluaciones de situaciones donde el contexto de la empresa es dinámico. Son utilizados en el entrenamiento o bien para la toma de decisiones.

Entre sus características más importantes se pueden mencionar (2):

- Interactividad controlada con un administrador.
- Interactividad controlada con un especialista.
- Interfaz gráfica para usuarios que realizan consultas.
- Se basan en una plataforma informática acorde (normalmente con infraestructura adecuada a la capacidad de cómputo requerida).
- Suelen integrar sistemas de información.
- Incorporan modelos que describen y predicen procesos productivos.
- Opcionalmente pueden asistir a los decisores en la selección de datos y modelos para identificar y resolver problemas y tomar decisiones.
- Proveen información textual y gráfica.

1.1.1 Proceso de toma de decisión

Se puede afirmar que un individuo o colectivo tiene un problema de decisión cuando se plantea un conjunto bien definido de alternativas o cursos de acción posibles, al menos dos, y un conflicto tal que es necesario elegir una de las alternativas, o bien establecer en ese conjunto unas preferencias.

El Análisis de Decisión Multicriterio se presenta como una valiosa herramienta para ayudar al decisor durante este proceso de toma de decisiones. Los métodos propuestos desde esta disciplina permiten abordar, de forma sistemática y ordenada, un problema en el que subyace una gran subjetividad. Ayudan a que todas las partes afectadas por el proceso de decisión participen en el mismo, suministran una gran cantidad de información, facilitan la búsqueda de consenso, permiten que el decisor aprenda sobre el propio problema de decisión y ayudan a racionalizar un proceso complejo. Se puede destacar, que de manera general las distintas definiciones tienen elementos comunes, entre los que se señalan los siguientes (3):

- Se elige o selecciona.
- En función de criterios.
- Entre un grupo de alternativas.
- Para lograr un objetivo.

1.2 TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution*)

Es un método de decisión multicriterio, cuyo objetivo es la ordenación de un conjunto finito de alternativas. Este método se basa en que una alternativa seleccionada debe tener la distancia más corta posible hacia la solución ideal positiva y estar lo más lejos posible respecto de la solución ideal negativa (4). Fue desarrollada por Hwang y Yon en 1981, recibiendo posteriores aportes de Zeleny (1982), Hall (1989). Fue mejorada por los propios autores en 1987 y más tarde conjuntamente con Lai y Liu en 1993. Una solución ideal se define como una colección de niveles ideales (o de puntajes) en todos los atributos considerados, pudiendo suceder que tal solución normalmente sea inalcanzable o que sea no factible. Esta noción se basa en la idea que el logro de tal meta se encuentra en la racionalidad de la elección humana. El vector compuesto por los mejores valores del j-ésimo atributo respecto a todas las alternativas posibles es quien recibe el nombre de solución ideal positiva. En contraposición, la solución ideal negativa estaría dada por el vector que contiene los peores puntajes alcanzables en los atributos. De este modo puede ocurrir que una alternativa seleccionada desde el punto de vista de su más corta distancia respecto de la solución ideal positiva deba competir con otra alternativa que se encuentra lo más lejos posible de la solución ideal negativa. Por ello, y a fin de definir la solución ideal, el método

TOPSIS define un índice de similaridad (o de proximidad relativa) que se construye combinando la proximidad al ideal positivo y la lejanía respecto al ideal negativo (5).

Como paso final del algoritmo TOPSIS se obtiene un conjunto de alternativas ordenadas según su índice de similaridad. Para llegar al resultado final el método TOPSIS sigue una serie de pasos o etapas que se muestran en la Figura 1.

1.2.1 Algoritmo del método TOPSIS

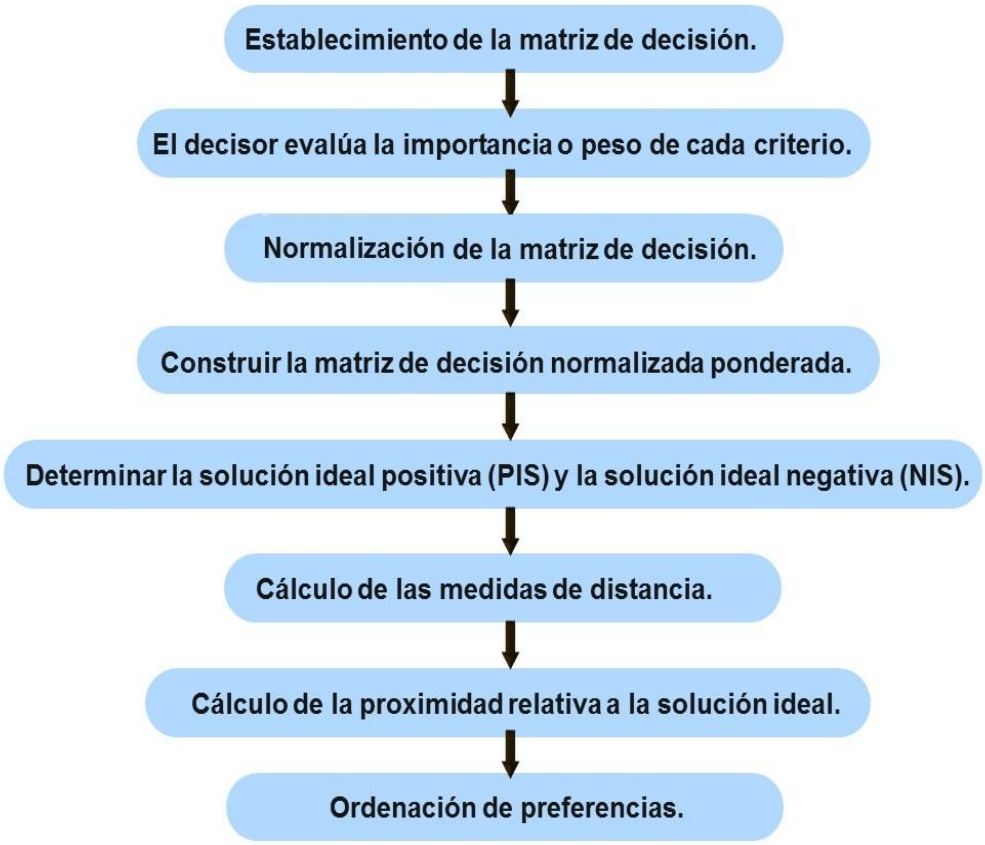

Figura 1: Pasos del algoritmo del método TOPSIS (Elaboración propia) (1) (4) (5).

Paso 1: Establecimiento de la matriz de decisión

El método TOPSIS evalúa la siguiente matriz de decisión que se refiere a *m* alternativas y *n* criterios:

Matriz de decisión

	w_1	w_2	...	w_j		w_n
	C_1	C_2	...	C_j	...	C_n
A_1	x_{11}	x_{12}	...	x_{1j}	...	x_{1n}
A_2	x_{21}	x_{22}	...	x_{2j}	...	x_{2n}
...	...	...	...	...	...	...
A_m	x_{m1}	x_{m2}	...	x_{mj}	...	x_{mn}

Figura 2: Matriz de Decisión (1).

Donde X_{ij} denota la valoración de la i-ésima alternativa en términos del j-ésimo criterio.

Paso 3: Normalización del matriz de decisión

En el método TOPSIS primero convierte las dimensiones de los distintos criterios en criterios no dimensionales. Un elemento n de la matriz de decisión normalizada $m \times n$ se calcula como:

$$r_{ij} = \frac{X_{ij}}{X_{ij}^{max}}$$

Donde cada elemento r_{ij} es el elemento normalizado de la matriz, donde X_{ij} es el elemento a normalizar de la matriz de decisión y X_{ij}^{max} es el máximo valor que alcanza un elemento en la matriz.

Paso 4: Construir la matriz de decisión normalizada ponderada

El valor normalizado ponderado de la matriz de decisión normalizada ponderada se calcula como (4):

$$v_{ij} = r_{ij} * w_j$$

Donde v_{ij} es el nuevo elemento de la matriz normalizada ponderada, r_{ij} es el elemento normalizado de la matriz y w_j es el peso del criterio que se corresponde con el elemento normalizado.

Paso 5: Determinar la solución ideal positiva (PIS) y la solución ideal negativa (NIS)

El conjunto de valores ideal positivo *A*+ y el conjunto de valores ideal negativo

A− se determina como (6):

$$A^+ = (x_1^+, x_2^+, \dots\dots, x_n^+)$$

Donde A^+ es un conjunto de valores que representa el ideal positivo y x_j^+ el máximo valor que toman los elementos de la matriz cuando se evalúan en el criterio j.

$$A^- = (x_1^-, x_2^-, \dots\dots, x_n^-)$$

Donde A^- es un conjunto de valores que representa el ideal negativo y x_j^- el mínimo valor que toman los elementos de la matriz cuando se evalúan en el criterio j.

Paso 6: Cálculo de las medidas de distancia

Para el cálculo de las distancias al ideal positivo (*A*+) y al ideal negativo (*A*−), el método TOPSIS propone la distancia euclidiana, distancia de Euler, distancia de Manhattan entre otras, se trabajó con la distancia ciudad o distancia Manhattan en un enfoque alterno del método TOPSIS; se calcula como (6):

$$d_{(p,q)} = \sum_{j} |p_j - q_j|$$

Donde $d_{(p,q)}$ es la distancia entre el punto p y el punto q, siendo $p = (p_1, p_2, \dots\dots, p_n)$ y $q = (q_1, q_2, \dots\dots, q_n)$.

$$d_i^+ = \sum_{j=i}^{n} d(v_{ij}, v_j^+)$$

Donde d_i^+ es la distancia de la alternativa i al ideal positivo.

$$d_i^- = \sum_{j=i}^{n} d(v_{ij}, v_j^-)$$

Donde d_i^- es la distancia de la alternativa i al ideal negativo.

Paso 7: Cálculo de la proximidad relativa a la solución ideal

$$IS_i = \frac{d_i^-}{d_i^- + d_i^+}$$

Donde IS_i es el índice de similaridad de la alternativa i.

Paso 8: Ordenación de preferencias

Se ordenan las alternativas de acuerdo con el *IS* en orden descendente, mientras mayor sea el IS mas preferida es la alternativa.

1.2.2 Sistemas que implementan TOPSIS

Se realizó una búsqueda para identificar aplicaciones informáticas que implementaran el método TOPSIS en alguna de sus variantes. El análisis de estos sistemas se hizo en función de caracterizar su comportamiento de acuerdo al volumen de datos que pueden procesar, las facilidades de configuración para diferentes problemas y contextos, así como otros elementos tecnológicos como la posibilidad de ser instalado en diferentes sistemas operativos.

Topsis Solver

TopsisSolver2012 es un *software* que implementa el método TOPSIS, es capaz de gestionar problemas de hasta 20 criterios y 20 alternativas, introduce por sí mismo el ideal negativo y positivo, funciona en Windows. Es una herramienta privativa y está implementada para una solución específica (7).

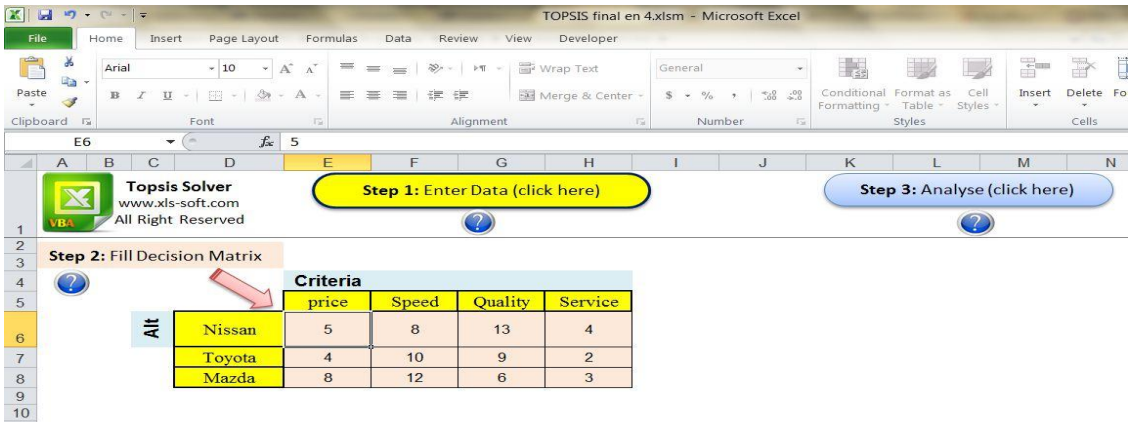

Figura 3: Topsis Solver (7).

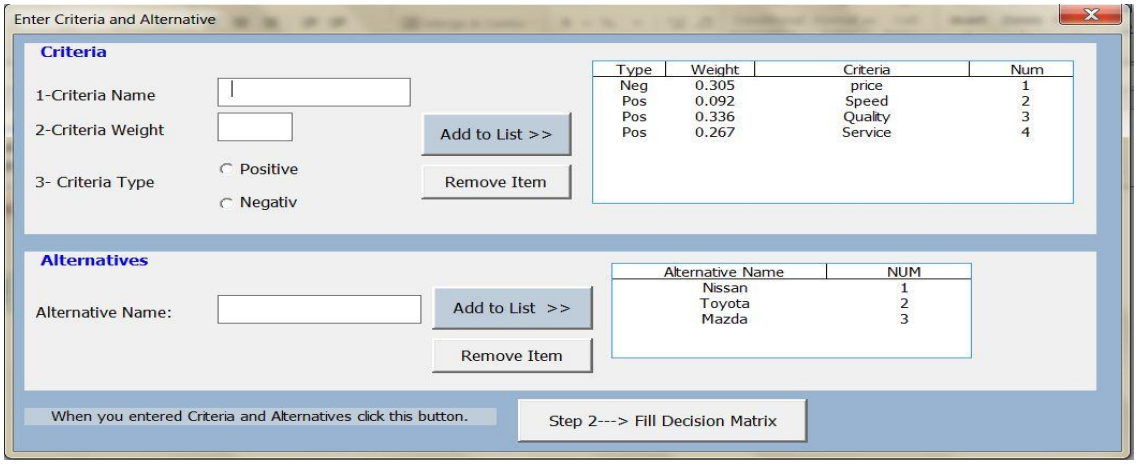

Figura 4: Topsis Solver (7).

SADRU-II

SADRU-II es un sistema de ayuda a la toma de decisiones para el ranking de universidades. Implementado en el 2009. El *software* se presenta en ejecutable (SADRU-II.exe). Es una herramienta libre y está implementada para una solución específica (8).

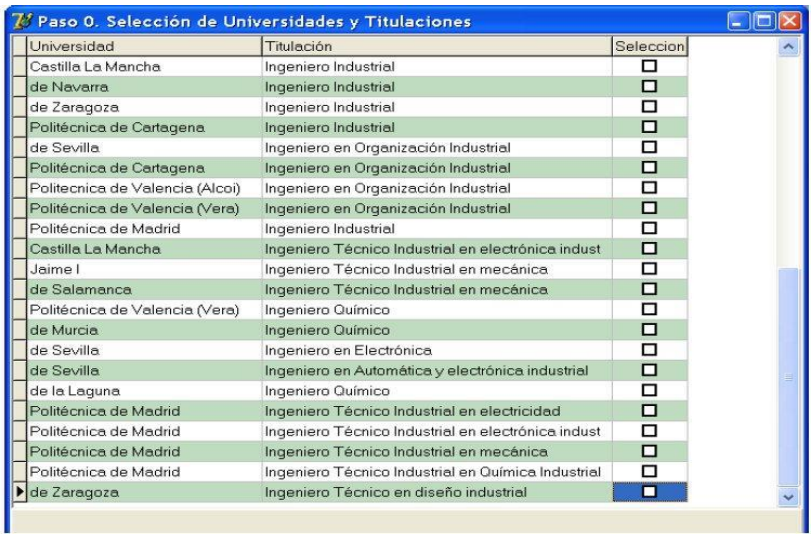

Figura 5: Sadru-II (8).

Topzis Aplication Program

Implementado del año 2010 al 2011 es una herramienta privativa desarrollada por Rodrigo Martínez y Cecilia Paterno de la Universidad Tecnológica Nacional, Facultad Regional de Córdoba.

La herramienta está organizada en una secuencia funcional de interfaces, en las cuales se plantea alguna etapa del proceso. Además, cada una de ellas ofrece la funcionalidad de navegar entre las distintas etapas del proceso de manera de verificar los datos o corregirlos en el caso que sea necesario (4).

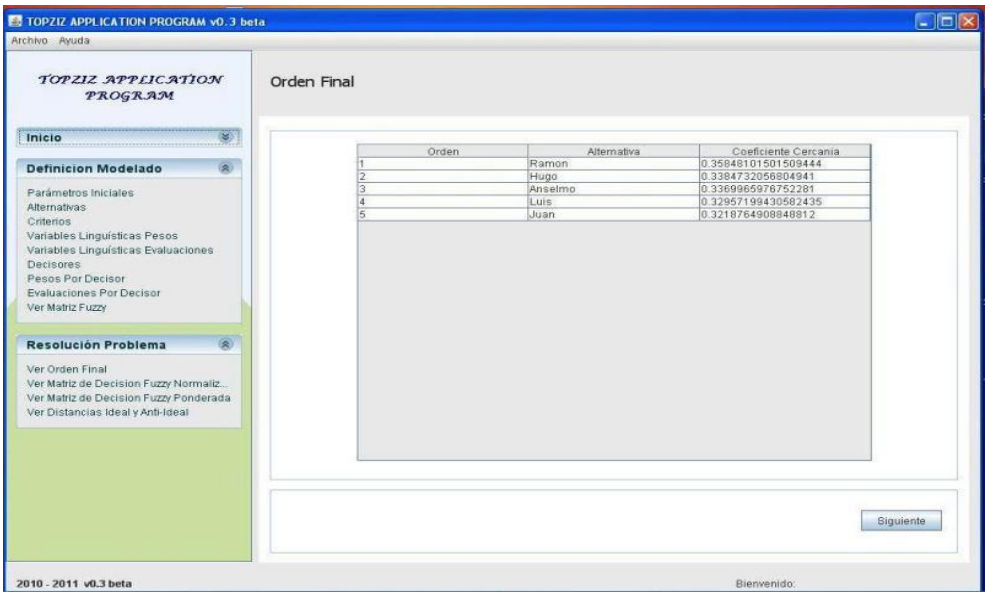

Figura 6: Topzis Aplication Program (4).

De las herramientas estudiadas se concluye que no se ajustan a las necesidades del sistema a desarrollar, debido a que en su mayoría son privativas, solo pueden ser instaladas en el sistema operativo *Windows.* Además están implementadas para soluciones específicas.

1.3 Herramientas y metodologías de desarrollo.

Para el desarrollo del sistema informático es necesario definir las herramientas y metodología a utilizar, para lo cual se hace un análisis de algunas de ellas y se define las que serán empleadas en la solución.

1.3.1 Metodologías para el desarrollo

Las Metodologías de desarrollo de *software* surgen ante la necesidad de utilizar procedimientos, técnicas, herramientas y soporte documental para el desarrollo de un producto, que facilite las tareas a desarrollar y su posterior mantenimiento (9). Dichas metodologías guían a los desarrolladores a crear un *software*; pero los requisitos de unos a otros son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del *software* (10).

Las metodologías tradicionales son aquellas orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán.
Las metodologías ágiles son aquellas orientadas a la interacción con el cliente y el desarrollo incremental del producto, mostrando versiones parcialmente funcionales del sistema al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Se centran, en el factor humano o en el producto del *software*. Las mismas están mostrando su efectividad fundamentalmente en proyectos que presentan requisitos cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero sin alterar la calidad (11). Las necesidades del cliente pueden variar desde el momento de contratación hasta el momento de su entrega; y es de mayor importancia satisfacer estas últimas que las primeras. Esto requiere procesos de *software* que en lugar de rechazar los cambios los incorpore. Las metodologías ágiles suministran una serie de normas y principios junto a técnicas que intentan hacer la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega.
Se ha definido que se utilizarán metodologías ágiles para el desarrollo del presente trabajo debido a que permiten disminuir costos, brindar flexibilidad a proyectos de *software*, generar poca documentación y mostrar productos funcionales en poco tiempo. Además de presentar un equipo de desarrollo pequeño y para el desarrollo de la aplicación es muy conveniente hacer al cliente parte del mismo.

1.3.1.1 SCRUM, XP, SXP

Dentro de las diferentes metodologías ágiles que existen se encuentran SCRUM (Gestión Ágil de Proyectos) y XP (Programación Extrema) debido a las características que presentan que se ajustan a las necesidades para el desarrollo del *software* a implementar, además del estudio de una personalización de estas dos metodologías que se ha utilizado en proyectos de la Universidad de las Ciencias Informáticas con el nombre de SXP. A continuación se muestran algunas de las características de estas metodologías.
Extreme Programing, o Programación Extrema (XP), fue desarrollada por Kent Beck. Consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. Este tipo de método se basa en una retroalimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes. Este tipo de programación es la adecuada para los proyectos

con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo. Está pensada para un grupo pequeño y muy integrado dónde la comunicación sea más factible que en grupos de desarrollo grandes. Sus características principales son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas: frecuentemente repetidas y automatizadas, incluyendo regresión de pruebas. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.
- Frecuente interacción del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir una nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código: es decir, reescribir ciertas partes del código para aumentar su legibilidad sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código. La programación extrema plantea que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo (12).

Por otra parte SCRUM es una metodología ágil y liviana utilizada para administrar y controlar el desarrollo de un software. Está centrada en priorizar el trabajo, maximizando la utilidad de lo que se construye. Los requisitos y las prioridades se examinan y ajustan durante el proyecto en intervalos cortos y regulares. Esta metodología busca entregar un producto que resuelva las necesidades, aumentando la satisfacción del cliente.

La gestión de un proyecto en SCRUM se concentra en definir qué características debe tener el producto a construir, transformando cualquier obstáculo que pudiera entorpecer la tarea del equipo de desarrollo. Este proceso busca que los equipos sean lo más prácticos y ágiles posible.

SCRUM posee un conjunto de reglas muy simples, basado en los principios de inspección continua, adaptación, auto-gestión e innovación. Teniendo como objetivo que el cliente se entusiasme y se comprometa con el proyecto, debido a que ve crecer el producto iteración tras

iteración, localizando las herramientas para alinear el desarrollo con las metas de negocio de su empresa. Por otro lado, el equipo encuentra un ámbito propicio para desarrollar sus capacidades profesionales y resultando un incremento en la motivación de los integrantes del mismo.
Las principales características de SCRUM son:

- Es una metodología de desarrollo ágil.
- Está pensada para equipos de desarrollos pequeños.
- Se especifican pocos artefactos eliminando documentación que puede ser innecesaria, dedicando más tiempo a la implementación.
- Permite la entrega de un producto funcional al finalizar cada Sprint.
- Da la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- Permite hacer una visualización del proyecto diaria.
- Tiene un alcance acotado y viable.
- Se aplica en equipos integrados y comprometidos con el proyecto y que se auto administran.
- No es una metodología de análisis, ni de diseño (aunque puede adaptarse para que lo sea) sino de gestión del trabajo (13).

En la UCI se ha definido la metodología SXP, a partir de las metodologías XP y SCRUM. Ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, y ayudando al líder del proyecto a tener un mejor control del mismo. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto (14).
La creación de SXP se centra principalmente en la utilización de SCRUM para lograr una correcta planificación y organización; mientras que XP respalda con sus prácticas todo el proceso de desarrollo y de esta forma se obtiene un proceso de *software* completo y además para guiar el proceso ingenieril, caracterizándose por presentar una documentación discreta y de mayor dinamismo. SCRUM propone actividades que son adecuadas para la gestión de proyectos

relativamente pequeños, sirviendo de soporte para acelerar el dinamismo identificado en XP. Las reuniones diarias con el cliente posibilitan su integración con el equipo de desarrollo.

En SXP se definen 4 fases principales:

- Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- Desarrollo, es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- Entrega, puesta en marcha.
- Mantenimiento, donde se realiza el soporte para el cliente.

De cada una de ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implementación, prueba, entrega de la documentación, soporte e investigación, utilizándose este último por el equipo de desarrollo cuando sea necesario, es decir, es un flujo que se puede mover y utilizarlo en cualquier parte del ciclo de vida del proyecto (15).

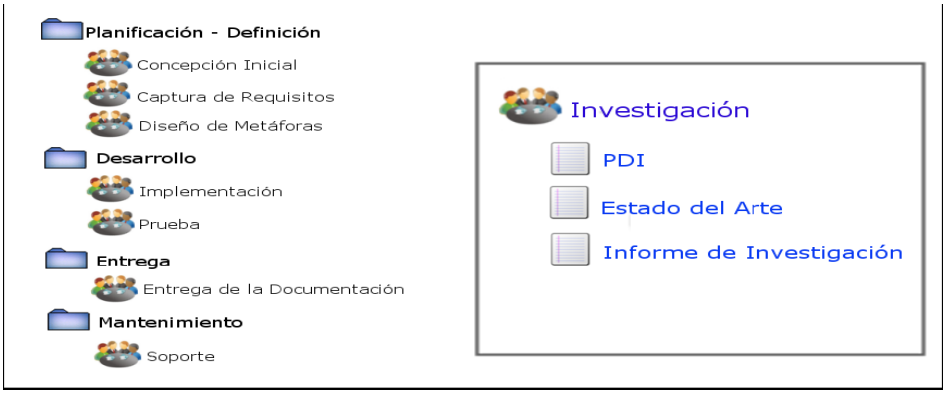

Figura 7: Fases y flujos de trabajo de SXP (15).

En la presente investigación se decidió usar esta personalización de las metodologías XP y SCRUM por sus características; ajustándose las mismas a las características propias del equipo de desarrollo, que el equipo está formado por 1 estudiante y un líder, el mismo necesita reunirse diariamente para discutir avances y posibles cambios, es de vital importancia para el equipo ganar en tiempo para no retrasar la entrega.

Las fases, actividades y los artefactos que se generan en ellas se representan en el esquema de la Figura 7, para lograr una mayor comprensión de la metodología a utilizar.

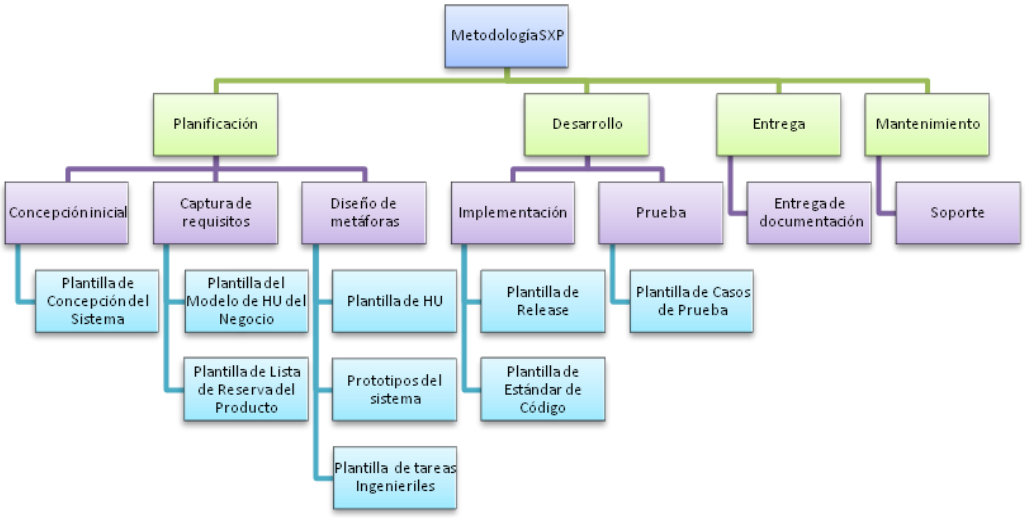

Figura 8: Esquema de la metodología SXP (Elaboración Propia) (15)

1.3.2 Herramientas Case

Las herramientas CASE (*Computer Aided Software Engineering*) constituyen un conjunto de programas y ayudas que facilitan las tareas de los analistas, ingenieros de *software* y desarrolladores durante el ciclo de vida de desarrollo de un *software*. También CASE se define como un conjunto de utilidades y técnicas que facilitan parcial o completamente el desarrollo de sistemas de información. Otros la entienden como una filosofía de desarrollo de *software* que representa una innovación en la organización, una unión entre herramientas de *software* automáticas y las metodologías de desarrollo de *software* (16).

1.3.2.1 Visual Paradigm

La herramienta CASE seleccionada para el desarrollo del proyecto es Visual Paradigm. La misma soporta la metodología de desarrollo seleccionada, pudiéndose por consiguiente, generar todos los diagramas y esquemas necesarios. Se integra con el entorno de desarrollo que será usado en la fase de implementación, cubre todas las fases del ciclo de vida de desarrollo del *software*, tiene soporte para aplicaciones web, permite la generación de código en el lenguaje Java, generación de base de datos entre otras funcionalidades. Ofrece una colección de herramientas para la captura de requisitos, planeación, pruebas, modelamiento de clases, modelado de datos y otros (17).

Se tomó en cuenta para la selección de la herramienta las características que tiene, fundamentalmente que se puede adquirir mediante licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones. Existe amplia experiencia en la universidad en el uso de esta herramienta CASE en los proyectos productivos.

1.3.3 UML-Lenguaje Unificado de Modelado

Se hace más fácil comprender un sistema si tenemos una forma de dividirlo en partes o

pequeños fragmentos, pudiendo representarse las mismas como modelos que de una forma sencilla abstraigan los aspectos esenciales del sistema que se analiza; siendo precisamente un paso fundamental para la creación de *software* la creación de modelos que organicen y expresen los aspectos más importantes de la vida real con que se relaciona y del sistema en cuestión. Los modelos se componen de otros modelos, diagramas y documentos que describen artefactos, siendo preciso mencionar al Lenguaje Unificado de Modelado (UML) (18).

UML es un lenguaje, que proporciona un vocabulario y reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. No especifica en sí mismo qué metodología o proceso usar, clasificando sus elementos en estructurales (cases, interfaces. colaboraciones, casos de uso, clases activas, componentes y nodos), de comportamiento (interacciones y máquinas de estado), de agrupación (paquetes) y de anotación. A su vez, hay cuatro tipos de relaciones: Dependencia, de asociación, de agrupación y de realización. Para construir un plano de *software* que tenga sentido, lo que se hace es combinar los elementos estructurales con sus respectivas relaciones, según sea el caso, obteniendo como resultado uno de los nueve diagramas que existen en UML, a saber: de clases, de objetos, de casos de uso, de secuencia, de colaboración, de estados, de actividades, de componentes y de despliegue (19).

Funciones de UML (19)**:**

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Se decidió usar como lenguaje de modelado UML porque es libre y por las facilidades que brinda para el desarrollo de la aplicación.

1.3.4 Lenguaje de programación

1.3.4.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por *Sun Microsystems* a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero

tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (20).

El mismo se presenta bajo licencia GNU GPL, por tanto se puede considerar el lenguaje Java como de Software Libre. Actualmente java se ha convertido en uno de los lenguajes más usados y más demandados por los desarrolladores. Este lenguaje permite aprovechar la flexibilidad de la Programación Orientada a Objetos (POO) en el diseño de sus aplicaciones. Algunas de las características del lenguaje son las siguientes (21):

- Orientado a objetos: Java ratifica el paradigma de la programación orientada a objetos (POO) desde el mismo punto en que ofrece numerosas bibliotecas de clases denominadas *APIs Core* Java que no son más que una colección de componentes.

- Interpretado: El código escrito en Java es primeramente compilado a un lenguaje intermedio denominado *bytecode*, que luego es interpretado por una máquina virtual *(Java Virtual Machine, JVM)* que sirve como una abstracción entre la máquina y el lenguaje permitiendo que se pueda ejecutar la aplicación en cualquier arquitectura.

Se considera que para el desarrollo de la aplicación Java es el lenguaje de programación más adecuado, por las características que posee el lenguaje, y porque se integra con las herramientas y tecnologías que se serán utilizadas.

1.3.5 Marcos de trabajo

1.3.5.1 Spring

Spring es un marco de trabajo de código abierto para el desarrollo de aplicaciones para la plataforma Java. Fue creado con el objetivo de guiar el desarrollo de aplicaciones empresariales complejas. La utilidad de Spring no se limita solo al desarrollo del lado del servidor, cualquier aplicación Java puede recibir el beneficio de usar Spring en aras de la simplicidad, facilidad de probar las aplicaciones y el bajo acoplamiento (22). Mientras que muchas aplicaciones fuerzan al código de la aplicación a estar al tanto del marco de trabajo implementando interfaces específicas del mismo o extendiendo las funcionalidades de clases específicas del mismo, Spring se enfoca en la dependencia del código de la aplicación del marco de trabajo configurando objetos de la aplicación que no importen las *APIs* del marco de trabajo (23). Spring interviene en todas las capas arquitectónicas de una aplicación Java y su diseño brinda flexibilidad arquitectónica. Para la selección de Spring como marco de trabajo se tuvo en cuenta que es un marco de trabajo de código abierto y por las facilidades que brinda para el desarrollo de la solución.

1.3.5.2 Hibernate

Hibernate es una herramienta para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos *XML* que permiten establecer estas relaciones. Está distribuido bajo los términos de la licencia *GNU LGPL*. Hibernate es una herramienta *ORM* completa que ha conseguido una excelente reputación en la comunidad de desarrollo posicionándose como uno de los productos de código abierto líder en este campo gracias a sus prestaciones, buena documentación y estabilidad (24). El equipo de desarrollo seleccionó Hibernate como marco de trabajo a utilizar para la capa de acceso a datos, ya que es una herramienta libre y satisface las necesidades para el desarrollo de la solución propuesta.

1.3.5.3 Vaadin

Vaadin es un marco de trabajo para el desarrollo de aplicaciones *Web Open Source* modernas, que posee una librería con numerosos componentes de usuario con los que construir aplicaciones web profesionales similares a los interfaces de una aplicación de escritorio. Este marco de trabajo se ejecuta del lado del servidor, lo que significa que la mayoría de la lógica y por tanto la mayor carga del trabajo recae en el servidor.

Vaadin minimiza casi a “cero” la cantidad de JavaScript y HTML requerido. La idea es, básicamente, pensar como si se estuvieran haciendo aplicaciones de escritorio y obtener resultados tremendamente parecidos a esta (25).

El equipo de desarrollo para la capa de presentación decidió utilizar el marco de trabajo Vaadin, por las facilidades que este le brinda al programador y por ser una herramienta de *software* libre.

1.3.6 Gestor de Base de Datos

Los servidores de bases de datos son dispositivos que almacenan grandes colecciones de datos de forma estructurada. Son utilizados en todo el mundo en una gran cantidad de aplicaciones. Permiten realizar almacenamiento, acceso y análisis de datos al estar los mismos almacenados de forma estructurada. La mayoría de los servidores ofrecen una interfaz de texto para interactuar con el servidor usando SQL o alguna de sus variantes (26).

1.3.6.1 PostgresSQL

PostgresSQL es un sistema de administración de base de datos relacional desarrollado en el departamento de Ciencias de Computación de la Universidad de California. Es un proyecto de código abierto y posee un excelente ajuste al estándar SQL. Es distribuido bajo una licencia similar a la *BSD* o *MIT*, que permite a los usuarios disponer del código (puede ser usado, modificado y distribuido por cualquiera libremente), incluso revender el producto compilado sin el

código fuente. La única restricción es la imposibilidad de responsabilizar legalmente a los desarrolladores por problemas con el *software*. PostgresSQL debería poder funcionar sin problemas en cualquier plataforma compatible con UNIX además de ejecutarse en los sistemas operativos Windows (27) (28). Está estructurado en una arquitectura cliente – servidor, se pueden realizar consultas con alto grado de complejidad, existencia y manejo de llaves foráneas, disparadores *(triggers)*, vistas, integridad transaccional, un diseño enfocado en soportar configurable y extensible, es escalable y tiene un alto rendimiento con muchas opciones para optimizarlo. Permite una alta concurrencia donde las operaciones de lectura y escritura no se bloquean entre sí. Esta característica es denominada "*Multi-Version Concurrency Control*" (28). Se decidió usar PostgresSQL, ya que es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia de *Distribución de Software Berkeley* (BSD, *Berkeley Software Distribution*, según sus siglas en inglés).Además Posee la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

1.3.7 Entorno de Desarrollo Integrado (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, los mismos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java y C# entre otros (29).

1.3.7.1 Eclipse

Eclipse es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Con respecto a las aplicaciones clientes, abastece al programador con marcos de trabajo muy ricos para las aplicaciones gráficas, definición y manipulación de modelos de *software*, aplicaciones web, entre otros.

La utilización de Eclipse brinda muchas ventajas como son (30):

- Emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la Plataforma de Cliente rico, a diferencia de otros entornos donde las funcionalidades están todas incluidas, las necesite o no el usuario.

- Este mecanismo de módulos le permite a Eclipse extenderse usando otros lenguajes de programación.
- Eclipse provee al programador con marcos de trabajo muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de Software, Aplicaciones web, entre otras.
- Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código.

El equipo de desarrollo decidió utilizar el Eclipse porque es una herramienta libre, porque es compatible con el lenguaje de programación Java y por las facilidades que brinda.

1.3.8 Servidor de Aplicaciones

1.3.8.1 Tomcat

Tomcat es un servidor Web con soporte para servlets y JSPs. Es un producto muy robusto, altamente eficiente y uno de los más potentes contenedores de Servlets existentes. El mismo incluye un pequeño servidor HTTP para atender las peticiones estáticas y redirigir el resto al motor de ejecución de *Servlets* (31).

Actualmente Tomcat puede funcionar como servidor de aplicaciones por sí solo pero también puede especificarse como el manejador de las peticiones de JSP y *servlets* recibidas por servidores Web populares, como el servidor HTTP Apache de la Fundación Apache. Es una solución de servidor sin coste de licencia y de fácil mantenimiento y que permite el establecimiento de clústeres fácilmente aprovechando el potencial del servidor HTTP Apache. Actualmente es el servidor más utilizado en Internet para el despliegue de aplicaciones web. De forma nativa no soporta *EJBs* ni *Web Services*, pero esto se puede lograr con su integración con otras soluciones que brindan estos servicios (32).

Presenta compatibilidad con las API más recientes de Java. Ocupa poco espacio, presentando su código binario un *megabyte* de tamaño, logrando con ello que se ejecute de manera rápida. Otra característica de Tomcat es que es muy fiable. Presenta gran éxito como producto de código libre. El equipo de desarrollo decidió trabajar con Tomcat por sus características y porque este satisface las necesidades para el desarrollo de la aplicación que se propone.

1.3.9 Controlador de versiones

1.3.9.1 SVN

El equipo tomó la decisión de hacer uso de SVN, pudiendo expresar del mismo primeramente que es software libre. Una característica importante de Subversion es que los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el

repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo (33).

Ventajas (34):

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente (tiene costo de complejidad constante y no lineal como en CVS).
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).

Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.

Conclusiones Parciales

En el presente capítulo después de un estudio de las distintas variantes del método TOPSIS se definió la variante a utilizar para su posterior diseño e implementación. Se caracterizaron algunos sistemas que implementan el método TOPSIS, comprobando que estos carecen de características y funcionalidades requeridas por el sistema a desarrollar. Luego del estudio de diferentes metodologías se demuestra que la metodología SXP es adecuada para el equipo de trabajo y el tamaño de la solución. Se seleccionó un conjunto de herramientas y tecnologías que facilitarán el desarrollo del sistema debido a su integración, facilidad de uso, y documentación.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se hace una descripción de la solución propuesta, así como la explicación de los principales artefactos de la metodología empleada para las etapas de Planificación y Definición. La implementación de la solución se basa en los principios y reglas de la metodología SXP, utilizando las tecnologías y herramientas definidas. Se exponen además los patrones empleados durante el desarrollo así como la concepción de la arquitectura con el objetivo de brindar una mayor comprensión de la propuesta de solución.

2.1. Descripción del problema

Por las características de un proceso de toma de decisión, la realización manual del método TOPSIS se torna engorrosa, está sujeta a errores y demora mucho tiempo. En muchas ocasiones la toma de decisiones está asociada a riesgos de gran escala, como pueden ser bienes materiales y hasta vidas humanas. Se ha demostrado que en un proceso de toma de decisión donde la realización manual del análisis de grandes volúmenes de información, trae como consecuencia errores en los resultados del proceso, los plazos de tiempo necesitados son muy grandes e incluso se puede tornar impracticable. De aquí que la premisa será como implementar el método TOPSIS como módulo de un sistema de toma de decisión que permita reducir los tiempos e impedir que la misma sea susceptible a errores.

2.2. Solución Propuesta

Partiendo de la necesidad de implementar el método TOPSIS en un sistema de soporte a la toma de decisión que sea capaz de adaptarse a diferentes tipos de problemas y que tenga en cuenta los criterios de los especialistas en el proceso de toma de decisión; se hace necesario el desarrollo de una aplicación informática que implemente el método TOPSIS en un sistema de soporte a la decisión. Esto permitirá el procesamiento de grandes volúmenes de información, agilizando el proceso de toma de decisión y evitando que el sistema sea susceptible a errores.

2.3 Concepción inicial del sistema

En la etapa de concepción del sistema se genera la plantilla de concepción del sistema, en ella se especifican los aspectos generales organizativos y de concepción del sistema, su objetivo, principales involucrados y otros aspectos que permiten la posterior organización del desarrollo del proyecto. Esta plantilla se muestra en el Anexo 1.

2.4 Captura de requisitos

Los requisitos de *software* se encuentran clasificados en dos grupos: funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir sin alterar la funcionalidad del producto y se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los requisitos no funcionales son propiedades o

cualidades que el producto debe tener; especifican propiedades como restricciones del entorno o de la implementación, rendimiento, dependencia de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. Los requisitos de manera general recogen las necesidades del cliente añadiendo el valor esperado por estos. Deben ser descritos de modo que sean comprendidos por los usuarios y clientes (35). Entre las técnicas existentes para la captura de requerimientos se empleó la entrevista y la lluvia de ideas para determinar en conjunto los requisitos del sistema. Como resultado de la aplicación de estas técnicas se obtuvo la plantilla de modelos de historias de usuarios del negocio y la lista de reserva del producto, estos elementos son descritos en los siguientes epígrafes. Una vez concluida la captura de requisitos y luego de varias revisiones estos fueron aprobados y validados por el cliente.

2.4.1 Historias de usuarios del negocio

En la plantilla del modelo de historias de usuarios del negocio se describen los actores y trabajadores del negocio, además se presenta un diagrama de historias de usuarios del negocio que permite ver la relación entre los usuarios y las actividades que se realizan, de ahí que se puedan obtener de ello los requisitos. Esta plantilla se describe en el anexo 2.

2.4.2 Lista de reserva del producto (LRP)

En la lista de reserva, artefacto generado en la captura de requisitos, se describen los mismos como funcionalidades que el sistema debe cumplir en su desarrollo. Estos son descritos en la siguiente tabla:

Prioridad	Ítem	Descripción	Estimación en sprint	Estimado por	Asignado a
Requisitos Funcionales Muy Alta					
	1	Crear Decisión	1	programador	Yanet Peña Táramo
	2	Gestionar Alternativa	1	programador	Yanet Peña Táramo
	3	Gestionar Criterio	1	programador	Yanet Peña Táramo
	4	Gestionar Pesos de los criterios	1	programador	Yanet Peña Táramo
	5	Evaluar alternativas respecto a criterios	1	programador	Yanet Peña Táramo
Alta					
	6	Autenticación de usuario	2	programador	Yanet Peña Táramo
	7	Generar Ranking Final	2	programador	Yanet Peña Táramo

Media					
	8	Gestionar Usuario	3	programador	Yanet Peña Táramo
	9	Generar Ranking Final por pasos	3	programador	Yanet Peña Táramo
Baja					
	10	Seleccionar cantidad especialistas	1	programador	Yanet Peña Táramo
	11	Seleccionar Decisión	2	programador	Yanet Peña Táramo
Requisitos No Funcionales					
Usabilidad					
	1	El *software* permitirá acceder a todas sus funcionalidades de manera sencilla y directa.		Programador	Yanet Peña Táramo
	2	El tiempo de entrenamiento requerido para que usuarios sean productivos operando el sistema es de 2 días.		Programador	Yanet Peña Táramo
	3	Debe poseer una interfaz agradable para el cliente.		Diseñador	Yanet Peña Táramo
Fiabilidad					
	4	El sistema estará disponible 24 horas del día, los siete días de la semana.		programador	Yanet Peña Táramo
	5	La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.		programador	Yanet Peña Táramo
Eficiencia					
	6	Tiempo de respuesta promedio de las peticiones que se realizan al servidor no deberá ser mayor de 3 segundos.		programador	Yanet Peña Táramo
Seguridad					
	7	Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.		programador	Yanet Peña Táramo
	8	El sistema debe mantener en todo momento la seguridad de la información asegurando la autenticidad de la misma.		programador	Yanet Peña Táramo
	9	El sistema debe garantizar la confidencialidad, integridad y disponibilidad de la información que se procese en el sistema.		programador	Yanet Peña Táramo

	10	El control de acceso se establecerá por roles que se le asignarán a los usuarios que interactúen con el sistema.	programador	Yanet Peña Táramo
Soporte				
	11	Soporte para grandes volúmenes de datos y velocidad de procesamiento.	programador	Yanet Peña Táramo
Restricciones de diseño				
	12	El lenguaje de programación es Java.	programador	Yanet Peña Táramo
	13	Los marcos de trabajo a utilizar son Hibernate, Spring y Vaadin.	programador	Yanet Peña Táramo
	14	La herramienta IDE de desarrollo utilizada será Eclipse STS.	programador	Yanet Peña Táramo
	15	La herramienta case utilizada es Visual Paradigm para modelado.	programador	Yanet Peña Táramo
	16	La herramienta gestor de base de datos es el PostgreSQL.	programador	Yanet Peña Táramo
Interfaz				
	17	El sistema tiene que ofrecer una interfaz amigable, fácil de operar.	Diseñador	Yanet Peña Táramo
	18	Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para que los usuarios puedan utilizar el sistema.	Diseñador	Yanet Peña Táramo

Tabla 1: Lista Reserva del Producto

2.5 Diseño de metáforas

A partir de la definición de las funcionalidades descritas en la LRP es posible establecer las historias de usuarios, prototipos del sistema y las tareas ingenieriles que permiten su desarrollo siendo estas las actividades que se realizan y se describen a continuación.

2.5.1 Historias de Usuario

Básicamente una historia es una lista priorizada de requisitos o funcionalidades, descritas usando la terminología del cliente. Estas historias de usuario están documentadas en el expediente de proyecto en la Plantilla de Historias de Usuarios. En el documento se presenta una de ellas.

Historia de Usuario	
Número: HU_1	**Nombre Historia de Usuario:** Autenticación de usuario

Modificación de Historia de Usuario Número: Ninguna	
Usuario: Yanet Peña Táramo	**Iteración Asignada:** Sprint 2
Prioridad en Negocio: *Alta*	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1

Descripción: *El usuario accede al sistema insertando un nombre y una contraseña, el sistema comprobará que estos datos son correctos, permitiendo al usuario acceso a las funcionalidades a las cuales tiene permisos. El usuario que se autentique puede ser especialista o administrador.*

Observaciones:

Prototipo de interfaces:

Tabla 2: Historia de Usuario Autenticar usuario

2.5.2 Tareas Ingenieriles

Partiendo de las historias de usuarios se establecen un grupo de tareas ingenieriles para cada una de ellas, sirviendo de guía para el posterior desarrollo de la solución propuesta.

Algunas de las tareas ingenieriles están relacionadas con todas las historias de usuario preparando el ambiente de implementación, estas son:

Tarea de Ingeniería	
Número Tarea: T_1	**Número Historia de Usuario:** todas
Nombre Tarea: Generación de la base de datos	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 15-enero-2013	**Fecha Fin:** 20-enero-2013
Programador Responsable: Yanet Peña Táramo	
Descripción: Se genera la base de datos a partir del diseño realizado, esta debe implementarse sobre el sistema gestor definido para el desarrollo de la aplicación.	

Tabla 3: Tarea Ingenieril 1

Tarea de Ingeniería	
Número Tarea: T_2	**Número Historia de Usuario:** todas
Nombre Tarea: Montaje del ambiente de desarrollo con la integración de los marcos de trabajo seleccionados para el desarrollo	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 15-enero-2013	**Fecha Fin:** 5-febrero-2013
Programador Responsable: Yanet Peña Táramo	

Descripción: Instalar y configurar los marcos de trabajo definidos para el desarrollo de manera que se pueda comenzar la implementación de la aplicación.

Tabla 4: Tarea Ingenieril 2

Tarea de Ingeniería	
Número Tarea: T_3	**Número Historia de Usuario:** todas
Nombre Tarea: Realizar el mapeo de la base de datos en el marco de trabajo Hibernate.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 5-febrero-2013	**Fecha Fin:** 20-febrero-2013
Programador Responsable: Yanet Peña Táramo	
Descripción: Realizar el mapeo de la base de datos en el marco de trabajo de acceso a datos seleccionado.	

Tabla 5: Tarea Ingenieril 3

Otras están asociadas a cada historia de usuario, en particular, para poder dar cumplimiento a la implementación de las funcionalidades definidas en ellas, por ejemplo las relacionadas con la HU_9 son:

Tarea de Ingeniería	
Número Tarea: T_14	**Número Historia de Usuario:** HU_9
Nombre Tarea: Diseñar la interfaz necesaria para Seleccionar Decisión	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 30-marzo-2013	**Fecha Fin:** 1-abril-2013
Programador Responsable: Yanet Peña Táramo	
Descripción: *Se diseña la interfaz necesaria para la implementación de la funcionalidad relacionada con la historia de usuario, teniendo en cuenta los prototipos diseñados en las HU.*	

Tabla 6: Tarea Ingenieril 14

Tarea de Ingeniería	
Número Tarea: T_15	**Número Historia de Usuario:** HU_9
Nombre Tarea: Implementar Seleccionar Decisión	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 2-abril-2013	**Fecha Fin:** 8-abril 2013
Programador Responsable: Yanet Peña Táramo	
Descripción: *Se implementa la funcionalidad vinculada con la historia de usuario.*	

Tabla 7: Tarea Ingenieril 15

Todas las tareas ingenieriles definidas se encuentran en el expediente de proyecto en la Plantilla de Tareas de Ingeniería.

2.6 Arquitectura

La arquitectura de *software* es el conjunto de técnicas metodológicas desarrolladas con el fin de facilitar la programación. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarse en el desarrollo de *software* dentro de un sistema informático. Establece todos los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema (36). Un patrón arquitectónico es una descripción de un problema particular y recurrente de diseño, que aparece en un contexto específico y presenta una solución demostrada. Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de *software* (37). El patrón arquitectónico seleccionado para la implementación de la

solución es Arquitectura N Capas, en este caso con N igual a 3. El mismo fue seleccionado después ya que se adapta a los requerimientos que debe cumplir el sistema. Este patrón consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de tareas, las cuales se clasifican de acuerdo a un nivel particular de abstracción. En otras palabras las capas dividen el *software*, teniendo cada una de estas un grupo de interfaces públicas que pueden ser invocadas por otras capas o *software*, disponiendo por consiguiente de un conjunto de servicios cohesionados. Del mismo se pueden desatacar atributos tales como la reusabilidad, la portabilidad y la facilidad de pruebas (38).

La arquitectura a la cual se hace referencia es la arquitectura en 3 niveles, en la cual existe un nivel intermediario. Esto significa que la arquitectura generalmente está compartida por:

1. Un cliente, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador Web) para la presentación
2. El servidor de aplicaciones (también denominado *software* intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo
3. El servidor de datos, que proporciona al servidor de aplicaciones los datos que requiere

2.7 Patrones de diseño

A un nivel menor de abstracción de los patrones arquitectónicos se encuentran los patrones de diseño. Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de *software*, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. Tienden a ser independientes de los lenguajes y paradigmas de programación y su aplicación no afecta necesariamente al sistema completo pero si a un subsistema o parte del mismo (37).

Para la definición de las clases del sistema, es importante la revisión de algunos patrones que permiten realizar un diseño adecuado y consistente. Para ello tiene suma importancia la utilización de los patrones GRASP.

2.7.1 Patrones GRASP (Patrones de Software para la Asignación General de Responsabilidad)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Estos patrones se describen a continuación:

- **Experto:** Asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad.

- **Creador:** Asignar a las clases la responsabilidad de crear instancias de otras clases.
- **Controlador:** Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.
- **Bajo Acoplamiento:** Asignar una responsabilidad para mantener bajo acoplamiento. El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.
- **Alta Cohesión:** Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Otros patrones utilizados:

Patrón Fachada: Es un patrón de diseño de tipo Estructural. Proporciona una interfaz unificada de alto nivel para un subsistema, que oculta las interfaces de bajo nivel de las clases que lo implementan. Con esto se consiguen dos objetivos fundamentales: hacer el subsistema más fácil de usar y desacoplar a los clientes de las clases del subsistema (39).

Patrón DAO: El problema que viene a resolver este patrón es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos, etc). De tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el *software* cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento (40).

En la descripción del diagrama de clases se especifica cómo se materializa el uso de estos patrones en el diseño realizado.

2.8 Diagrama de Clases

El propósito de este diagrama es el de representar los objetos fundamentales del sistema, es decir, los que percibe el usuario y con los que espera tratar para completar su tarea en vez de objetos del sistema o de un modelo de programación (41).

El diagrama de clases del diseño representa los métodos y atributos de cada una de las clases del sistema, para mostrar de forma simple la colaboración y las tareas de cada una de ellas en relación al sistema que conforman. Ver Figura 9.

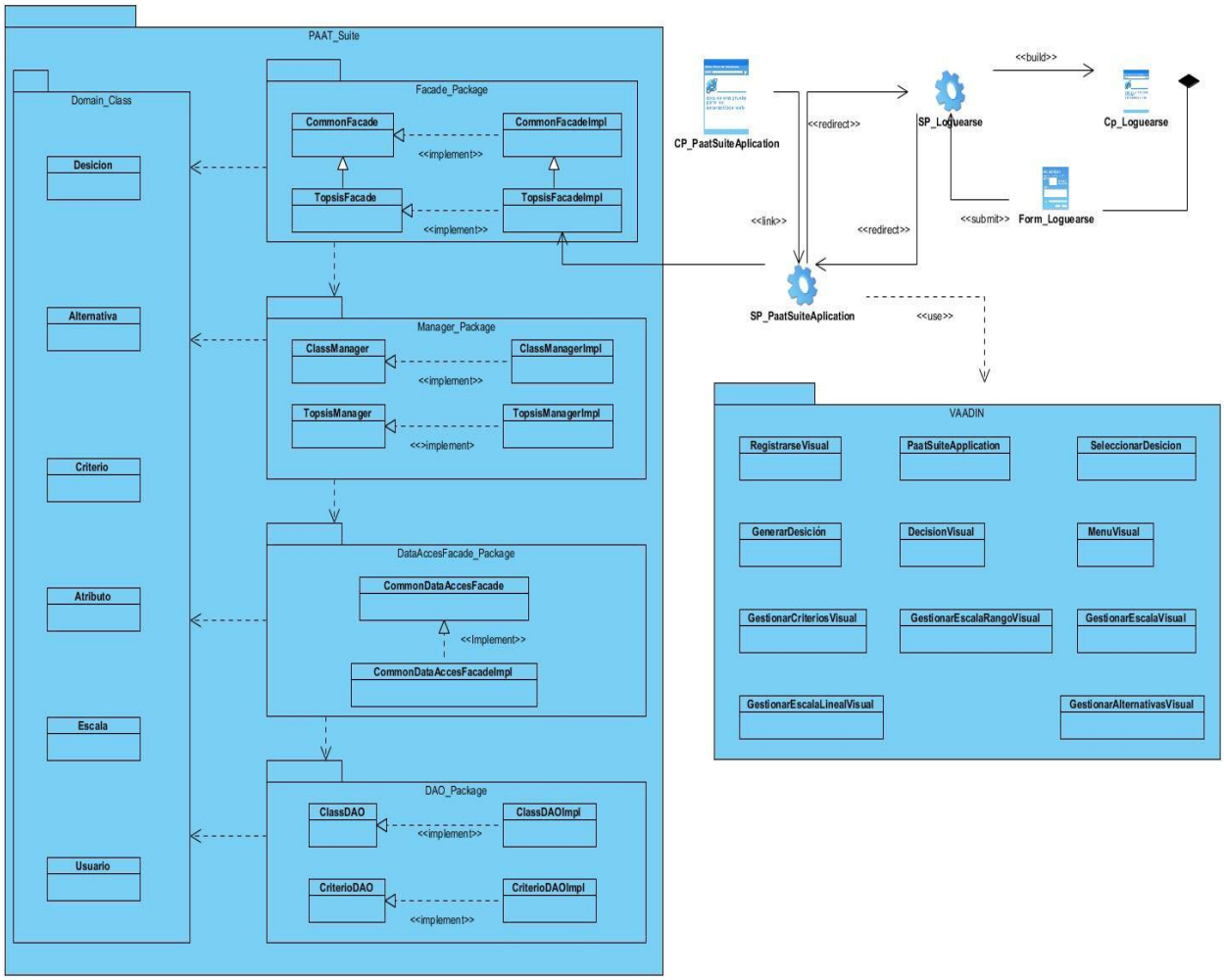

Figura 9: Diagrama de clases

2.9 Modelo de Datos

Es un conjunto de conceptos que nos permiten describir los datos, las relaciones entre ellos, la semántica y las restricciones de consistencia (42).

Existen 3 tipos de modelos de datos (42):

- **Modelos externos o lógicos basados en objetos:** permiten representar los datos que necesita cada usuario con las estructuras propias del lenguaje de programación que se vaya a usar.
- **Modelos globales o lógicos basados en registros:** ayuda a escribir los datos para el conjunto de usuarios.
- **Modelos físicos o de datos:** orientado a la máquina.

Ver Figura 10.

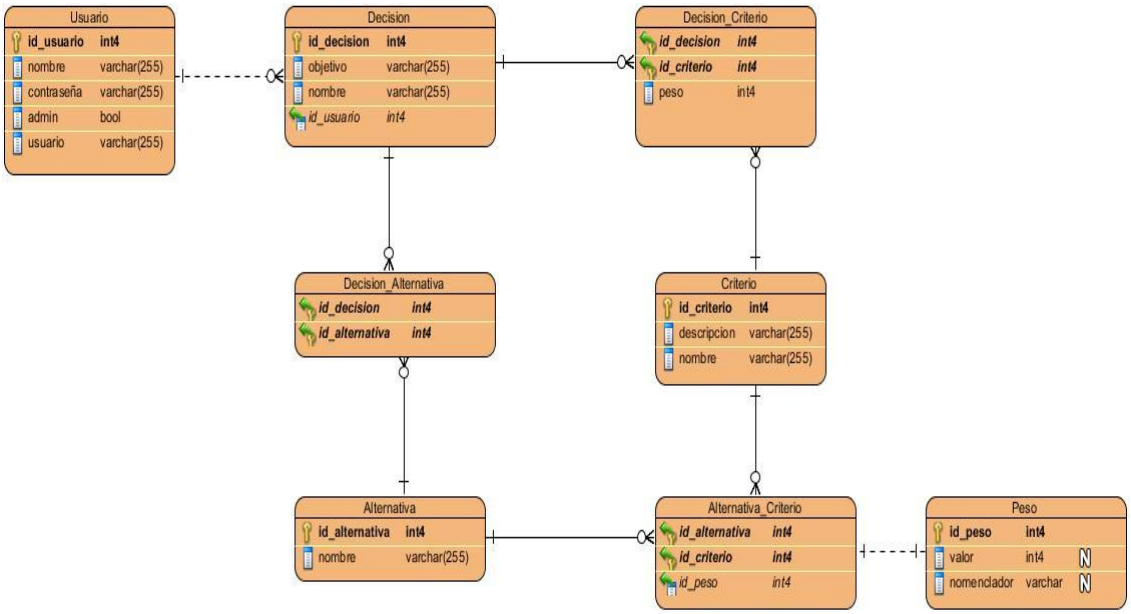

Figura 10: Modelo de datos

Se describe el modelo de datos en el Anexo 5, cada una de las tablas y sus atributos, un ejemplo de la descripción de estos modelos es la tabla alternativa_criterio.

Nombre Tabla: alternativa_criterio		
Descripción: En esta tabla se almacenan los datos correspondientes a la relación entre la tabla decisión y la tabla criterio.		
Atributos	**Tipo**	**Descripción**
id_alternativa	Int	Es la llave foránea de la relación existente entre la tabla alternativa y la tabla criterio.
id_criterio	Int	Es la llave foránea de la relación existente entre la tabla alternativa y la tabla criterio.
Id_peso	Int	Es la llave foránea de la relación existente entre la tabla alternativa_criterio y la tabla peso.

Tabla 8: Tabla alternativa_criterio.

2.10 Diseño del método TOPSIS

Para facilitar la implementación de método TOPSIS se realizó un diseño de clases que se presenta a continuación.

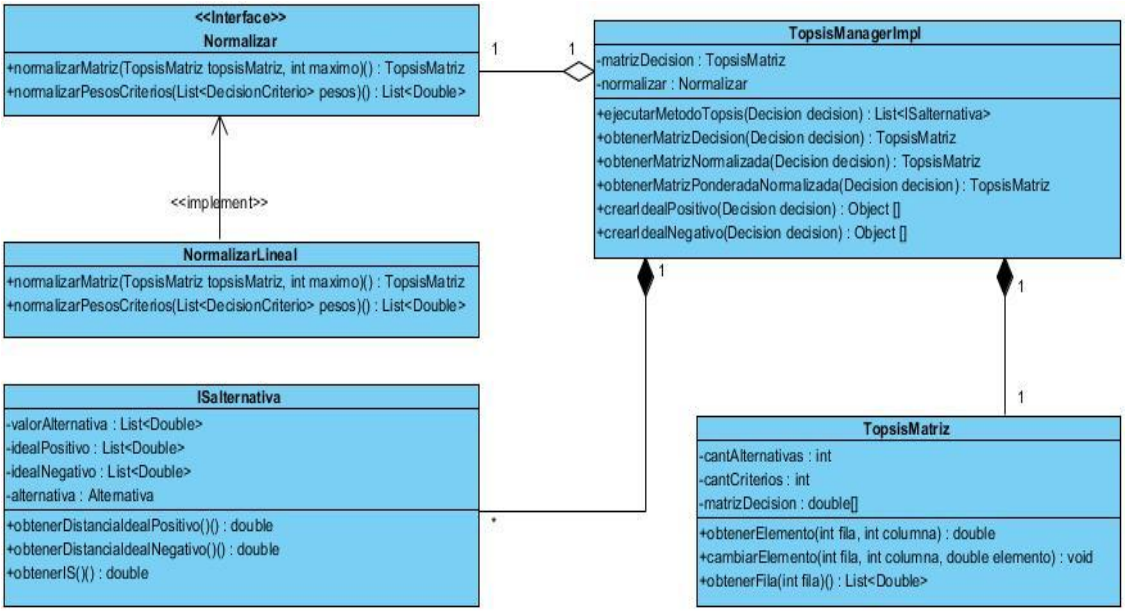

Figura 11: Diseño de clases de TOPSIS

Conclusiones Parciales

En el presente capítulo se realizaron los artefactos Modelo de Usuario, Plantilla de Usuario y la Plantilla de Tareas de ingeniería; preparando las bases para las fases restantes del proceso. Además se realizó el Modelo de Datos y el Diagrama de Clases, representando las entidades relevantes del sistema y las clases involucradas en el desarrollo del mismo así como sus relaciones.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

3.1 Implementación

3.1.1. Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo:

- Dispositivos
- Procesadores
- Memoria

Los nodos se interconectan mediante soportes bidireccionales que pueden a su vez estereotiparse. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos. Las instancias de los nodos pueden contener instancias de ejecución, como instancias de componentes y objetos. El modelo puede mostrar dependencias entre las instancias y sus interfaces, y también modelar la migración de entidades entre nodos u otros contenedores (43).

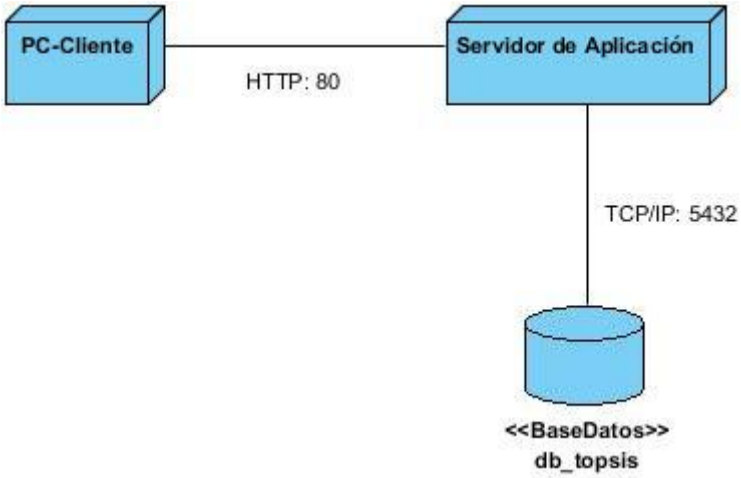

Figura 12: Diagrama de despliegue

3.1.2 Diagrama de componentes

Un diagrama de componentes muestra la organización y dependencias entre los diferentes componentes de un *software*. Son usados para mostrar la implementación estática de un sistema. Incluye el modelado de elementos concretos tales como ejecutables, bibliotecas de

clases, tablas, archivos y documentos. En esencia, son diagramas de clases enfocados en los componentes de un sistema. Son importantes para la visualización, especificación y documentación de sistemas basados en componentes y para implementación de sistemas ejecutables (44).

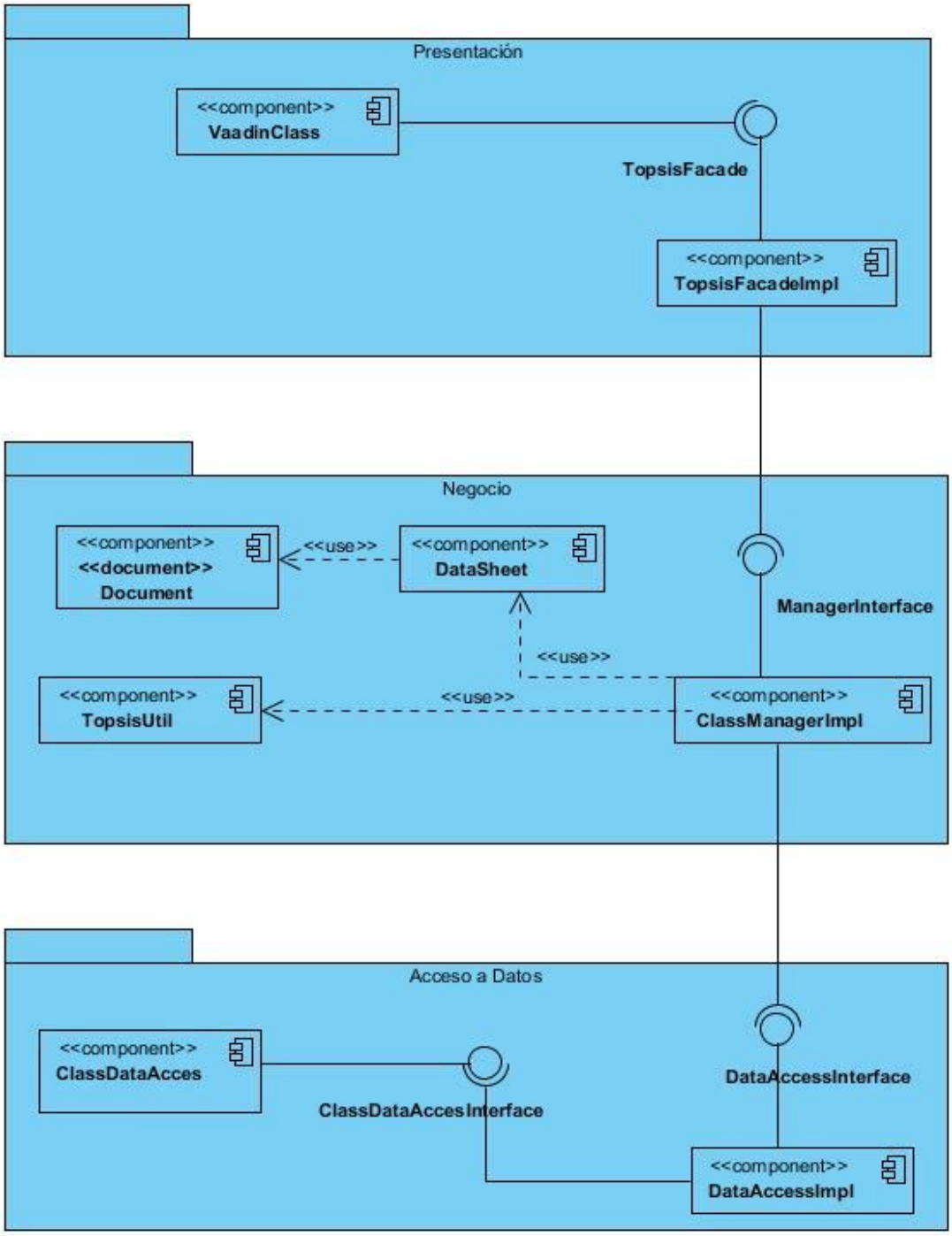

Figura 13: Diagrama de componentes

3.2 Diseño de los casos de prueba

Las pruebas que se le realizan al sistema para la validación de sus funcionalidades son pruebas de caja negra. Las pruebas de caja negra comprueban que las funciones del *software* son

operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Permiten detectar funcionamiento incorrecto o incompleto, errores de interfaz, errores de acceso a estructuras de datos externas, problemas de rendimiento y/o errores de inicio y terminación.

Estas pruebas son llevadas a cabo sobre la interfaz del *software*, actuando sobre ella como una caja negra, proporcionando entradas y estudiando las salidas para ver si son o no las esperadas. Conociendo la función para la que fue diseñado, se hacen pruebas que demuestren que cada función es operativa y al mismo tiempo se buscan errores en cada una (45).

Las pruebas se encuentran en el Anexo 6 de Casos de Prueba de Aceptación. En la Tabla 9 de muestra un ejemplo de los casos de pruebas de aceptación.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-1 **Nombre Historia de Usuario:** Autenticación de usuario.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y el usuario introduce un nombre de usuario y la contraseña y el sistema comprueba que estén registrados en el sistema.
Condiciones de Ejecución: • Que el sistema esté disponible. • Que el usuario y la contraseña estén correctos.
Entrada / Pasos de ejecución: Al introducirse el usuario y la contraseña, se comprueba que los mismos sean correctos.
Resultado Esperado: Que el usuario entre al sistema y tenga acceso a las funcionalidades correspondientes a sus permisos.
Evaluación de la Prueba: Satisfactoria

Tabla 9: Caso de Prueba de Aceptación P-1.

3.3 Resultados de las pruebas funcionales

Para la realización de las pruebas de caja negra a la solución fueron analizadas todas las funcionalidades referentes al módulo TOPSIS. Se hizo una descripción de cada una de las pruebas. Se realizaron un total de tres iteraciones para poder alcanzar los resultados satisfactorios desde el punto de vista funcional, atendiendo al correcto comportamiento del mismo ante diferentes situaciones (entradas válidas y no válidas). A continuación se realiza una descripción de las no conformidades detectadas durante la ejecución de las pruebas de caja negra. En las mismas se encontraron mayormente errores de faltas ortográficas, de validaciones, y errores en las interfaces.

En la primera iteración se encontraron un total de 8 no conformidades (NC):

- Faltas ortográficas 2 NC
- Errores en las interfaces 4 NC
- Errores de validación 2 NC

En la segunda iteración se encontraron un total de 5 no conformidades (NC):

- Faltas ortográficas 1 NC
- Errores en las interfaces 3 NC
- Errores de validación 1 NC

En la tercera iteración no se encontraron no conformidades (NC):

- Faltas ortográficas 0 NC
- Errores en las interfaces 0 NC
- Errores de validación 0 NC

La Figura 14 muestra el resultado de estas pruebas según las NC detectadas durante su ejecución:

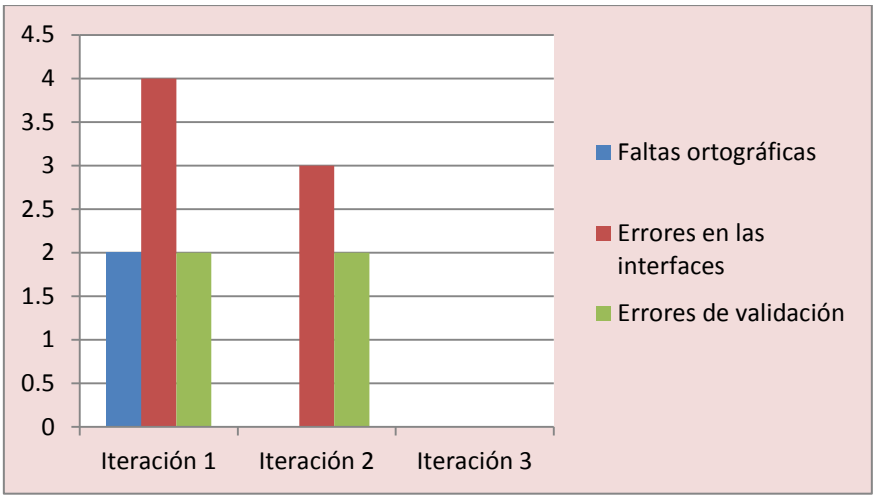

Figura 14: Errores detectados en las pruebas funcionales

3.4 Resultados de la aplicación de las métricas para la validación

Las Métricas de diseño permiten medir de forma cuantitativa la calidad de los atributos internos del *software*. Esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema (46). La solución informática incluirá estas pruebas haciendo uso de las siguientes métricas:

Tamaño Operacional de la Clase (TOC): Está dada por el número de funcionalidades asignadas a cada una de las clases del sistema informático propuesto en la investigación evaluando los siguientes atributos de calidad:

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.

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:

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.

3.4.1 Resultados Métrica TOC

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

Tabla 10: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad Responsabilidad, Complejidad de implementación y Reutilización.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
TopsisFacade	12	Media	Media	Media
TopsisFacadeImpl	12	Media	Media	Media
TopsisManager	10	Media	Media	Media
TopsisManagerImpl	12	Media	Media	Media
TopsisMatriz	5	Baja	Baja	Alta
ISalternativa	7	Baja	Baja	Alta
Normalizar	2	Baja	Baja	Alta
NormalizarLineal	2	Baja	Baja	Alta
CommonDataAccessFacade	14	Media	Media	Media
CommonDataAccessFacadeImpl	14	Media	Media	Media

Tabla 11: Evaluación de las clases del sistema mediante la métrica TOC

En la Figura 15 se muestra una gráfica con agrupaciones por intervalos de la cantidad de clases según la cantidad de procedimientos:

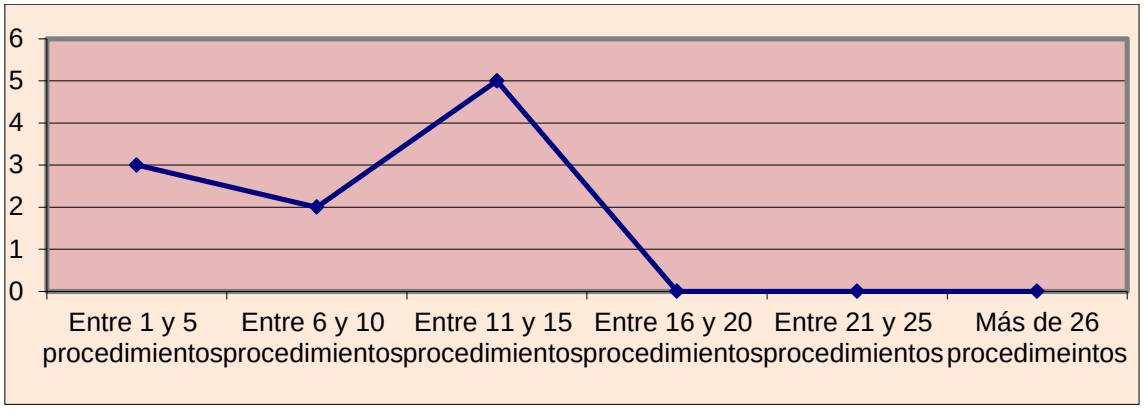

Figura 15: Representación de la cantidad de clases agrupadas en intervalos según la cantidad de procedimientos.

A continuación las gráficas que representan los resultados obtenidos en cada atributo de calidad.

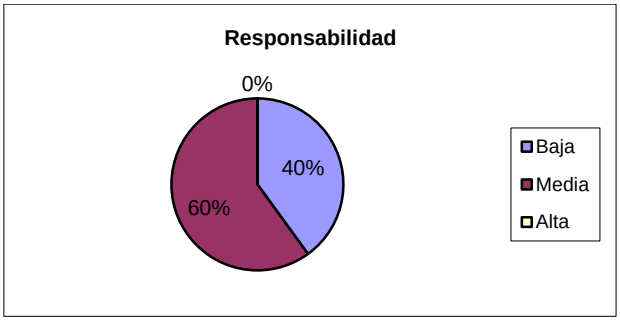

Figura 16: Representación en por ciento (%) de los resultados obtenidos en el atributo Responsabilidad.

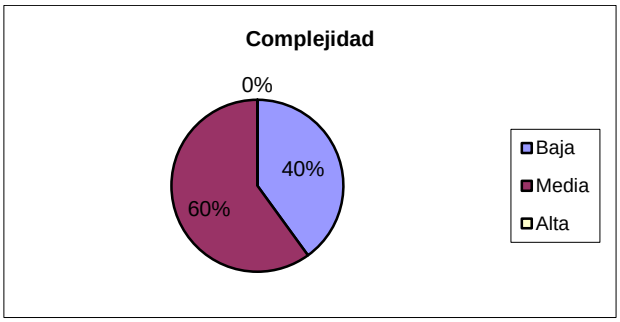

Figura 17: Representación en por ciento (%) de los resultados obtenidos en el atributo Complejidad de implementación.

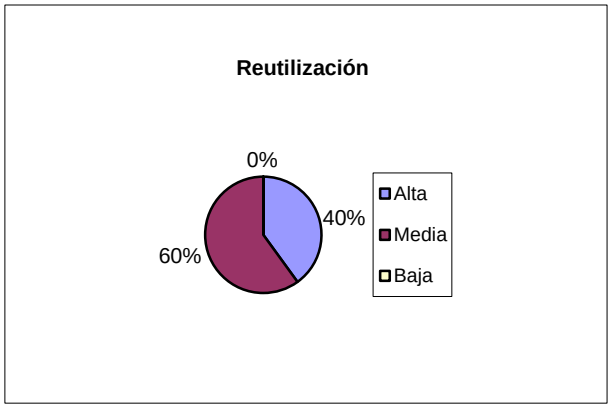

Figura 18: Representación en por ciento (%) de los resultados obtenidos en el atributo Reutilización.

3.4.2 Resultados Métrica RC

	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de Mantenimiento	Baja	<= Promedio
	Media	Entre Promedio. y 2* Promedio.
	Alta	> 2* Promedio.
Reutilización	Baja	>2* Promedio.
	Media	Entre Promedio. y 2* Promedio.
	Alta	<= Promedio.
Cantidad de Pruebas	Baja	<= Promedio.
	Media	Entre Promedio. y 2* Promedio.
	Alta	> 2* Promedio.

Tabla 12: Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad Acoplamiento, Complejidad del mantenimiento, Cantidad de pruebas y Reutilización.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
TopsisFacade	0	Ninguno	Baja	Alta	Baja
TopsisFacadeImpl	2	Medio	Alta	Baja	Alta
TopsisManager	0	Ninguno	Baja	Alta	Baja
TopsisManagerImpl	4	Alto	Alta	Baja	Alta
TopsisMatriz	0	Ninguno	Baja	Alta	Baja
ISalternativa	0	Ninguno	Baja	Alta	Baja
Normalizar	0	Ninguno	Baja	Alta	Baja
NormalizarLineal	1	Bajo	Media	Media	Media
CommonDataAccessFacade	0	Ninguno	Baja	Alta	Baja
CommonDataAccessFacadeImpl	1	Bajo	Media	Media	Media

Tabla 13: Clases del Sistema evaluadas en los atributos de calidad según la métrica RC.

A continuación se muestra una gráfica con agrupaciones por intervalos de la cantidad de clases

según las dependencias entre ellas.

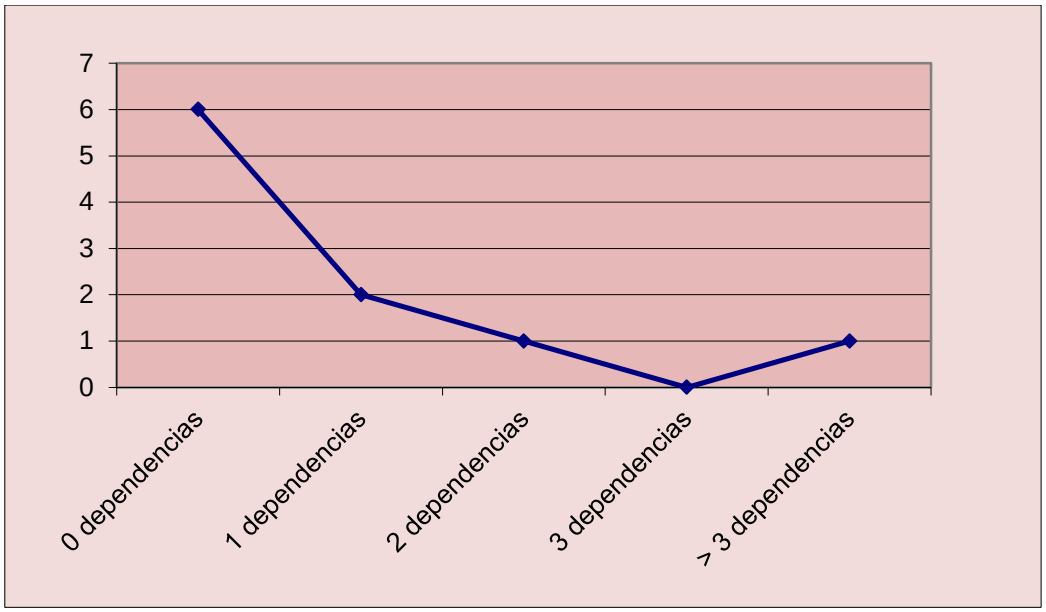

Figura 19: Intervalos de las clases agrupadas según las dependencias entre ellas.

La Figura 20 muestra los valores en por ciento obtenidos al evaluar el diseño en el atributo Acoplamiento.

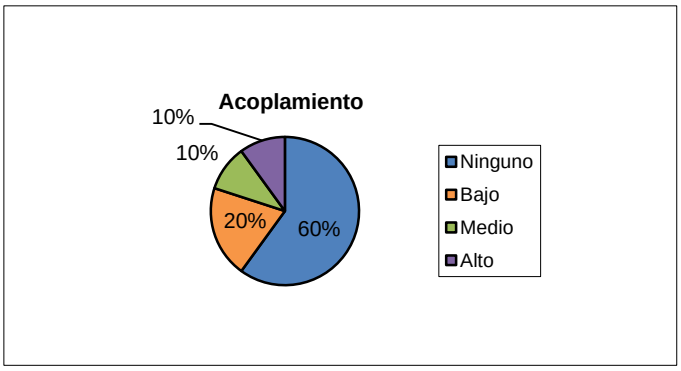

Figura 20: Representación en porcientos (%) de los atributos obtenidos en el atributo Acoplamiento.

La Figura 21 muestra los valores en por ciento obtenidos al evaluar el diseño en el atributo Mantenimiento.

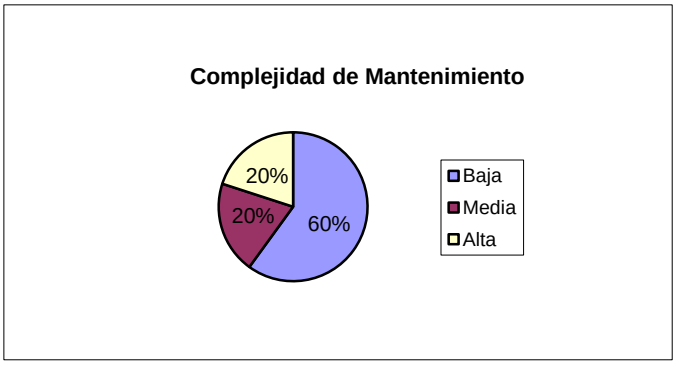

Figura 21: Representación en porcientos (%) de los atributos obtenidos en el atributo Complejidad de Mantenimiento.

La Figura 22 muestra los valores en por ciento obtenidos al evaluar el diseño en el atributo Cantidad de Pruebas.

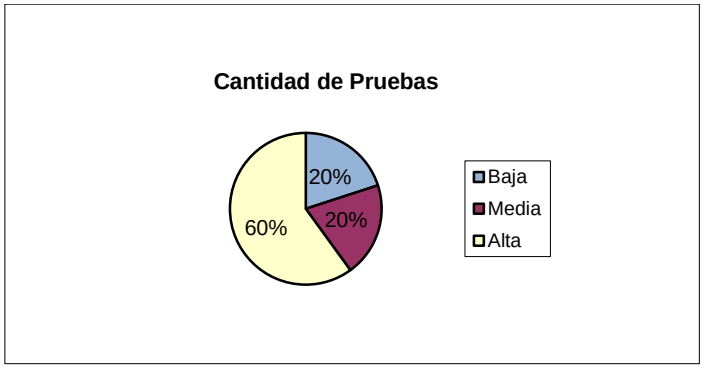

Figura 22: Representación en porcientos (%) de los atributos obtenidos en el atributo Cantidad de Pruebas.

La Figura 23 muestra los valores en por ciento obtenidos al evaluar el diseño en el atributo Reutilización.

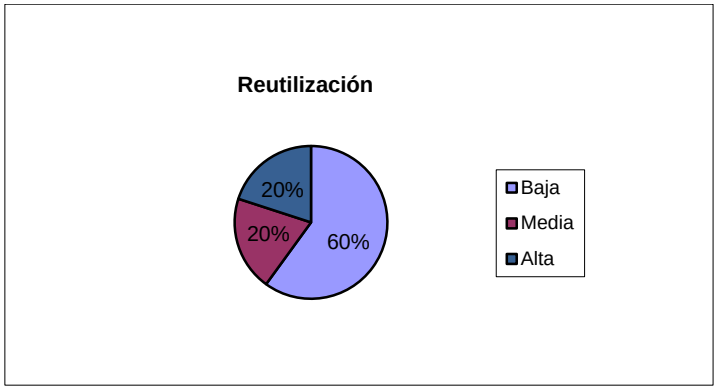

Figura 23: Representación en porcientos (%) de los atributos obtenidos en el atributo Reutilización.

Al aplicar las métricas de diseño al sistema se puede concluir de acuerdo a la métrica TOC que las clases del sistema poseen media responsabilidad, media complejidad, media reutilización y de acuerdo a la métrica RC, bajo acoplamiento, baja complejidad de mantenimiento, alta cantidad de pruebas y baja reutilización.

3.5 Validación de las variables de la investigación

Para hacer la validación de las variables del problema, se realizó un caso de estudio aplicando el método TOPSIS de forma manual y utilizando el *software* MultiDecisionPAAT, y se compararon los resultados de cada una de las variables en ambos casos. Las variables que se evaluaron fueron: demora en el proceso de decisión, susceptibilidad a errores y complejidad de la confección. Los resultados se muestran en las siguientes gráficas:

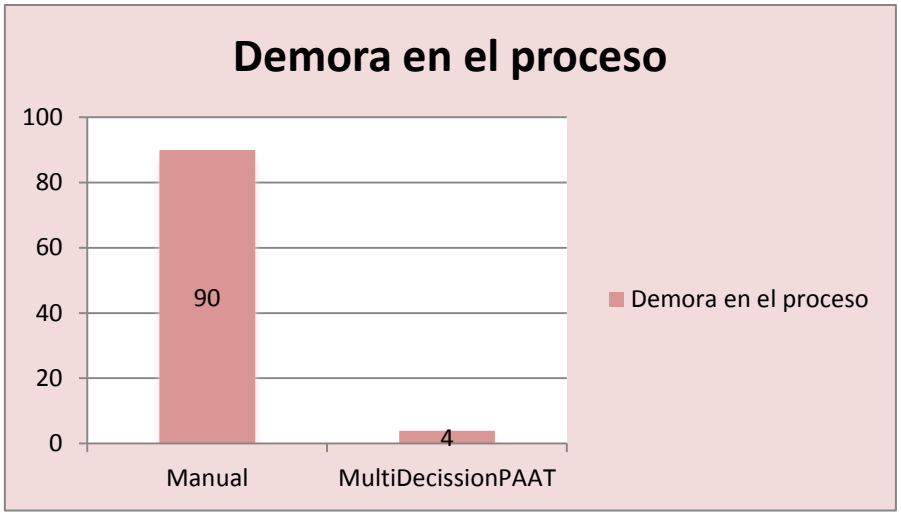

Figura 24: Demora en el proceso de decisión utilizando el método TOPSIS.

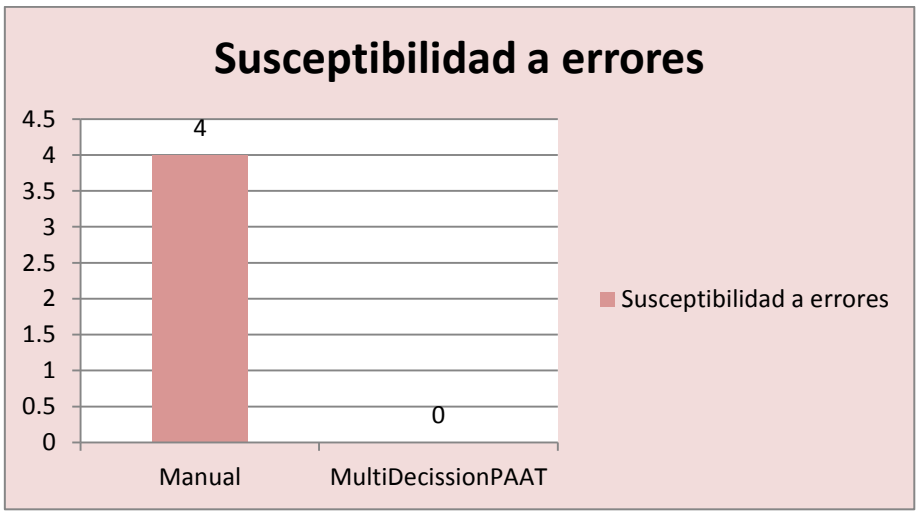

Figura 25: Susceptibilidad a errores utilizando el método TOPSIS.

Con el análisis de las gráficas se puede afirmar que el sistema MultiDecisionPAAT ofrece los mejores resultados para las variables analizadas en comparación con la aplicación de método de forma manual, en cuanto a la susceptibilidad a errores y la demora en el proceso de decisión.

Conclusiones parciales

En el presente capítulo se realizó el diagrama de componentes y de despliegue, mostrando una vista de la aplicación a nivel de componentes y su distribución. Se diseñaron los casos de pruebas para realizar pruebas funcionales a la aplicación y se aplicaron las métricas para la validación de diseño Tamaño Operacional de Clases (TOC) y Relación entre Clases (RC) obteniéndose resultados satisfactorios. Con las pruebas y validaciones realizadas se puede concluir que la herramienta desarrollada cumple con las especificaciones y requisitos definidos por los clientes en la etapa de concepción del sistema.

CONCLUSIONES

Durante el desarrollo de la presente investigación, se dio cumplimiento a los objetivos planteados, teniendo como principales conclusiones las siguientes:

El uso de la metodología SXP y la selección del conjunto de herramientas y tecnologías adecuadas facilitaron el desarrollo del sistema desarrollado. Además se definió una variante del método TOPSIS permitiendo la integración de este a un sistema de apoyo a la toma de decisión. Dentro del flujo de trabajo de la metodología se realizaron los artefactos Modelo de Usuario, Plantilla de Usuario, Plantilla de Tareas de Ingeniería, Modelo de Datos y Diagrama de Clases, Diagrama de Componentes y Diagrama de Despliegue; los cuales permitieron el análisis, diseño e implementación del método TOPSIS. Se realizó la validación del sistema implementado mediante pruebas de funcionalidad y la aplicación de métricas. Esta validación arrojó como resultados que la aplicación desarrollada satisface los requerimientos especificados en las historias de usuario, así como el diseño de las clases e implementación de acuerdo a las métricas aplicadas.

RECOMENDACIONES

Para una futura versión del sistema se recomienda que el sistema brinde la posibilidad de generar reportes en varios formatos y que el sistema permita registrar los expertos que evalúan las alternativas. Además se recomienda implementar las variantes difusas del método.

BIBLIOGRAFÍA

1. **Wei, Jianli.** *TOPSIS Method for Multiple Attribute Decision Making with Incomplete Weight Information in Linguistic Setting.* 2010.
2. **Lopez de Luise, Daniela, Aguirre, Federico and Curlat, Maria Elena .** *Sistema de Soporte de Decisiones para inversiones en transporte.* Buenos Aires : s.n., 2010.
3. **Lamata Jiménez, Teresa.** *Métodos para la comparación de alternativas mediante un Sistema de Ayuda a la Decisión (S.A.D.) y "Soft Computing".* Cartagena : s.n., 2009.
4. **Ercole, Raúl Alberto, Alberto, Catalina Lucía and Carignano, Claudia Etna.** *DECISIONES CON TOPSIS DIFUSO.* Bahía Blanca : s.n., 2011.
5. **Maurtua Ollaguez, Diego Edher.** *Criterios de Selección de Personal mediante el uso del proceso de análisis jerárquico.* Lima : s.n., 2006.
6. **Perez Mackeprang, Carlos.** *EVALUACIÓN DE LOS SISTEMAS DE INVESTIGACIÓN CIENTÍFICA Y DESARROLLO EXPERIMENTAL (I+D) EN PAISES IBEROAMERICANOS.* ARGENTINA : s.n., 2003.
7. Excel VBA Software. [Online] enero 25, 2013. http://www.xls-soft.com/31/download-topsis-solver-2012/.
8. **García Cascales, Maria del Socorro.** *Métodos para la comparacion de alternativas mediantes un Sistema de Ayuda a la Decisión (SAD) y "Soft Computing".* Cartagena : s.n., 2009.
9. **David.** *Metodologías de desarrollo de software.* 2008.
10. **Palacio, Juan.** *Gestión de proyectos* . s.l. : ScrumManager, 2008.
11. **Pérez Carrillo, Isaías, Pérez González, Rodrigo and Rodríguez Martín, David Aureliano.** *Metodologías de desarrollo de software.* 2008.
12. **Sommerville, I.** *Ingeniería de Software.* s.l. : Pearson Educación, 2002.
13. **Figueroa, R.G, Cabrera, Armando and Solís, Camilo.** *Metodologías tradicionales vs metodologías ágiles.*
14. Ecured. [Online] [Cited: enero 27, 2013.] http://www.ecured.cu/index.php/Metodologia_Agil_de_Desarrollo_SXP.
15. **Peñalver, G., Meneses, A. and García, S.** *SXP, para el desarrollo de software libre.* Antofagasta : s.n., 2010.
16. **INSTITUTO NACIONAL DE ESTADISTICA E INFORMATICA.** *Herramientas Case.* Peru : Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática, 1999. 875-99-OI-OTDETI-INEI.
17. **Visual Paradigm International.** Visual Paradigm International. *Sitio web de Visual Paradigm.* [Online] [Cited: Febrero 23, 2012.] http://www.visual-paradigm.com/product/vpuml/.
18. DocIRS. [Online] 2002. [Cited: Enero 22, 2013.] http://www.docirs.cl/uml.htm.
19. The Unified Modeling Language. [Online] 1997. [Cited: Enero 22, 2013.] http://www.uml.org.
20. **Holzner, Steven.** *La Bilbia de Java 2.* España : Anaya Multimedia, 2003. 84-415-1037-7.
21. Enciclopedia Libre Universal en Español. [Online] febrero 12, 2013. http://enciclopedia.us.es/index.php/Java_(lenguaje_de_programaci%C3%B3n).
22. **Walls, Craig and Breidenbach, Ryan.** *Spring in action.* s.l. : Manning Publications Co., 2008. 1-933988-13-4.
23. **Johnsonet al., Rod.** *Professional Java Development with the Spring Framework.* s.l. : John Wiley & Sons, 2005. 0764574833.
24. JBoss Community. [Online] enero 10, 2013. http://www.hibernate.org/.
25. *Exploración del Framework Vaadin.* Bucaramanga : s.n., 2011.

26. **S. Mullins, Craig.** *Database Administration: The Complete Guide to Practices and Procedures.* s.l. : Addison Wesley, 2002. 0-201-74129-6.
27. **The PostgreSQL Global Development Group.** *PostgreSQL 9.0.1 Documentation.* California : Regents of the University of California, 2010.
28. **Riggs, Simon and Krosing, Hannu.** *PostgreSQL 9 Administration Cookbook.* Birmingham : Packt Publishing Ltd., 2010. 978-1-849510-28-8.
29. EcuRed. [Online] 2011. [Cited: Enero 24, 2013.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.
30. Introducción al Software Libre.Capítulo 9. [Online] [Cited: febrero 2, 2013.] http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html.
31. **Mateu, Carles.** *Desarrollo de aplicaciones web.* Barcelona : Eureca Media, SL, 2004. 84-9788-118-4.
32. **Java, Consultor.** Consultor Java. [Online] [Cited: Febrero 19, 2012.] http://www.consultoriajava.com/tools/tomcat.shtml.
33. Uso Práctico de CVS para control de versiones. [Online] 2003. [Cited: marzo 3, 2013.] http://www.tuxpan.com/fcatrin/files/cvs.html.
34. Inter Grafic Designs. [Online] [Cited: enero 10, 2013.] http://www.intergraphicdesigns.com/blog/2008/09/29/resumen-sobre-ventajas-de-utilizar-subversion/.
35. **S. Pressman, Roger.** *Ingeniería del Software. Un enfoque práctico.* s.l. : McGraw Hill Higher Education, 2002. 9701054733.
36. **Fernandez Lanvin, Daniel.** *Definición de una arquitectura software para el diseño de aplicaciones web.* Oviedo : s.n., 2009.
37. **Buschmann, Frank, et al., et al.** *Pattern – Oriented Software Architecture. A System of Patterns.* Inglaterra : John Wiley & Sons, Inc. 978-0471958697.
38. **Bachmann, Felix, et al., et al.** *Software Architecture Documentation in Practice: Documenting Architectural Layers.* Pittsburgh : Carnegie Mellon University, 2000.
39. **Nieto Diego, Javier and Ramos Fernández, Pablo.** *El Patrón Fachada: Universidad de Salamanca.*
40. **Lago, R. .** Patrón "Data Access Object". [Online] 2007. http://www.proactiva-calidad.com/java/patrones/DAO.html..
41. **Zapata, María Antonia.** *Diseño estructural:Diagrama de Clases.* 2006.
42. Aurea. [Online] [Cited: marzo 5, 2013.] http://aurea.es/wp-content/uploads/modelodedatos.pdf.
43. **Campderrich Falgueras, Benet.** *Ingeniería del Software.* 2003.
44. **Booch, Grady, Rumbaugh, James and Jacobson, Ivar.** *The Unified Modeling Language User Guide.* Massachusetts : Addison-Wesley Longman Inc., 1998. 0-201-57168-4.
45. **Rojas, Johanna and Barrios, Emilio.** Universidad Distrital Francisco José de Caldas. [Online] 2007. [Cited: febrero 3, 2013.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node26.html.
46. Ecured.Métricas del diseño. [Online] 2012. http://www.ecured.cu/index.php/M%C3%A9trica_de_dise%C3%B1o.
47. *Enfoque alterno de la Técnica TOPSIS al utilizar la distancia de Mahalanobis.* **Villanueva Ponce, Rodrigo and García Alcaraz, Jorge Luis.** 2, Ciudad Juárez : s.n., 2012, Vol. 4.

ANEXOS

Anexo 1 Plantilla de Concepción Inicial del Sistema

1. **Polo productivo**

   Macro proyecto de investigación: Sistema para la ayuda a la toma de decisión multicriterio.

2. **Clasificación del proyecto**

Desarrollo de aplicación.

3. **Tipo de proyecto**

Nacional

4. **Resumen:**

Este documento además de reflejar la visión general del producto a implementar, también recoge los diferentes roles que intervendrán en el desarrollo del *software*, así como las responsabilidades que tendrán en dicho proceso. Se documenta el tipo de proyecto al que pertenece así como la especificación del Polo Productivo y su clasificación. Se recoge además cuales herramientas serán utilizadas para el desarrollo de la aplicación, el alcance que va a tener, una descripción de los involucrados en el negocio, cuales son los motivos de la necesidad del desarrollo del *software* y la propuesta de solución.

5. **Surgimiento**

   La necesidad de realizar el sistema surge luego de haber realizado un análisis en las áreas donde la toma de decisión juega un papel fundamental. En estos lugares donde el decisor tiene que analizar una gran cantidad de información y realizar el proceso de forma manual. Un gran porciento de las veces este se realizaba bajo condiciones erróneas, debido a que el mismo se hace de forma manual y en ocasiones donde el cúmulo de información era grande se tornaba impracticable. Esto traía como consecuencia que el proceso lejos de brindar criterios eficientes y confiables, se volvía dudoso, lo que provocaba que en muchas ocasiones no se realizara el mismo.

6. **¿Qué es?**

El sistema es un sistema de apoyo a la toma de decisión multicriterio que podrá adaptarse y ser utilizado en diferentes problemas donde sea necesario la tomar de decisiones y se maneje grandes volúmenes de datos.

7. **Metodología a utilizar**

SXP es la metodología a utilizar que consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Basada completamente en los valores y principios de las metodologías ágiles expuestos en el Manifiesto Ágil. La creación de SXP con la unión de las

metodologías XP y SCRUM se centra principalmente en que con la utilización de SCRUM para la gestión, se logra una correcta planificación y organización; mientras que XP respalda con sus prácticas todo el proceso de desarrollo y de esta forma se obtiene un proceso de *software* completo.

8. Involucrados

Hermes Miguel Velázquez Domínguez y Lizandra Arza Pérez expertos en la metodología y el modelo.

9. Roles.

Rol	Responsabilidad	Nombre
Líder del Proyecto	Asegurar que el proyecto se está llevando a cabo de acuerdo con las prácticas y que todo funcione según lo planeado. Su principal trabajo es remover los impedimentos y definir, así como reducir los riesgos del producto.	Lizandra Arza Pérez
Gerente	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la definición de objetivos y requerimientos, así como en la selección de los miembros del Equipo del Proyecto. Tiene la responsabilidad de controlar el progreso del *software* y trabaja junto con el Líder de Proyecto en la reducción de la Lista de reserva del producto, así como en la de riesgos.	Lizandra Arza Pérez
Cliente	El cliente contribuye a definir las historias de usuario y los casos de prueba de aceptación, para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio y participa en la concepción inicial del sistema.	Lizandra Arza Pérez y Hermes Miguel Velázquez Hernández.
Programadores	El programador define las tareas de ingeniería y produce el código del sistema. Además selecciona el estándar de programación a utilizar, controlando incluso la gestión recambios. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.	Yanet Peña Táramo
Analista	Escribe la concepción del sistema y las historias de usuario. Crea el Modelo de historia de usuario del negocio y la LRP. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.	Yanet Peña Táramo
Diseñadores	Son los encargados del diseño del sistema, así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.	Yanet Peña Táramo
Encargado de Pruebas	Escribe los casos de prueba de aceptación. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Yanet Peña Táramo
Arquitecto	Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas.	Yanet Peña Táramo
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas, además aportan ideas y experiencias para el beneficio del sistema en desarrollo.	Iskael Díaz Márquez

Tabla 14: Roles involucrados en el proceso de desarrollo del *software*.

10. Misión

Brindar los criterios decisión en ambiente s donde la confección manual del mismo se torna impracticable y es susceptible a errores.

11. Alcance

El desarrollo de las funcionalidades necesarias para hacer el análisis de la información y dar los criterios para la toma de decisión.

Anexo 2 Modelo de historias de usuarios del negocio

1. Actores del negocio

Actor	Descripción
Especialista	Este personal se beneficia con el negocio al ser ellos los interesados de que la selección se realice los más acertada posible, permitiendo así que se realice de una forma ágil el trabajo de selección.

Tabla 15: Actores del negocio.

2. Trabajadores del negocio

Trabajador	Descripción
Administrador del sistema	Será el encargado de realizar toda la gestión de los usuarios. Además podrá acceder a todas las funcionalidades del sistema.

Tabla 16: Trabajadores del negocio.

3. Diagrama de Historia de Usuario

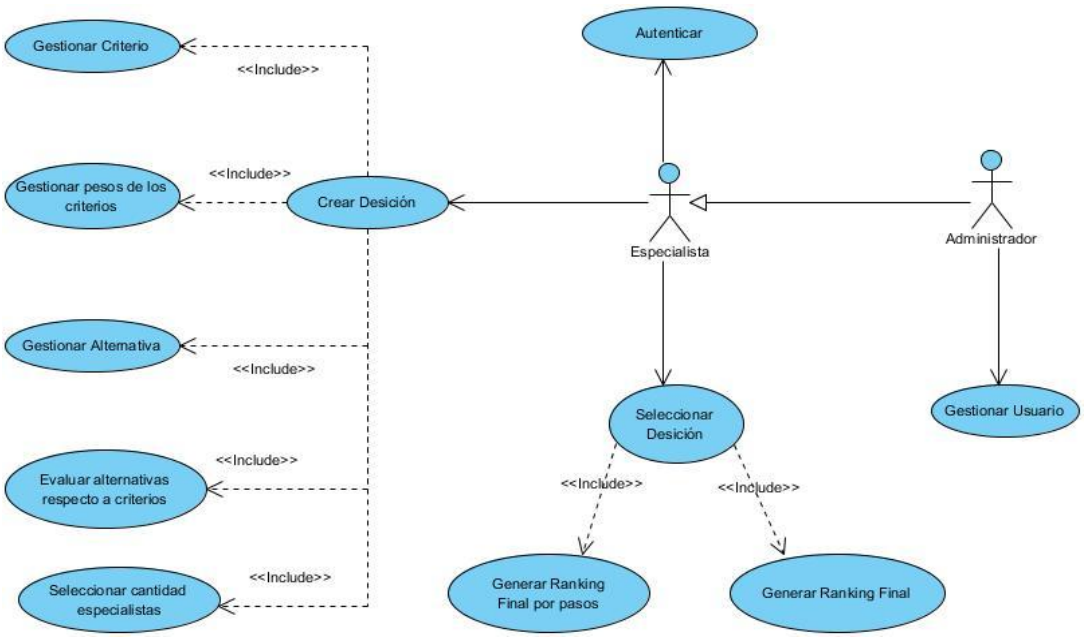

Figura 26: Historias de Usuario del Negocio

Anexo 3 Descripción del Modelo de Datos

Nombre Tabla: decision		
Descripción: En esta tabla se almacenan los datos correspondientes a una decisión.		
Atributos	**Tipo**	**Descripción**
id_decision	String	Es el identificador de la decisión. Es la llave primaria de la tabla.
objetivo	String	Describe el objetivo de la decisión. Describe la decisión
nombre	String	Es el nombre de la decisión.
id_usuario	Int	Es el identificador del usuario que toma la decisión.

Tabla 17: Tabla decisión.

Nombre Tabla: alternativa		
Descripción: En esta tabla se almacenan los datos correspondientes a una alternativa		
Atributos	**Tipo**	**Descripción**
id_alternativa	String	Es el identificador de la alternativa. Es la llave primaria de la tabla.
nombre	String	Es el nombre de la alternativa

Tabla 18: Tabla alternativa.

Nombre Tabla: criterio		
Descripción: En esta tabla se almacenan los datos correspondientes a un criterio.		
Atributos	**Tipo**	**Descripción**
id_criterio	Int	Es el identificador del criterio. Es la llave primaria de la tabla.
descripción	String	Describe en que consiste el criterio.
nombre	String	Es el nombre del criterio.

Tabla 19: Tabla criterio.

Nombre Tabla: peso		
Descripción: En esta tabla se almacenan los datos correspondientes a un peso.		
Atributos	**Tipo**	**Descripción**
id_peso	Int	Es el identificador del peso. Es la llave primaria de la tabla.
valor	Int	Es el valor que tiene el peso.
nomenclador	String	Es el nomenclador del peso.

Tabla 20: Tabla peso.

Nombre Tabla: decision_alternativa		
Descripción: En esta tabla se almacenan los datos correspondientes a un peso.		
Atributos	**Tipo**	**Descripción**
id_decision	Int	Es la llave foránea de la relación existente entre la tabla decisión y la tabla alternativa.
Id_alternativa	Int	Es la llave foránea de la relación existente entre la tabla decisión y la tabla alternativa.

Tabla 21: Tabla decision_alternativa.

Nombre Tabla: decision_criterio		
Descripción: En esta tabla se almacenan los datos correspondientes a la relación la tabla decisión y la tabla criterio.		
Atributos	**Tipo**	**Descripción**
id_decision	Int	Es la llave foránea de la relación existente entre la tabla decisión y la tabla criterio.
id_criterio	Int	Es la llave foránea de la relación existente entre la tabla decisión y la tabla criterio.
peso	Int	Valor del atributo de la alternativa.

Tabla 22: Tabla decision_criterio.

Nombre Tabla: alternativa_criterio		
Descripción: En esta tabla se almacenan los datos correspondientes a la relación entre la tabla decisión y la tabla criterio.		
Atributos	**Tipo**	**Descripción**
id_alternativa	Int	Es la llave foránea de la relación existente entre la tabla alternativa y la tabla

		criterio.
id_criterio	Int	Es la llave foránea de la relación existente entre la tabla alternativa y la tabla criterio.
Id_peso	Int	Es la llave foránea de la relación existente entre la tabla alternativa_criterio y la tabla peso.

Tabla 23: Tabla alternativa_criterio.

Nombre Tabla: usuario		
Descripción: En esta tabla se almacenan los datos correspondientes a la relación entre la tabla decisión y la tabla criterio.		
Atributos	**Tipo**	**Descripción**
nombre	String	Es la llave foránea de la relación existente entre la tabla alternativa y la tabla criterio.
contrasenna	String	Es la llave foránea de la relación existente entre la tabla alternativa y la tabla criterio.
admin	Boolean	Es la llave foránea de la relación existente entre la tabla alternativa_criterio y la tabla peso.
usuario	String	

Tabla 24: Tabla usuario.

Anexo 4 Descripción de las clases

Nombre Clase: TopsisManagerImpl	
Tipo de clase: Controladora	
Atributos	**Tipo**
matrizDecision	TopsisMatriz
normalizar	Normalizar
alternativas	List<DecisionAlternativa>
criterios	List<DecisionCriterio>
dataAccesFacade	CommonDataAccesFacade
Nombre Responsabilidad	**Descripción**
ejecutarMetodoTopsis	Estable el ranking de alternativas, comenzando por la mejor.
obtenerMatrizDecision	Crea la matriz de decisión a utilizar en la ejecución del método.
obtenerMatrizNormalizada	Normaliza la matriz de decisión.
obtenerMatrizPonderadaNormalizada	Pondera con los pesos de los criterios la matriz normalizada.
crearIdealPositivo	Crea una alternativa que representa el ideal positivo de la decisión.
crearIdealNegativo	Crea una alternativa que representa el ideal negativo de la decisión.

Tabla 25: Clase TopsisManagerImpl.

Nombre Clase: Decision	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
nombre	String
objetivo	String
usuario	Usuario

Tabla 26: Clase Decision.

Nombre Clase: Alternativa	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
nombre	String

Tabla 27: Clase Alternativa.

Nombre Clase: Criterio	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
nombre	String
descripcion	String

Tabla 28: Clase Criterio.

Nombre Clase: Peso	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
valor	Integer
nomenclador	String

Tabla 29: Clase Peso.

Nombre Clase: Usuario	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
nombre	String
nombreUsuario	String
contrasenna	String
administrador	boolean

Tabla 30: Clase Usuario.

Nombre Clase: AlternativaCriterio	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
valor	Integer
alternativa	Alternativa
criterio	Criterio

Tabla 31: Clase AlternativaCriterio.

Nombre Clase: DecisionCriterio	
Tipo de clase: Entidad	
Atributos	
	Tipo
id	Integer
peso	Integer
decision	Decision
criterio	Criterio

Tabla 32: Clase DecisionCriterio.

Nombre Clase: TopsisMatriz	
Tipo de clase: Util	
Atributos	
	Tipo
matrizDecision	Double[][]
cantAlternativas	int
cantCriterios	int

Tabla 33: Clase TopsisMatriz.

Nombre Clase: ISalternativa	
Tipo de clase: Entidad	
Atributos	
	Tipo
alternativa	Alternativa
IS	double
idealPositivo	List<Double>
idealNegativo	List<Double>
valorAlternativa	List<Double>

Tabla 34: Clase ISalternativa.

Anexo 5 Caso de Estudio

Resuelto de forma manual

Un equipo de desarrollo desea elegir un lenguaje de programación entre un conjunto de lenguajes y teniendo en cuenta algunos elementos como son el dominio del equipo de desarrollo en cada lenguaje de programación, el uso a nivel mundial de los lenguajes, la compatibilidad con el ambiente de desarrollo y las ventajas que brinda cada lenguaje en lo que va a trabajar el equipo de desarrollo.

Objetivo de la decisión: Escoger un lenguaje de programación para el equipo de desarrollo.

Descripción: Se necesita escoger el lenguaje de programación más adecuado para el equipo de desarrollo.

Criterios:

C1: Dominio del equipo de desarrollo en el lenguaje de programación

C2: El uso a nivel mundial del lenguaje.

C3: Compatibilidad con el ambiente de desarrollo.

C4: Ventajas que brinda el lenguaje.

Alternativas:

Alt1: Java

Alt2: PHP

Alt3: C++

Alt4: C#

Luego que se determina la cantidad de expertos que participarán en la decisión se recogen las evaluaciones de cada uno de ellos.

En este caso participarán 4 expertos. Sus evaluaciones se recogen en las siguientes matrices:

Experto 1

	C1	C2	C3	C4
Alt1	5	1	4	1
Alt2	3	3	3	3
Alt3	4	2	4	4
Alt4	3	4	2	4

Experto 2

	C1	C2	C3	C4
Alt1	4	4	4	1
Alt2	5	5	5	5
Alt3	4	4	1	4
Alt4	3	2	5	4

Experto 3

	C1	C2	C3	C4
Alt1	5	4	3	2
Alt2	3	4	3	3
Alt3	4	1	2	4
Alt4	3	5	4	4

Experto 4

	C1	C2	C3	C4
Alt1	2	1	5	4
Alt2	3	3	3	3
Alt3	4	2	2	4
Alt4	5	3	4	1

Luego con la media aritmética se obtiene la matriz de decisión:

(5+4+5+2)/4= 4	(1+4+4+1)/4=2.5	(4+4+3+5)/4=4	(1+1+2+4)/4=2
(3+5+3+3)/4=3.5	(3+5+4+3)/4=3.75	(3+5+3+5)/4=4	(3+5+3+3)/4=3.5
(4+4+4+4)/4=4	(2+4+1+2)/4=2.25	(4+1+2+2)/4=2.25	(4+4+4+4)/4=4
(3+3+3+5)/4=3.5	(4+2+5+3)/4=3.5	(2+5+4+4)/4=3.75	(4+4+4+1)/4=3.25

	C1	C2	C3	C4
Alt1	4	3	4	2
Alt2	4	4	4	4
Alt3	4	2	2	4
Alt4	4	4	4	3

Luego con la siguiente fórmula se obtiene la matriz de decisión normalizada:

$$r_{ij} = \frac{X_{ij}}{X_{ij}^{max}}$$

	C1	C2	C3	C4
Alt1	0.8	0.6	0.8	0.4
Alt2	0.8	0.8	0.8	0.8
Alt3	0.8	0.4	0.4	0.8
Alt4	0.8	0.8	0.8	0.6

Pesos de los criterios:

C1	C2	C3	C4
1	2	4	4

Normalización de los pesos de los criterios:

C1	C2	C3	C4
0.25	0.5	1	1

Matriz de decisión normalizada ponderada:

	C1	C2	C3	C4
Alt1	0.2	0.3	0.8	0.4
Alt2	0.2	0.4	0.8	0.8
Alt3	0.2	0.2	0.4	0.8
Alt4	0.2	0.4	0.8	0.6

Ideal Positivo:

0.2	0.4	0.8	0.8

Ideal Negativo:

0.2	0.2	0.4	0.4

Distancias al ideal positivo:

Alt1	0	0.1	0	0.4
Alt2	0	0	0	0
Alt3	0	0.2	0.4	0
Alt4	0	0	0	0.2

Distancias al ideal negativo:

Alt1	0	0.1	0.4	0
Alt2	0	0.2	0.4	0.4
Alt3	0	0	0	0.4
Alt4	0	0.2	0.4	0.2

	A+	A-
Alt1	0.5	0.5
Alt2	0	1
Alt3	0.6	0.4
Alt4	0.2	0.8

El índice de Similaridad se calcula con la siguiente fórmula:

$$IS_i = \frac{d_i^-}{d_i^- + \ d_i^+}$$

Índices de Similaridad:

Alt1	0.5/(0.5+0.5)=0.5
Alt2	1/(1+0)=1
Alt3	0.4/(0.4+0.6)=0.4
Alt4	0.8/(0.8+0.2)=0.8

Ranking Final de Alternativas:

	IS
Alt2	1
Alt4	0.8
Alt1	0.5
Alt3	0.4

En el sistema

Inicialmente en el sistema se insertan el nombre y el objetivo de la decisión.

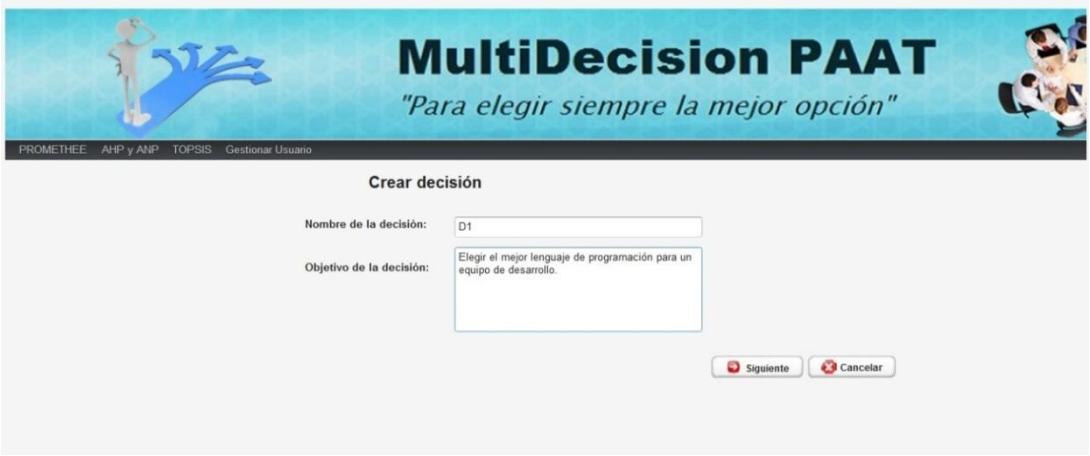

Figura 27: Insertar nombre y objetivo de una decisión en el método TOPSIS.

Luego se insertan los criterios de la decisión.

Figura 28: Insertar criterios de la decisión en el método TOPSIS.

Luego se insertan los pesos de cada criterio.

Figura 29: Insertar pesos de criterios de la decisión en el método TOPSIS.

Luego se insertan las alternativas.

Figura 30: Insertar alternativas en la decisión en el método TOPSIS.

Luego se escoge la cantidad de especialistas.

Figura 31: Escoger cantidad de especialistas en la decisión en el método TOPSIS.

Luego los expertos evalúan las alternativas respecto a los criterios.

Figura 32: Evaluar las alternativas respecto a los criterios.

Luego de que la decisión se guarda, se genera la matriz de decisión.

Figura 33: Se genera matriz de decisión.

Luego se calcula la matriz de decisión normalizada.

Figura 34: Matriz de decisión normalizada.

Luego se pondera la matriz de decisión normalizada.

Figura 35: Matriz de decisión normalizada ponderada.

Luego se determinan el ideal positivo y el ideal negativo.

Figura 36: Ideal positivo y negativo.

Luego se calculan las distancias de cada alternativa al ideal positivo y al ideal negativo.

Figura 37: Distancias al ideal positivo, negativo y el IS.

Finalmente se genera el ranking final.

Figura 38: Ranking Final del caso de estudio.

Anexo 6 Plantilla Caso de Prueba de Aceptación

Caso de Prueba de Aceptación
Código Caso de Prueba: P-1 **Nombre Historia de Usuario:** Autenticación de usuario.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y el usuario introduce un nombre de usuario y la contraseña y el sistema comprueba que estén registrados en el sistema.
Condiciones de Ejecución: • Que el sistema esté disponible. • Que el usuario y la contraseña estén correctos.
Entrada / Pasos de ejecución: Al introducirse el usuario y la contraseña, se comprueba que los mismos sean correctos.
Resultado Esperado: Que el usuario entre al sistema y tenga acceso a las funcionalidades correspondientes a sus permisos.
Evaluación de la Prueba: Satisfactoria

Tabla 35: Caso de Prueba de Aceptación P-1.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-2 **Nombre Historia de Usuario:** Gestionar Usuario.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y se verifica que el usuario autenticado sea administrador para que pueda adicionar, modificar y eliminar usuario.
Condiciones de Ejecución: • Que el usuario autenticado sea administrador.
Entrada / Pasos de ejecución: Al elegir la opción de gestionar usuario, el sistema debe dar la posibilidad de adicionar, modificar y eliminar usuario.
Resultado Esperado: Comprobar en la base de datos que se haya adicionado, eliminado o modificado un usuario correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 36: Caso de Prueba de Aceptación P-2.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-3 **Nombre Historia de Usuario:** Crear Decisión.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y se inserta el nombre y el objetivo de la decisión.
Condiciones de Ejecución: • Que se haya seleccionado la opción crear decisión.
Entrada / Pasos de ejecución: Se introduce el nombre y el objetivo de la decisión y se presiona el botón siguiente.
Resultado Esperado: Que cuando el usuario introduce el nombre y el objetivo de la decisión se pase al próximo paso de una secuencia lógica de pasos cuando se de en el botón siguiente.
Evaluación de la Prueba: Satisfactoria

Tabla 37: Caso de Prueba de Aceptación P-3.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-4 **Nombre Historia de Usuario:** Gestionar Criterio.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y se introduce el nombre y la descripción de un criterio y luego adiciona el mismo.
Condiciones de Ejecución: • Que se haya seleccionado la opción gestionar criterio.
Entrada / Pasos de ejecución: Luego que el usuario introduzca el nombre y la descripción del criterio este adiciona el mismo a la tabla de criterios.
Resultado Esperado: Que los criterios adicionados se muestren en la tabla criterio y que al presionar el botón siguiente se pase al próximo paso de la secuencia lógica de pasos.
Evaluación de la Prueba: Satisfactoria

Tabla 38: Caso de Prueba de Aceptación P-4.

Caso de Prueba de Aceptación

Código Caso de Prueba: P-5	**Nombre Historia de Usuario:** Gestionar pesos de los criterios.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo	
Descripción de la Prueba: Se ejecuta la prueba y se asignan los pesos de los criterios.	
Condiciones de Ejecución: • Que se haya seleccionado la opción gestionar pesos de los criterios.	
Entrada / Pasos de ejecución: Al elegir la opción gestionar peso de los criterios, el sistema debe dar la posibilidad de seleccionar el peso de cada criterio.	
Resultado Esperado: Que al seleccionar la opción siguiente el sistema pase al próximo paso de la secuencia lógica de pasos.	
Evaluación de la Prueba: Satisfactoria	

Tabla 39: Caso de Prueba de Aceptación P-5.

Caso de Prueba de Aceptación	
Código Caso de Prueba: P-6	**Nombre Historia de Usuario:** Gestionar Alternativa.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo	
Descripción de la Prueba: Se deben llenar los datos en el archivo en el formato establecido. Luego se carga la información.	
Condiciones de Ejecución: • Que se seleccione la opción gestionar alternativa. • Se comprueba que el archivo a cargar exista, que esté en el formato establecido y que contenga la información requerida.	
Entrada / Pasos de ejecución: Al elegir la opción de gestionar alternativa el usuario debe ir a la dirección especificada y entrar los datos requerido. Luego el usuario carga la información, permitiendo eliminar alternativas.	
Resultado Esperado: Que las alternativas cargadas se muestren en la tabla de alternativas y que al usuario seleccionar la opción siguiente el sistema pase al siguiente paso de la secuencia lógica.	
Evaluación de la Prueba: Satisfactoria	

Tabla 40: Caso de Prueba de Aceptación P-6.

Caso de Prueba de Aceptación	
Código Caso de Prueba: P-7	**Nombre Historia de Usuario:** Seleccionar cantidad de especialistas.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo	
Descripción de la Prueba: Se ejecuta la prueba y el usuario selecciona la cantidad de especialistas.	
Condiciones de Ejecución: • Que se haya seleccionado la opción seleccionar cantidad de especialistas.	
Entrada / Pasos de ejecución: Al elegir la opción seleccionar cantidad de especialistas, el sistema debe permitir seleccionar la cantidad de especialistas.	
Resultado Esperado: Que al presionar en la opción siguiente el sistema pase al paso próximo en la secuencia lógica de pasos.	
Evaluación de la Prueba: Satisfactoria	

Tabla 41: Caso de Prueba de Aceptación P-7.

Caso de Prueba de Aceptación	
Código Caso de Prueba: P-8	**Nombre Historia de Usuario:** Evaluar alternativas respecto a criterios.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo	
Descripción de la Prueba: Se deben llenar los datos en los archivos en el formato establecido. Luego se carga la información.	
Condiciones de Ejecución: • Que se seleccione la opción evaluar alternativas respecto a criterios. • Se comprueba que el archivo a cargar exista, que esté en el formato establecido y que contenga la información requerida.	
Entrada / Pasos de ejecución: Los especialistas deben llenar los archivos en el formato especificado. Luego se carga la información.	
Resultado Esperado: Que se muestre en la tabla la matriz con la media aritmética de las evaluaciones de los especialistas. El sistema debe permitir seleccionar la opción finalizar. Se comprueba que la información recogida en este paso y los pasos anteriores de la secuencia lógica estén correctamente en la base de datos.	
Evaluación de la Prueba: Satisfactoria	

Tabla 42: Caso de Prueba de Aceptación P-8.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-9 **Nombre Historia de Usuario:** Seleccionar decisión.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se muestren todas las decisiones que se encuentran en la base de datos. Debe permitir seleccionar cualquiera de las decisiones existentes.
Condiciones de Ejecución: • Que se seleccione la opción seleccionar decisión. • Que exista al menos una decisión.
Entrada / Pasos de ejecución: El usuario selecciona la decisión que desea.
Resultado Esperado: Que el sistema permita seleccionar una decisión.
Evaluación de la Prueba: Satisfactoria

Tabla 43: Caso de Prueba de Aceptación P-9.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-10 **Nombre Historia de Usuario:** Generar ranking final.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y se ejecuta el generar ranking final.
Condiciones de Ejecución: • Que se seleccione la opción generar ranking final. • Que se haya seleccionado la decisión a ejecutar.
Entrada / Pasos de ejecución: Al elegir la opción generar ranking final, el sistema muestra una tabla con las alternativas ordenadas de acuerdo a su preferencia según el algoritmo de solución basado en el método TOPSIS.
Resultado Esperado: Que el sistema muestre una tabla con las alternativas ordenadas correctamente.
Evaluación de la Prueba: Satisfactoria

Tabla 44: Caso de Prueba de Aceptación P-10.

Caso de Prueba de Aceptación
Código Caso de Prueba: P-10 **Nombre Historia de Usuario:** Generar ranking final por pasos.
Nombre de la persona que realiza la prueba: Yanet Peña Táramo
Descripción de la Prueba: Se ejecuta la prueba y se ejecuta el generar ranking final por pasos.
Condiciones de Ejecución: • Que se seleccione la opción generar ranking final por pasos. • Que se haya seleccionado la decisión a ejecutar.
Entrada / Pasos de ejecución: Al elegir la opción generar ranking final, el sistema muestra una secuencia de pasos, estos son : -Mostrar la matriz de decisión. -Mostar la matriz normalizada. -Mostar la matriz normalizada ponderada. -Mostrar el Ideal Positivo y el Ideal Negativo. -Mostrar una tabla con las distancias de cada alternativa al Ideal Positivo y al Ideal Negativo. -Mostrar una tabla con las alternativas ordenadas.
Resultado Esperado: Que el sistema muestre correctamente la secuencia de pasos para obtener el orden de las alternativas.
Evaluación de la Prueba: Satisfactoria

Tabla 45: Caso de Prueba de Aceptación P-11.

Anexo 7 Interfaces de Usuario

Figura 39: Interfaz de Usuario Registrarse.

Figura 40: Interfaz de Usuario Inicio Registrar Decisión.

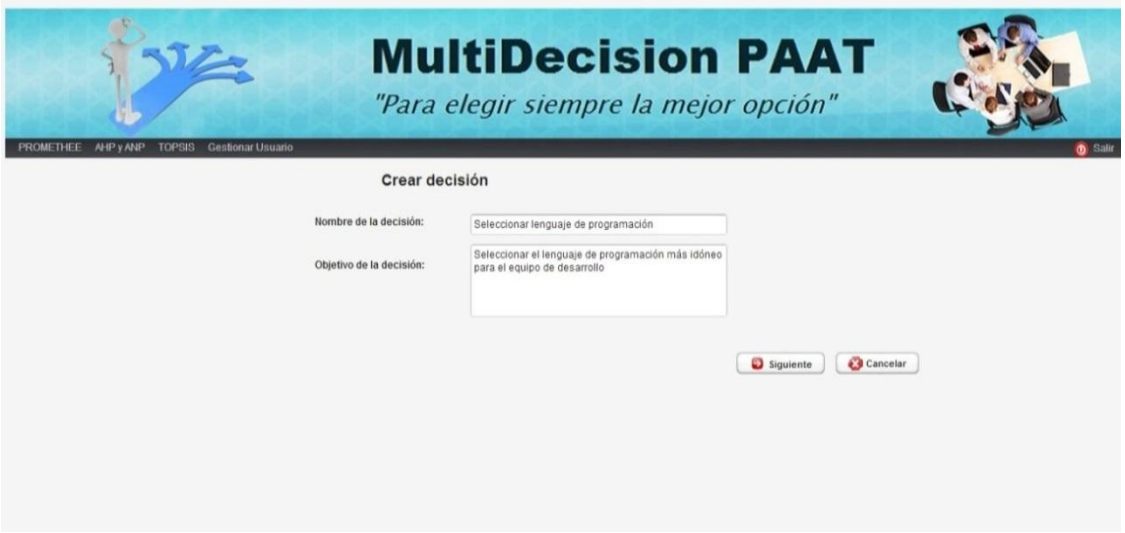

Figura 41: Interfaz de Usuario Crear Decisión.

Figura 42: Interfaz de Usuario Gestionar Criterios.

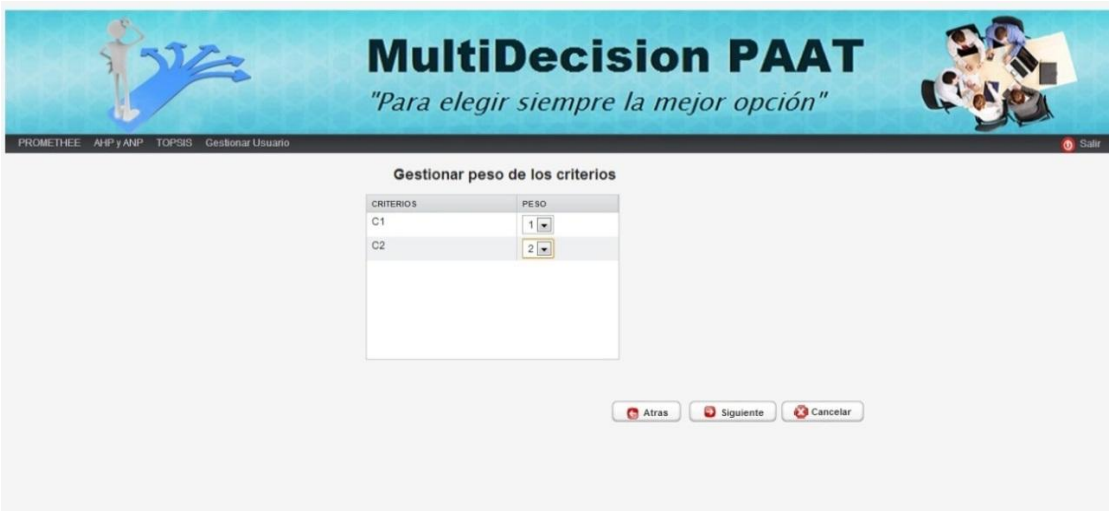

Figura 43: Interfaz de Usuario Gestionar peso de los criterios.

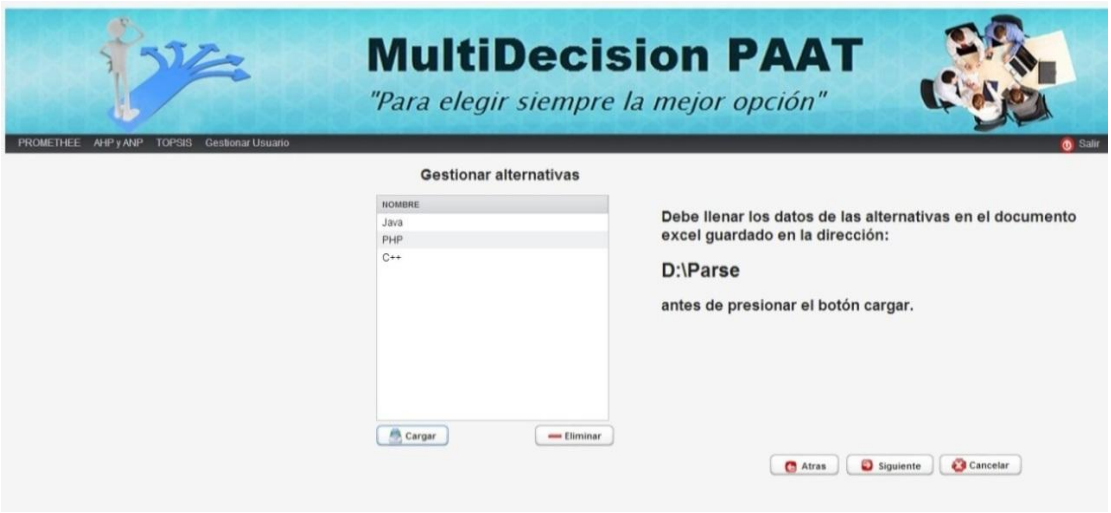

Figura 44: Interfaz de Usuario Gestionar alternativas.

Figura 45: Interfaz de Usuario Seleccionar cantidad de especialistas.

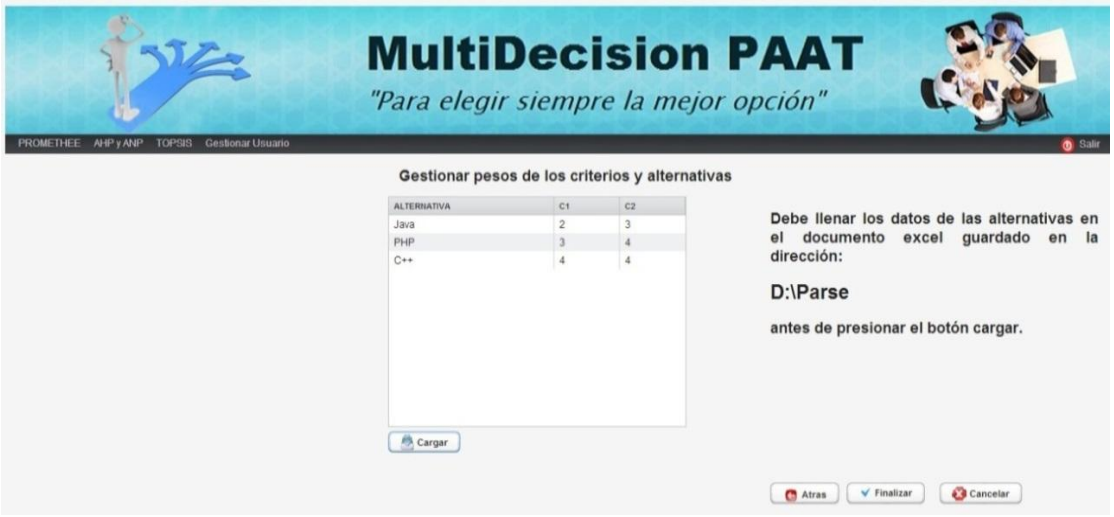

Figura 46: Interfaz de Usuario Gestionar pesos de los criterios y las alternativas.

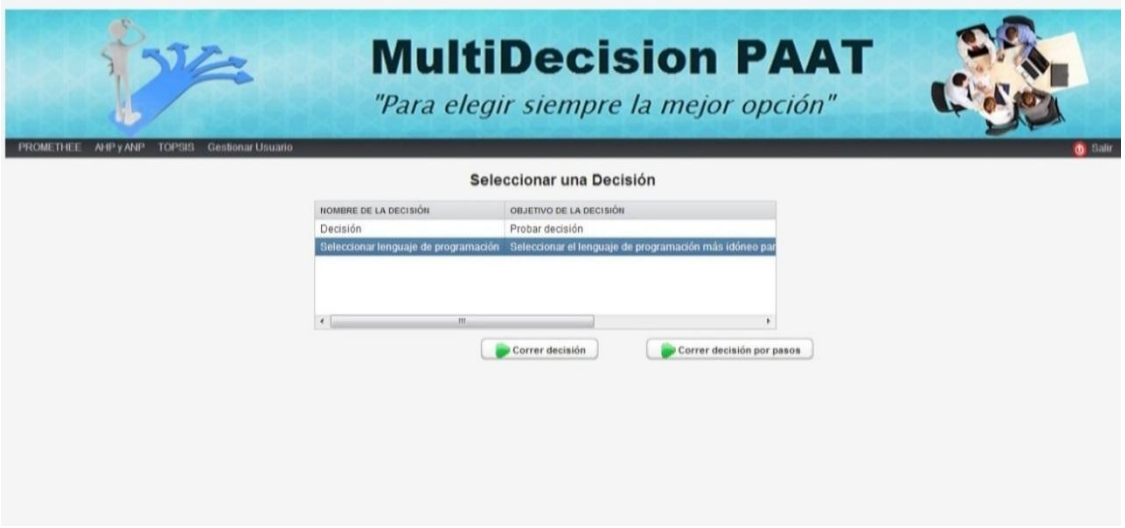

Figura 47: Interfaz de Usuario Seleccionar decisión.

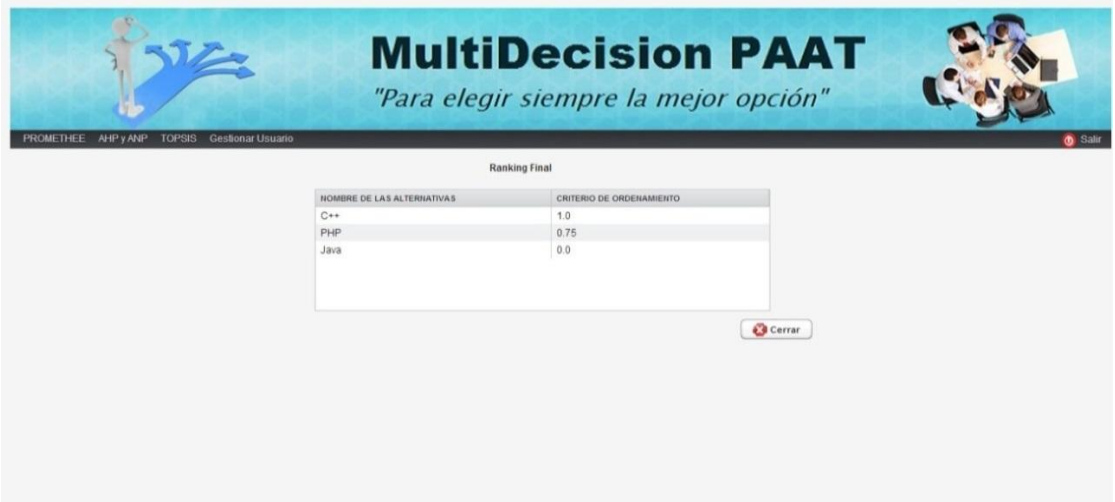

Figura 48: Interfaz de Usuario Ranking Final

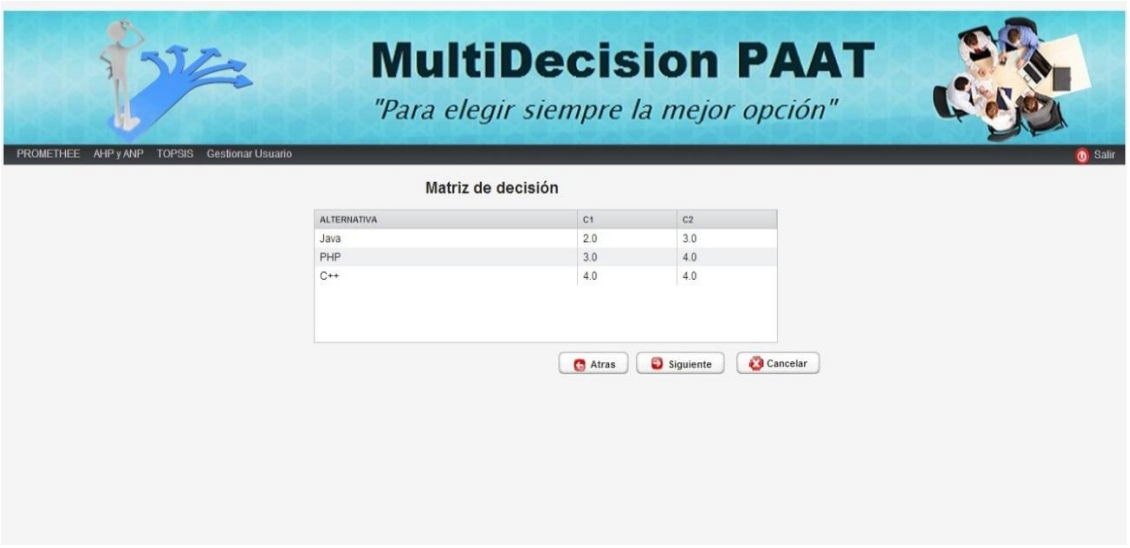

Figura 49: Interfaz de Usuario Generar Matriz de decisión.

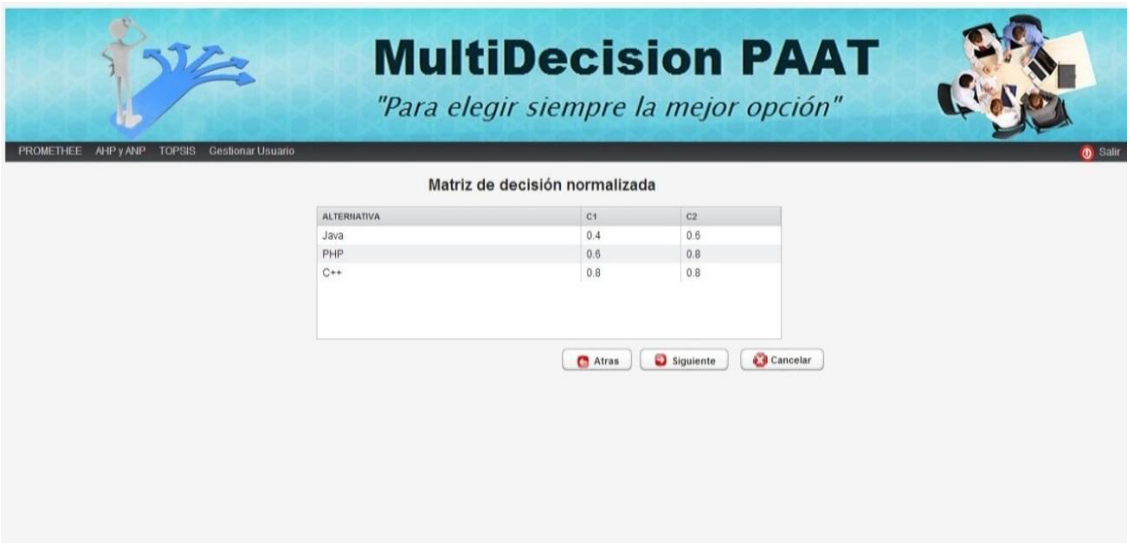

Figura 50: Interfaz de Usuario Generar Matriz de decisión normalizada.

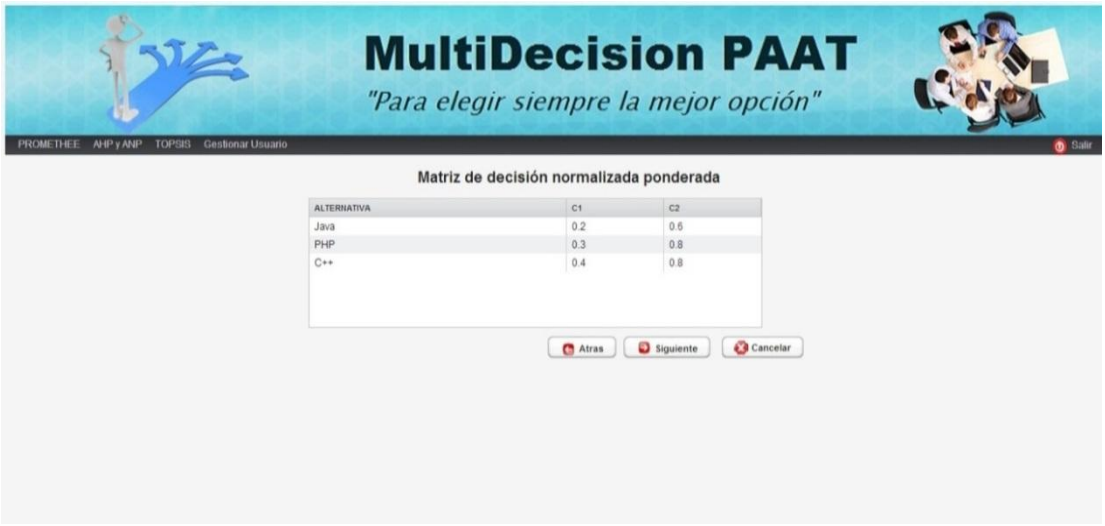

Figura 51: Interfaz de Usuario Generar Matriz de decisión normalizada ponderada.

Figura 52: Interfaz de Usuario Gestionar Usuarios