Escuela Politécnica Superior (Jaén)
Ingeniería en Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Proyecto Fin de Carrera

PLATAFORMA WEB PARA LA RESOLUCIÓN Y GESTIÓN DE PROBLEMAS DE TOMA DE DECISIÓN MULTICRITERIO: PROMETHEE Y AHP

Alumno: Juan Jesús Cárdenas Labella

Tutor: Prof. D. Francisco Mata Mata
Dpto: Informática

Julio, 2011

INFORME DEl TUTOR DEL PROYECTO

Contenido

CAPÍTULO I

Introducción

1.1. Introducción al proyecto

La Real Academía Española (RAE) define el término *Decisión* como la determinación o resolución que se toma o se da en una situación dudosa. La estructura de una decisión puede ser muy variada (una opinión, una regla, etc.), pero esencialmente se trata de un elemento no físico que se traduce en la ejecución de una o varias tareas tras el estudio de una serie de factores, lo que se conoce como el proceso de toma de decisión.

Por tanto, de la definición anterior se puede deducir que la *Toma de Decisiones* es el proceso mediante el cual se toma una decisión, es decir, se realiza la elección de la mejor alternativa posible entre varias disponibles, a efectos de resolver un problema determinado. Para ello es posible realizar previamente un estudio detallado atendiendo a las características del problema, posibles alternativas y factores que la persona que toma la decisión (decisor) quiera tener en cuenta. La elección de una u otra alternativa viene dada por el proceso de selección utilizado en cada caso y por la clase de problema que se pretende resolver.

Normalmente los problemas que surgen en el área de la toma de decisiones tratan de situaciones que se producen en el mundo real, por lo que pueden presentarse en distintos contextos: a nivel familiar, laboral, empresarial, etc.

Los procesos de toma de decisiones pueden ser de diversa índole atendiendo a la estructura y el tipo del problema a resolver así como a los criterios que el decisor pretende ajustar. El área que atañe a este proyecto fin de carrera es el de la toma de decisiones con varios criterios, denominada toma de decisiones multicriterio.

Durante el proceso de toma de decisiones, además de las características que describen el problema que se pretende resolver y de las posibles alternativas a seleccionar, se distinguen una serie de criterios que el decisor pretender maximizar o minimizar, que son los que van a determinar la naturaleza de la decisión que finalmente se toma. En ocasiones la toma de decisiones hace referencia a un sólo criterio, es decir, que para la elección de la mejor alternativa a un determinado problema sólo se tiene en cuenta un factor. Normalmente este tipo de problemas de decisión son los más sencillos de resolver, pero es evidente que esto no siempre es así, de hecho en la gran mayoría de las situaciones de la vida real intervienen más de un factor o criterio que son tenidos en cuenta a la hora de tomar una decisión. Es de este modo como surgen los problemas de toma de decisiones multicriterio.

Dentro de la toma de decisión multicriterio existen distintos modelos para resolver problemas, siendo PROMETHEE y AHP unos de los más extendidos. A continuación se proporciona una breve explicación sobre las bases y el funcionamiento de cada método.

El modelo PROMETHEE hace uso de una noción de criterio generalizado para construir un orden entre el conjunto de alternativas. Fue introducido por J. P. Brans en la década de los 80 y consta principalmente de dos fases: la construcción de una clasificación entre las posibles acciones y la explotación de esta clasificación para obtener la acción que maximice/

minimice los criterios. Además como todos los parámetros definidos tienen un significado económico el decisor puede modificarlos fácilmente [2].

Una vez definido el conjunto de criterios a tener en cuenta y las posibles alternativas a tomar, cada alternativa se evalúa en relación a un criterio determinado, lo que permite hablar de preferencias de acciones atendiendo a un factor específico. Es por ello que en este modelo hay que definir para cada criterio su correspondiente función de preferencia que permita establecer las relaciones entre acciones. De hecho, se distinguen seis posibles funciones de preferencia para cada criterio. Cada función va acompañada de una serie de parámetros que el decisor debe fijar y que permiten definir el contexto del problema [1]. Asimismo, es necesario asociar a cada criterio un peso específico que representa la importancia del mismo con respecto al problema.

A raíz de las preferencias que se generan entre las acciones surgen tres conceptos: flujo entrante, flujo saliente y flujo neto, que determinan dichas preferencias teniendo en cuenta todos los criterios para una acción y por tanto indican que alternativa es la mejor sobre las demás.

Dentro de este modelo existen dos variantes, PROMETHEE I y II, las cuales han sido tratadas en este proyecto [1, 2].

- PROMETHEE I: se obtiene un preorden parcial.
- PROMETHEE II: se consigue un orden total.

Por otro lado, el modelo AHP hace uso de matrices de comparaciones pareadas entre criterios y alternativas para construir una clasificación de valores de la que obtiene la mejor alternativa. Estas comparaciones se pueden llevar a cabo directamente o usando una escala que refleje las preferencias [5]. Fue diseñado e introducido por Thomas L. Saaty a finales de los años 70.

AHP crea una estructura jerárquica descendente usando para ello los datos que se incluyen en el problema: posibles objetivos, alternativas, criterios y subcriterios si es que los hubiera. El uso de esta estructura tiene dos utilidades: proporcionar una visión completa de la compleja relación entre los factores del problema y permitir al decisor comparar los elementos de los mismos niveles ya que son homogéneos.

Para llevar a cabo las comparaciones entre los elementos de la estructura jerárquica en el modelo AHP se proporciona una escala verbal que tiene una directa correspondencia con unos valores numéricos. Tras realizar una serie de comparaciones entre los criterios y entre las alternativas respecto de cada criterio, se determina cual es la mejor alternativa.

Este modelo se descompone en las siguientes fases [4]:

1. Descomponer el problema complejo en una jerarquía.

2. Establecer prioridades entre los distintos niveles de la jerarquía.

3. Establecer las prioridades de los datos proporcionados por el grupo de solicitantes.

Ambos modelos están muy extendidos e implican la realización de complejos cálculos, de ahí que se haya considerado interesante desarrollar un aplicación web no sólo para resolver este tipo de problemas, sino también para gestionarlos de una forma eficaz y eficiente a través de un grupo de expertos.

1.2. Propósito

El propósito del proyecto es desarrollar e implementar una aplicación basada en tecnologías web que permita resolver y gestionar problemas de Toma de Decisión Multicriterio mediante los modelos PROMETHEE y AHP. La aplicación está diseñada para ser implantada inicialmente en un entorno educativo aunque puede ser extendida a cualquier otro tipo de contexto.

1.3. Objetivos

1. Localizar y estudiar información bibliográfica sobre la Toma de Decisión Multicriterio y la resolución de problemas según los modelos PROMETHEE y AHP.

2. Analizar, diseñar e implementar los modelos PROMETHEE y AHP para resolver problemas de toma de decisión multicriterio de carácter general.

3. Conseguir una simulación representativa y fiable de ambos modelos.

4. Integrar los modelos en una única aplicación web usable y modular para futuras ampliaciones.

5. Desarrollar un sistema de gestión de problemas de toma de decisión multicriterio a través de expertos.

6. Verificación y validación de la plataforma web.

7. Documentar todo el proyecto haciendo uso de una memoria.

1.4. Resultados esperados

El resultados esperado es el desarrollo e implementación de una plataforma web que permita:

a) Plantear y resolver problemas de Toma de Decisión Multicriterio por medio de la aplicación de los modelos PROMETHEE y AHP.

b) Desarrollar e implementar las funcionalidades necesarias que permitan gestionar los problemas planteados.

1.5. Resumen

Los objetivos que persigue este proyecto se desarrollan a través de la memoria del siguiente modo:

Tras la búsqueda y posterior estudio de la bibliografía adecuada para desarrollar este proyecto, se presentan en el segundo y tercer capítulo los conocimientos adquiridos que permiten introducirse en el mundo de la toma de decisiones, para conocerlo y entenderlo con más detalle. En el segundo capítulo se explica desde un punto de vista teórico la toma de decisión, prestando una especial atención a la toma de decisiones multicriterio, que es el área que atañe a este proyecto. El tercer capítulo se centra en dos modelos concretos de resolución de problemas con múltiples criterios, como son PROMETHEE y AHP. Primero habla del tipo de información que se utiliza en el proceso de toma de decisión, luego continúa con una presentación para conocer ambos modelos y finaliza con la explicación de su método de aplicación.

El cuarto capítulo recoge algunos detalles generales sobre los que se fundamenta el proyecto. Inicialmente se hace una descripción de las etapas de Ingeniería del *Software* en las que se divide el desarrollo de la aplicación y del contexto donde se va a hacer uso de ella. También se incluye la planificación temporal del proyecto, la especificación de los requerimientos y un apartado acerca de las tecnologías de desarrollo empleadas para la creación del *software*.

El quinto capítulo abarca la fase de análisis del sistema. Incorpora los perfiles de usuario, basados en las personas que van a hacer uso de la aplicación, además de los casos de uso y los escenarios. Por último, se hace un análisis de tareas que detalla las interacciones del usuario con el objetivo de ayudar en el diseño de la interfaz.

En el sexto capítulo se detalla todo el diseño del sistema. Dentro del mismo se incluye el diseño de los objetos, que se compone del diagrama de paquetes y los diagramas de clases. el diseño de los los datos, que se fundamenta en el modelo entidad-relación, y el diseño de la interfaz de usuario.

El séptimo capítulo comprende las fases de implementación y prueba del sistema. La parte de implementación abarca algunas de las tecnologías empleadas en la aplicación y las herramientas de desarrollo para su implementación. La parte de verificación, validación y prueba se compone de un conjunto de pruebas de caja negra.

En el octavo capítulo se exponen las conclusiones obtenidas tras la realización del proyecto. Incluyen la justificación del proyecto, sus características más destacables, algunas de las tecnologías más importantes sobre las que se sustenta y las posibles futuras ampliaciones.

La memoria también se compone de dos anexos. El primer anexo es un manual de instalación de la aplicación, que se divide en dos secciones: el servidor de aplicaciones y la base de datos. El segundo anexo es un manual de usuario que explica cómo hacer uso de la aplicación y que está dividido en dos partes: una para el administrador y otra para los expertos.

Finalmente, la memoria incorpora una sección donde se recoge la bibliografía consultada durante la realización del proyecto fin de carrera.

CAPÍTULO II

Antecedentes

2.1. La toma de decisiones

En los epígrafes que se presentan a continuación se pretende dar una visión más profunda y detallada sobre la toma de decisiones, desde una breve reseña histórica hasta la evaluación del proceso de decisión, pasando por sus elementos, etapas y ambientes de decisión entre otros.

2.1.1. Antecedentes

El problema de la toma de decisiones es muy antiguo, tanto como la vida misma. Todo ser vivo, por muy simple que sea, se enfrenta a lo largo de su vida a un buen número de problemas de decisión que debe resolver, en ocasiones estos problemas serán más o menos complejos y dependiendo del proceso de razonamiento utilizado por el ser vivo la decisión que finalmente se toma será más o menos acertada.

Normalmente los seres vivos más simples tienden a realizar procesos de toma de decisiones más simples, mientras que según aumenta la complejidad del individuo se incrementa la complejidad de la decisión tomada.

En el ámbito de la toma de decisiones intervienen dos factores importantes como son el *instinto* y la *memoria* [7]. El primero de ellos es un factor determinante que está directamente relacionado con la intuición del ser vivo y por tanto depende en gran medida de éste, por ejemplo, las crías de los insectos conocen de un modo innato complejas formas de conductas como lo es la construcción de nidos. La memoria de un ser vivo formada por sus experiencias a lo largo de la vida, lo que ha aprendido, también es un factor determinante a la hora de tomar decisiones, queda reflejado en el ejemplo del laberinto que tiene que recorrer un ratón para encontrar la comida cuando el resto de caminos conllevan un castigo, al final el ratón aprende que camino es el correcto.

Conociendo todo lo anterior se hace comprensible que, al ser los humanos seres vivos más complejos, sean necesarias nuevas formas de tomar decisiones. Es así como durante la civilización griega surge el concepto de *razonamiento* como mecanismo para tomar decisiones, distinguiéndose dos claras vertientes en su utilización:

a) Como proceso mediante el que se llega a la elección de una forma de actuar: *toma de decisiones*.

b) Como método para convencer a los demás de que se ha actuado correctamente: *persuasión*.

A partir del siglo XVII, y debido a los descubrimientos científicos, los seres humanos empiezan a dudar y descubren una concepción algo diferente sobre el razonamiento y es lo que se hace llamar como el *método científico*. El concepto de ciencia cuando es utilizada por el ser humano está asociado a la realización de experimentos.

El método científico se puede considerar como la interacción entre la intuición y la lógica interpretación de los resultados obtenidos por la experiencia, lo que produce nuevas teorías [7]. A continuación se detalla por medio de un esquema las fases del método científico.

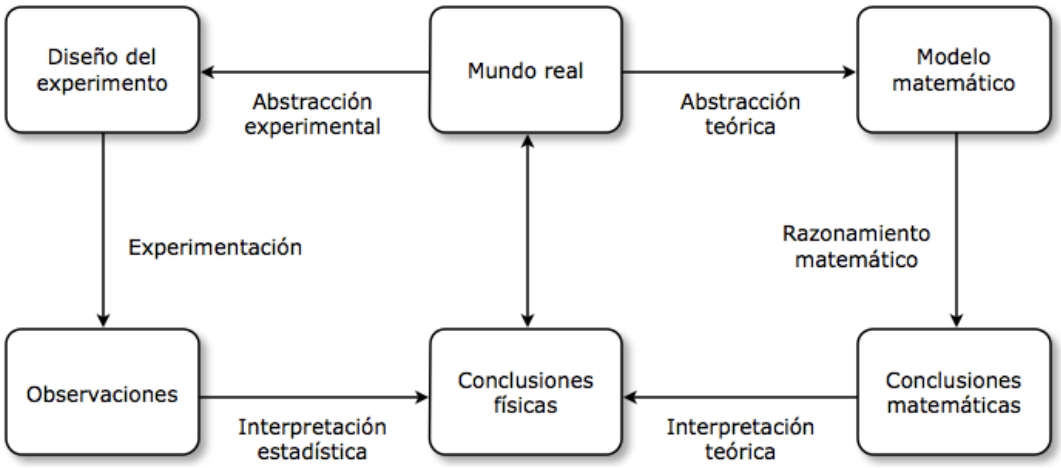


Figura 2.1. Fases del método científico.

Es el método científico el que sienta las bases que han permitido a los seres humanos hacer uso de los procesos de toma de decisiones que existen actualmente y que se utilizan para elegir las mejores decisiones que además quedan respaldadas por un proceso de razonamiento adecuado que permita demostrar a los demás que la elección realizada es la más correcta.

2.1.2. Descripción

La *Toma de Decisiones* es el área que abarca las tareas y procesos que los seres humanos llevan a cabo y se trata de uno de los aspectos más importantes en su vida debido al gran número de decisiones que hay que tomar a lo largo de ésta.

Dentro de este ámbito es de obligado aviso definir los factores que intervienen, de forma general, durante la toma de decisiones. En primer lugar, existe algún problema o situación que se pretende resolver por medio de la elección más oportuna, a través de un proceso de toma de decisión. Dicho problema tiene asociadas una serie de características que lo hacen único y todas ellas influyen de un modo decisivo sobre qué decisión tomar. Del mismo modo, se presentan un conjunto de alternativas o acciones a poder realizar que no es más que el conjunto de decisiones que se pueden tomar. En determinados casos es posible que la persona que toma la decisión requiera el cumplimiento de ciertos criterios que pretende maximizar o minimizar lo que hace más complejo el proceso de razonamiento.

Por tanto, el proceso de toma de decisiones consiste en la elección de la mejor alternativa posible entre varias disponibles, a efectos de resolver un problema determinado. En oca-

siones el problema es tan complejo que obliga a tener en cuenta muchos detalles lo que hace de la toma de decisiones un proceso bastante complejo y nada trivial. Es por ello que existe la posibilidad de aplicar algún modelo de toma de decisiones que lleve a cabo un razonamiento más minucioso que guíe al decisor sobre que alternativa tomar y además permita justificar científicamente el porqué de su elección. En este caso, la elección de una u otra alternativa viene dada por el modelo de selección utilizado en cada caso y por la clase de problema que se pretende resolver.

Todo proceso de toma de decisiones se apoya sobre cincos conceptos básicos:

- ***Información***

Ésta se recoge con el fin de definir las limitaciones de las alternativas, recogiendo tanto aspectos que están a favor de ellas como en contra. Sin embargo si la información no puede obtenerse, la decisión entonces debe basarse en los datos disponibles, los cuales caen en la categoría de información general.

- ***Conocimientos***

Si quien toma la decisión tiene conocimientos, ya sea de las circunstancias que rodean el problema o de una situación similar, entonces éstos pueden utilizarse para seleccionar un curso de acción favorable. En caso de carecer de conocimientos, es necesario buscar consejo en quienes están informados.

- ***Experiencia***

Cuando un individuo aporta información para solucionar un problema de forma particular, ya sea con resultados buenos o malos, esta experiencia le proporciona información para la solución del próximo problema similar. Si ha encontrado una solución aceptable, con mayor razón tenderá a repetirla cuando surja un problema parecido. Si se carece de experiencia entonces hay que experimentar; pero sólo en el caso de que las consecuencias de un mal experimento no sean desastrosas.

- ***Análisis***

No puede hablarse de un método en particular para analizar un problema. En ausencia de un método para analizar matemáticamente un problema es posible estudiarlo con otros métodos diferentes. Si estos otros métodos también fallan, entonces debe confiarse en la intuición, ya que si los otros ingredientes de la toma de decisiones no señalan el camino a tomar, entonces ésta es la única opción disponible.

- ***Juicio***

El juicio es necesario para combinar la información, los conocimientos, la experiencia y el análisis, con el fin de seleccionar el curso de acción apropiado.

Por otra parte, hay que señalar que el proceso de toma de decisiones presenta las siguientes características:

- Como mínimo deben ser posibles dos formas de actuar que se denominan *alternativas* o *acciones*, que se considerarán excluyentes entre sí, de forma que realizar alguna de ellas imposibilita la ejecución de las demás.

- Entre las alternativas y por medio de un proceso de decisión *se elige una* que es la que se lleva cabo.

- La elección de una alternativa a de hacerse de forma que cumpla un determinado *fin*.

En la toma de decisiones se distinguen claramente, desde hace cierto tiempo, dos escuelas mayoritarias: la *normativa* y la *descriptiva* [12].

- Normativa: está orientada a la salida. Se basa en el paradigma de la racionalidad sustantiva, que indica como deberían tomarse las decisiones y qué métodos utilizar para ello.

- Descriptiva: está orientada al proceso. Se basa en el paradigma de la racionalidad procedimental, que indica como se tomas las decisiones.

En la pasada década ha ido apareciendo y planteándose una tercera vía: la *escuela prescriptiva* o *constructiva* que está orientada al conocimiento y se basa en nuevos paradigmas de racionalidad, que indican como mejorar los procesos de decisión.

La toma de decisiones es un área que se aplica en distintas disciplinas, tales como, las Ciencias Sociales, la Economía, la Ingeniería, etc. Esta amplia gama de campos de aplicación tiene como consecuencia la existencia de diferentes modelos de Toma de Decisión, aunados bajo la disciplina denominada Teoría de la Decisión.

2.1.3. La decisión

El término *Decisión* se define como la combinación de las facultades analíticas de observación, conocimiento e intuición que tienen los seres vivos y su aplicación al estudio de un determinado problema para con posterioridad seleccionar y ejecutar una o varias tareas. Las decisiones puede presentarse de diversas formas puesto que tienen una estructura muy variada.

Las decisiones pueden clasificarse atendiendo a múltiples criterios, a continuación se presentan algunos de ellos. Dependiendo de la naturaleza del objetivo que se persigue y del contexto en el que se toma la decisión:

- Decisiones aleatorias: se tratan de decisiones en las que sólo interviene el azar.

- Decisiones paramétricas: son decisiones cuyo comportamiento es previsible puesto que los elementos del contexto son constantes.

- Decisiones estratégicas: en este tipo de decisiones los elementos que las condicionan no son muy estables.

Atendiendo a la posible especificación previa de las decisiones se distinguen:

- Decisiones programadas: son aquellas que pueden ser especificadas previamente por un conjunto de reglas o procedimientos de decisión.

- Decisiones no programadas: no tiene el conjunto previo de reglas o procedimientos.

Según el tipo de modelo de decisión se encuentran:

- Decisiones prescriptivas: modelo de decisión que le dice al decisor como hacer la decisión. Supone un comportamiento completamente racional de la persona que toma decisiones, esto es, siempre escoge la alternativa óptima.

- Decisiones descriptivas: modelo de decisión que describe como hacen sus decisiones los que actualmente están encargados de tomar decisiones. El criterio normal de decisión es el de satisfacción, por lo que se limita a encontrar la primera alternativa que satisfaga todas las restricciones.

2.1.4. Elementos

En todo problema de decisión se pueden distinguir una serie de elementos característicos:

- El *decisor*, encargado de realizar la elección de la mejor forma de actuar de acuerdo con sus intereses.

- Las *alternativas* o *acciones*, que son las diferentes formas de actuar posibles, de entre las cuales se seleccionará una. Deben ser excluyentes entre sí.

- Los posibles *estados de la naturaleza*, término mediante el cual se designan a todos aquellos eventos futuros que escapan al control del decisor y que influyen en el proceso.

- Las *consecuencias* o *resultados* que se obtienen al seleccionar las diferentes alternativas bajo cada uno de los posibles estados de la naturaleza.

- La *regla de decisión* o *criterio*, que es la especificación de un procedimiento para identificar la mejor alternativa en un problema de decisión.

2.1.5. Etapas de la toma de decisiones

Tomar una buena decisión consiste en trazar el objetivo que se quiere conseguir, reunir toda la información relevante y tener en cuenta las preferencias del que tiene que tomar la decisión. Si se quiere tomar de forma correcta hay que ser conscientes de que una buena decisión es un proceso que necesita tiempo y planificación. Por ello la única manera de tomar una buena decisión es a través de la aplicación de un buen procedimiento o modelo de toma de decisiones, el cual permita ahorrar tiempo, esfuerzo y energía.

En todo proceso de toma de decisiones se pueden advertir una serie de etapas que se dan de forma generalizada ante cualquier problema que se pretenda resolver, sin embargo se pueden distinguir fases distintas dependiendo del enfoque del autor.

De acuerdo al modelo de Herbert A. Simon [20], las fases principales en la toma de decisiones son:

1. ***Inteligencia***

Obtención de los datos de entrada, procesamiento y análisis de aspectos que pueden identificar problemas u oportunidades.

2. ***Diseño***

Inventar, desarrollar y analizar posibles cursos de acción. Esto involucra procesos para entender el problema, para generar soluciones y para probar la factibilidad de las soluciones.

3. ***Selección***

Seleccionar una alternativa o curso de acción entre los disponibles. La selección se realiza e implementa.

Otros autores consideran que las etapas que constituyen el proceso de toma de decisiones son:

1. ***Identificación del problema***

En esta fase hay que identificar el problema que se desea resolver.

2. ***Generación de las alternativas***

Esta etapa consiste en identificar y desplegar el conjunto de todas las alternativas posibles que permitan solucionar el problema.

3. Evaluación de alternativas

En esta etapa se realiza un análisis cada una de las alternativas identificadas en la fase anterior. Hay que evaluar cada una de ellas de forma crítica, con el fin de obtener una serie de ventajas e inconvenientes.

La evaluación de una alternativa se puede realizar atendiendo a dos escalas de preferencia: factores cuantitativos o factores cualitativos. En el epígrafe siguiente se explica con más detalle en qué consiste cada término.

Para evaluar y comparar los factores se debe reconocer el problema y luego analizar qué factor se aplica, clasificar los términos de importancia, comparar su probable influencia sobre el resultado y tomar una decisión.

4. Selección de la mejor alternativa

Cuando el decisor ha considerado las posibles consecuencias de sus alternativas, se presupone que ya está en condiciones de tomar la decisión. Se pueden considerar tres objetivos diferentes a la hora de realizar la elección:

- Maximizar: el objetivo es tomar la mejor decisión posible.
- Satisfacer: es la elección de la primera opción que sea mínimamente aceptable o adecuada, y de esta forma se satisface una meta o criterio buscado.
- Optimizar: es el mejor equilibrio posible entre distintas metas.

5. Implementación de la decisión

El proceso no finaliza cuando la decisión se toma; esta debe ser implementada. Bien puede ser que quienes participen en la toma de una decisión sean quienes procedan a implementarla, o como en otras ocasiones, se delegue dicha responsabilidad en otras personas. Debe existir la comprensión total de la elección que se ha tomado, las razones que la motivan y sobre todo debe existir el compromiso de su implementación con éxito. Para tal fin, las personas que participan en esta fase del proceso, deberían estar involucradas desde las primeras etapas que anteriormente se han mencionado.

Cuando una decisión es tomada, ésta probablemente generará ciertos problemas durante su ejecución, por lo tanto los expertos deben dedicar el tiempo suficiente al reconocimiento de los inconvenientes que se pueden presentar así como también ver la oportunidad potencial que éstos pueden representar.

6. Evaluación de la decisión

Es la última etapa del proceso de toma de decisiones. Se recopila toda la información sobre los resultados obtenidos pudiendo tratarse de un proceso de retroalimentación positiva o negativa.

Si la retroalimentación es positiva, entonces indica que se puede continuar sin problemas y que incluso se podría aplicar la misma decisión a otras áreas de la organización. Si por el contrario, la retroalimentación es negativa, podría ser que tal vez la implementación requiera de más tiempo, recursos, esfuerzos o pensamiento, o que indique que la decisión fue equivocada, para lo cual se debe volver al principio del proceso (re)definición del problema. Si esto ocurriera, sin duda se tendría más información y probablemente más sugerencias que ayudarían a evitar los errores cometidos en el primer intento.

2.1.6. Escalas de preferencia

En todo problema de decisión, según se ha visto, existen un conjunto de alternativas como soluciones a un problema determinado, sobre las que hay que realizar una elección. Para poder decidir por que opción decantarse el decisor necesita previamente establecer comparaciones entre ellas, de manera que pueda determinar cuando una alternativa es preferida sobre otra. Atendiendo a la naturaleza del problema, de los criterios o de las alternativas se pueden utilizar valoraciones cualitativas o cuantitativas.

- ***Cualitativas***

Se trata de valoraciones que son difíciles de medir por lo que no se les puede asignar valores numéricos. Normalmente hacen referencia a valoraciones relacionadas con la medición de sensaciones subjetivas por parte de las personas. Este carácter subjetivo implica que en la mayoría de los casos sea difícil asignar valores precisos, decantándose por el uso de valores aproximados. Por ejemplo, son factores cualitativos las relaciones de trabajo, el riesgo del cambio tecnológico o el clima político internacional.

- ***Cuantitativas***

Se trata de valoraciones que se pueden medir numéricamente por lo que se le puede asignar valores numéricos. Por ejemplo, son factores cuantitativos el tiempo, los costes o los beneficios.

La escalas cuantitativas son mejores que las cualitativas desde un punto de vista matemático ya que permiten realizar un tratamiento numérico del problema, y por tanto, las operaciones entre números es directa.

2.1.7. Ambientes de decisión

Se definen como el contexto o la situación donde se toman las decisiones. Se pueden clasificar atendiendo al conocimiento y control que se tenga sobre las variables que intervienen o influencian el problema, ya que la decisión final que se tome va a estar condicionada por dichas variables.

Por tanto, los problemas de decisión se pueden clasificar dependiendo del ambiente en el que se toma la decisión: ambiente de certidumbre, riesgo o incertidumbre [7].

- ***Ambiente de certidumbre***

Un problema de decisión está definido en un ambiente de certidumbre cuando son conocidos con exactitud todos los elementos o factores que intervienen en el problema, es decir, el ambiente es conocido con certeza, por lo que cada acción conduce invariablemente a un resultado bien definido.

Es necesario construir una medida de eficiencia que se pueda utilizar para determinar cuál es la mejor alternativa y qué función de esta medida, conocida como función objetivo, debe utilizarse como criterio para determinar cual es la mejor solución. Este tipo de modelos no implican probabilidades, de ahí que generalmente sean más fáciles de construir y resolver que los probabilísticos, sin embargo, algunos modelos probabilísticos no tienen una solución exacta de ahí que se utilicen los modelos determinísticos como aproximaciones de situaciones más complejas.

El criterio de selección se basa en la alternativa que produzca el mejor resultado (optimización), para ello se asignan valores precisos de utilidad a cada una de las alternativas presentes en el problema.

- ***Ambiente de riesgo***

Un problema de decisión está definido en un ambiente de riesgo cuando alguno de los elementos o factores que intervienen están sujetos a las leyes del azar, es decir, cuando cada decisión puede dar lugar a una serie de consecuencias a las que es posible asignarle una distribución de probabilidad.

En estos ambientes el decisor no conoce con seguridad el resultado asociado a cada una de las elecciones posibles que se le plantean, sino que cada alternativa lleva asociada un conjunto de resultados posibles los cuales tienen unas probabilidades, conocidas por quien toma la decisión, de suceder cuando el decisor escoge dicha alternativa. El conjunto de las decisiones no está totalmente ordenado respecto de los resultados que éstas producen, ya que sólo se conocen los posibles resultados que ocurrirían con ciertas probabilidades, y dado que la decisión ha de tomarse antes de conocer el resultado, las decisiones no pueden ser valoradas con un número como ocurría en el ambiente de certidumbre.

El criterio de selección se basa en la alternativa que produzca el mejor resultado esperado, esto es, el producto del resultado por su correspondiente probabilidad de ocurrencia. Estos problemas son resueltos utilizando la Teoría de la Probabilidad.

✦ ***Ambiente de incertidumbre***

Un problema de decisión está definido en un ambiente de incertidumbre cuando la información disponible sobre las distintas alternativas puede ser incompleta, vaga o imprecisa, lo que implica que la utilidad asignada a cada alternativa tenga que ser valorada de forma aproximada. En este ambiente no es posible asignar una distribución de probabilidad a cada alternativa, bien porque la distribución sea desconocida o porque no tenga sentido hablar de ella. Se posee información deficiente para tomar la decisión ya que algún elemento que forma parte del problema no es conocido con exactitud, lo que puede deberse a diferentes causas:

- Cuando se trabaja con valores aproximados.
- Cuantificación de aspectos cualitativos.
- Cuando se desconoce un valor concreto.

Los métodos clásicos no son adecuados para tratar situaciones en las que la incertidumbre se debe a la aparición de información vaga e imprecisa. Esto ha generado la necesidad de recurrir a la definición de nuevos modelos basados en la Teoría de los Conjuntos Difusos para modelar la incertidumbre.

Son muchos los criterios propuestos para su utilización en ambientes de incertidumbre: Wald, Maximax, Hurwicz, Sagave y Laplace.

2.1.8. Evaluación de los resultados

Tras la elección de la decisión y su posterior puesta en marcha es necesario evaluar si el problema se resuelve correctamente, es decir, si la decisión está teniendo el resultado esperado o no.

Si el resultado no es el esperado se debe mirar si es porque es necesario dar un poco más de tiempo para obtener los resultados o si definitivamente la decisión no fue la acertada, en este caso se debe iniciar el proceso de nuevo para hallar una nueva decisión.

El nuevo proceso que se inicie en caso de que la solución haya sido errónea, contará con más información y se tendrá conocimiento de los errores cometidos en el primer intento.

Además se debe tener conciencia de que estos procesos de decisión están en continuo cambio, es decir, las decisiones que se tomen continuamente van a tener que ser modificadas, por la evolución que tenga el sistema o por la aparición de nuevas variables que lo afecten.

2.2. Toma de decisiones multicriterio

Con el desarrollo de este epígrafe se pretende dar una visión detallada sobre la toma de decisiones multicriterio, mostrando los pilares sobre los que se basa y como aparece. A su vez, se distinguen las etapas en las que se divide el proceso, los diferentes enfoques de los que consta, los problemas para llevarla a la práctica y una pequeña reflexión sobre qué enfoque aplicar según el problema a resolver.

2.2.1. Antecedentes

La toma de decisiones es una de las actividades de los seres vivos en la que mejor se aprecia su nivel de evolución y organización. Para los seres humanos, decidir es uno de los tópicos que más se han tratado desde todos los puntos de vista (filosóficos, sociológicos, psicológicos, económicos, etc.) y que mejor refleja su conocimiento, procedimiento y su grado de libertad.

En el pasado la toma de decisiones se efectuaba basándose en el binomio *experiencia-intuición*. A medida que la complejidad de los problemas ha ido creciendo, es decir, conforme las situaciones contempladas han sido menos estructuradas e intervienen numerosos escenarios, actores y factores, el binomio ha sido el de *información-razonamiento*, aunque en los últimos años se está considerando el de *conocimiento-razonamiento* [12].

Estos dos últimos términos sintetizan los aspectos más destacados a la hora de resolver un problema de toma de decisiones en los que la incorporación del factor humano en el proceso de resolución es fundamental para su correcta solución.

El término r*azonamiento* hace referencia a su sentido más clásico, esto es, a la aplicación del método científico en la resolución de problemas. Por otro lado, el término *conocimiento* hace referencia a las creencias, ideas, reglas y procedimientos generalmente ciertos en un dominio particular, es decir, a la interpretación dada a la información existente dentro de un dominio específico [12].

Este binomio, *conocimiento-razonamiento*, refleja la integración de lo *racional* del proceder científico en la toma de decisiones con lo *emocional* del comportamiento humano.

2.2.2. Crítica al paradigma de decisión tradicional

Los procesos de toma de decisiones se han venido analizando tradicionalmente siguiendo un paradigma que puede esquematizarse de la siguiente forma [9]. En primer lugar, se establece el conjunto de soluciones factibles al problema que se está analizando. A continuación, basándose en un cierto criterio, se asocia a cada alternativa un número que representa el grado de deseabilidad que tiene cada solución para el decisor, es decir, se ordenas las alternativas. Finalmente, utilizando técnicas matemáticas más o menos sofisticadas, se pro-

cede a buscar entre las soluciones factibles aquella que posee mayor grado de deseabilidad, la solución óptima.

Los problemas abordados por medio de la programación matemática siguen esta misma estructura. Las soluciones posibles son aquellas que satisfacen las restricciones del problema. Estas decisiones posibles se ordenan con arreglo a un cierto criterio que representa las preferencias del decisor, está función de criterio recibe el nombre de función objetivo. Por último, recurriendo a técnicas matemáticas se elige la solución óptima.

Esta estructura paradigmática posee una gran solidez desde el punto de vista lógico, ahora bien, desde un punto de vista empírico presenta importantes debilidades que lo desvía de los procesos reales de toma de decisiones [9]. En muchos casos de la vida, los decisores no están interesados en ordenar las soluciones factibles con arreglo a un único criterio, sino que desean efectuar esta tarea en relación a diferentes criterios que reflejen sus particulares preferencias. Por ejemplo, una gran empresa desea establecer sus decisiones óptimas no sólo en base al criterio del beneficio, sino que otros criterios como volumen de ventas, riesgo, etc. también les interesan.

2.2.3. Paradigma multicriterio

Desde el comienzo de los años 70, surge en la toma de decisiones el *paradigma multicriterio*. En sus orígenes intentó solventar algunas de las limitaciones del enfoque clásico, permitiendo la consideración de múltiples criterios. En la actualidad sus aspiraciones son mucho mayores.

En el pasado se entendía que el objetivo de la toma de decisiones multicriterio era la resolución de problemas de decisión complejos donde los criterios y objetivos pueden ser múltiples, pero en la actualidad, se considera que el objetivo primordial de la decisión multicriterio es el de asistir en el proceso de toma de decisiones.

Se entiende por *Decisión Multicriterio* el conjunto de aproximaciones, métodos, modelos, técnicas y herramientas dirigidas a mejorar la calidad integral de los procesos de decisión seguidos por los individuos y sistemas, es decir, mejorar la efectividad, eficacia y eficiencia de los procesos de decisión e incrementar el conocimiento de los mismos permitiendo trabajar con problemas complejos en los que intervienen múltiples criterios o factores, normalmente en conflicto entre sí [12]. De esta manera, las técnicas de decisión multicriterio permiten una resolución más realista y efectiva de los problemas sin necesidad de recurrir, como ocurre en otros enfoques más tradicionales, a la rígida reducción a una escala monetaria.

Los criterios que aparecen en este tipo de problemas pueden ser de diversa naturaleza: de carácter social, económico, financiero, ecológico y tecnológico.

Los problemas de decisión que pertenecen al ámbito económico, político, financiero, industrial o social, son casi siempre problemas de decisión multicriterio. De este modo el

problema de selección u ordenamiento de un conjunto de alternativas factibles sometidas a una evaluación multicriterio, no resulta un problema sencillo de resolver. Normalmente no existe una alternativa que sea la mejor para todos los criterios de forma simultánea, por tanto se suele elegir una alternativa óptima como solución de compromiso.

Dado que la mayoría de los problemas de decisión poseen una naturaleza multicriterio carece de sentido seleccionar una decisión considerando únicamente un criterio, en su lugar, deben considerarse simultáneamente todos los criterios muchos de los cuales se encuentran en conflicto entre sí, es decir, el acercamiento a la consecución de algún criterio implica el alejamiento de otro.

La solución de un problema multicriterio no depende solamente de la naturaleza del problema sino también del propio decisor. Cada decisor asigna un importancia relativa a cada uno de los criterios seleccionados.

Debido a la naturaleza conflictiva de los criterios muchas de las alternativas de un problema multicriterio son incomparables entre sí. Esto obliga al decisor a investigar bajo qué criterios las alternativas presentan un buen comportamiento y bajo cuáles su desempeño es deficiente, para proceder luego a efectuar la mejor elección de acuerdo a su propio esquema de preferencias. Existen modelos que se utilizan para la resolución del problema de decisión multicriterio que soportan trabajar con alternativas que no se pueden comparar, mientras que hay otros que no.

2.2.4. Conceptos

Como primer paso para la exposición del paradigma muticriterio se va a introducir una serie de conceptos y definiciones [23]. Hay que indicar que algunos de los términos que se van a explicar a continuación tienen el mismo significado semántico y en el contexto de algunos problemas se pueden utilizar como sinónimos. Sin embargo dentro del ámbito de las decisiones multicriterio es necesario establecer ciertas matizaciones.

- ***Atributo***

Este concepto se refiere a una propiedad de las soluciones a un problema dado, es decir, las características que definen las alternativas. Pueden expresarse mediante una función matemática.

- ***Objetivo***

Representa cómo se considera que el valor de un atributo mejora, es decir, si se desea que el atributo aumente (maximizarlo) o que disminuya (minimizarlo). Por consiguiente, los objetivos implican la maximización o minimización de las funciones que corresponden a los atributos.

- ***Meta***

Una meta es la asociación de un atributo con un valor numérico, que representa un nivel satisfactorio para el valor del atributo. Las metas se expresan mediante desigualdades matemáticas.

Un ejemplo rápido para entender mejor los tres términos anteriores puede ser el siguiente: el beneficio es un atributo, obtener el máximo beneficio posible es un objetivo y alcanzar un beneficio al menos igual a un determinado nivel de aspiración es una meta.

- ***Criterio***

Finalmente, el término criterio se utiliza como un término general que engloba los tres conceptos precedentes. En otras palabras, los criterios constituyen los atributos, los objetivos o metas que se consideran relevantes para un cierto problema de decisión. Por consiguiente, la teoría de la decisión muticriterio constituyen un marco general en el que subyacen diferentes atributos, objetivos o metas.

2.2.5. Esquema general del proceso

La Figura 2.2 recoge, de forma general, las principales fases de un proceso de toma de decisiones multicriterio [10].

El analista se halla presente en todas las fases. Asimismo, el decisor interviene prácticamente en todo momento, bien para recopilar la información o bien para validar el trabajo del analista. Aunque el esquema, inicialmente, puede dar la impresión de un desarrollo lineal, la mayoría de las fases son susceptibles de llevarse a cabo de forma recursiva ya que numerosas razones pueden provocar una vuelta hacia fases ya superadas.

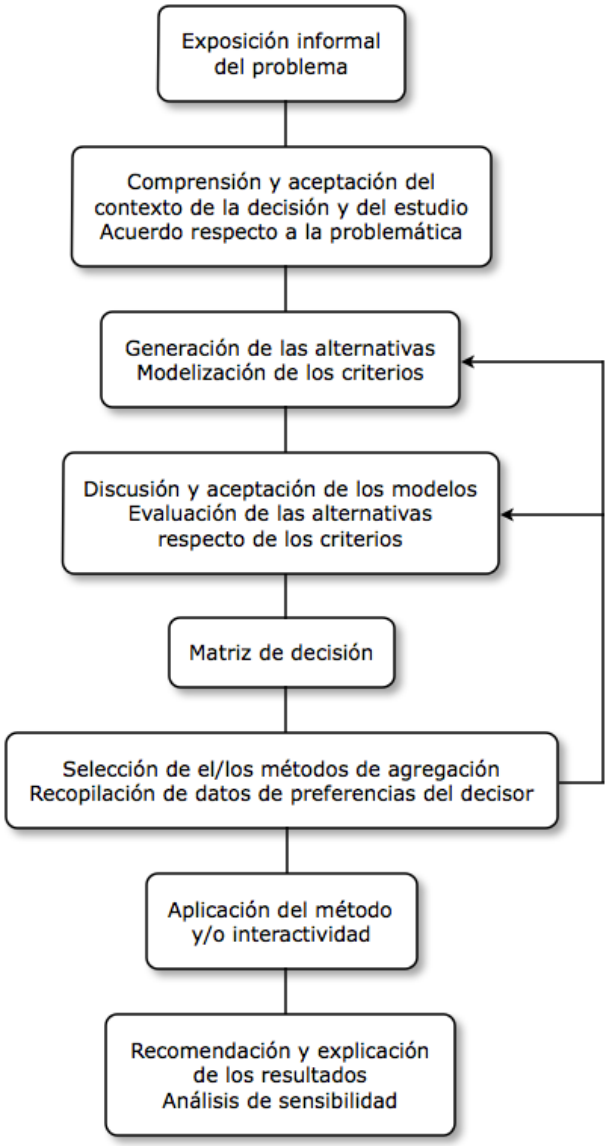


Figura 2.2. Fases de un proceso de toma de decisiones.

2.2.6. Formulación de un problema multicriterio

La formulación de un problema multicriterio puede expresarse del siguiente modo [11]:

$$Max\,\{f_1(a), f_2(a), ..., f_k(a) \;/\; a \in A\}$$

donde:

A es el conjunto de alternativas factibles.

y $\{f_i(a), j = 1, ..., k\}$ un conjunto de criterios de evaluación para la alternativa a.

Si bien se expresa como un problema de maximización, lo normal es que algunos criterios deban maximizarse y otros minimizarse al mismo tiempo, lo cual no representa ningún obstáculo para su consideración.

Dentro de un problema multicriterio como el anterior la relación de dominancia se define del siguiente modo:

$$\begin{cases} \begin{cases} \forall\ j\colon f_j(a) \geq f_j(b) \\ \exists\ h\colon f_h(a) > f_h(b) \end{cases} & \Leftrightarrow aPb\ (a\ es\ preferida\ sobre\ b) \\ \forall\ j\colon f_j(a) = f_j(b) & \Leftrightarrow aIb\ (a\ es\ indiferente\ con\ b) \\ \begin{cases} \exists\ s\colon f_s(a) > f_s(b) \\ \exists\ r\colon f_r(a) < f_r(b) \end{cases} & \Leftrightarrow aRb\ (a\ es\ incomparable\ con\ b) \end{cases}$$

Así se pueden identificar las alternativas dominadas, indiferentes o incomparables entre sí. Aquellas alternativas que no son dominadas se denominan soluciones o alternativas eficientes, sin embargo, la identificación de las alternativas eficientes no resuelve el problema al decisor, ya que es imposible concluir sin alguna información sobre sus preferencias.

2.2.7. Modelado de alternativas, criterios y pesos

Suponiendo que el problema de decisión está bien identificado y consensuado entre los diferentes agentes implicado, llega el momento de modelar tres de los factores más importantes a la hora de abordar un problema de decisión multicriterio como son las alternativas, los criterios y los pesos [10].

- ***Alternativas***

El paradigma multicriterio más simple supone que las alternativas ya están prefijadas antes de poner en marcha el método, pero en la práctica puede que esto no ocurra. Es por ello que cuando las alternativas están por construir se parte generalmente de un pequeño número de ideas del decisor. Son estas las ideas que es preciso ir enriqueciendo a fin de llegar a un conjunto suficiente de opciones. El enriquecimiento se realiza en dos direcciones:

a) En primer lugar, es necesario ampliar la elección de alternativas.

b) En segundo lugar, hay que añadir información en torno a cada alternativa.

- ***Criterios***

Para resolver el problema multicriterio de un modo correcto es interesante que la familia de criterios elegidos sea coherente, para ello es necesario que se cumplan tres propiedades:

- Exhaustividad: no se ha olvidado ninguno de los atributos que permite discriminar entre las alternativas, es decir, si no existen pares de alternativas empatadas según todos los criterios de la familia. Es primordial que los criterios definidos recojan adecuadamente los atributos realmente importantes en la decisión.

- Coherencia: las preferencias globales del decisor son coherentes con las preferencias según cada criterio. Es decir, que sean a y b dos alternativas indiferentes para el decisor, entonces la mejora de a según un criterio y/o el empeoramiento de b según otro criterio, entraña para el decisor que a es preferible sobre b. Esta propiedad suele cumplirse si el decisor es racional.

- No redundancia: una familia que verifica las propiedades de exhaustividad y coherencia es no redundante si la supresión de uno sólo de de los criterios provoca que la familia restante ya no satisfaga las exigencias de coherencia y de exhaustividad. El mayor riesgo de la redundancia es la posibilidad de dar mayor importancia a un criterio sin advertir que interviene dos veces bajo formas más o menos próximas.

✦ ***Pesos***

La determinación de los pesos no es en absoluto una tarea sencilla de realizar. Los pesos deben determinarse conjuntamente y al mismo tiempo que las utilidades relativas a cada criterio. Hay dos aspectos a tener en cuenta:

- Por una parte, una visión global que haga depender a los pesos del conjunto de criterios y de las relaciones que puedan existir entre ellos.

- Y por otro lado, la conexión entre los pesos y las escalas utilizadas para medir la utilidad de cada alternativa.

2.2.8. Principales enfoques multicriterio

Una vez definidos los diferentes conceptos que aparecen en el ámbito de la toma de decisiones multicriterio es posible efectuar una primera aproximación metodológica a los diferentes enfoques multicriterio [9].

Se puede decir que cuando el contexto en el que se va a tomar la decisión está definido por una serie de objetivos a optimizar que deben de satisfacer un determinado conjunto de restricciones, entonces el enfoque multicriterio a considerar es la programación multiobjetivo. Desde un punto de vista práctico, la toma de decisiones multiobjetivo está asociada con problemas donde las alternativas no están determinadas a priori, por lo que el número de alternativas no es finito, siendo el propósito del decisor obtener la mejor alternativa con los recursos limitados de los que dispone.

Sin embargo, la utilidad del enfoque multiobjetivo se reduce considerablemente cuando los problemas de decisión alcanzan un tamaño relativamente elevado. Así, si se está analizando un problema de toma de decisiones con muchos atributos y un conjunto de restricciones ciertamente complejo, el problema puede resultar computacionalmente intratable. Por tanto, el uso de este tipo de enfoques ante problemas tan complejos debe evitarse para dar paso a enfoques que tengan una menor solidez teórica pero una operatividad muy superior. Dentro de esta línea es donde puede encuadrarse la toma de decisiones por metas.

En los dos enfoques multicriterio vistos hasta el momento el conjunto de alternativas a considerar por parte del decisor es infinito. Este tipo de problemas de decisión suelen denominarse problemas continuos dado el carácter matemáticamente continuo de las decisiones posibles. Frente a los problemas de decisiones continuos se encuentran los de tipo discreto, en los que el número de alternativas a considerar por parte del decisor es finito y normalmente no muy elevado. En efecto, existen multitud de contextos de decisión en los que el número de alternativas deben evaluarse en base a varios atributos y son en ellos donde se considera el enfoque multiatributo que es el que se va a abordar en este proyecto.

Existen una gran cantidad de métodos para la toma decisiones multiatributo, cada uno de ellos con sus características y campos más adecuados de aplicación. Es por ello que se pueden distinguir dos grandes categorías [13]:

- Métodos con solución a priori: son aquellos en los que la información es obtenida a priori a partir de los datos suministrados por los decisores.

- Métodos interactivos: son métodos progresivos, caracterizados porque el decisor se va desplazando de una solución a la siguiente de forma iterativa, según la información facilitada en cada etapa por él acerca de sus preferencias sobre las soluciones presentadas.

A continuación, se muestra la estructura de un problema multicriterio discreto, por lo que se introducen los siguientes elementos:

a) Un conjunto de m puntos (E_1, ..., E_m) que representan las posibles alternativas o elecciones que puede alcanzar el decisor.

b) Un conjunto de n puntos (A_1, ..., A_n) que representan los atributos o criterios relevantes para el correspondiente problema.

c) Un conjunto de m x n puntos(R_{11}, ..., R_{mn}) que representan el resultado alcanzado por cada alternativa en cada uno de los atributos considerados.

Por tanto, un problema de decisiones multicriterio puede representarse por medio de una matriz, conocida como matriz decisional y que presenta la siguiente estructura:

		Atributos		
		A_1	...	A_n
Alternativas	E_1	R_{11}	...	R_{1n}
	...	...	...	...
	E_m	R_{m1}	...	R_{mn}

Para abordar un problema multicriterio discreto, en primer lugar, hay que determinar respecto de cada atributo la correspondiente función de utilidad de cada alternativa para el decisor. Una vez conocidas las funciones de utilidad individuales, se calcula la función de utilidad multiatributo para cada alternativa teniendo en cuenta todos los criterios. De esta forma se consigue una ordenación completa del conjunto finito de alternativas, de modo que para cualquier par de alternativas bien la primera será preferida sobre la primera, la segunda sobre la primera o ambas resultarán indiferentes entre sí.

2.2.9. Dificultades prácticas de los métodos de decisión multicriterio

En los problemas que se analizan bajo distintos criterios que están en conflicto debido al grado de incertidumbre en el que se desarrollan, la decisión multicriterio es uno de los instrumentos más útiles y prácticos para poder obtener la mejor solución de compromiso. Además, permiten un sencillo planteamiento del problema lo que facilita su comprensión aún sin tener elevados conocimientos sobre el tema.

Sin embargo, la puesta en práctica de estos métodos de decisión multicriterio conllevan muchos y diferentes problemas que afectan a los distintos elementos que forman parte del problema de decisión, y no siempre son percibidos [14].

A continuación se detallan las mayores dificultades que se presentan a la hora de aplicar los modelos de decisión multicriterio:

- ***El modelo***

El primer paso a la hora de enfrentarse con un problema real es crear el modelo. El diseño del modelo precisa de la enumeración completa de las alternativas, de todos los criterios, las direcciones de mejora según el criterio sea a maximizar o minimizar, sus escalas de medida y los umbrales de preferencia o indiferencia según el tipo de criterio generalizado.

Los objetivos que pretende alcanzar el decisor generalmente son contradictorios, por lo tanto es necesario para poder obtener una solución estar dispuesto a conseguir algo menos de un objetivo para conseguir un poco más de otro.

Existe toda una fase previa de recogida y clasificación de la información y los datos disponibles, para posteriormente realizar la elección del método multicriterio a aplicar, ya que existe un amplio panorama de métodos de decisión cada uno de ellos con sus ventajas, inconvenientes, su posterior interpretación y discusión.

Los modelos que se pueden establecer deben ser sencillos y no alejarse de la realidad que representan, ya que de esta forma son más comprensibles para el decisor y le será más fácil llevarlos a la práctica. El analista tendrá que presentar todas las conclusiones posibles sobre el modelo, aunque éstas sean contradictorias entre sí, puesto que el decisor debe tener en cuenta todos los posibles resultados para realizar la elección.

- ***El decisor y el analista***

Uno de los primeros elementos de este modelo es el decisor. Este elemento es, en la realidad, una persona o conjunto de personas encargadas de tomar la decisión, aunque ellas mismas no conozcan el método de resolución y las técnicas más apropiadas, lo cual corresponde al analista.

El decisor tiene que elegir libremente y sin ningún tipo de coacción el modelo que quiere aplicar para resolver un problema concreto, sin embargo, pueden existir distintos decisores que se enfrentan al mismo problema y el modelo que plantean es diferente, ya que la evaluación de cada alternativa bajo cada uno de los criterios estará condicionada al criterio utilizado y por lo tanto, la solución a un mismo problema podría ser diferente y no tendría una solución única.

El analista es el segundo elemento que entra a formar parte de modelo. Él no expresa sus opiniones, sino que se limita a tratar la información disponible de la manera más objetiva posible, pero está claro que es la persona que realiza directamente la modelización, es decir, toma la iniciativa a la hora de interpretar la información suministrada por el decisor y la plasma en un modelo que elige él mismo.

- ***Las alternativas***

El siguiente elemento importante es el conjunto de alternativas posibles, dentro del que hay que elegir una. Este conjunto debe estar compuesto por alternativas mutuamente excluyentes entre sí, es decir, no deben tener ningún elemento común, de tal forma que al obtener la solución si existen dos o más alternativas indiferentes entre sí, es porque proporcionan al decisor la misma utilidad y podrá elegir cualquiera de ellas. Si las alternativas con incomparables, entonces el decisor elegirá la alternativa mejor en función del criterio al que se asigna una mayor importancia.

Las alternativas tendrán que ser un conjunto de elementos bien definidos desde el espacio de resultados, además el decisor debe elegir una de ellas no pudiendo tomar una alternativa intermedia entre dos de las propuestas. Si además se necesitara introducir una nueva alternativa dentro de conjunto, el análisis debe volver a realizarse desde un principio.

- ***Los criterios***

La manera según la cual el decisor expresa su preferencia respecto de cada alternativa a elegir es siempre a través de la evaluación de cada una de éstas bajo cada criterio. Estos criterios pueden ser, en principio, dependientes entre sí, lo que daría lugar a que unos criterios sean más o menos importantes que otros. El problema se plantea a la hora de calcular exactamente la correlación o dependencia que existe entre los diferentes criterios y cómo afecta al problema trabajar con criterios que no son independientes.

Como ya se indicó anteriormente las escalas de medida de las alternativas (escalas de preferencia) con los diferentes criterios pueden ser cualitativas o cuantitativas. Es muy importante señalar que cuando los criterios son cualitativos puede ser difícil hacer valoraciones precisas y/o exactas.

La representación de los diferentes puntos de vista, aspectos, factores y características por medio de una familia de criterios es una de las etapas más delicadas de las formulación de un problema de decisión. Los criterios que aparecen en un problema de decisión son de diversa naturaleza para poder reflejas los distintos ámbitos económico, político, social, cultura, medioambiental, etc.

- ***Las ponderaciones***

Las ponderaciones o pesos de los criterios que intervienen en la mayoría de los métodos multicriterio son de vitales para poner de manifiesto la importancia relativa de éstos, y se obtienen de forma bastante subjetiva por parte del decisor junto a la ayuda del analista.

Es preciso tener en cuenta el método utilizado para ponderar los criterios, ya que en determinados casos basta con cambiar la escala de alguno de los criterios para tener que cambiar su peso relativo. A veces la estimación de las ponderaciones no se hace en términos de importancia (aunque es lo más frecuente), sino teniendo en cuenta la ganancia de un criterio que permite compensar la pérdida de otro.

- ***Los umbrales***

En algunos modelos de resolución de problemas multicriterio es necesario asignar un criterio generalizado junto con los correspondientes umbrales, tanto de indiferencia como de preferencia estricta, asignados a cada uno de los criterios para establecer la preferencia o indiferencia de cada una de las alternativas.

- ***Validación del modelo y análisis de sensibilidad***

Esta etapa consiste en analizar los modelos ideados considerando como de buena es la solución que proporcionan ante los problema reales para los que se han diseñado. Es decir, un modelo está validado si su comportamiento representa de forma adecuada el del sistema real, sujeto a toda las condiciones impuestas.

El estudio de las variaciones de las soluciones ante pequeños cambios en los valores de los parámetros escogidos es fundamental para todo análisis.

Asociados a los pesos, se tienen sus correspondientes intervalos de estabilidad que indican entre qué valores pueden variar las ponderaciones sin que la elección final se vea afectada. Sin embargo, en los métodos de decisión multicriterio, a diferencia de otros métodos, no sólo interesa cuál es la mejor alternativa sino también cuál es el ordenamiento total o parcial que existe entre ellas.

CAPÍTULO III

PROMETHEE y AHP

3.1. Toma de decisiones con información precisa

Hasta el momento, se ha visto que para resolver problemas de decisión existen disponibles una gran variedad de procesos de toma de decisiones para aquellas personas que quieran utilizarlos. Estos modelos facilitan que los decisores tomen una decisión de forma razonada, mediante la utilización de un proceso construido sobre una base matemática sólida que la sustente. La elección de un método u otro depende de las características propias inherentes al problema y del método que el decisor pretenda aplicar, teniendo en cuenta que cada técnica lleva asociada sus propias particularidades.

Con indiferencia del método que se aplique, todos ellos requieren cierta información proporcionada por el experto que se usa para hallar una solución al problema, y el modo en que se expresa dicha información es lo que se conoce como modelado de preferencias. El modelado de preferencias es una de las actividades inevitables en los problemas de toma de decisión, independientemente del área en la que se esté trabajando. Los expertos, en base a su conocimiento, experiencias y creencias han de emitir sus valoraciones sobre el conjunto de alternativas y establecer un orden de preferencia sobre la idoneidad de cada una de ellas como solución al problema.

Los expertos pueden decidir sobre la utilización de modelos de representación de preferencias que les resulten cercanos a sus disciplinas o campos de trabajo. Además, adaptar el modelado de preferencias al contexto en el que se desarrolla el problema de decisión consigue que los expertos se sientan más seguros a la hora de valorar sus preferencias, y por tanto que la solución final tenga mayor garantía de éxito [24].

Hasta ahora, a lo largo del proyecto no se ha explicado cómo puede ser este tipo de información. En ocasiones se conocen con certeza valoraciones precisas sobre las preferencias respecto a las alternativas y sobre los juicios emitidos para la evaluación de los criterios, y es en estos casos cuando la información proporcionada por los expertos consiste en valores numéricos precisos. Otra veces no se conocen con certeza estas valoraciones precisas y es cuando se utilizan valoraciones aproximadas como pueden ser valores en el intervalo [0, 1], otros valores intervalares o términos lingüísticos [22]. Cuando en los procesos de toma de decisiones se hace uso de valoraciones aproximadas se dice que la información utilizada es difusa. Por tanto, aquellos problemas donde los expertos modelan sus preferencia haciendo uso de información difusa se desarrollan en un ambiente de incertidumbre, de acuerdo a su definición. Este proyecto se centra únicamente en el tratamiento del modelado de preferencias con información precisa.

3.2. PROMETHEE

En los siguientes epígrafes, se procede a explicar el método PROMETHEE haciendo una introducción sobre él, indicando a su vez, los requerimientos que deben cumplirse para para poder aplicarlo y dando a conocer de forma detallada las dos fases de las que consta.

En la primera fase, se habla de la construcción de una relación de superación, haciendo especial hincapié en la noción de criterio generalizado y en algunos criterios ya existentes. En la segunda fase, se detalla cómo hacer uso de la relación de superación para clasificar las alternativas y obtener la solución al problema multicriterio.

3.2.1. Introducción

Dentro de los Métodos de Relaciones de Superación, ocupan un lugar muy destacado los llamados métodos Preference Ranking Organization METHod for Enrichment Evaluations (PROMETHEE) para la ayuda a la toma de decisiones multicriterio. Fue introducido por J. P. Brans [1, 2, 3] en la década de los 80.

Estos métodos nacen con el propósito de ayudar al decisor en los problemas de selección o de ordenamiento de alternativas posibles sometidas a una evaluación multicriterio, donde además los criterios se encuentran generalmente en conflicto entre sí.

Las principales características de estos métodos son su sencillez, limpieza y estabilidad. Además se hace uso de una noción generalizada de criterio que se usa para construir la clasificación entre alternativas. Todos los parámetros que se definen en este tipo de métodos tienen un carácter económico, lo que facilita que el decisor pueda fijarlos, de ahí que se hayan utilizado mucho en entornos económicos para análisis financiero.

Los métodos PROMETHEE pertenecen a la familia de los métodos de clasificación introducida por B. Roy, que incluyen dos fases [2]:

- La construcción de una clasificación que relacione las alternativas.

- La utilización de dicha clasificación para dar una solución al problema muticriterio.

Durante la primera fase, se considera una evaluación de la clasificación basada en la noción generalizada de criterio. Se define así un índice de preferencia para cada pareja de alternativas y se obtiene un grafo de clasificación que indica la preferencia del decisor entre las alternativas.

En la segunda fase, el uso de la relación de clasificación se realiza considerando para cada acción un flujo entrante y un flujo saliente en el grafo, siendo éstos los que van a permitir resolver el problema de decisión.

Inicialmente, se ofrecen dos posibilidades para resolver el problema de ordenamiento de alternativas: bien es posible obtener un preorden parcial (PROMETHEE I) o bien obtener un preorden completo (PROMETHEE II), ambos aplicados sobre un conjunto finito de acciones [1, 2, 3, 11]. Aunque dentro de PROMETHEE existen otros modelos que también se pueden aplicar como pueden ser PROMETHEE III, IV, V y VI [2, 11], únicamente los dos anteriores son los que interesan para el desarrollo de este proyecto.

3.2.2. Requerimientos del modelo

Para poder aplicar el método PROMETHEE a un problema multicriterio se deben tener en cuenta los siguientes requerimientos:

1. El decisor debe ser capaz de cuantificar la importancia de cada uno de los criterios.

2. El decisor debe poder cuantificar la importancia de cada alternativa respecto a cada criterio.

3. El decisor debe asumir que, dado que las importancias de los criterios están interrelacionadas, la modificación de una de ellas influye significativamente en el resto.

4. Debe tener sentido calcular la diferencia entre las diferentes evaluaciones de un criterio.

5. PROMETHEE no puede aplicarse a problemas con discordancia pues asume transitividad en la ordenación.

6. La adición o eliminación de una alternativa puede provocar cambios tanto en el preorden proporcionado por el método como en la solución final propuesta.

3.2.3. La información adicional

Existen un buen número de métodos de apoyo a la toma de decisiones multicriterio muy diversos entre sí, cada uno con sus respectivas características. Todos estos métodos afrontan el tratamiento del problema de un modo más o menos similar, pero varían entre sí de acuerdo al tipo de información adicional que requieren.

La ventaja de los métodos PROMETHEE frente al resto de sus competidores es que requiere información adicional muy clara y precisa, información que puede ser fácilmente obtenida por el decisor con la permanente y activa colaboración del analista [11].

Los métodos PROMETHEE fueron diseñados y llevados a la práctica para tratar problemas multicriterio donde el conjunto de alternativas es un conjunto finito de alternativas factibles. Es posible que surjan alternativas adicionales a medida que se obtiene mayor cantidad de información durante el proceso de decisión. Asimismo, como ocurre con el conjunto de acciones, también pueden aparecer nuevos criterios de evaluación o la eliminación de otros.

La información adicional requerida por los métodos PROMETHEE consiste en [11]:

- Información entre los distintos criterios (intercriterio).

- Información propia de cada criterio (intracriterio).

La información entre los distintos criterios consiste en el establecimiento de pesos o ponderaciones que reflejen la importancia relativa de cada uno de ellos. Así, un criterio será más importante que otro cuando su peso sea mayor. Los pesos se suponen siempre positivos y no existe ninguna objeción para considerar pesos normalizados.

La labor de determinación de los pesos no resulta trivial ni sencilla debido a que la componente subjetiva existente es muy fuerte. La selección de los pesos es llevada a cabo por el decisor, y es el ámbito donde él puede expresar libremente sus preferencias conforme a la estructura de las mismas en su mente.

La información propia de cada criterio se refiere a la forma en que el decisor percibe la escala específica en la que será expresado cada uno de ellos.

3.2.4. La relación de superación de PROMETHEE

La relación de superación de PROMETHEE es la clasificación que se construye tras la aplicación del modelo y que se encarga de evaluar las alternativas, para poder decidir cual de ellas es la solución más óptima al problema de decisión.

En primer lugar, se explica en qué consiste la noción clásica de criterio y cómo a partir de ella se extiende a la noción generalizada de criterio que se usa durante el desarrollo de este modelo. Después, se introducirán un conjunto de seis criterios generalizados que se recomiendan aplicar según el tipo de criterio que se pretenda representar. Y finalmente se presenta el cálculo del índice de preferencia multicriterio.

3.2.4.1. Noción clásica de criterio

La noción clásica de criterio define sobre un conjunto de alternativas K una estructura de preferencias {P, I}, donde P e I denotan la preferencia y la indiferencia respectivamente. Se define la noción clásica de criterio como [2, 11]:

$$\begin{cases} aPb \ sii \ f(a) > f(b) \\ aIb \ sii \ f(a) = f(b) \end{cases}$$

donde:

$a, b \in K$, son dos alternativas del conjunto total.

f es un criterio.

$f(a)$ y $f(b)$ son las evaluaciones de las alternativas de acuerdo al criterio f.

Este modelado de preferencias del decisor implica no hacer distinciones de preferencia para desviaciones entre los valores de $f(a)$ y $f(b)$. Además, la indiferencia se hace transitiva. Este modelo no se ajusta bien a la realidad, y es por ello por lo que algunos autores han introducido una extensión de la noción de criterio que refleje mejor el mundo real.

3.2.4.2. Extensión de la noción de criterio

Sea *K* el conjunto de alternativas generado para un problema de decisión multicriterio y $\Re$ el conjunto de los números reales, se define la noción de criterio generalizado como una aplicación *f* de *K* en $\Re$ [1, 2, 11]:

$$f: K \rightarrow \Re$$

donde *f* debe maximizarse o minimizarse en función del criterio que representa y el conjunto de los números reales puede sustituirse por cualquier otro conjunto ordenado. Es el decisor quien debe definir previamente la función f para cada criterio considerado.

Por tanto, para cada alternativa $a \in K$ se define *f(a)* como la evaluación de la alternativa *a* respecto del criterio *f*. De modo que hasta este momento se tiene:

- Un conjunto de alternativas como posibles soluciones al problema multicriterio (a_i).
- Un conjunto de criterios que se quieren maximizar o minimizar (f_i).
- Cada alternativa se puede evaluar en relación a un criterio determinado ($f_i(a_i)$).

Gracias a la evaluación que se puede realizar sobre cualquier alternativa atendiendo a un criterio determinado por medio de una función definida por el decisor, es posible comparar dos alternativas cualesquiera ($a, b \in K$), siendo posible expresar el resultado de esta comparación en términos de preferencia. Así, se puede considerar una función de preferencia entre alternativas que viene determinada por la siguiente aplicación:

$$P: K \; x \; K \rightarrow (0, 1)$$

Los resultados de esta función de preferencia respecto de cualesquiera dos alternativas $a, b \in K$ es la siguiente [2]:

- P(a, b) = 0: indiferencia entre *a* y *b*, o no preferencia de *a* sobre *b*.
- P(a, b) ~ 0: débil preferencia de *a* sobre *b*.
- P(a, b) ~ 1: fuerte preferencia de *a* sobre *b*.
- P(a, b) = 1: estricta preferencia de *a* sobre *b*.

En la práctica esta función opera como la diferencia entre el valor de las dos evaluaciones de las acciones, es decir:

$$P(a, b) = P(f(a) - f(b))$$

Una gráfica que representa la función de preferencia anterior tiene que ser una función no decreciente, igual a cero para valores negativos de *d = f(a) - f(b)*:

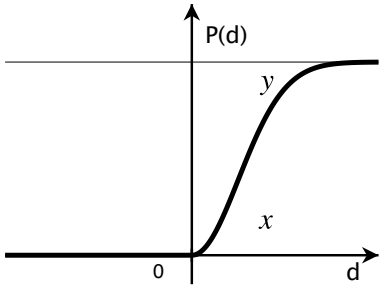


Figura 3.1. Función de preferencia P(d).

Pero hay que tener en cuenta que cuando se comparan dos alternativas atendiendo a un criterio determinado, éste se puede pretender maximizar o minimizar, por lo que el cálculo de *d* puede variar. Si el criterio se pretende maximizar el valor de *d* queda como:

$$d = f(a) - f(b)$$

Mientras que si se pretende minimizar el criterio el valor de *d* viene determinado por:

$$d = f(b) - f(a)$$

En los métodos PROMETHEE, para cada criterio se considera una noción generalizada de éste que viene definida por *f* y su correspondiente función de preferencia *P*.

3.2.4.3. Criterios generalizados recomendados

Para dar una mejor visión de la función anterior se puede definir una función *H(d)* que está directamente relacionada con la función de preferencia *P*. Consiste en generalizar *P* haciéndola una función simétrica respecto del eje Y, de este modo *H(d)* no sólo puede mostrar la preferencia de la primera alternativa sobre la segunda, sino también de la segunda sobre la primera. Entonces *H(d)* queda definida como [2]:

$$H(d) = \begin{cases} P(a,b) \; sii \; d \geq 0 \\ P(b,a) \; sii \; d < 0 \end{cases}$$

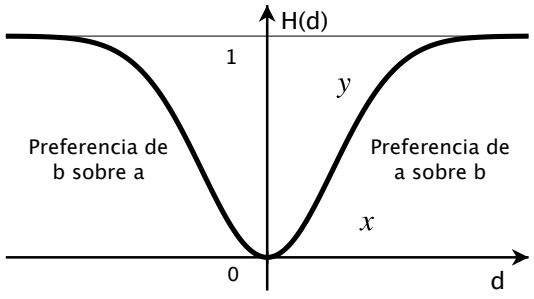


Figura 3.2. Función H(d).

Observando esta función se determina que para calcular la preferencia de *a* sobre *b* el valor de *d* debe ser obligatoriamente mayor o igual a cero, ya que en caso de ser negativo la preferencia de *a* sobre *b* sería cero. Ocurre lo mismo pero de forma análoga para calcular la preferencia de *b* sobre *a*.

La correspondencia entre cada criterio y su respectiva noción de criterio generalizado es llevada a cabo por el decisor. Para la elección de cada criterio se recomienda uno de los seis criterios generalizados que se proponen a continuación, cada uno con su correspondiente función de preferencia *P*. En general, estos seis criterios generalizados suelen ser suficientes en la mayoría de los casos prácticos [1, 2].

1. Criterio usual

Hay indiferencia entre dos alternativas *a* y *b* únicamente si *f(a) = f(b)*, en cualquier otro caso el decisor tiene una estricta preferencia por la acción que tiene la mayor evaluación.

No requiere que el decisor especifique ningún parámetro en particular. La generalización de este criterio se corresponde con el significado usual de criterio introducido por B. Roy.

$$H(d) = \begin{cases} 0 \; sii \; d = 0 \\ 1 \; sii \; d \neq 0 \end{cases}$$

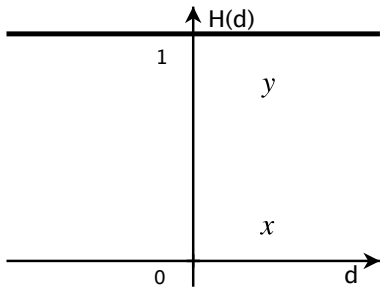


Figura 3.3. Criterio usual.

2. Quasi-criterio

En este caso el decisor determina un único parámetro *q* que representa un umbral de indiferencia. Las acciones *a* y *b* son indiferentes siempre y cuando la diferencia entre *f(a)* y *f(b)* esté dentro del umbral de indiferencia, es decir, entre *-q* y *q*. En cualquier otro caso hay una estricta preferencia de la acción con mayor evaluación sobre la otra.

$$H(d) = \begin{cases} 0 & sii\ -q \le d \le q \\ 1 & sii\ d < -q\ or\ d > q \end{cases}$$

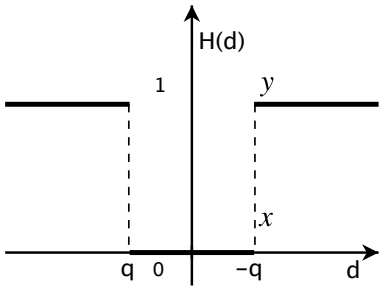


Figura 3.4. Quasi-criterio.

3. Criterio con preferencia lineal

De nuevo un decisor determina un parámetro p que representa ahora el umbral de preferencia, que en este caso sigue una función lineal. Si d es menor que p, la preferencia del decisor incrementa linealmente con d, en cambio si d es mayor que p se da una relación de estricta preferencia de la acción que mayor evaluación tiene. Se define del mismo modo para el caso simétrico.

$$H(d) = \begin{cases} \dfrac{|d|}{p} & -p \le d \le p \\ 1 & d < -p\ or\ d > p \end{cases}$$

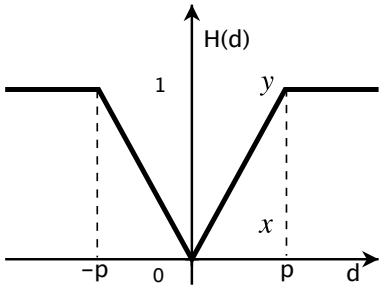


Figura 3.5. Criterio con preferencia lineal.

4. Criterio con nivel

En esta ocasión se define un rango de indiferencia q y un rango de preferencia p, si el valor de d se sitúa entre q y p se produce una ligera preferencia ($H(d)=1/2$). Si el valor es menor que q hay una situación de indiferencia y si el valor está por encima de p hay una estricta preferencia. Si el decisor lo considera necesario puede definir más de dos rangos (p y q).

$$H(d) = \begin{cases} 0 & sii\ |d| \le q \\ \dfrac{1}{2} & sii\ q < |d| \le p \\ 1 & sii\ p < |d| \end{cases}$$

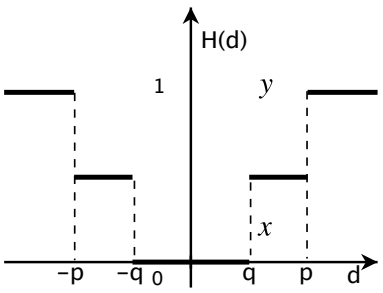


Figura 3.6. Criterio con nivel.

Hasta el momento se han descrito dos tipos de umbrales:

- Umbral de indiferencia (q): que representa el mayor valor de d, por debajo del cual hay indiferencia.
- Umbral de preferencia (p): que representa el menor valor de d, por encima del cual hay estricta preferencia.

En la práctica estos dos umbrales no tienen porque ser necesariamente iguales.

5. Criterio con preferencia lineal y área de indiferencia

En este caso el decisor considera que la preferencia incrementa linealmente entre la indiferencia hasta la estricta preferencia entre los rangos q y p. En esta ocasión el decisor tiene que definir dos parámetros.

$$H(d) = \begin{cases} 0 & sii\ |d| \le q \\ \dfrac{|d| - q}{p - q} & sii\ q < |d| \le p \\ 1 & sii\ p < |d| \end{cases}$$

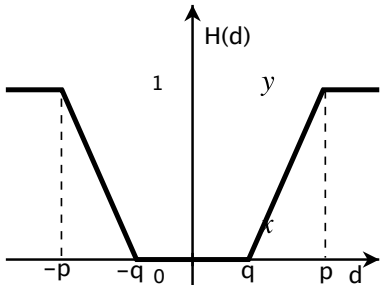


Figura 3.7. Criterio con preferencia lineal y área de indiferencia.

6. Criterio gaussiano

El único parámetro que el decisor debe determinar es σ, el cual se estima fácilmente según la distribución Normal. El valor de σ es la distancia entre el origen y el punto de inflexión de la curva. Esta función es continua lo que garantiza la estabilidad de los resultados.

$$H(d) = 1 - e^{-\frac{d^2}{2\sigma^2}}$$

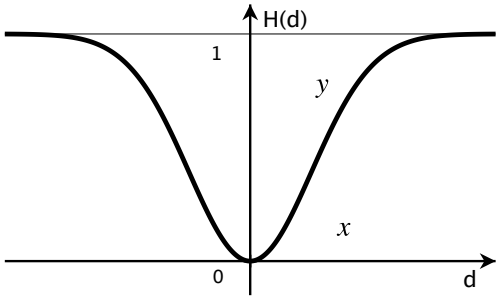
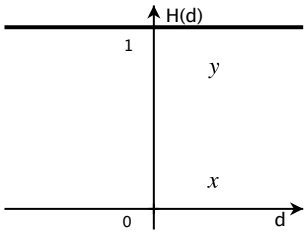
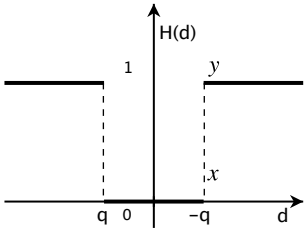
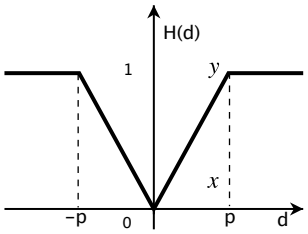
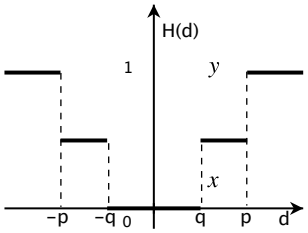
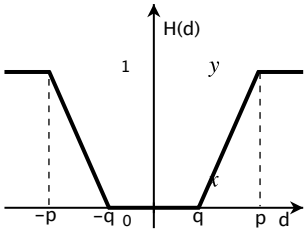
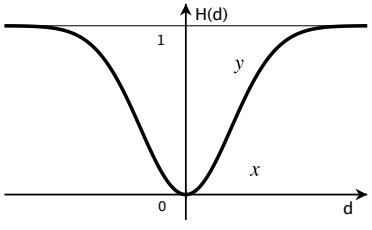


Figura 3.8. Criterio gaussiano.

A continuación, se presenta una tabla que resume los seis tipos de criterios generalizados vistos con anterioridad, así como los parámetros que el decisor tiene que ajustar:

Tipos de criterios generalizados		Parámetros
1. Criterio usual		-
2. Quasi-criterio		q
3. Criterio con preferencia lineal		p
4. Criterio con nivel		q, p
5. Criterio con preferencia lineal y área de indiferencia		q, p
6. Criterio gaussiano		σ

3.2.4.4. Índice de preferencia multicriterio

Para calcular el índice de preferencia multicriterio, en primer lugar, el decisor debe especificar la función de preferencia P y los pesos π_i de cada criterio considerado para el problema de decisión. Los pesos representan una medida de la importancia relativa de cada criterio para el decisor, por lo tanto si todos los criterios tuvieran la misma importancia su peso sería el mismo.

Sean $a, b \in K$ dos alternativas cualesquiera, se define para cada pareja de ellas el índice de preferencia multicriterio Π como la media ponderada de las funciones de preferencia P de cada de alternativas respecto de cada criterio considerado [1, 2, 11]:

$$\Pi(a,b) = \frac{\sum_{i=1}^{k} \pi_i P_i(a,b)}{\sum_{i=1}^{k} \pi_i}$$

$\Pi(a,b)$ representa la preferencia por parte del decisor de a sobre b considerando simultáneamente todos los criterios. Su valor se sitúa entre 0 y 1 e indica:

- $\Pi(a,b) \approx 0$: denota una ligera preferencia de a sobre b para todos los criterios.
- $\Pi(a,b) \approx 1$: denota una fuerte preferencia de a sobre b para todos los criterios.

Si a domina b entonces $\Pi(b, a) = 0$ pero eso no implica que $\Pi(a, b) = 1$, porque a puede ser preferente sobre b para cada criterio sin que la preferencia sea estricta.

Cada índice de preferencia determina una relación de valores entre cada par de alternativas de K. Eso implica que entre dos nodos (alternativas), a y b, haya dos arcos que toman como valores $\Pi(a, b)$ y $\Pi(b, a)$ formando lo que denomina el grafo de valores.

3.2.5. La clasificación de PROMETHEE

Para explicar como el modelo PROMETHEE realiza la clasificación de las alternativas, en primer lugar, es necesario construir un grafo de valores que las relacione. A continuación, se explica detalladamente en qué consiste y cómo se construye este grafo, para posteriormente ser utilizado en la aplicación de los métodos PROMETHEE I y/o PROMETHEE II según determine el decisor [1, 2, 3, 11].

3.2.5.1. Los flujos del grafo

A partir de los índices de preferencia multicriterio se puede construir un grafo valuado, donde los nodos representan las alternativas del problema y los arcos tienen los valores de sus respectivos índices:

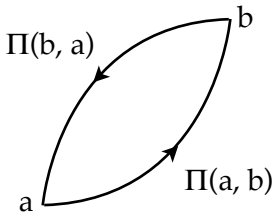


Figura 3.9. Flujos.

Para cada nodo del grafo se definen tres medidas: el flujo de salida, el flujo de entrada y el flujo neto.

✦ ***Flujo de salida***

Para cada nodo a del grafo se define un flujo de salida o positivo que no es más que la suma de los valores de los arcos que salen del nodo *Π(a, b)*:

$$\phi^{+}(a) = \sum_{b \in K} \Pi(a,b)$$

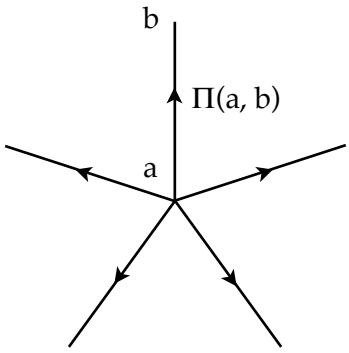


Figura 3.10. Flujo de salida.

Cuanto mayor sea el valor del flujo de salida más domina *a* al resto de alternativas de *K*.

✦ ***Flujo de entrada***

De forma similar, se define para cada nodo a del grafo un flujo de entrada o negativo como la suma de los valores de los arcos que llegan al nodo *Π(b, a)*:

$$\phi^{-}(a) = \sum_{b \in K} \Pi(b, a)$$

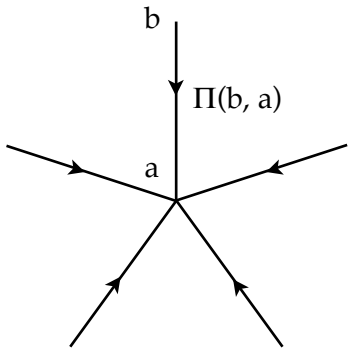


Figura 3.11. Flujo de entrada.

Cuanto menor sea el valor del flujo de entrada más domina *a* al resto de alternativas de *K*.

- ***Flujo neto***

También se define para cada nodo *a* del grafo el flujo neto, que es la diferencia entre el flujo de salida y de entrada del nodo:

$$\phi(a) = \phi^{+}(a) - \phi^{-}(a)$$

3.2.5.2. PROMETHEE I

La mejor alternativa sería aquella cuyos flujos salientes sean mayores y sus entrantes menores. El flujo saliente de cada alternativa induce el siguiente preorden:

$$\begin{cases} aP^{+}b & sii\ \phi^{+}(a) > \phi^{+}(b) \\ aI^{+}b & sii\ \phi^{+}(a) = \phi^{+}(b) \end{cases}$$

Del mismo modo el flujo entrante de cada alternativa induce el siguiente preorden:

$$\begin{cases} aP^{-}b & sii\ \phi^{-}(a) < \phi^{-}(b) \\ aI^{-}b & sii\ \phi^{-}(a) = \phi^{-}(b) \end{cases}$$

El método PROMETHEE I obtiene un preorden parcial (P_I, I_I, R) considerando la intersección de los dos preórdenes anteriores:

$$\begin{cases} aP_Ib & (a\ es\ preferida\ sobre\ b) & sii\ aP^+b\ y\ aP^-b \\ & & o\ aP^+b\ y\ aI^-b \\ & & o\ aI^+b\ y\ aP^-b \\ aI_Ib & (a\ es\ indiferente\ sobre\ b) & sii\ aI^+b\ y\ aI^-b \\ aRb & (a\ y\ b\ son\ incomparables) & en\ otro\ caso \end{cases}$$

Este preorden parcial es el propuesto por PROMETHEE I para que el decisor seleccione la mejor alternativa al problema multicriterio. Como se observa en el preorden parcial, en este método pueden existir alternativas que sean incomparables. Esta característica lejos de ser una desventaja proporciona al decisor información adicional sobre el problema, indicando que alternativas son incomparables debido a su naturaleza. Se considera que dos alternativas a y b son incomparables cuando a es buena bajo un conjunto de criterios para los cuales b es débil y viceversa.

3.2.5.3. PROMETHEE II

El método PROMETHEE II obtiene un preorden completo (P_{II}, I_{II}) que se induce a través de los flujos netos de los nodos considerándose:

$$\begin{cases} aP_{II}b & (a\ es\ preferida\ sobre\ b) & sii\ \phi(a) > \phi(b) \\ aI_{II}b & (a\ es\ indiferente\ sobre\ b) & sii\ \phi(a) = \phi(b) \end{cases}$$

Aunque es más fácil para el decisor utilizar el preorden completo para resolver el problema multicriterio ya que no aparecen alternativas que sean incomparables entren sí, el preorden parcial contiene información más completa y realista.

3.3. AHP

A continuación, se procede a hacer un análisis en profundidad sobre el método de toma de decisiones multicriterio AHP, haciendo una pequeña introducción acerca de su funcionamiento, sobre qué aspectos se basa y que ventajas presenta. A su vez, se presentan las etapas que lo componen y de un modo más profundo las bases matemáticas en las que se fundamentan, para finalizar mostrando la estructura que sigue el modelo y cómo se evalúa.

3.3.1. Introducción

El modelo Analytic Hierarchy Process (AHP) o proceso analítico jerárquico es una teoría general sobre juicios y valoraciones, basada en escalas de razón, que trata de sintetizar la naturaleza humana dentro del ámbito de la toma de decisiones. Fue diseñado e introducido por Thomas L. Saaty [4, 5, 6] a finales de los años 70.

El motivo por el que surge el modelo AHP es que gran parte del conocimiento y comportamiento humano puede explicarse en términos de comparaciones relativas expresadas en forma de proporción (ratios). De hecho, a los aspectos intangibles no se les puede asignar directamente un valor numérico, sin embargo, sí pueden ser medidos relativamente y tener sentido en función de otros criterios que permitan entenderlos mejor. Además, en cuanto al modo de expresar la realidad, los seres humanos habitualmente usan principios de orden jerárquicos para capturar y generalizar la información. Asimismo, se requieren escalas de razón para poder comprender el mundo humano.

El modelo AHP se utiliza para obtener escalas de proporción a partir de comparaciones pareadas ya sean discretas o continuas [5]. Estas comparaciones pueden llevarse a cabo utilizando medidas reales o bien escalas que reflejen la preferencia. El proceso de comparaciones pareadas no consiste en asignar números para ordenar las alternativas, una cosa es asignar un número a una magnitud medible y otra cosa es derivar un número a partir de las comparaciones entre intangibles homogéneos basadas en proximidad.

El proceso requiere que el decisor proporcione evaluaciones subjetivas respecto a la importancia relativa de cada uno de los criterios y que, después, especifique su preferencia con respecto a cada una de las alternativas de decisión y para cada criterio. El resultado de AHP es una jerarquización con prioridades que muestran la preferencia global para cada una de las alternativas de decisión.

El AHP, mediante la construcción de un modelo jerárquico, permite de una manera eficiente y gráfica organizar la información respecto de un problema, descomponerla y analizarla por partes, visualizar los efectos de cambios en los niveles y sintetizar.

Como tantas otras técnicas científicas, el proceso analítico jerárquico puede considerarse, según la orientación dada al mismo, de muy diversas maneras. Su contribución es importante en niveles operativos, tácticos y estratégicos, sirviendo para mejorar la eficiencia, eficacia y efectividad del sistema. En resumen, AHP puede entenderse como:

a) Una técnica que permite la resolución de problemas multicriterio, incorporando en el modelo los aspectos tangibles e intangibles, así como el subjetivismo y la incertidumbre inherente en el proceso de toma de decisiones.

b) Una teoría matemática de la medida generalmente aplicada a la dominación de la influencia entre alternativas respecto a un criterio.

c) Una filosofía para abordar, en general, la toma de decisiones.

3.3.2. Fundamentos

El proceso AHP se basa en los siguientes fundamentos [17, 18]:

- La estructuración del modelo jerárquico: representación del problema mediante identificación de meta, criterios, subcriterios y alternativas.
- Priorización de los elementos del modelo jerárquico.
- Comparaciones pareadas entre los elementos.
- Evaluación de los elementos mediante asignación de pesos.
- Ranking de las alternativas de acuerdo con los pesos dados.
- Síntesis.
- Análisis de sensibilidad.

3.3.3. Ventajas

El proceso analítico jerárquico presenta una serie de ventajas si se compara con algunos otros modelos de toma de decisiones multicriterio [17, 18]:

- Presenta un sustento matemático.
- Permite desglosar y analizar un problema por partes.
- Permite medir criterios cuantitativos y cualitativos mediante una escala común.
- Permite verificar la razón de consistencia y hacer las correcciones, si fuese el caso.
- Genera una síntesis y da la posibilidad de realizar análisis de sensibilidad.

3.3.4. Etapas del modelo

Saaty propone en su formulación inicial del proceso analítico jerárquico tres etapas que se incluyen en su metodología. Estas tres etapas son: modelización, valoración y priorización y síntesis [12].

1. Modelización

En esta primer etapa se construye un modelo o estructura en la que queden representados todos los aspectos considerados relevantes en el proceso de resolución: actores, escenarios, factores, elementos e independencias.

En su formulación inicial, AHP supone cuatros axiomas (se ven más adelante) y utiliza como estructura para modelizar el problema una jerarquía, en la que los elementos de un nivel no dependan de los descendientes ni hermanos. En el nivel superior de la jerarquía (nivel

0) se coloca la meta global considerada para el problema, y en los sucesivos niveles los demás aspectos relevantes. En el nivel 1 se colocan los criterios, que a su vez pueden dividirse en subcriterios, los cuales se sitúan en los niveles inferiores hasta que finalmente en el último nivel emplazan las alternativas. Esta jerarquía puede complicarse tanto como requiera el problema. La jerarquía resultante debe ser completa, representativa, no redundante y minimal. Su construcción es la parte más creativa del proceso de resolución.

En general, la modelización estructural del problema puede efectuarse en tres bloques. El bloque superior, modelizado mediante una red que recogería la parte menos estructurada y desconocida del problema, incluyendo los escenarios y actores así como sus interdependencias. El bloque intermedio, modelizado mediante una jerarquía recogería la parte semiestructurada del problema, incluyendo los atributos relevantes organizados en diferentes niveles de criterios. Por último, la parte inferior del modelo estructural, modelizada mediante una jerarquía (medidas relativas) o una tabla de valoraciones (medidas absolutas), recoge la parte más estructurada del problema.

2. Valoración

En esta segunda etapa, se incorporan las preferencias de los actores mediante los juicios incluidos en las denominadas matrices de comparaciones pareadas. Estas matrices cuadradas reflejan la dominación relativa de un elemento frente a otro respecto a un atributo en común.

Cuando se dispone de una unidad de medida, o escala, para evaluar la características considerada (aspecto tangible), se suele tomar la citada unidad y establecer el número de veces que el elemento en cuestión la contiene. En este caso las prioridades de las alternativas respecto al atributo se obtienen directamente. Sin embargo, si no se dispone de escala para la característica considerada (aspecto por el momento intangible) lo que se suele hacer para obtener las prioridades es recurrir a procedimientos relativos, comparando los elementos entre sí de forma pareada.

En la práctica, de los dos elementos comparados, se toma como referencia el que posee en menor medida la característica en estudio y se da una medida de las veces que el mayor incluye al menor.

Como es lógico, las medidas en diferentes escalas (tangibles o intangibles) no pueden agregarse directamente, luego para tratarlas conjuntamente se consideran todos los aspectos como si fueran intangibles, recurriendo a las comparaciones pareadas para derivar las prioridades relativas. Cuando se dispone de una escala (aspecto tangible), se toman como juicios las razones entre las mediciones, en cambio, si no se dispone de una escala (aspecto intangible), se usan como juicio los correspondientes a las comparaciones pareadas entre los elementos considerados.

Saaty propone la utilización de una escala fundamental para establecer los valores (juicios) correspondientes a las citadas comparaciones, considerando un rango entre 1 y 9, evi-

tando así el problema que se plantea cuando se realizan comparaciones relativas entre elementos con valores que van de cero a infinito como en las fórmulas matemáticas habituales.

3. Priorización y síntesis

Es la última etapa y proporciona las diferentes prioridades consideradas en la resolución del problema: prioridades locales, prioridades globales y prioridades totales. En general, se entiende por prioridad una unidad abstracta válida para cualquier escala en la que se integran las preferencias que el individuo tiene al comparar aspectos tangibles e intangibles.

Las prioridades locales son las de los elementos que cuelgan de un nodo común, están medidas en escalas de razón de las magnitudes relativas, y se obtienen a partir de la matriz recíproca de comparaciones pareadas. El proceso matemático seguido en su obtención es el método de autovector principal por la derecha propuesto por Saaty o el de la media geométrica por filas propuesto por Aguaron y Moreno-Jiménez [19].

Una de las grandes ventajas del proceso analítico jerárquico es que permite evaluar el grado de consistencia del decisor a la hora de introducir los juicios en las matrices recíprocas de comparaciones pareadas. Para evaluar la consistencia del decisor se calcula la denominada razón de consistencia (RC), un índice no estadístico que viene dado como el cociente entre el índice de consistencia (IC) y el índice de consistencia aleatorio (ICA). En apartados posteriores se explica con detalle como se calculan estos tres parámetros.

Posteriormente las prioridades locales obtenidas resolviendo el problema del autovector en cada uno de los nodos considerados en el problema, son transformadas en prioridades globales, esto es, conocida la importancia relativa, prioridad o peso de los elementos de un nivel respecto al atributo en común que sirve para compararlo, interesa determinar la importancia de estos elementos respecto a la meta global o misión fijada para el problema. La forma de transformar las prioridades locales en globales consiste en aplicar el principio de composición jerárquica.

El proceso de cálculo finaliza obteniendo para cada alternativa comparada en el problema su prioridad final en el mismo. Para obtener la prioridad final o total de una alternativa se agregan las prioridades globales obtenidas para esa alternativa en los diferentes caminos que la une con la meta global. El método habitualmente empleado en AHP para la agregación es el aditivo, pero alternativamente se han propuesto otros procedimientos de síntesis entre los más conocidos se encuentra el método de agregación multiplicativo.

3.3.5. Estructuración del modelo jerárquico

Como ya se indicó con anterioridad, una de las partes más relevantes de AHP consiste en la estructuración de la jerarquía del problema, etapa en la cual éste se desglosa en sus principales componentes.

La jerarquía básica está formada por: objetivo general, criterios, subcriterios (si fueran necesarios) y alternativas.

Los pasos a seguir para la estructuración del modelo jerárquico son: identificación del problema, definición del objetivo, identificación de los criterios e identificación de las alternativas [4, 5, 6, 12, 18].

3.3.5.1. Identificación del problema

Es la situación que se desea resolver mediante la selección de una de las alternativas de las que se dispone. Dichas alternativas serán comparadas unas con otras mediante la evaluación de criterios establecidos que permitan conocer los pros y contras incorporados por cada una de ellas.

Normalmente para identificar el problema se requiere cierto tiempo, lo cual puede darse después de una serie de discusiones en las que se han listado muchos problemas y es necesario decidir cual seleccionar para su análisis.

3.3.5.2. Definición del objetivo

Un objetivo es una dirección identificada para mejorar una situación existente. El objetivo está es un nivel independiente y los otros elementos de la jerarquía que serán subobjetivos o criterios, subcriterios y alternativas apuntan en conjunto a la consecución del mismo.

Hay objetivos de largo, mediano y corto plazo y esta diferenciación influirá directamente en la construcción del modelo jerárquico.

El objetivo o los objetivos serán establecidos por el decisor. Hay que tener en cuenta que la definición de los objetivos puede ser una tarea difícil porque algunas veces son contrapuestos entre las personas. No obstante, los objetivos determinados finalmente deben representar las necesidades e intereses generales.

3.3.5.3. Identificación de los criterios

Son los factores relevantes que afectan significativamente a los objetivos y deben expresar las preferencias de los implicados en la toma de decisión.

Se deben incluir aspectos cuantitativos y cualitativos a tener en cuenta en la toma de decisión. Normalmente hay aspectos cualitativos que pueden incidir fuertemente en la decisión, pero que no son incorporados debido a su complejidad para definirles algún esquema de medición que revele su grado de aporte en el proceso de toma de decisiones.

3.3.5.4. Identificación de las alternativas

Las alternativas corresponden a propuestas factibles mediante las cuales se podrá alcanzar el objetivo general del problema. Cada una de las alternativas presenta una serie de características que definen los pros y contras que aportan.

3.3.5.5. Árbol de jerarquías

Consiste en elaborar una representación gráfica del problema en términos de la meta global, los criterios y las alternativas de decisión [4, 18]. Esta gráfica recibe el nombre de árbol de jerarquías e ilustra la estructura del problema.

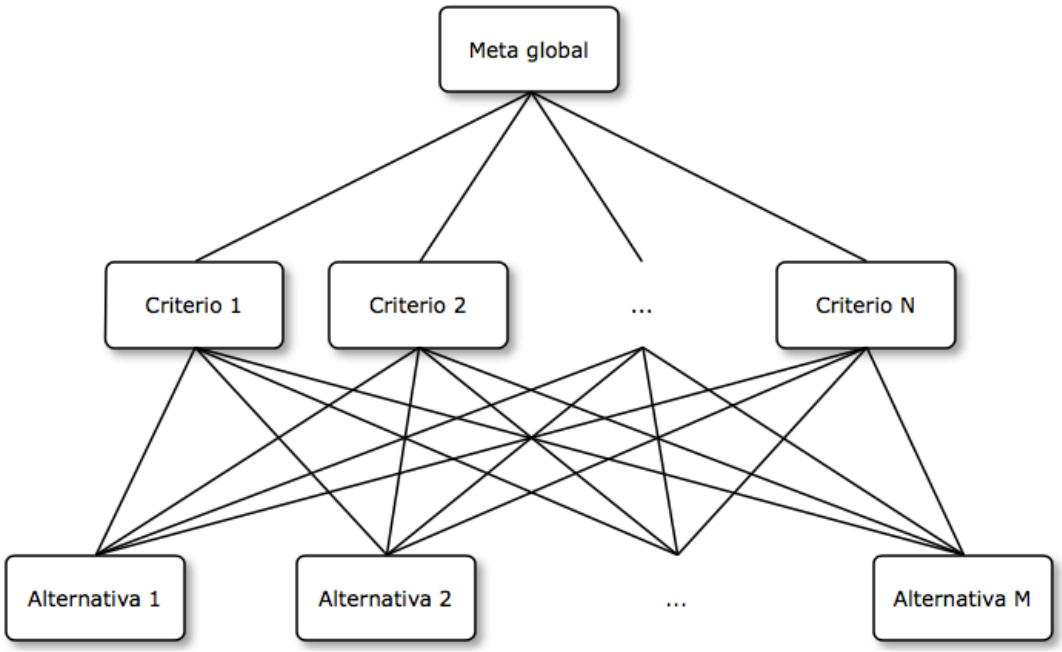


Figura 3.12. Árbol de jerarquías.

De forma general y muy resumida, se puede decir que el soporte que da comienzo al método AHP consiste en hacer que el decisor especifique sus opiniones con respecto a la importancia relativa de cada uno de los criterios en términos de su contribución al logro de la meta global.

Cuando se construye la jerarquía, se puede hacer de dos modos: de arriba hacia abajo o de abajo hacia arriba [18].

- La construcción de arriba hacia abajo se inicia con la identificación de los criterios más globales, es decir desde lo más general hasta lo más particular. De esta manera, todos los aspectos generales recopilados en la definición del problema están presentes en ese primer nivel a manera de criterios. Cada criterio identificado debe ir

acompañado de una descripción de lo que significa. Si se requiere, de los criterios pueden desprenderse subcriterios. Estos últimos deben guardar una relación jerárquica con el criterio del que se desprenden.

- En la construcción de abajo hacia arriba el proceso se desarrolla a la inversa. Primero se generan todas las características que permiten diferenciar entre las alternativas y posteriormente se construye el modelo jerárquico agrupando aquellas características que mantienen un factor común a manera de criterios o subcriterios, según sea el caso, hasta llegar al objetivo general.

El sentido en que se comienza a construir depende de los datos disponibles e inclusive del decisor. Si en la elaboración están definidas las alternativas y se conocen sus pros y contras, se puede iniciar el modelo ascendente. En caso contrario, se recomienda utilizar el modelo descendente puesto que es un enfoque para situaciones de planeación estratégica en donde los objetivos están más claros que las alternativas.

3.3.6. Evaluación del modelo

En la evaluación se examinan los elementos del problema aisladamente por medio de comparaciones entre pares. Las evaluaciones o juicios son emitidos por cada analista o grupo de interés.

De esta forma, el éxito en esta etapa dependerá de la inclusión de los grupos de interés o decisores que se verán representados en el modelos construido y podrán evaluar el modelo consensuado de acuerdo con sus intereses y necesidades propios.

Los pasos a seguir para la evaluación de los componentes del modelo jerárquico son: establecimiento de las prioridades y emisión de los juicios y evaluaciones.

3.3.6.1. Establecimiento de las comparaciones

El proceso analítico jerárquico utiliza comparaciones pareadas para establecer medidas de prioridad tanto para los criterios como para las alternativas de decisión.

3.3.6.2. Emisión de los juicios y evaluaciones

Los juicios son la base del proceso llevado a cabo por AHP. Los juicios pueden estar guiados por información científica, técnica y la dada por la experiencia y conocimientos del decisor útiles para evaluar los diferentes componentes del modelo. Es precisamente esta situación la que hace diferente a AHP de otros métodos, puesto que dentro de la evaluación del modelo se toman en cuenta los juicios u opiniones de cada individuo o grupo de interés involucrado en el proceso de toma de decisión.

Esta evaluación se realiza por medio de comparaciones entre pares que permiten conocer y medir las preferencias de los actores respecto a los diferentes componentes del modelo (criterios, subcriterios y alternativas).

Cada persona expresa su preferencia en términos de importancia asignando un valor numérico, el cual mide la intensidad de su preferencia. El método AHP dispone de una escala creada por el propio Saaty que mide los juicios emitidos por el decisor.

3.3.7. Resultado final

Una vez realizada el total de comparaciones se obtiene el resultado final consensuado, es decir, el ordenamiento de las alternativas. Este resultado está basado entonces, en las prioridades, en la emisión de juicios y evaluación hecha a través de las comparaciones del modelo jerárquico, llevada cabo por los actores.

3.3.7.1. Síntesis y análisis de sensibilidad

El AHP logra combinar todos los juicios u opiniones en un todo, en el cual las alternativas quedan organizadas desde la mejor hasta la peor.

El análisis de sensibilidad permite visualizar y analizar la sensibilidad del resultado, es decir la ordenación de las alternativas respecto de posibles cambios en la importancia de los criterios supuestos.

El análisis de sensibilidad debe responder a la pregunta ¿Qué pasa si...?, facilitando el análisis en aquellos procesos de toma de decisión en los que se requiere volver a aplicar el AHP en un corto o mediano plazo porque son procesos dinámicos que requieren ser revisados y ajustados en el tiempo ya que su entorno está en continuo cambio.

3.3.8. Base matemática

El método AHP trata directamente con preferencias entre pares de elementos en función de un atributo o criterio común representado en la jerarquía de decisión. En los siguientes apartados se detalla el proceso matemático seguido para llevar a cabo el proceso analítico jerárquico.

3.3.8.1. Establecimiento de prioridades

El proceso analítico jerárquico pide al decisor que indique una preferencia o prioridad respecto a cada alternativa de decisión en términos de la medida en la que contribuya a cada criterio [18].

Teniendo la información sobre la importancia relativa y las preferencias, se utiliza el proceso matemático denominado síntesis, para resumir la información y para proporcionar una jerarquización de prioridades de las alternativas, en términos de la preferencia global.

3.3.8.2. Comparaciones pareadas

Las comparaciones pareadas son la base fundamental de AHP. El proceso analítico jerárquico utiliza una escala subyacente con valores de 1 a 9 para calificar las preferencias relativas de los dos elementos que se comparan [4, 5, 12]. A continuación se presentan las escalas numéricas que se recomiendan para las preferencias verbales expresadas por el decisor.

Escala numérica	Escala verbal	Explicación
1	Igual importancia.	Los dos elementos contribuyen igualmente a la propiedad o criterio.
3	Moderadamente más importante un elemento que el otro.	El juicio y la experiencia previa favorecen a un elemento frente al otro.
5	Fuertemente más importante un elemento que el otro.	El juicio y la experiencia previa favorecen fuertemente a un elemento frente al otro.
7	Mucho más fuerte la importancia de un elemento que la del otro.	Un elemento domina fuertemente. Su dominación está probada en la práctica.
9	Importancia extrema de un elemento frente al otro.	Un elemento domina al otro con el mayor orden de magnitud posible.

Los valores intermedios como son el 2, 4, 6 y 8 suelen utilizarse en situaciones intermedias y las cifras decimales en estudios de gran precisión.

3.3.8.3. Matriz de comparaciones pareadas

Las matrices de comparaciones pareadas son matrices cuadradas que contienen comparaciones entre pares de alternativas o criterios [4, 5, 6, 12, 18].

Sea A una matriz de dimensión $n \times n$, donde $n \in Z^+$ y sea $a_{i,j}$ el elemento (i, j) de A con $i=1, 2, ..., n$ y $j=1, 2, ..., n$, se dice que A es una matriz de comparaciones pareadas de n alter-

nativas si $a_{i,j}$ es la medida de la preferencia de la alternativa de la fila *i* cuando se la compara con la alternativa de la columna *j*. Cuando *i=j*, entonces el valor de $a_{i,j}$ será igual a 1 puesto que se está comparando la alternativa consigo misma. Además se cumple que $a_{i,j}=1/a_{j,i}$ por lo que una matriz de comparaciones pareadas sigue la siguiente estructura:

$$A=\begin{pmatrix} 1 & a_{12} & \dots & a_{1n} \\ 1/a_{12} & 1 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1/a_{1n} & 1/a_{2n} & \dots & 1 \end{pmatrix}$$

De forma análoga, se dice que *C* es una matriz de comparaciones pareadas de *n* criterios si $c_{i,j}$ es la medida de la preferencia del criterio de la fila *i* cuando se le compara con el criterio de la columna *j*.

$$C=\begin{pmatrix} 1 & c_{12} & \dots & c_{1n} \\ 1/c_{12} & 1 & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1/c_{1n} & 1/c_{2n} & \dots & 1 \end{pmatrix}$$

Todo esto se sustenta debido a que AHP está basado en una fuerte base matemática la cual queda reflejada en cuatro axiomas [5, 18]:

1. Referido a la condición de juicios recíprocos: si *A* es una matriz de comparaciones pareadas se cumple que $a_{i,j}=1/a_{j,i}$.

2. Referido a la condición de homogeneidad de los elementos: los elementos que se comparan son del mismo orden de magnitud o jerarquía.

3. Referido a la condición de estructura jerárquica o estructura dependiente: existe dependencia jerárquica en los elementos de dos niveles consecutivos.

4. Referido a la condición de expectativas de orden de rango: las expectativas deben estar representadas en la estructura en términos de criterios y alternativas.

3.3.8.4. Síntesis

Una vez elaborada la matriz de comparaciones pareadas se puede calcular la prioridad de cada uno de los elementos que se comparan. A esta parte del método AHP se la conoce como síntesis o sintetización [4, 5, 18]. El proceso matemático preciso para realizar la sintetización requiere del cálculo de valores y vectores característicos. Un procedimiento que proporciona una buena aproximación de esta etapa es el que sigue:

1. Sumar los valores de cada columna de la matriz de comparaciones pareadas.

2. Dividir cada elemento de la matriz entre el total de su columna; a la matriz resultante se le denomina matriz de comparaciones pareadas normalizadas.

3. Calcular el promedio de los elementos de cada fila de la matriz de comparaciones pareadas normalizadas.

3.3.8.5. Vector y matriz de prioridades

Cuando se realiza el proceso de síntesis para la matriz de comparaciones pareadas de criterios se obtienen las prioridades de cada uno de ellos en términos de la meta global. Estas prioridades se representan en forma de vector de prioridades quedando:

$$\begin{array}{c} \\ \text{Criterio 1} \\ \text{Criterio 2} \\ \vdots \\ \text{Criterio m} \end{array} \begin{array}{c} \text{Meta Global} \\ \begin{pmatrix} P'_1 \\ P'_2 \\ \vdots \\ P'_m \end{pmatrix} \end{array}$$

donde m es el número de criterios y P'_i es la prioridad del criterio i con respecto a la meta global para $i=1, 2, ..., m$.

Del mismo modo, una vez finalizado el proceso de síntesis para las matrices de comparaciones pareadas de alternativas, se puede definir una matriz que recoja las prioridades respecto a cada criterio:

$$\begin{array}{c} \\ \text{Alternativa 1} \\ \text{Alternativa 2} \\ \vdots \\ \text{Alternativa n} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Criterio 1} & \text{Criterio 2} & \dots & \text{Criterio m} \end{array} \\ \begin{pmatrix} P_{11} & P_{12} & \dots & P_{1m} \\ P_{21} & P_{22} & \dots & P_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nm} \end{pmatrix} \end{array}$$

donde $P_{i,j}$ es la prioridad de la alternativa i con respecto al criterio j, para $i=1, 2, ...,n$ y $j=1,2, ..., m$.

La prioridad global para cada alternativa de decisión viene dada por el vector columna que resulta del producto de la matriz de prioridades de las alternativas con el vector de prioridades de los criterios [4, 5, 6, 12, 18]:

$$\begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1m} \\ P_{21} & P_{22} & \cdots & P_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nm} \end{pmatrix} \begin{pmatrix} P'_1 \\ P'_2 \\ \vdots \\ P'_m \end{pmatrix} = \begin{pmatrix} Pg_1 \\ Pg_2 \\ \vdots \\ Pg_n \end{pmatrix}$$

donde Pg_i es la prioridad global respecto de la meta global de la alternativa para *i=1, 2, ..., n*.

3.3.8.6. Consistencia

Una consideración importante en términos de calidad de la decisión final se refiere a la consistencia de los juicios emitidos por el decisor en el transcurso de las comparaciones pareadas. Hay que tener presente que lograr la consistencia perfecta es muy difícil y es de esperar cierta inconsistencia en casi cualquier conjunto de comparaciones pareadas.

AHP ofrece un método para medir el grado de consistencia entre las opiniones que proporciona el decisor. Si el grado de consistencia es aceptable, el proceso puede continuar, pero en caso contrario, quien toma las decisiones debe reconsiderar y posiblemente modificar sus juicios sobre las comparaciones pareadas antes de continuar.

Para determinar si una matriz de comparaciones pareadas es consistente el proceso analítico jerárquico calcula una medida conocida como la razón de consistencia (RC), que viene determinada por le cociente entre el índice de consistencia de A (IC) y el índice de consistencia aleatorio (IA) [4, 5, 6, 18]:

$$RC = \frac{IC}{IA}$$

A su vez, el índice de consistencia de A (IC) se calcula como sigue:

$$IC = \frac{n_{max} - n}{n - 1}$$

Para calcular n_{max} en primer lugar hay que calcular la suma de las columnas de la matriz A y posteriormente el vector de pesos de la misma matriz. Una vez hecho esto, n_{max} viene determinado por:

$$n_{max} = \sum_{i=1}^{n} a_i w_i$$

donde a_i es la suma de la columna *i* de la matriz *A* y w_i es el peso de la alternativa o criterio *i* de esa misma matriz de comparaciones pareadas.

Finalmente, hay que calcular el índice de consistencia aleatoria (IA), que representa el índice de consistencia de una matriz de comparaciones pareadas generada de forma aleatoria. Se puede mostrar que IA depende del número de elementos que se comparan, y toma los siguientes valores:

Número de elementos que se comparan	1	2	3	4	5	6	7	8	9	10
Índice aleatorio de consistencia	0	0	0,59	0,89	1,11	1,24	1,32	1,40	1,45	1,49

La razón de consistencia (RC) está diseñada de manera que los valores que exceden de 0,10 son señal de juicios inconsistentes, por lo que es probable que en estos casos el decisor desee reconsiderar y modificar los valores originales de la matriz de comparaciones pareadas. Según el valor de RC se considera que:

- Si RC $\leq$ 0,10 la matriz tiene una consistencia razonable.
- Si RC > 0,10 la matriz es inconsistente.

3.3.9. Aplicaciones

El proceso analítico jerárquico es una de las técnicas muticriterio con mayor implantación práctica en casi todos los ámbitos de la toma de decisiones. Sin entrar a estudiar en detalle cuáles son las causas que han motivado su gran aplicabilidad, mencionar que entre éstas, cabe citar las mismas ideas que sugirieron su metodología, esto es: la flexibilidad de la técnica, la adecuación a numerosas situaciones reales referidas a la selección multicriterio entre alternativas, su facilidad de uso, la posibilidad de aplicarla en decisión individual y en grupo, y por último, la existencia de *software* amigable para su aplicación.

Los ámbitos donde se hace uso del proceso analítico jerárquico son muy variados y diversos, pero a continuación se mencionan algunas de las áreas y tópicos donde se aplica el método AHP [12]:

- Sociedad, ciencia y educación.
- Economía y transporte.
- Localización y asignación de recursos.
- Decisiones empresariales y marketing.
- Aplicaciones ambientales.

- Producción.
- Sanidad.
- Decisión en grupo y resolución de conflictos internacionales.
- Planificación urbana.
- Desarrollo y evaluación del *software*.
- Selección de personal.
- Establecimiento de ponderaciones para otros modelos de toma de decisiones (por ejemplo: PROMETHEE).

3.3.10. Puntos críticos

Teniendo en cuenta que hasta la fecha no se ha podido probar que una técnica multicriterio sea mucho mejor que el resto en cualquier contexto de decisión, en todas ellas se puede encontrar aspectos positivos y negativos, bien desde el punto de vista teórico o práctico.

En el caso de AHP, como les sucede a todas las aproximaciones multicriterio discretas, existen una serie de controversias en los diferentes pasos de su metodología [12]. Entre éstas cabe destacar:

1. La justificación de la independencia exigida en la modelización jerárquica.
2. La escala fundamental usada para expresar los juicios relativos a las comparaciones pareadas.
3. Los procedimientos de priorización, síntesis y evaluación de la consistencia empleados.
4. La interpretación de las prioridades totales obtenidas en el procedimiento.
5. El problema del cambio de rango.

El problema de cambio de rango consiste en la posibilidad de cambio de la ordenación inicial obtenida para las alternativas consideradas, al añadir o eliminar de la ordenación inicial obtenida alguna alternativa "irrelevante" (una copia o cuasi-copia de otras). Según Saaty, el cambio de rango está permitido e incluso es natural en dos casos: en los problemas de asignación de recursos y cuando la aparición de alguna copia introduce en el problema la idea de abundancia o escasez de una determinada alternativa. Al margen de los dos casos anteriores, la introducción de una nueva alternativa puede hacer variar la estructura de preferencias del decisor, o poner de manifiesto la alguna inconsistencia en los juicios. Si la nueva

alternativa es una copia de las iniciales y se mantienen los juicios emitidos con anterioridad, estas valoraciones refuerzan las relaciones marcadas por la correspondiente alternativa.

Respecto a otro tópicos citados dentro de los aspectos controvertidos, mencionar que se han propuesto distintos procedimientos para calcular y sintetizar las prioridades, así como diferentes escalas para incorporar las preferencias a través de juicios.

En cuanto a la existencia de otros procedimientos de síntesis, el más extendido es el denominado forma multiplicativa ponderada, sin embargo Saaty afirma que éste método presenta cuatro grandes limitaciones: no devuelve valores en la misma escala de medida, supone que siempre la matriz de juicios es consistente, no generaliza el caso de interdependencia y retroalimentación y siempre preserva el ranking de las alternativas.

Respecto a la justificación de la independencia entre "hermanos", exigida para poder realizar la modelización jerárquica, al igual que ocurre en otras técnicas multicriterio no es sencilla. En AHP la independencia se justifica por la forma de modelizar y valorar.

Hay otro punto cuestionado en la modelización jerárquica efectuada en AHP, como es la influencia que el número de descendientes de cada nodo tiene en la prioridad final de los elementos considerados. Si todas las alternativas son evaluadas en función de todos los subcriterios este problema no suele presentarse, pues cada alternativa alcanzaría su proporción que se distribuye a lo largo de la jerarquía. En cambio, si las alternativas son evaluadas en función de parte de los subcriterios este problema puede afectar al resultado final. Para evitarlo se sugiere reescarlar el peso de los criterios con el número relativo de elementos bajo el mismo.

CAPÍTULO IV

Detalles del proyecto

4.1. Descripción

Una vez presentado el propósito y los objetivos del proyecto, así como explicado el modelo teórico sobre el que se fundamenta, se procede a detallar y explicar el proceso de Ingeniería del *Software* que se sigue para desarrollar la Plataforma web para la resolución y gestión de Problemas de Toma de Decisión Multicriterio: PROMETHEE y AHP.

El proyecto aborda el desarrollo de una plataforma web basada en la arquitectura cliente-servidor por medio del uso de la tecnología Java para su implementación.

El término *Ingeniería del Software* se define como la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del *software* [28]. Otro modo de definirla es como una disciplina de la ingeniería que comprende todos los aspectos de la producción del *software*, desde las etapas iniciales de la especificación del sistema hasta el mantenimiento del mismo.

Un proceso de *software* es un conjunto de actividades y resultados asociados que producen un producto *software* [25]. Estas actividades son llevadas a cabo por los ingenieros de *software*. Existen seis actividades fundamentales de procesos que son comunes para todos los procesos del *software*:

- Especificación de los requerimientos: los clientes e ingenieros definen el propósito del sistema, las propiedades y restricciones sobre su operación.
- Análisis: se define el *software* a producir obteniendo un modelo del sistema correcto, completo, consistente, claro y verificable.
- Diseño: se definen las estrategias a seguir para conseguir obtener un *software* que cumpla con sus objetivos.
- Implementación: el modelo se traduce a código siguiendo las estrategias definidas.
- Prueba: el *software* se verifica y valida para asegurar que es lo que el cliente requiere.
- Mantenimiento: el *software* se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

En los posteriores capítulos se profundiza en cada una de las actividades del proceso del *software*, explicando en qué consisten y cómo se han llevado a cabo en el ámbito de este proyecto. Obviamente, al tratarse de un proyecto fin de carrera cuyo fin es académico, carece de sentido realizar la actividad de mantenimiento.

4.2. Contexto de uso de la aplicación

En primer lugar, para la realización de un buen análisis y posterior diseño de la aplicación es necesario explicar el contexto en el que se va a hacer uso de la misma, es decir, bajo que entorno se va a utilizar, quien va hacer uso de ella y qué tareas va a realizar.

Puesto que el proyecto fin de carrera sobre la toma de decisiones muticriterio que nos atañe pertenece a un ámbito totalmente académico, no hay que perder de vista que el contexto dónde se hace uso de él tiene el mismo carácter.

A todo esto hay que añadir que el *software* que se desarrolla se trata de una plataforma web basada en la arquitectura cliente-servidor, por lo que cualquier usuario con conexión a Internet y un navegador web puede acceder a ella. Sin embargo, a pesar de los múltiples lugares desde los que se puede hacer uso de la aplicación, su uso queda restringido en principio a la universidad.

Asimismo, se pueden distinguir dos categorías generales de usuarios que hacen uso de la aplicación web:

a) Administrador: consiste en una o varias personas que se encargan de gestionar de un modo global la aplicación.

b) Expertos: dentro de la categoría de los expertos se pueden encontrar como posibles usuarios a profesores, investigadores, becarios, alumnos y en general cualquier persona perteneciente a la comunidad universitaria que necesite aplicar alguno de los modelos de toma de decisión multicriterio que se automatizan en este proyecto.

Normalmente este último tipo de usuarios están ligados al ámbito de la toma de decisiones y por tanto, poseen al menos unos conocimientos básicos sobre los modelos que se desarrollan, en qué consisten, cómo funcionan y cuándo se aplican, sin embargo, el administrador no es necesario que posea este tipo de conocimientos. Además, ambas categorías conocen los ordenadores, al menos de forma básica, ya que tienen experiencia previa con ellos y saben utilizarlos para navegar a través de diversas páginas web. Por último, decir también que de forma general los usuarios que usan la aplicación no presentan discapacidad psicomotriz alguna que les impida usar el ordenador y la aplicación de forma normal.

Una vez determinados los posibles usuarios de forma muy general y estrechamente relacionados con éstos se encuentran los lugares donde generalmente se utiliza la aplicación: despachos, laboratorios y aulas de prácticas de la universidad. Todos ellos se caracterizan por ser habitaciones normalmente amplias, con espacio suficiente para situar el ordenador y con asientos para poder interaccionar con la aplicación de forma cómoda y relajada. En estos lugares existen unas buenas condiciones de luminosidad ya sea natural o artificial y en ellos se dispone, para el usuario que hace uso de la aplicación, de al menos un ordenador relativamente moderno que consta como mínimo de monitor, ratón, teclado, acceso a Internet y un navegador web.

Las tareas que los administradores pueden llevar a cabo con la aplicación son la gestión de los problemas y de los expertos existentes en el sistema. En cuanto a los expertos, pueden resolver problemas de decisión multicriterio por medio de la aplicación del método PROMETHEE y/o AHP, así como gestionar de forma más detallada los problemas que definan.

4.3. Planificación temporal

Dentro de este apartado se muestra una estimación temporal de la realización del proyecto. Primero se realiza un análisis de las tareas del proyecto y después se estima la planificación temporal por medio de un diagrama de Gantt.

4.3.1. Estructura de descomposición del proyecto

La estructura de descomposición del proyecto refleja la división de las tareas que se van a llevar a cabo durante el desarrollo del mismo y sirven de guía para la estimación temporal. Viene determinada por la Figura 4.3.

4.3.2. Diagrama de Gantt

El diagrama de Gantt asigna una fecha y tiempo estimado a cada una de las tareas y muestra de forma gráfica en qué orden y cuándo se van completando.

	Nombre	Duración	Inicio	Fin	Predecesoras
0	**Plataforma web para la resolución y**	230d	**05/07/2010**	**20/05/2011**	
1	Búsqueda de bibliografía	1s	05/07/2010	09/07/2010	
2	Estudio de bibliografía	3s	12/07/2010	30/07/2010	1
3	Fin de la localización de información	0d	30/07/2010	30/07/2010	2
4	Aprendizaje de los modelos	1s	02/08/2010	06/08/2010	3
5	Análisis de la aplicación	4s	09/08/2010	03/09/2010	4
6	Diseño de la aplicación	6s	06/09/2010	15/10/2010	5
7	Diseño de la interfaz web	4s	18/10/2010	12/11/2010	6
8	Aprendizaje de Struts 2 e Hibernate	4s	15/11/2010	10/12/2010	3,7
9	Implementación	20s	13/12/2010	29/04/2011	8
10	Verificación y validación	2s	02/05/2011	13/05/2011	9
11	Prueba global de la aplicación	1s	16/05/2011	20/05/2011	10
12	Memoria del proyecto	230d	05/07/2010	20/05/2011	
13	Fin del proyecto	0d	20/05/2011	20/05/2011	11,12

Figura 4.1. Diagrama de Gantt. Parte I.

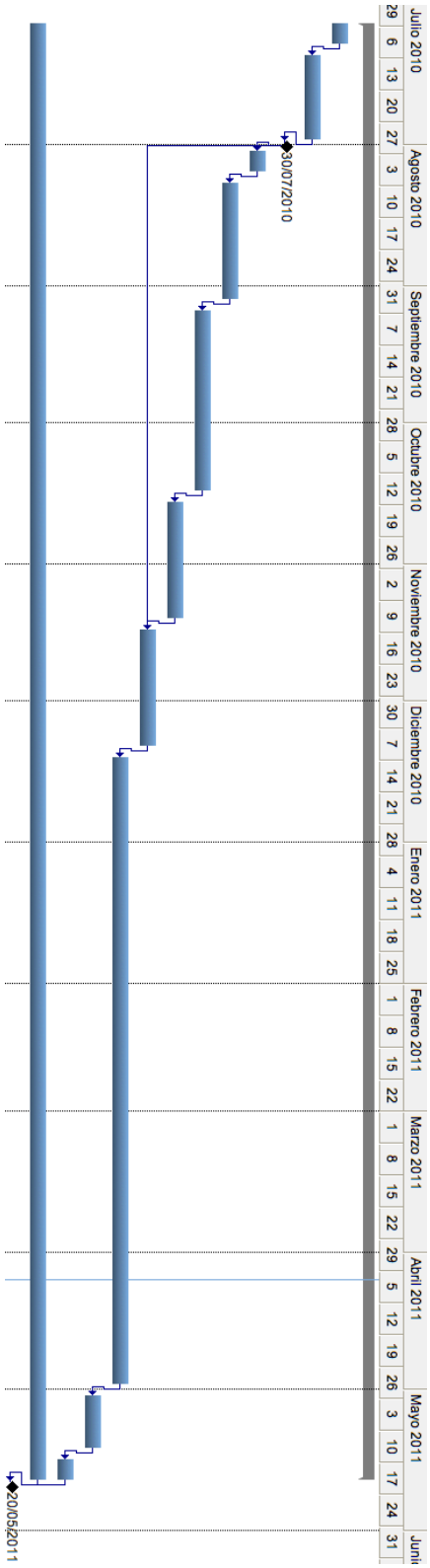


Figura 4.2. Diagrama de Gantt. Parte II.

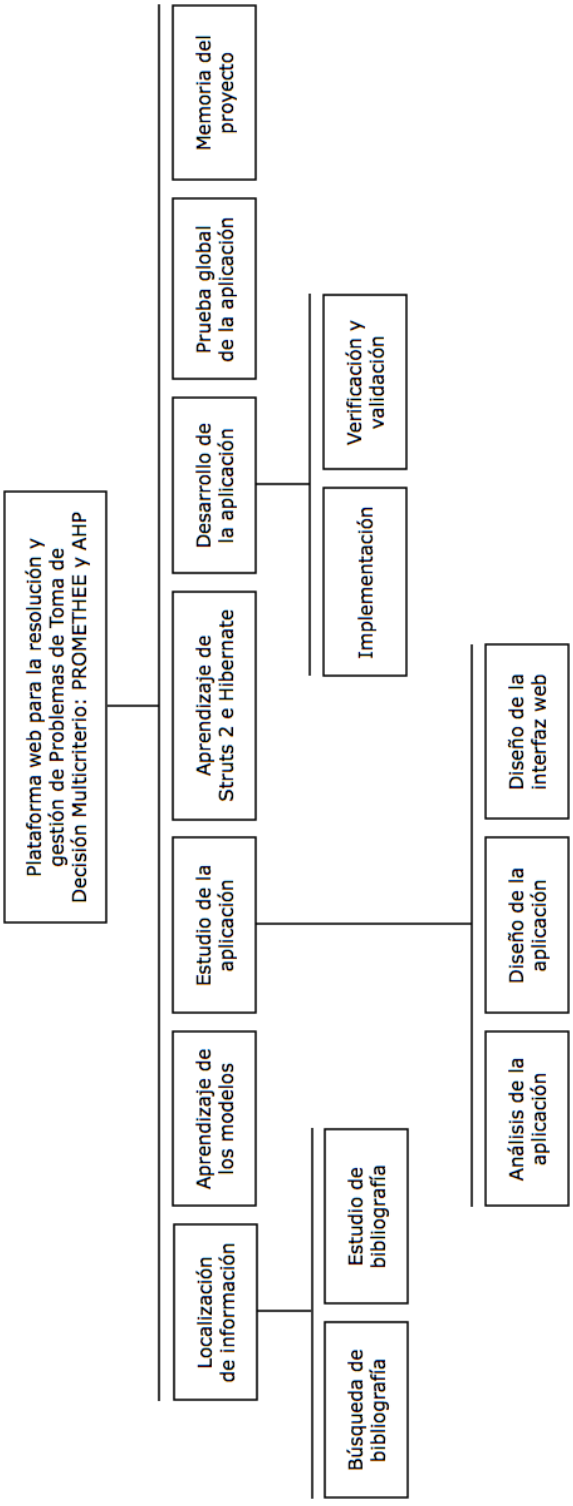


Figura 4.3. Estructura de descomposición del proyecto.

4.4. Especificación de requerimientos

El primer paso en la Ingeniería del *Software* consiste en determinar el propósito del proyecto, las propiedades que debe satisfacer y las restricciones a las que está sometido, es decir, la especificación de requerimientos. Este es, sin duda, un paso de vital importancia dentro del desarrollo de un proyecto *software* ya que, sin conocer el propósito del proyecto y todas las limitaciones a las que debe hacer frente, difícilmente se podrá realizar una buena aplicación *software* que cumpla con sus objetivos.

Cuando un cliente solicita un nuevo sistema, tiene algunas nociones sobre lo que el sistema debe hacer. Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas [25]. Otro modo de definir el concepto de requerimiento es como una característica del sistema o una descripción de lo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del mismo. Estos requerimientos reflejan las necesidades de los usuarios de la aplicación que ayude a resolver algún problema concreto para ellos.

El término requerimiento no se utiliza de forma constante en la industria del *software*. En algunos casos, un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En otras ocasiones se trata de una definición detallada y formal de una función del sistema.

En todo proyecto de Ingeniería del *Software* los requerimientos se determinan por medio del trabajo con los clientes, de hecho, el cliente debe participar desde el comienzo y en todo momento en las etapas del proceso de Ingeniería del *Software*. La extracción de los requerimientos se lleva a cabo por medio de entrevistas con los clientes, formulando preguntas, realizando estudios de la situación actual del sistema, estudios de viabilidad, haciendo demostraciones de sistemas similares e incluso desarrollando prototipos de todo o partes del sistema a los clientes.

En determinadas ocasiones resulta muy útil separar los requerimientos en tres categorías atendiendo a la importancia que se les da [26, 28]:

- Requerimientos que deben ser absolutamente satisfechos.
- Requerimientos que son muy deseables pero no indispensables.
- Requerimientos que son posibles pero que podrían eliminarse.

Puesto que el proyecto que se desarrolla es totalmente académico los clientes que solicitan el producto en realidad no existen como tal, sino que se trata de los usuarios finales para los que el *software* se diseña. Es por ello que el propósito del proyecto *software* es conocido desde el comienzo del mismo y consiste en:

Implementar una plataforma web basada en la arquitectura cliente-servidor que hace uso de tecnología JAVA y que sirva para la resolución y gestión de problemas de Toma de Decisión con Múltiples Criterio por medio de la utilización de los modelos PROMETHEE y AHP.

Normalmente los requerimientos del sistema se pueden clasificar en dos categorías [25, 26]:

- Requerimientos funcionales: son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar ante entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistema también pueden declarar explícitamente lo que el sistema no debe hacer.

- Requerimientos no funcionales: son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad, normalmente apenas se aplican a características o servicios individuales del sistema.

En los dos epígrafes siguientes se presentan los requerimientos, tanto funcionales como no funcionales, de la aplicación que se desarrolla en este proyecto. Sin embargo, estas definiciones se refinan a lo largo de la fase de análisis del sistema en la que se descubren nuevas necesidades que permiten completar los requerimientos.

4.4.1. Requerimientos funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer [25]. Son aquellos que se encargan de describir las funcionalidades que el sistema debe proporcionar a los usuarios del mismo para cumplir sus expectativas. Un requerimiento funcional describe una interacción entre el sistema y su ambiente, además de como debe comportarse el sistema ante un determinado estímulo [26].

Estos requerimientos dependen del tipo de *software* que se desarrolle, de los posibles usuarios del *software* y del enfoque general tomado al redactar los requerimientos. Se pueden expresar de diferentes formas, pero siempre definen los recursos específicos que el sistema puede proporcionar.

La especificación de requerimientos funcionales de un sistema debe ser completa y consistente. La completitud significa que todos los servicios solicitados por el usuario deben estar definidos, mientras que la consistencia hace referencia a que los requerimientos no deben tener definiciones contradictorias. En la práctica, para sistemas grandes y complejos es prácticamente imposible alcanzar los requerimientos de completitud y consistencia porque

es muy fácil cometer errores u omisiones cuando se redactan especificaciones para sistemas grandes y complejos.

A continuación, se definen los requerimientos funcionales para el sistema que se desarrolla en este proyecto. Los requerimientos se han clasificado en varias categorías según el tipo de usuario y el modelo de toma de decisiones multicriterio con los que se relacionen.

✦ *Administrador y Experto*

1) Identificar a los usuarios

Para proporcionar seguridad, confidencialidad, así como para distinguir entre el administrador, los expertos y resto de entidades no dadas de alta en el sistema, éste debe proporcionar soporte para la identificación de los usuarios mediante un nombre y una contraseña.

2) Consultar problemas

La aplicación debe permitir al administrador consultar cualquier problema que se encuentre almacenado en el sistema, mientras que los expertos únicamente pueden consultar los problemas creados por ellos mismos. El sistema tiene que mostrar todos los datos del problema consultado y el resultado final obtenido al aplicar el método de resolución.

3) Eliminar problemas

El administrador debe tener potestad para eliminar del sistema el problema que desee. Igualmente, la aplicación debe permitir que los expertos puedan borrar cualquier problema creado por ellos mismos.

4) Modificar contraseña

El sistema deber permitir que tanto el administrador como los expertos puedan modificar su contraseña por otra de su elección cuando lo consideren oportuno.

✦ *Administrador*

1) Crear expertos

Se deben poder introducir nuevos expertos en el sistema que no existan con anterioridad y esta tarea tiene que ser desarrollada únicamente por el administrador. El sistema debe dar soporte a este requerimiento.

2) Modificar expertos

Ante expertos que ya existen en el sistema, la aplicación tiene que permitir que el administrador sea capaz de modificar, cuando requiera, los datos pertenecientes al experto que él seleccione.

3) Eliminar expertos

Del mismo modo que ocurre con la modificación de expertos, un administrador debe ser capaz de eliminar, cuando él decida, cualquier experto que exista previamente en el sistema.

✦ *Experto*

1) Crear problemas

Cualquier usuario que pertenezca a la categoría de experto debe ser capaz de crear tantos problemas como desee, visibles únicamente por él mismo y el administrador. Para finalizar la creación del problema, éste se resuelve por medio de la aplicación de alguno de los métodos de toma de decisión estudiados en este proyecto.

2) Modificar problemas

La aplicación tiene que soportar que un experto pueda modificar cualquier elemento que forme parte de un problema previamente almacenado en el sistema y que haya sido creado antes por dicho experto. Únicamente no se permite modificar el tipo del problema ya que entonces no se trataría de modificar un problema, sino de crear uno nuevo.

✦ *PROMETHEE y AHP*

1) Resolver problemas de decisión aplicando uno de los modelos implementados

Se trata de la principal funcionalidad del sistema y es que éste debe ser capaz de permitir resolver problemas de toma de decisiones multicriterio bien por medio de la utilización del modelo PROMETHEE o AHP.

2) Problemas con un único objetivo

Los problemas que se pretenden resolver con la utilización de este sistema sólo pueden tener un único objetivo.

3) Permitir tantas alternativas como se requieran

En este tipo de problemas es el decisor quien decide el número de alternativas que se desean tener en cuenta, por ello el sistema tiene que permitir aceptar tantas como se requieran.

4) Soportar tantos criterios como sean necesarios

En los problemas de decisión multicriterio pueden surgir tantos criterios como el decisor quiera tener en cuenta, por lo que el problema debe soportar este requerimiento.

5) Los criterios pueden maximizarse o minimizarse de forma independiente

En todo problema de decisión con múltiples criterios lo normal es que los criterios sean contradictorios entre sí, que unos se pretendan maximizar y otros minimizar, por lo que el sistema debe dar soporte a este requerimiento.

6) Expresar las valoraciones utilizando información numérica

En ambos modelos es necesario definir una serie de valoraciones emitidas por los expertos. Éstas deben expresarse en términos numéricos.

7) Mostrar una clasificación ordenada entre las alternativas

Tras finalizar el modelo elegido el sistema tiene que mostrar una clasificación ordenada de todas las alternativas propuestas. En el caso de PROMETHEE II y AHP esta clasificación es inequívoca y va acompañada de los valores que se obtienen para cada alternativa. Para el caso de PROMETHEE I, como existe un ordenamiento parcial, se realiza una comparativa entre aquellas alternativas que tenga sentido hacerlo.

8) Permitir realizar un análisis de sensibilidad

Cuando se muestran los resultados obtenidos por el método elegido, el sistema debe permitir al usuario realizar, si éste lo desea, un análisis de sensibilidad que posibilite la modificación de los datos que introdujo.

✦ ***PROMETHEE***

1) Utilizar PROMETHEE I y PROMETHEE II

Dentro de la familia de los métodos PROMETHEE existen múltiples modelos que se pueden aplicar. Para este sistema en concreto si se decide seleccionar el método PROMETHEE se deben aplicar siempre los modelos I y II, mostrando al final del proceso los resultados de ambos y facilitando así la comparación entre ellos.

2) Soportar seis tipos de criterios generalizados

Dentro del método de toma de decisiones multicriterio PROMETHEE existen multitud de nociones generalizadas de criterio que se pueden utilizar, tantas como el decisor pueda imaginar. Este sistema debe dar soporte por tanto a las seis más utilizadas en la ámbito de la toma de decisiones: criterio usual, quasi-criterio, criterio con preferencia lineal, criterio con nivel, criterio con preferencia lineal y área de indiferencia y criterio gaussiano.

3) Comparativas entre alternativas

Se debe dar la posibilidad al decisor de realizar comparativas entre dos alternativas elegidas por él. Las comparativas consisten en la visualización con detalle del cálculo del índice de preferencia entre esas dos alternativas.

✦ *AHP*

1) Permitir definir subcriterios

En el método AHP, el sistema debe permitir definir para cada criterio tantos subcriterios como el decisor estime necesarios. De forma análoga, es posible determinar para cada subcriterio sus propios subcriterios, tantos como se requieran. Este proceso se puede repetir tantas veces como se quiera.

2) Calcular y mostrar la razón de consistencia de todas las matrices

Para cada matriz de comparaciones pareadas que se construya durante el proceso de AHP, es necesario calcular y mostrar al usuario un valor que indica si la matriz es consistente. Esta razón depende de los juicios emitidos por el experto.

4.4.2. Requerimientos no funcionales

Los requerimientos no funcionales, como su mismo nombre sugieren, son aquellos requerimientos que no se refieren directamente a las funcionalidades específicas que proporciona el sistema, sino a las propiedades emergentes de éste [25]. De forma alternativa, describen las restricción sobre el sistema y restringen los requerimientos funcionales [26].

Son tan importantes como los propios requerimientos funcionales y pueden incluso a llegar a ser críticos para la aceptación del sistema. Los usuarios del sistema normalmente pueden encontrar formas de trabajar alrededor de una función del sistema que realmente no cumple sus necesidades, sin embargo, el incumplimiento de un requerimiento no funcional puede significar que el sistema entero sea inutilizable.

Los requerimientos no funcionales rara vez se asocian con características particulares del sistema. Más bien, estos requerimientos especifican o restringen las propiedades emergentes del sistema o producto como el rendimiento, protección, disponibilidad, plataforma y fiabilidad entre otras, así como el diseño de la interfaz gráfica de usuario además de todas las restricciones impuestas por la organización como las políticas de empresa, estándares, legalidad vigente y demás.

Los requerimientos no funcionales no sólo se refieren al sistema *software* a desarrollar, sino que alguno de estos requerimientos pueden restringir el proceso que se debe utilizar para desarrollar el sistema.

A continuación, se definen los requerimientos no funcionales para el sistema que se desarrolla en este proyecto.

1) Arquitectura cliente-servidor

El *software* que se desarrolla sigue la arquitectura cliente-servidor, por el cual el cliente realiza peticiones a otro programa, en este caso el servidor, que proporciona las respuestas. De este modo el cliente se desentiende por completo de dónde está el servidor situado, qué programas contiene y cuáles se ejecutan para generar la respuesta, el cliente sólo sabe que envía una petición y que recibe una respuesta pero desconoce el proceso seguido para obtenerla.

2) Patrón Modelo-Vista-Controlador (MVC)

La aplicación se diseña siguiendo el patrón de diseño MVC, que no es más que un patrón de arquitectura del *software* que separa los datos de una aplicación (modelo), la interfaz de usuario (vista), y la lógica de control (controlador) en tres componentes distintos. De esta manera, se permite conservar las múltiples partes del sistema con un bajo acoplamiento y una alta cohesión, facilitando con ello el mantenimiento de la aplicación y sus futuras ampliaciones y renovaciones.

3) Usabilidad

Es necesario que la aplicación sea lo más usable posible. La usabilidad se define como la medida en que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico.

La efectividad es la precisión con la que los usuarios alcanzan los objetivos especificados. La eficiencia hace referencia a los recursos empleados y que son necesarios para alcanzar con precisión esos objetivos. Y la satisfacción es la ausencia de incomodidad y la actitud positiva del usuario hacia la aplicación, por lo tanto es un factor subjetivo.

4) Facilidad de uso

La aplicación debe ser fácil de utilizar e intuitiva. Se entiende por facilidad de uso como la disposición para utilizar la aplicación sin necesidad de requerir un gran trabajo o esfuerzo.

5) Facilidad de aprendizaje

Interesa que el aprendizaje acerca del funcionamiento de la aplicación sea lo más rápido y sencillo posible. Se define como la facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema. Está relacionada con las siguientes propiedades:

- Predecibilidad: una vez conocida la aplicación, se debe saber en cada momento a que estado se pasará en función de la tarea que se realice. Debe ser suficiente el conocimiento de la historia de interacción para determinar el resultado de una interacción futura.

- Sintetización: los cambios de estado tras una acción deben ser fácilmente captados por parte del usuario. Así puede evaluar fácilmente el efecto que han tenido sobre el sistema acciones anteriores.

- Generalización: las tareas semejantes se resuelven de modo parecido. Los usuarios frecuentemente tratan de extender su conocimiento con interacciones específicas a situaciones que son similares pero con las que no se han encontrado anteriormente.

- Familiaridad: el aspecto de la interfaz tiene que resultar conocido y familiar para el usuario. Es la correlación que existe entre los conocimientos que ya posee un nuevo usuario del sistema y los que son necesarios para su uso.

- Consistencia: todos los mecanismos que se utilizan deben ser usados siempre de la misma manera, siempre que se utilicen y sea cual sea el momento en que se haga.

6) Rapidez

La aplicación debe ser lo más rápida posible para minimizar la espera que debe soportar el usuario, impidiendo en la medida de lo posible la aparición de ansiedad por parte de éste. Esta característica se define como la cualidad que indica cómo de veloz es el sistema, es decir, cómo de deprisa opera.

7) Robustez

El sistema que se desarrolla tiene que alcanzar ciertos niveles de robustez. Esta cualidad se define como el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos o también como la capacidad del sistema para tolerar fallos. Está relacionada con los siguientes factores:

- Navegabilidad: el sistema tiene que ser observable. El usuario debe poder observar el estado del sistema sin que esta observación repercuta de forma negativa en él.

- Valores por defecto: ayudan al usuario a recordar de un modo pasivo que formato seguir.

- Persistente: relacionado con la duración del efecto de un acto de comunicación con el usuario.

- Recuperación de información: la aplicación debe ser capaz de poder deshacer alguna operación crítica que cambie el estado del sistema y permitir volver a un estado anterior.

- Ajuste de la tarea al usuario: las tareas que lleva a cabo la aplicación deben hacerse centrándose en el usuario.

- Tiempo de respuesta: es el tiempo que necesita el sistema para reflejar los cambios de estado al usuario. En general, es deseable que haya tiempos de respuesta instantáneos o muy cortos.

8) Fiabilidad

Es necesario que el sistema sea totalmente fiable para que el usuario se sienta seguro y tenga la certeza de que los resultados obtenidos son correctos. La fiabilidad se define como la probabilidad de que el sistema funcione o desarrolle una sus funciones bajo condiciones fijas y durante un período de tiempo determinado, es decir, la probabilidad de que el sistema funcione bien.

9) Flexibilidad

El sistema debe garantizar un cierto nivel de flexibilidad en la comunicación entre el usuario y el sistema. La flexibilidad atiende a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema. Una aplicación flexible tiene las siguientes características:

- Control del diálogo por parte del usuario: la interacción entre usuario y sistema se puede considerar como un diálogo. Hay que intentar que el usuario tenga el control de la interacción siempre que sea posible.

- Migración de tareas: relacionada con la transferencia de control entre usuario y sistema. Tanto el usuario como el sistema deben poder realizar una tarea en exclusiva, pasarla al otro, o realizarla de forma compartida.

- Capacidad de sustitución: permite que valores equivalentes puedan ser sustituidos los unos por los otros.

- Capacidad de adaptación: la aplicación debe adaptarse automáticamente al usuario.

10) Portabilidad

La aplicación debe ser totalmente portable ya que se desconoce la plataforma sobre la que se va a ejecutar en el servidor. Esta característica se logra al construir el sistema haciendo

uso de la tecnología Java, por lo que teniendo instalada en el servidor una Máquina Virtual Java (JVM) es suficiente. La portabilidad se define como la capacidad que posee un *software* para ejecutarse en diferentes plataformas.

11) Persistencia

Es necesario que la aplicación sea capaz de almacenar datos sobre los problemas y expertos de forma permanente, permitiendo recuperarlos, modificarlos y eliminarlos cuando se requiera. Los datos deben almacenarse de forma que se eviten redundancias y se mantenga su integridad referencial.

12) Didáctica

Al tratarse de una aplicación que se ejecuta en el ámbito académico una buena propiedad que se le exige es que sea muy didáctica. Por tanto, la aplicación debe mostrar al usuario de forma visual y detallada qué pasos se siguen para resolver el problema de toma de decisión.

4.4.3. Requerimientos de *software* y *hardware*

Para poder ejecutar la aplicación es necesaria una serie de recursos *software* y *hardware* que deben tener tanto el equipo desde el que se va a acceder a la aplicación (cliente) como el equipo donde se va alojar y ejecutar la aplicación (servidor). Algunos de estos requerimientos son comunes a ambos ordenadores mientras que otros son específicos de alguno de ellos. A continuación, se detallan los requerimientos.

1) Sistema operativo

Cualquier sistema operativo de los actuales en el mercado es válido tanto para poder acceder a la aplicación desde el cliente como para almacenarla y ejecutarla en el servidor.

2) Navegador web

El equipo del cliente debe poseer una aplicación que le permita navegar por las páginas web que genera el *software*. Es altamente recomendable que el navegador siga los estándares de HTML y JavaScript. Sólo es requerido por el cliente.

3) Máquina Virtual Java (JVM)

Al construirse la aplicación haciendo uso de la tecnología Java, para poder ejecutar el *software* es necesario que el equipo encargado de esta tarea (servidor) posea como mínimo una Máquina Virtual Java. Sólo es requerido por el servidor.

4) Servidor de páginas web

Como el *software* que se construye es una aplicación web, finalmente la información que se intercambian el cliente y el servidor son páginas web. Por tanto, el servidor debe disponer de un *software* que le permita alojar la aplicación y enviar las páginas de respuesta al cliente. Sólo es requerido por el servidor.

5) Sistema de gestión de bases de datos

Como en el servidor se almacenan datos de forma persistente, es necesario utilizar algún *software* que permita guardar la información de forma adecuada y disponer de ella cuando se precise. Para ello se hace uso de un sistema que se encargue de gestionar la base de datos. Sólo es requerido por el servidor.

6) Conexión a Internet

Tanto el equipo del cliente como el del servidor deben tener acceso a Internet para poder intercambiar información. Cualquier Proveedor de Servicios de Internet (ISP) del mercado proporciona una conexión más que suficiente para la comunicación que se establece.

7) Memoria principal

El equipo del cliente debe tener la suficiente memoria como para permitir usar un navegador web que acceda a diversas páginas mediante el envío de peticiones al servidor. El equipo del servidor necesita algo más de memoria para soportar la ejecución de la aplicación por parte de múltiples clientes y enviar las respuestas a los mismos. Generalmente, con que ambos equipos tengan la memoria necesaria como para que el sistema operativo pueda operar correctamente, ésta es suficiente para hacer un buen uso de la aplicación.

8) Almacenamiento

El cliente sólo requiere la capacidad necesaria para tener instalado el sistema operativo y el navegador de páginas web. El servidor necesita algo más de almacenamiento puesto que debe tener instalado, además del sistema operativo, el servidor de páginas web, la JVM y el sistema de gestión de base de datos. En cualquier caso, la mayoría de los equipos disponibles en la actualidad poseen una capacidad de almacenamiento más que suficiente para utilizar el *software*.

9) Memoria de vídeo

Tanto el equipo del cliente como el del servidor requieren muy poca memoria de vídeo, la suficiente para soportar un entorno gráfico mínimo. Ésta depende principalmente de la interfaz gráfica del sistema operativo instalado (cuanto más moderna sea más memoria se necesita), ya que la aplicación que se desarrolla apenas requiere de este recurso. Incluso si el

servidor no va acompañado de entorno gráfico se puede reducir aún más las necesidad de esta memoria.

10) Frecuencia de reloj

Los equipo deben ser lo suficientemente rápidos como para ejecutar la aplicación en el menor tiempo posible y soportar a su vez el resto del *software* que se requiere para su utilización. Cualquier microprocesador del mercado es capaz de cumplir con esta labor.

4.5. Tecnologías de desarrollo

A la hora de desarrollar una aplicación existen multitud de arquitecturas, patrones de diseño y tecnologías que se pueden utilizar. Uno de los objetivos prioritarios en la realización de este proyecto es desarrollar una aplicación que realice su cometido de forma eficaz y eficiente, que maximice la usabilidad y experiencia del usuario, al mismo tiempo que facilita su modificación y mantenimiento. Para conseguirlo, es necesario seguir ciertas directrices y tecnologías.

Este epígrafe recoge las arquitecturas y tecnologías empleadas en el desarrollo de la aplicación, explicando su funcionamiento y el motivo de su uso. El uso de la arquitectura cliente-servidor viene dado por la naturaleza del proyecto, mientras que la elección del patrón Modelo-Vista-Controlador y de los *frameworks* Struts 2 e Hibernate viene determinada por el estudio de la bibliografía pertinente.

Aunque el contenido de este epígrafe está muy relacionado con la implementación del sistema, se introduce antes para garantizar una completa comprensión de las elecciones tomadas en la etapa de diseño y de los diagramas asociados a éstas.

4.5.1. Arquitectura cliente-servidor

En Internet la gran mayoría de las comunicaciones se realizan siguiendo la arquitectura cliente-servidor. Por ello, el uso de esta arquitectura en la aplicación viene determinado por el desarrollo de una plataforma web, que como tal, se sustenta en el uso de Internet como medio de comunicación entre el usuario y la aplicación.

La arquitectura cliente-servidor se define como un modelo de computación entre ordenadores, que permite a los usuarios finales obtener acceso a recursos remotos de forma transparente.

En este tipo de arquitectura, como su propio nombre indica, se distinguen dos entes, el cliente y el servidor. En ambos casos se tratan de ordenadores con funciones claramente delimitadas. La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

Tal y como se muestra en la Figura 4.4, el modelo de comunicación entre ambos consiste en que el cliente envía una petición al servidor solicitando un determinado recurso o servicio. Por su parte, el servidor se encarga de recibir la petición, analizarla y en función de ésta, realizar algún tipo de procesamiento para el cual fue diseñado. Finalmente el servidor envía una respuesta al cliente proporcionando el servicio.

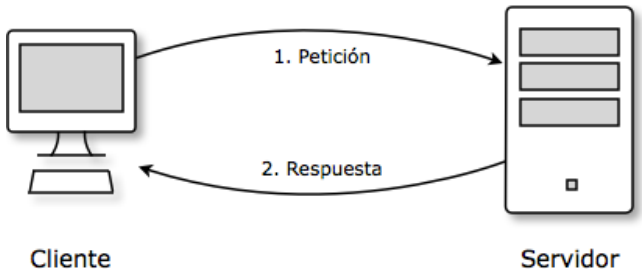


Figura 4.4. Arquitectura cliente-servidor.

El cliente se encarga de la interacción con el usuario. Por una parte, permite recoger sus requerimientos y enviarlos al servidor, mientras que por el otro, se encarga de mostrar los datos de la respuesta recibida.

El servidor tiene la tarea de recibir las peticiones de múltiples clientes que solicitan recursos. Normalmente gestiona las funciones relacionadas con la lógica de la aplicación y el acceso a los recursos de datos, tal y como ocurre en la aplicación desarrollada en este proyecto.

El uso de esta arquitectura lleva asociada una serie de ventajas e inconveniente. Algunas de las ventajas son:

- Se centraliza el control de los accesos, recursos e integridad de los datos al realizar estas tareas el servidor.
- Se puede aumentar la capacidad de cliente y servidor por separado.
- Fácil mantenimiento, al estar bien separadas las funcionalidades y responsabilidades en varios ordenadores.
- Favorece el rápido desarrollo de aplicaciones.

Por otro lado, estos son algunos de sus inconvenientes:

- Gran congestión de tráfico cuando múltiples clientes realizan peticiones simultáneas al servidor.
- Cuando el servidor no está en línea no se pueden satisfacer las peticiones de los clientes.

- El *hardware* y *software* del servidor son determinantes.
- La seguridad es una preocupación muy importante.

4.5.2. Modelo-Vista-Controlador

Cuando se desarrollan sistemas interactivos es muy importante mantener separadas la lógica de la aplicación de la interfaz de usuario que proporciona el acceso a dicha lógica. Haciéndolo así, es posible cambiar la interfaz sin necesidad de modificar la lógica y viceversa.

El uso de este enfoque conlleva varias ventajas:

- Portabilidad. Para permitir que la misma aplicación sea usada en diferentes sistemas es mejor considerar su desarrollo separado de su interfaz, ya que ésta es dependiente del sistema.
- Reusabilidad. La separación incrementa la probabilidad de que los componentes puedan ser reusados para ahorrar costes de desarrollo.
- Múltiples interfaces. Para mejorar la flexibilidad de una aplicación se pueden desarrollar varias interfaces de usuario para acceder a la misma funcionalidad.
- Personalización. La interfaz de usuario puede ser personalizada tanto por el diseñador como por el usuario para incrementar su efectividad sin tener que alterar los programas subyacentes.

Separar la aplicación en dos componentes: la interfaz y la lógica de la aplicación, plantea un problema, determinar cómo deben comunicarse ambos componentes. Existen varias arquitecturas que tratan de resolver este problema y una de ellas es la arquitectura Modelo-Vista-Controlador (MVC) [29, 33, 34].

El MVC es un patrón de arquitectura de *software* que separa la lógica de la aplicación, la interfaz de usuario y la lógica de control en tres componentes: el modelo, la vista y el controlador respectivamente. El uso de este patrón favorece un bajo acoplamiento y una alta cohesión entre las tres partes. Además, disminuye la duplicación de código, centraliza el control y facilita la modificación y el mantenimiento de la aplicación.

A continuación se presenta una breve descripción de cada componente:

1. Modelo. El modelo implementa la lógica de la aplicación, es decir, almacena todos los datos, el estado de la aplicación y tiene los métodos que manipulan esos datos. El modelo no es consciente de la vista y el controlador. Informa a la vista cuando cambia y le permite consultar su estado. También permite al controlador acceder a las funcionalidades de la aplicación que encapsula.

2. Vista. La vista es la interfaz de usuario. Se encarga de mostrar al usuario una representación visual del modelo, sus datos y estado, tomándolos directamente de éste cuando cambia. También contiene los elementos de la interfaz que permiten al usuario interactuar con la aplicación. Sin embargo, no es tarea de la vista decidir cómo se deben comportar esos elementos.

3. Controlador. El controlador define el comportamiento de la aplicación. Se sitúa entre la vista y el modelo. Se encarga de recoger las acciones del usuario y traducirlas en acciones para el modelo.

El principal objetivo es mantener el modelo, la vista y el controlador lo más independientes posible, de forma que se puedan cambiar sin necesidad de realizar ninguna modificación en los demás, lo que en la práctica se consigue con el uso del patrón Estrategia [29, 33].

El diagrama de la Figura 4.5 muestra la interacción entre los tres componentes cuando el usuario realiza alguna acción.

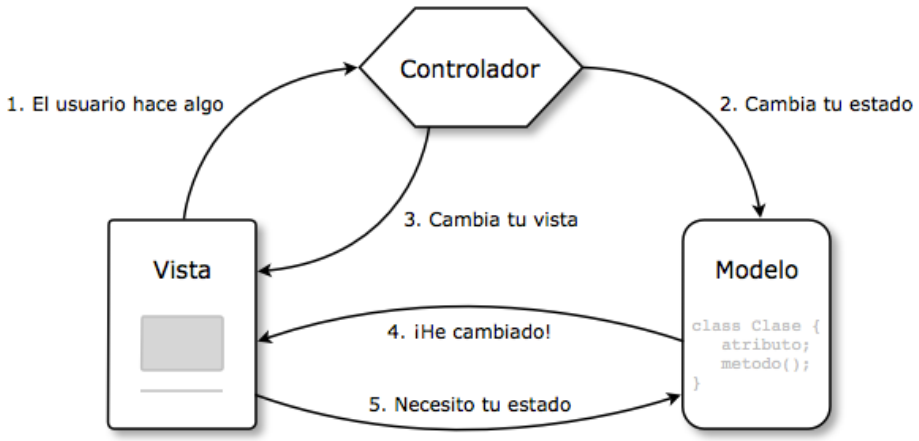


Figura 4.5. Patrón Modelo-Vista-Controlador.

El usuario sólo interactúa con la vista, que comunica al controlador las acciones del usuario. Por su parte, el controlador interpreta estas acciones y las convierte en órdenes para el modelo, generalmente por medio del uso del patrón Orden [29, 33], al mismo tiempo que puede ordenar a la vista que cambie como resultado de esta acción. Una vez el modelo finaliza las órdenes, notifica a la vista que ha cambiado, tras lo cual la vista solicita el nuevo estado del modelo a través del patrón Observador [29, 33].

Hasta el momento se ha mostrado el patrón Modelo-Vista-Controlador desde un punto de vista general, sin embargo, en el paradigma web este patrón sufre algunas modificaciones para poder adaptarse a la nueva arquitectura, aunque su filosofía se mantiene.

A continuación, se procede a describir el patrón Modelo-Vista-Controlador para aplicaciones web desarrolladas para la plataforma Java, puesto que es el lenguaje utilizado para desarrollar la plataforma web de este proyecto.

Dentro del MVC orientado a la web se pueden distinguir dos enfoques: el Modelo 1 y el Modelo 2.

- Modelo 1. El navegador web accede directamente a las Java Server Pages (JSP), que son las que se encargan de acceder al modelo de la aplicación y de seleccionar la siguiente vista que se va a mostrar. En el Modelo 1 el control está descentralizado, porque cada JSP decide la siguiente página que se va a mostrar, y además, las páginas se encargan de procesar sus propias entradas.

- Modelo 2. Introduce un Servlet controlador entre el navegador y los JSP o Servlets solicitados. El controlador centraliza la lógica recibiendo las peticiones y solicitando acciones al modelo, además de controlar la selección de la vista, lo que desacopla los JSP de los Servlets. Las aplicaciones que siguen este enfoque son más fáciles de mantener y ampliar, ya que cada vista no hace referencia a otra directamente. El controlador proporciona un único punto de acceso que facilita la implementación de otras capas (seguridad, registro de actividades, etc.) y además desacopla la vista del modelo.

Por estas razones se recomienda el Modelo 2 para aplicaciones interactivas de cierta envergadura y que puedan ser ampliadas en el futuro. Además, existen multitud de *frameworks* como Struts 2 o JavaServer Faces que facilitan mucho la implementación de este modelo.

El Modelo 1 puede proporcionar un diseño más ligero para aplicaciones pequeñas y estáticas. Es recomendable para aplicaciones que tienen un flujo de páginas simple y cambian poco con el tiempo. Si la aplicación creciera mucho y se volviera más compleja, debería reconstruirse para adaptarla al Modelo 2.

Debido a la envergadura, complejidad y a la intención de facilitar posibles ampliaciones y reutilizaciones de este proyecto, se utiliza para su desarrollo el patrón MVC Modelo 2 a través del uso del *framework* Struts 2.

✦ ***Diseño del controlador, la vista y el modelo***

El MVC Modelo 2 utiliza Servlets para procesar las peticiones y seleccionar las vistas. Aplicando el patrón *Front Controller* [34] estos dos procesos se centralizan en un único componente, de forma que el usuario envía las peticiones y recibe respuesta desde un único punto, el controlador.

Cuando el controlador recibe una petición debe procesarla para saber que acción ejecutar. Aplicando el patrón Orden [29, 33], las acciones van a ser clases que encapsulan la opera-

ción solicitada llamando al modelo y que heredan de una clase abstracta llamada Acción. A su vez, el controlador consta de un mapa *hash*, de forma que para cada petición inicie una acción concreta.

La vista se encarga de mostrar los datos producidos por el modelo. Los componentes de la vista suelen ser generalmente páginas JSP, además de contenido estático como HTML, PDFs, etc.

Una aplicación suele tener vistas con una distribución común (*layout*). En estas ocasiones se suele hacer uso de plantillas, que permiten crear una vista a partir de la inclusión de subvistas, haciendo uso del patrón Vista Compuesta [34].

El modelo representa los datos y la lógica de la aplicación. Algunas aplicaciones Java complejas implementan el modelo por medio de Enterprise Beans, por sus beneficios de escalabilidad, concurrencia, balanceo de carga, etc. Sin embargo, otras aplicaciones mas simples hacen uso de componentes JavaBeans. Los JavaBeans proporcionan un rápido acceso mientras que los Entreprise Beans proporcionan acceso a datos y lógica de negocio que se van a compartir.

4.5.3. Struts 2

Struts 2 es un *framework* que facilita el desarrollo de aplicaciones web para la plataforma Java que siguen el Modelo 2 del patrón Modelo-Vista-Controlador, haciendo que el proceso sea más rápido y sencillo. Principalmente proporciona parte de la estructura del controlador y ayuda en su desarrollo, aunque también facilita algunas tareas relacionadas con la vista.

Una de las partes más importantes de Struts 2 es el fichero struts.xml, que constituye el fichero de configuración de su núcleo. En él se definen las relaciones entre el resto de componentes de Struts, de ahí su importancia.

El *framework* suministra una clase llamada FilterDispatcher, que realiza la función de controlador principal, encargándose de recibir las peticiones, para después procesarlas y decidir que acción debe ejecutarse. Para ello, el FilterDispatcher usa el fichero struts.xml, donde se define para cada petición la acción que se va a ejecutar. Esta clase también tiene como función seleccionar la siguiente vista que se va a mostrar. El FilterDispatcher se define en el fichero web.xml del proyecto.

Otro componente son las acciones, que se corresponden con las principales unidades de trabajo de Struts. Las acciones son clases que instancia el FilterDispatcher y cuya misión es encapsular la llamada al modelo. Todas las acciones heredan de una clase abstracta denominada ActionSupport que proporciona el *framework*. Desde las acciones es posible acceder a la sesión del usuario y en primera instancia, son las encargadas de decirle al FilterDispatcher

la siguiente página que se va a mostrar. Se definen en el fichero struts.xml asociándose cada acción a una petición.

Los resultados son los recursos que usan las acciones para indicarle al FilterDispatcher la siguiente vista que se muestra. Se definen también en el fichero struts.xml, dentro de cada acción, de forma que una acción puede devolver varios resultados y éstos llevar a una página JSP diferente.

Otro elemento son los interceptores. Se trata de clases que permiten ejecutar código antes y/o después de la ejecución de una acción. Se pueden agrupar en pilas, de modo que se ejecuten varios de forma consecutiva según el orden en el que se hayan definido. Se usan para cuestiones de seguridad, validación, internacionalización, conversión de tipos, etc. Struts incorpora un buen número de interceptores listos para usar según la funcionalidad que se requiera, pero también permite definir interceptores propios. Se declaran en el fichero struts.xml y deben asociarse con las acciones a las que van ligados.

Struts 2 también proporciona una librería de etiquetas que se usan en las páginas JSP. Su función es la de recoger los datos que introduce el usuario y mostrar a éste el estado del modelo. Se utilizan para que el código de las páginas sea coherente, puesto que siguen el mismo formato que las etiquetas de HTML. Al final del proceso, cuando se genera la página, las etiquetas se transforman en HTML que al fin y al cabo es el lenguaje que sabe interpretar el navegador.

Por último, hablar del ActionContext, que consiste en un área de almacenamiento global donde se guardan todos los datos asociados con el procesamiento de la petición. Dentro de este área existen un espacio de especial interés denominado ValueStack. Se trata de un lugar donde se almacenan los datos recogidos del usuario al realizar una petición y el estado del modelo cuando se muestra una página. Es decir, es la pieza que permite el paso de información de la vista al controlador y del modelo a la vista.

Una vez vistos los principales componentes de Struts 2 por separado, llega el momento de mostrar su funcionamiento en conjunto, tal y como se observa en la Figura 4.6.

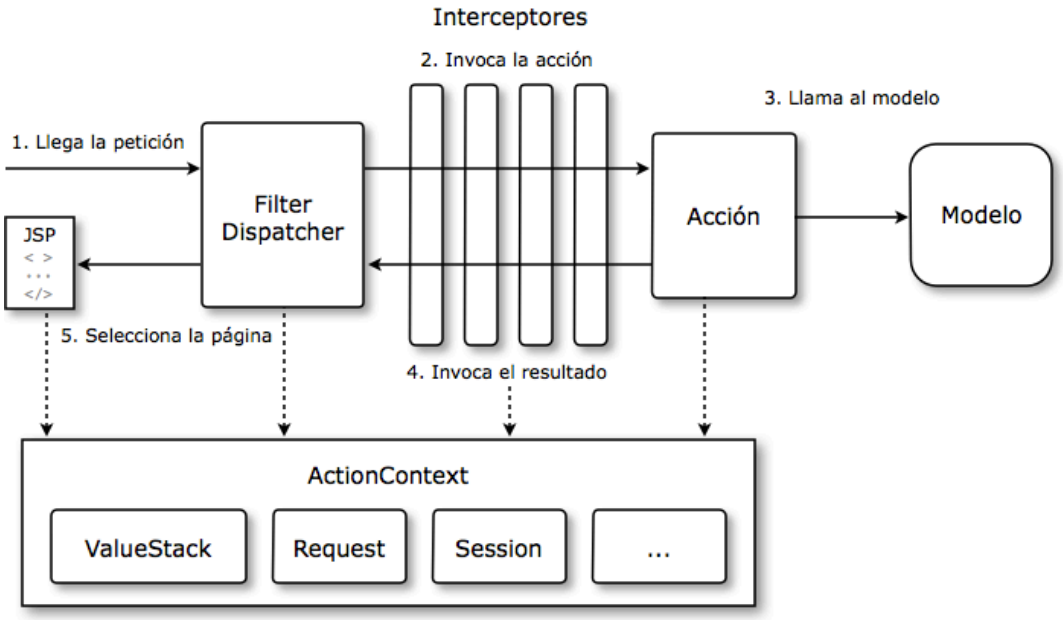


Figura 4.6. Funcionamiento de Struts 2.

Inicialmente el *framework* lee el fichero de configuración struts.xml. A partir de aquí, cuando un usuario envía una petición el FilterDispatcher la recibe, comprueba qué acción está asociada con la petición y los interceptores que acompañan a la acción.

Después el FilterDispatcher instancia la acción, utilizando los datos de la ValueStack si los requiriera, pero no la inicia todavía. Antes, los interceptores tienen que ejecutar su código asociado a la pre-ejecución de la acción, si es que lo tuvieran. Una vez finalizan todos los interceptores, el FilterDispatcher inicia la acción que es la encargada de realizar la llamada al modelo.

Una vez finalizado el procesamiento del modelo, se coloca su estado en la ValueStack y la acción devuelve un resultado al FilterDispatcher. De nuevo, los interceptores ejecutan su código, aunque en esta ocasión el asociado a la post-ejecución de la acción, si lo tuvieran, pero en orden inverso a cuando se ejecutaron por primera vez.

Luego, el FilterDispatcher recibe el resultado de la acción, mira los resultados asociados a ella y en función de éstos se muestra una página determinada.

Finalmente, la página muestra el estado actual del modelo por medio de la librería de etiquetas, que acceden a la ValueStack donde residen los datos.

El uso de Struts 2 ha facilitado el desarrollo de algunas características importantes de la aplicación:

- Comprobación de la autentificación y del rol del usuario para acceder a una determinada acción.
- Instanciación del modelo.
- Gestión de la sesión de Hibernate.
- Validación de las entradas del usuario.
- Internacionalización de la aplicación.
- Formato de fechas y números.
- Conversión automática de tipos.

4.5.4. Hibernate

Hibernate es un *framework* de mapeo objeto-relacional para la plataforma Java, entendiéndose por mapeo objeto-relacional como la técnica que permite transformar datos entre bases de datos relacionales y lenguajes de programación orientados a objetos. Es decir, Hibernate se encarga de mapear atributos entre los objetos y la base de datos relacional.

Desde hace tiempo y hasta ahora, el modelo de programación más usado es el orientado a objetos, del mismo modo que la persistencia utilizada en aplicaciones de cierta envergadura son las bases de datos relacionales. Debido a la naturaleza de ambas técnicas, se hace evidente que un objeto no se puede almacenar directamente en una base de datos relacional, sino que debe sufrir una serie de transformaciones que permitan almacenar sus atributos por separado. Este proceso suele ser tedioso, y empeora cuanto más complejos son los objetos y las relaciones entre ellos. Hibernate viene a facilitar precisamente este procedimiento, evitando al desarrollador tener que realizar las transformaciones manualmente y reduciendo la cantidad de código. Además, también proporciona facilidades para la consulta y recuperación de datos.

Otra de las ventajas de usar este *framework* es la independencia de la base de datos relacional que se utiliza. Esto se debe a que Hibernate soporta la mayoría de las bases de datos del mercado, por lo que él mismo se encarga de generar el código propio de éstas. Para ello, sólo hay que cambiar la propiedad del dialecto en el fichero de configuración de Hibernate.

Para establecer el funcionamiento de Hibernate se hace uso de dos tipos de ficheros. Por un lado está el fichero hibernate.cfg.xml, donde se definen los parámetros de su configuración, entre los que se encuentran el *driver* de conexión, la URL de la base de datos, el nombre de usuario y la contraseña para acceder a ella, el dialecto de la base de datos y los ficheros de mapeo de los objetos. El otro tipo de fichero define como deben mapearse los atributos del objeto en la base de datos. Su formato es <nombre_del_fichero>.hmb.xml y generalmente suele haber uno por cada objeto que se representa como una tabla en la base de datos.

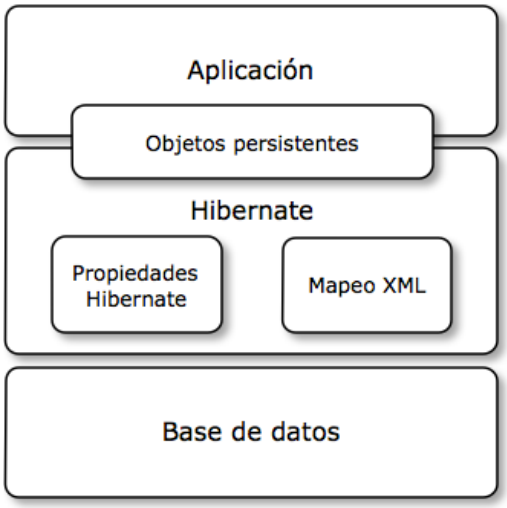


Figura 4.7. Arquitectura de Hibernate.

A continuación, se procede a describir el proceso de funcionamiento del *framework*. En Hibernate, todo acceso a la base de datos se realiza a través de una sesión, que es única y se comparte para cada acceso. Una vez obtenida la sesión se inicia una transacción, dentro de la cual se realizan un conjunto de operaciones de lectura y escritura. Después, se acomete la transacción, momento en el que realmente la base de datos sufre modificaciones. Si se produjese algún problema mientras se acomete la transacción habría que realizar un *rollback*. Por último, se cierra la sesión.

La aplicación desarrollada en este proyecto, siguiendo el patrón *Open Session in View* [38], cuenta con un interceptor encargado de abrir la sesión e iniciar la transacción antes de que se ejecute una acción que requiere acceso a la base de datos. Cuando finaliza la acción, el mismo interceptor es el que acomete la transacción, cierra la sesión y si fuera necesario realiza el *rollback*. Los Objetos de Acceso a Datos (patrón DAO [34]) mantienen una referencia a la sesión de Hibernate, y son ellos los que contienen las operaciones de lectura y escritura que se realizan durante la transacción. Por su parte, los Objetos de Transferencia (patrón TO [34]) son los objetos que se mapean en la base de datos tal y como se definen en sus respectivos ficheros de mapeo.

El acceso a bajo nivel a la base de datos se realiza a través de conexiones reutilizables. Para cada acceso se requiere de una conexión, por lo que si no quedan conexiones disponibles en un momento determinado (debido a accesos concurrentes a la base de datos) Hibernate debe coger una nueva de un *pool* de conexiones, lo que es un proceso muy costoso. Hibernate proporciona un *pool* de conexiones muy básico que no se debe utilizar en producción, por esta razón en este proyecto se hace uso de una librería especializada llamada c3p0.

4.5.5. Interacción y aportaciones

Ahora que se conoce el funcionamiento de la arquitectura cliente-servidor, el patrón Modelo-Vista-Controlador y los *frameworks* Strust 2 e Hibernate de forma individual, llega el momento de ver la interacción entre todos ellos de forma conjunta, lo cual queda reflejado en la Figura 4.8.

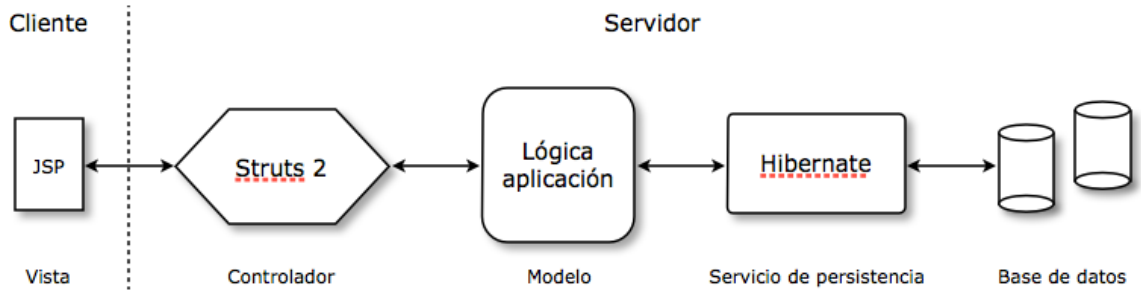


Figura 4.8. Interacción entre las tecnologías de desarrollo.

En el lado del cliente, las páginas JSPs desempeñan la función de la vista, de forma que cuando el usuario interactúa con ellas y realiza una acción se envía una petición al servidor. Una vez llega la petición al servidor, Struts 2 en el rol del controlador la recibe y en función de ésta realiza una llamada a la lógica de la aplicación, que representa el modelo. Si la lógica requiere de acceso a los datos hace uso de Hibernate, que desempeñando el papel del servicio de persistencia es el que accede a la base de datos.

Una vez Hibernate finaliza el acceso a los datos envía la información pertinente a la lógica de la aplicación, que a su vez la transmite a Struts 2. Finalmente, Struts 2 procesa la información y envía una respuesta al cliente, que generalmente consiste en una nueva página JSP con unos datos determinados.

A continuación, se detallan de forma más concreta las aportaciones que el uso de estas arquitecturas y tecnologías aportan a la aplicación.

El uso de la arquitectura cliente-servidor hace posible que el administrador y los expertos puedan conectarse a la plataforma web desde cualquier lugar. Para acceder sólo es necesario disponer de un ordenador con conexión a Internet y un navegador web.

El patrón MVC permite la separación entre vista y modelo a través del controlador, por lo que la interfaz del experto y el administrador pueden sufrir modificaciones sin necesidad de tocar la lógica de negocio. Del mismo modo, las funcionalidades relacionadas con la gestión de expertos y la resolución y gestión de problemas se pueden modificar sin tener que realizar cambios en la interfaz del experto o del administrador. Por tanto, todo esto permite que la modificación y mantenimiento de la aplicación sea más sencillo, gracias a que el seguimiento de este patrón produce un código con bajo acoplamiento y alta cohesión.

Para facilitar la aplicación de este patrón se hace uso del *framework* Struts 2, que se encarga de proporcionar buena parte del controlador y facilitar el resto de su desarrollo, aunque también facilita algunas tareas relacionadas con la vista.

La utilización del *framework* Hibernate aporta independencia del sistema de gestión de base de datos utilizado. Utilizar un sistema de gestión de base de datos u otro es tan sencillo como cambiar cinco parámetros en el fichero de configuración de Hibernate. Por ejemplo, para cambiar de una base de datos MySQL a otra Oracle sólo habría que cambiar esos parámetros.

CAPÍTULO V

Análisis del sistema

La etapa de análisis tiene como objetivo conseguir un modelo del sistema correcto, completo, consistente, claro y verificable. Para alcanzar esta meta, en primer lugar hay que hacer uso de los requerimientos definidos en la etapa anterior y a partir de ellos construir una especificación del sistema más concreta, siempre en estrecha colaboración con los usuarios.

En nuestro caso, la fase de análisis del sistema consta de varias herramientas. Inicialmente se procede a hacer un estudio detallado sobre las personas que van a hacer uso de la aplicación, lo que se conoce como perfiles de usuario. Muy ligado a los perfiles de usuario y para complementar la información proporcionada por ellos, se hace uso de las personas. Consisten en ejemplos concretos de los perfiles de usuario y tratan de reflejar lo más fielmente posible a las personas reales (aunque no lo sean) que van a hacer uso de la aplicación.

Después, se detallan los casos de uso y los escenarios. Los casos de uso se encargan de describir las funcionalidades que el sistema debe proporcionar, mientras que los escenarios son casos de usos concretos donde los datos toman valores reales y su función es mostrar todas las interacciones posibles entre el sistema y sus usuarios.

Finalmente, y de cara a ayudar en el desarrollo de la interfaz de usuario se define el análisis de tareas. Se define como el proceso de analizar la forma en que los usuarios realizan su trabajo, y está compuesto por dos elementos: uno es la estructura que posee cada tarea de interacción y el segundo la arquitectura de la interacción considerando todas las tareas juntas.

5.1. Perfiles de usuario

La primera etapa de la fase de análisis del sistema es la creación de los perfiles de usuario. Estos perfiles tratan de determinar con detalle quienes son los usuarios potenciales de la aplicación, sus características y obtener conclusiones de usabilidad para realizar una buena aplicación [31, 32].

La creación de perfiles de usuario es muy necesaria puesto que para crear una buena aplicación uno de los conceptos claves que deben cumplirse es el de usabilidad y este término requiere que el diseño realizado se centre en el usuario desde el comienzo y en todo el proceso. Para ello se precisa conocer bien a los usuarios potenciales, lo que se consigue por medio de estos perfiles.

La información necesaria para la construcción de los perfiles de usuario se obtiene a partir de diversas fuentes de datos. Por un lado, parte de la información se obtiene directamente a través de los propios usuarios, por medio de cuestionarios y entrevistas. El resto de datos se obtienen observando a los usuarios potenciales, usando manuales y textos sobre el dominio del problema y entrevistando a expertos en el dominio del mismo. Cada técnica tiene sus correspondientes utilidades, ventajas e inconvenientes por lo que es muy importante saber cuál usar en cada situación.

Cada perfil está formado por cuatro elementos. El título debe reflejar claramente a qué categoría de usuarios se refiere el perfil. La descripción general debe describir la categoría de forma clara. Las características es un listado de propiedades que mejor describen a la categoría. Y las conclusiones de usabilidad son resoluciones sobre como debería ser la aplicación.

A continuación, atendiendo a los usuarios potenciales que van a hacer uso de la aplicación se procede a detallar los perfiles de usuario, entre los que se distinguen dos: el administrador y el experto. Dentro de este último perfil de usuario se pueden distinguir tres tipos de personas que pueden desarrollar el papel de un experto, como son los profesores e investigadores, becarios y alumnos.

5.1.1. Administrador

- ***Descripción general***

El perfil de usuario del administrador está constituido normalmente por una única persona o por un grupo muy reducido de ellas, debido a que las acciones que pueden desarrollar suelen ser de gran importancia y no deben estar al alcance de cualquiera.

El administrador es una persona contratada por la universidad y que desarrolla casi siempre su trabajo en la misma, salvo en algunas ocasiones que lo hace desde su domicilio. Es el encargado de realizar tareas que poca gente está en disposición de hacer, relacionadas generalmente con la gestión de la aplicación, además suele tener acceso a casi cualquier funcionalidad de la misma.

- ***Características del usuario***

Este tipo de usuario hace normalmente un uso ocasional de la aplicación, por lo que es posible que no recuerde bien sus experiencias anteriores, pero el uso de aplicaciones similares le ayuda durante el proceso. Los administradores suelen trabajar principalmente en su despacho y en los laboratorios, por lo que el ambiente es poco ruidoso y está muy bien iluminado.

Los administradores se suelen caracterizar por ser personas que tienen bastante experiencia en el uso de ordenadores ya que su trabajo se centra principalmente en el tratamiento con éstos. Además, tienden a estar acostumbrados a usar programas en los que desarrollan tareas de gestión similares a las que se les asigna en este proyecto. A priori, este tipo de personas no tiene porque conocer el dominio de la toma de decisiones aunque podría darse el caso de que tuviera ciertas nociones básicas.

Su actitud frente a los sistemas informáticos es muy buena, pues está acostumbrado a trabajar con ordenadores y cree que el uso de éstos automatiza procesos tediosos en su trabajo. Los objetivos que llevan a un administrador a usar la aplicación son poder identificarse para gestionar los expertos que se almacenan en el sistema o bien para consultar o eliminar

problemas creados por cualquier experto. Le interesa poder realizar las tareas de una forma rápida, clara y eficaz, puesto que normalmente tiene bastante trabajo.

Se desconoce que tenga alguna discapacidad que le impida utilizar el *software* de forma habitual, a excepción de que es muy posible que necesite lentes de contacto porque pueda tener la vista cansada ya que pasa mucho tiempo delante de los ordenadores y los administradores no suelen ser jóvenes.

- ***Conclusiones de usabilidad***

A pesar de ser usuarios ocasionales no hay que mostrar especial atención a la facilidad de aprendizaje puesto que están acostumbrados a realizar tareas similares en otras aplicaciones, en lugar de eso hay que esforzarse en que la aplicación sea intuitiva de usar, que tenga nexos comunes a otras que utilice. Sin embargo, la facilidad de uso sí es más importante y hay que mostrarle más interés ya que el administrador suele tener bastante trabajo y quiere invertir el menor tiempo posible en realizar sus tareas.

Se puede hacer uso de terminología técnica relacionada con la informática ya que tiene bastante conocimiento en éste área, en cambio hay que evitar utilizar términos muy específicos vinculados a la toma de decisiones porque lo más normal es que los desconozca.

Ya que el administrador pasa mucho tiempo delante de ordenadores y como es altamente probable que necesite lentes de contacto hay que prestar atención al tipo y tamaño de letra, de forma que ambas características faciliten su lectura sin mucha dificultad.

5.1.2. Expertos: Profesores e investigadores

- ***Descripción general***

En este perfil de usuario se recoge al colectivo formado por profesores e investigadores. Se trata de personas que están en una situación laboral activa, contratadas por parte de la universidad y que están en posesión de al menos un título universitario.

Se han unido en una sola categoría porque en vistas a la aplicación no hay diferencia alguna entre ellos. Los profesores se distinguen por ser personal docente que imparte clase a personas matriculadas en la universidad, mientras que los investigadores llevan a cabo trabajos de investigación, pero nada impide que un profesor sea a su vez investigador o viceversa.

- ***Características del usuario***

Este tipo de usuario se caracteriza por hacer un uso medianamente frecuente de la aplicación, bien para resolver de forma automática y rápida problemas de toma de decisiones multicriterio o bien para usarla de forma didáctica con los alumnos a los que imparten clase. El ambiente donde utilizan el *software* es generalmente su despacho o laboratorio por lo

que el ruido es mínimo y hay unas buenas condiciones de luminosidad ya sea natural o artificial. En el caso de estar en el despacho las interrupciones telefónicas podrían ser frecuentes y casi siempre que se hace uso de la aplicación las dos manos están disponibles.

Tanto los profesores como los investigadores están acostumbrados a trabajar con sistemas informáticos puesto que la mayoría de ellos están en posesión de un título universitario relacionado con este ámbito, por lo que su conocimiento en este área es muy elevado. Todos creen que el uso de ordenadores facilita su trabajo por lo que su actitud y motivación hacia los ordenadores es bastante alta. Gracias a la experiencia en el uso de sistemas informáticos y a la actitud positiva en su utilización el proceso de aprendizaje es bastante rápido. Asimismo, estos usuarios son expertos en el dominio de la toma de decisiones por lo que están familiarizados con los procedimientos aplicados y la terminología técnica.

Los objetivos que llevan a estos usuarios a usar la aplicación es que quieren resolver problemas multicriterio de forma rápida y sencilla, puesto que muy posiblemente este proceso sea una parte minúscula de una tarea de mayor envergadura. Para resolver este problema prefieren usar dos métodos ampliamente conocidos, comunes y de efectividad justificada.

Muchos de estos usuarios pasan bastante tiempo delante del ordenador por lo que pueden tener deficiencias visuales derivadas de un uso prolongado del mismo. Además hay que tener en cuenta que un pequeño porcentaje de la población posee problemas de percepción del color.

- ***Conclusiones de usabilidad***

Dadas las características del personal docente, el uso de la aplicación debe requerir las acciones mínimas indispensables que disminuyan su tiempo de utilización y la generación de la solución tiene que ser lo más rápida posible. La facilidad de aprendizaje no es vital para estos usuarios, sin embargo, sí que lo es la de uso ya que no quieren invertir mucho tiempo en utilizar la aplicación.

El uso del terminología técnica no es problema para estos usuarios puesto que son expertos en el dominio del problema. Pero no hay que perder de vista que los profesores pueden usar la aplicación con fines docentes.

Debido a que un elevado número de personas puede tener algún tipo de deficiencia visual por el uso prolongado del ordenador, la fuente utilizada en la aplicación debe ser lo suficientemente grande y clara como para que permita leer el texto con el menor esfuerzo posible. Asimismo es altamente recomendable evitar el uso de colores en el texto para no complicar la lectura a posibles usuarios con problemas de daltonismo.

5.1.3. Expertos: Becarios

- ***Descripción general***

Los becarios son personas que están en disposición de una beca ofertada por la universidad, por lo que están contratados de forma temporal. Suelen ser personas que han terminado sus estudios universitarios recientemente o están en proceso de ello y suelen solicitar la beca para, en un futuro, establecerse de forma fija en la universidad o bien para adquirir experiencia laboral en el ámbito académico.

- ***Características del usuario***

Estos usuarios comparten algunas características con los profesores e investigadores ya que trabajan muy ligados a ellos. El uso de la aplicación es poco frecuente y es probable que no recuerden muy bien lo aprendido. Este tipo de usuarios está bajo supervisión de alguna clase de personal docente al que pueden recurrir si se les presenta algún problema. Generalmente hacen uso de la aplicación en despachos compartidos con otros becarios, por lo que tienen unas buenas condiciones de luminosidad y es posible que en ocasiones el nivel de ruido sea moderadamente elevado por el número de personas trabajando al mismo tiempo. Las interrupciones suelen ser frecuentes por parte de otros compañeros y casi siempre disponen de ambas manos para el uso de la aplicación.

Su experiencia en el uso de sistemas informáticos es muy alta ya que la mayoría pertenecen al ámbito de la informática, pero rara vez han usado una aplicación que les automatice y facilite la resolución de problemas de decisión, aunque conocen de su existencia. Su experiencia en el dominio del problema es media, suelen conocer más o menos los modelos que se aplican y parte de la terminología usada ya que la mayoría de los becarios que usan la aplicación poseen conocimientos acerca de la toma de decisiones. Si desconocen determinados aspectos del dominio, tienen a su alcance un amplio material bibliográfico al que pueden recurrir sin olvidar el personal docente del que están a cargo.

Su actitud y motivación en el uso de ordenadores es muy positiva, pues saben que el uso de éstos facilita muchos aspectos de su vida y reducen en gran medida el tiempo y esfuerzo en el trabajo. Además suelen ser personas jóvenes que usan ordenadores desde pequeños por lo que se desenvuelven bien en el uso de éstos. Los objetivos de uso son poder resolver problemas de toma de decisión multicriterio sin necesidad de conocer perfectamente los modelos aplicados y obtener, tras la introducción de un conjunto de datos, la solución al problema de un modo rápido y claro.

Estos usuarios hacen un uso prolongado del ordenador por lo que pueden presentar dificultades visuales, pero también hay que tener en cuenta que son personas jóvenes por lo que hay menor predisposición a presentar síntomas que están relacionados con la edad.

✦ ***Conclusiones de usabilidad***

Como usan con poca frecuencia la aplicación y no suelen acordarse de los conceptos aprendidos, la aplicación debe facilitar el aprendizaje en la medida de lo posible. También, debido a las posibles interrupciones que pueden sufrir esta categoría de usuarios, debe permitir retomar el proceso de toma de decisión de forma fácil.

Se permite el uso de cierta terminología técnica pero no excesiva. Además, es recomendable que la aplicación vaya guiando a los usuarios durante el proceso con el objetivo de que éstos afiancen conceptos y rememoren el proceso seguido.

5.1.4. Expertos: Alumnos

✦ ***Descripción general***

Los alumnos son personas matriculadas en la universidad y que pueden acceder a esta aplicación porque cursan alguna asignatura relacionada con la toma de decisiones durante el desarrollo de las clases prácticas. Pueden pertenecer a diversas carreras.

✦ ***Características del usuario***

Esta categoría de usuarios hace un uso muy esporádico de la aplicación, de hecho la totalidad de las veces que la utilizan es porque están realizando alguna práctica relacionada con la asignatura que están cursando. Por ello el número de ocasiones en que deben utilizar la aplicación es muy reducido y a priori apenas saben como hacer uso de la aplicación. Como están a cargo de un profesor éste puede guiar a los alumnos en todo momento. El lugar donde se hace uso de la aplicación es un aula de prácticas por lo que el ambiente presenta unas buenas condiciones de luminosidad y en ocasiones puede llegar a ser muy ruidoso. Las interrupciones suelen ser frecuentes ya que los alumnos tienden a perder con facilidad la atención y a menudo hablan entre ellos.

La experiencia en el uso de sistemas informáticos es muy variada, puede ir desde usuarios con conocimientos muy básicos de informática hasta usuarios expertos, según la carrera que cursen. Pero en general, nadie ha usado con anterioridad un *software* que resuelva problemas de decisión y al ser personas jóvenes tienen cierta experiencia en el uso de ordenadores aunque sea limitada. El conocimiento del dominio del problema suele ser bajo, ya que son alumnos que están aprendiendo sobre la toma de decisiones con el transcurso de la asignatura, aunque conocen algunos conceptos y procesos pero no de forma exhaustiva. Si se tienen dificultades pueden recurrir en todo momento al profesor de la asignatura.

Son personas jóvenes, por lo que saben que los ordenadores son muy útiles en el ámbito laboral y académico, sin embargo, no hay que perder de vista que usan la aplicación debido a la asignatura que cursan, por lo que su motivación será escasa y su actitud algo reacia ante el *software*. Sus objetivos son los de realizar la práctica y sentir que aprenden con la utili-

zación de la aplicación, ya que ésta les permite aprender nuevos conceptos y procesos de un modo visual.

- ***Conclusiones de usabilidad***

Dadas las características de esta categoría, inicialmente los alumnos desconocen como utilizar la aplicación, por lo que debe ser intuitiva y tiene que facilitar el aprendizaje y uso de la misma.

La aplicación debe dar soporte a una utilización con fines didácticos, por lo que es importante que sea muy visual y que detalle el proceso a seguir, indicando en todo momento en que fase se encuentra y como ésta se lleva a cabo, para facilitar la comprensión, seguimiento y aprendizaje de los modelos aplicados.

Debido a las continuas interrupciones que se pueden sufrir, la aplicación debe permitir retomar el proceso de toma de decisión de forma fácil.

5.2. Casos de uso

Un caso de uso es un diagrama que se genera durante el desarrollo de un sistema informático y es una herramienta sumamente útil. Un caso de uso describe una funcionalidad particular que el sistema debe realizar o exhibir, modelando el diálogo que un usuario u otra entidad tiene con el sistema que se desarrolla [26]. Esta entidad se denomina actor y puede ser un usuario, un dispositivo u otro sistema. En un caso de uso concreto pueden existir más de un actor.

Los casos de uso documentan el comportamiento del sistema desde el punto de vista del usuario [27]. Cada caso de uso describe un posible escenario de la interacción de la entidad externa con el sistema. Los casos de uso son particularmente útiles para comunicarse con los clientes, diseñadores y probadores, además, son relativamente fáciles de comprender de forma intuitiva, incluso sin conocer la notación específica.

El modelado de los casos de uso ayuda con tres de los aspectos más difíciles del desarrollo:

- La captura de requisitos.
- La planificación de las iteraciones del desarrollo.
- La validación de los sistemas.

Los diagramas de casos de uso constan de cuatro elementos principales: actores, casos, inclusiones y extensiones.

- Actor: es el rol con el que una entidad actúa. Se representa mediante el símbolo de un muñeco.

- Caso: es la representación de algún aspecto de la funcionalidad del sistema que resulta visible para el actor, cuya perspectiva es lo que se refleja en el caso de uso. Se representa mediante un óvalo con un nombre en su interior.

- Inclusión: es la reutilización de un caso de uso que ya se encuentra definido o el hecho de que en el proyecto se ha definido un nuevo componente reutilizable. Se representa mediante el estereotipo «include».

- Extensión: como su nombre indica extiende un caso de uso para explicar una perspectiva distinta o más profunda, representan casos invocados excepcionalmente o rara vez. Se especifican mediante el estereotipo «extend».

Los actores y casos se conectan entre sí por medio de líneas que indican que el actor puede interactuar con el sistema para realizar parte de la tarea. Sin embargo, una relación de comunicación entre un actor y un caso no significa que necesariamente alguien de ese rol tenga que estar implicado en la ejecución de una tarea, simplemente significa que puede ser, dependiendo de las circunstancias.

De forma general, un caso de uso se representa como un dibujo de los objetos involucrados, acompañado de una breve descripción textual. La descripción de un caso de uso consta de seis elementos: nombre, actores, condiciones de entrada, flujo de eventos, condiciones de salida y requerimientos especiales.

Para este proyecto se han determinado dos actores que se encargan de realizar la interacción con el sistema tomando los roles del administrador y del experto. Una vez definidos los actores del sistema, hay que crear los distintos casos de uso y es importante que cada uno de los requerimientos funcionales ya definidos aparezca en al menos uno de los casos de uso. A continuación, se detallan los diagramas de caso de uso y su correspondiente descripción.

5.2.1. Frontera del sistema

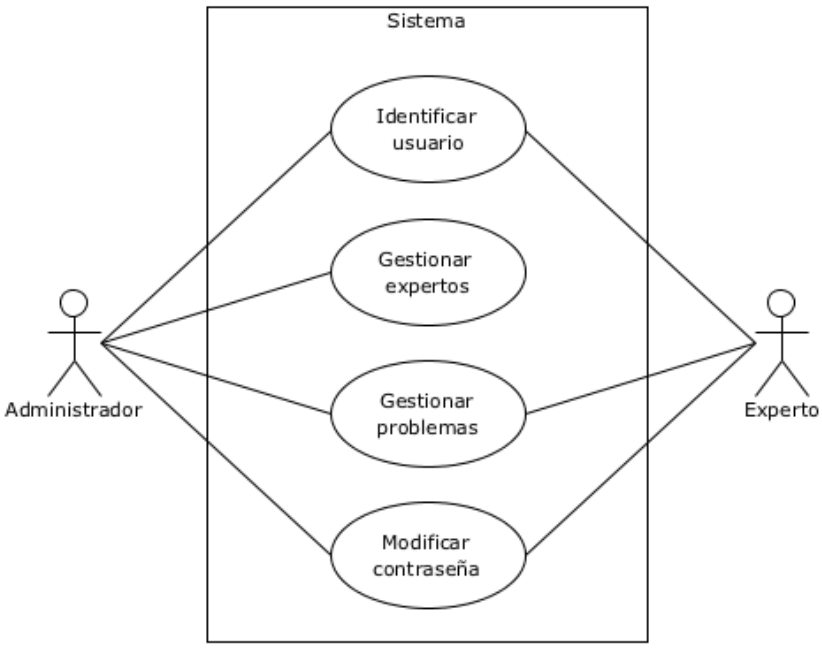


Figura 5.1. Frontera del sistema.

5.2.2. Identificar usuario

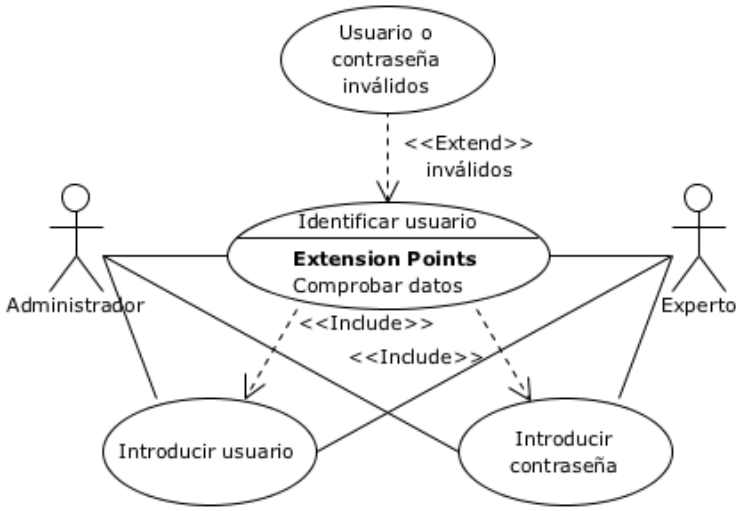


Figura 5.2. Identificar usuario.

- ***Nombre***

  Identificar usuario.

- ***Actores***

  Administrador y experto.

- ***Condiciones de entrada***

Tanto el administrador como el experto que quiera acceder al sistema deben estar registrados para poder hacerlo. Además, deben conocer y proporcionar al sistema su nombre de usuario y contraseña válidos.

- ***Flujo de eventos***

  El caso de uso es iniciado por el administrador o el experto.

  1. El usuario tiene que identificarse para poder acceder a las funcionalidades que proporciona el sistema.

  2. El sistema solicita el usuario y la contraseña.

  3. El usuario introduce su usuario y su contraseña.

  4. El sistema comprueba que los datos introducidos sean correctos y concede acceso al sistema. Puede ocurrir la *Excepción 1*.

  Excepciones:

  - Excepción 1: se produce cuando se introduce un usuario o una contraseña inválidos. Se informa mediante un mensaje y se le deniega el acceso al sistema hasta que introduzca unos correctos.

- ***Condiciones de salida***

El administrador o el experto se han identificado y han entrado en el sistema, permitiéndoles acceder a las funcionalidades que tengan asignadas.

5.2.3. Gestionar expertos

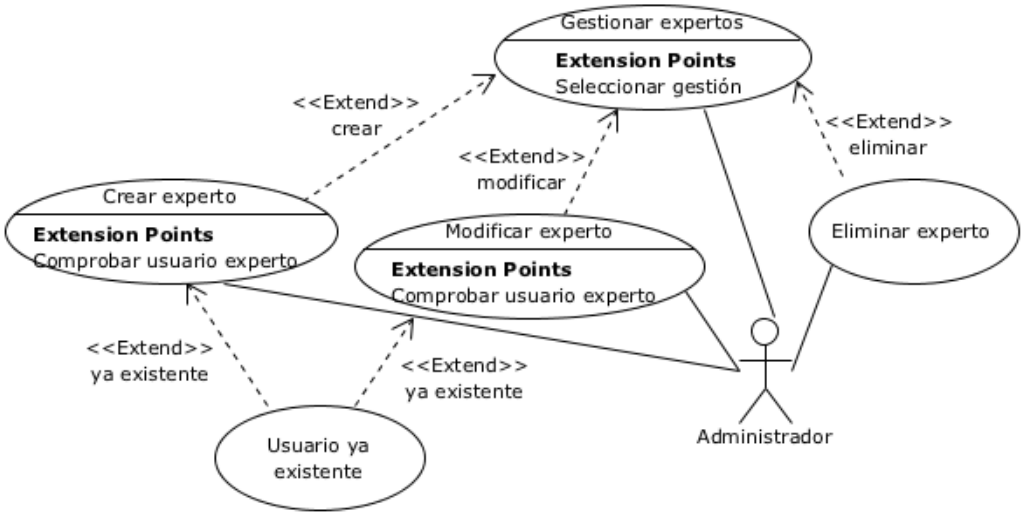


Figura 5.3. Gestionar expertos.

- ***Nombre***

  Gestionar expertos.

- ***Actores***

  Administrador.

- ***Condiciones de entrada***

El administrador tiene que estar identificado en el sistema. El administrador ha de conocer los datos que sean necesarios para crear, modificar o eliminar un experto.

- ***Flujo de eventos***

  El caso de uso es iniciado por el administrador.

  1. El usuario se dispone a realizar algún tipo de gestión relacionada con los expertos, por lo que ha seleccionado esta opción.

  2. El sistema muestra todos los expertos.

  3. Si el usuario pretende crear un experto se activa el *Subflujo 1*. Si lo que quiere es modificarlo se activa el *Subflujo 2*. Si quiere eliminarlo se activa el *Subflujo 3*.

Subflujo 1:

1. El usuario elige crear un experto.
2. El sistema solicita los datos del experto.
3. El usuario introduce los datos.
4. El sistema comprueba que no exista un experto con el mismo nombre de usuario y almacena los datos del nuevo experto. Puede ocurrir la *Excepción 1*.

Subflujo 2:

1. El usuario elige el experto que quiere modificar.
2. El sistema muestra los datos actuales del experto y permite su modificación.
3. El usuario cambia los datos que desea modificar.
4. El sistema comprueba que no exista otro experto con el mismo nombre de usuario y actualiza los datos actuales del experto. Puede ocurrir la *Excepción 1*.

Subflujo 3:

1. El usuario selecciona el experto y lo elimina.
2. El sistema elimina el experto elegido.

Excepciones:

- Excepción 1: se produce cuando el nombre de usuario del experto que se pretende crear o el nuevo nombre de usuario de un experto que se modifica ya está asignado a otro. Se informa mediante un mensaje y se impide la creación o modificación del experto.

✦ ***Condiciones de salida***

Según la elección del administrador, se crea un nuevo experto, se modifican los datos de uno ya existente o se elimina un experto del sistema.

5.2.4. Gestionar problemas

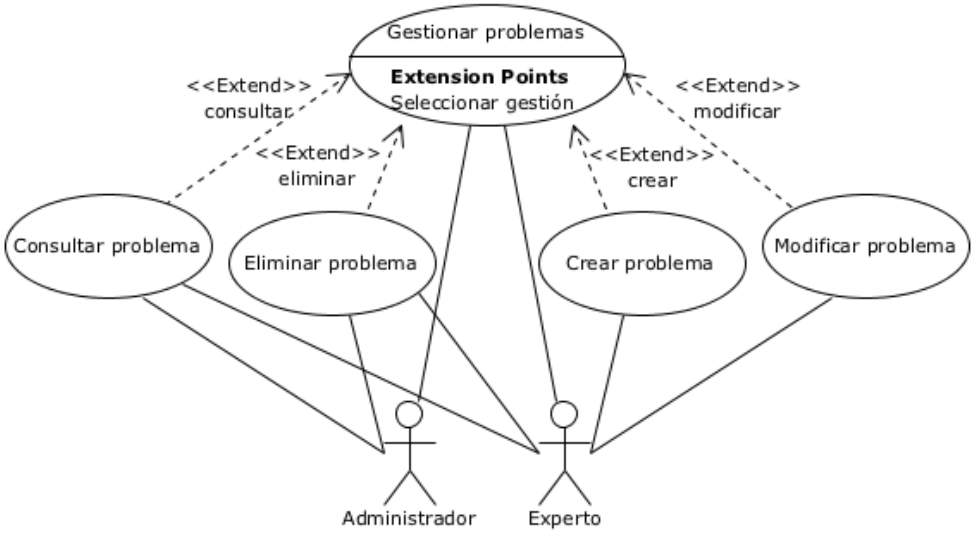


Figura 5.4. Gestionar problemas.

- ***Nombre***

  Gestionar problemas.

- ***Actores***

  Administrador y experto.

- ***Condiciones de entrada***

El administrador y el experto tienen que estar identificados en el sistema. Ambos han de conocer el problema que quieren consultar o eliminar. El experto tiene que saber los datos necesarios para crear un problema o modificarlo.

- ***Flujo de eventos***

  El caso de uso es iniciado por el administrador o el experto.

  1. El usuario se dispone a realizar algún tipo de gestión relacionada con los problemas, por lo que ha seleccionado esta opción.

  2. Si el usuario es el administrador, el sistema muestra todos los problemas existentes. Si es un experto, muestra todos los problemas que le pertenecen.

3. Si el usuario pretende consultar un problema se activa el *Subflujo 1*. Si lo que quiere es eliminarlo se activa el *Subflujo 2*. También puede crear un problema o modificarlo tal y como se describe en sus correspondientes casos de uso.

Subflujo 1:

1. El usuario elige el problema que quiere consultar.

2. El sistema muestra los datos del problema seleccionado.

Subflujo 2:

1. El usuario selecciona un problema y lo elimina.

2. El sistema elimina el problema elegido.

- ***Condiciones de salida***

Según la elección del administrador o el experto, se puede consultar, crear, modificar o eliminar un problema.

5.2.5. Crear problema

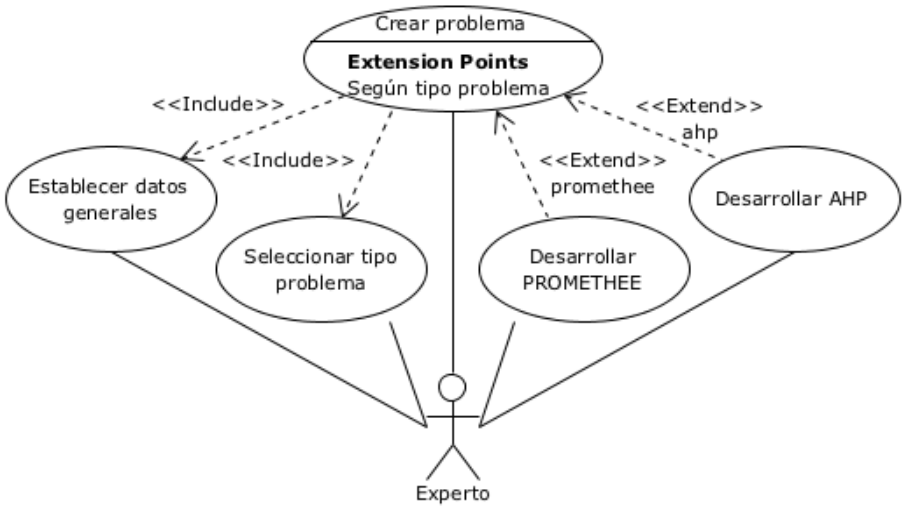


Figura 5.5. Crear problema.

- ***Nombre***

Crear problema.

- ***Actores***

  Experto.

- ***Condiciones de entrada***

El experto tiene que estar identificado en el sistema. El experto ha de conocer los datos que sean necesarios para crear un problema.

- ***Flujo de eventos***

  El caso de uso es iniciado por el experto.

  1. El usuario selecciona la opción de crear problema.
  2. El sistema solicita los datos generales del problema.
  3. El usuario introduce los datos generales.
  4. El sistema necesita conocer el tipo de problema que se va a crear.
  5. El usuario indica el tipo de problema.
  6. En función del tipo seleccionado, el sistema crea el problema según el modelo PROMETHEE o AHP.
  7. El sistema almacena los datos y resultados del problema.

- ***Condiciones de salida***

Se crea un nuevo problema, aplicando el método seleccionado sobre los datos proporcionados por el experto y se almacena en el sistema.

5.2.6. Modificar problema

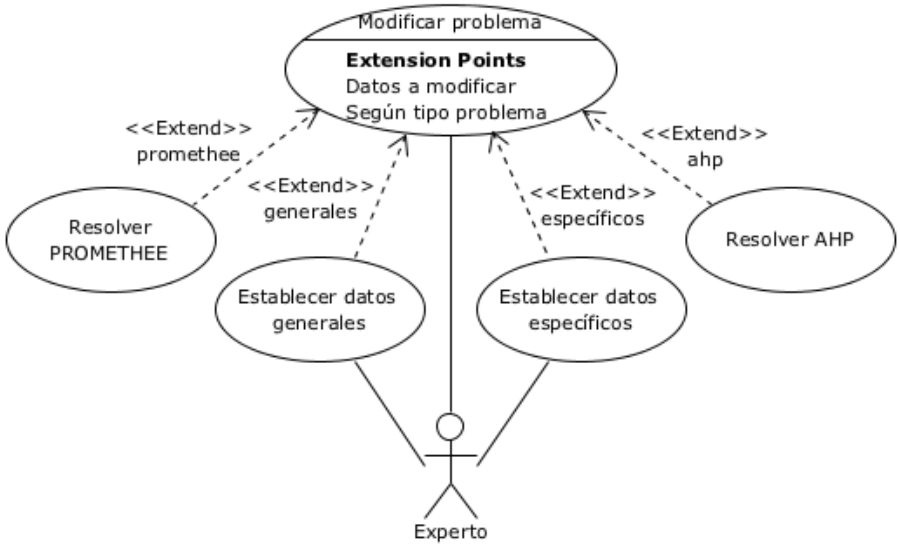


Figura 5.6. Modificar problema.

- ***Nombre***

  Modificar problema.

- ***Actores***

  Experto.

- ***Condiciones de entrada***

  El experto tiene que estar identificado en el sistema. El experto ha de conocer los datos que quiera modificar en el problema y sus nuevos valores.

- ***Flujo de eventos***

  El caso de uso es iniciado por el experto.

  1. El usuario selecciona el problema que quiere modificar.
  2. El sistema muestra los datos actuales del problema y permite su modificación.
  3. El usuario modifica los datos generales o específicos del problema que considere pertinentes.
  4. Según el tipo de problema que se está modificando se aplica un método de resolución determinado.

5. El sistema actualiza los nuevos datos y resultados del problema modificado.

✦ ***Condiciones de salida***

Se modifica un problema ya existente, aplicando el método correspondiente sobre los nuevos datos proporcionados por el experto y se actualiza en el sistema.

5.2.7. Crear PROMETHEE

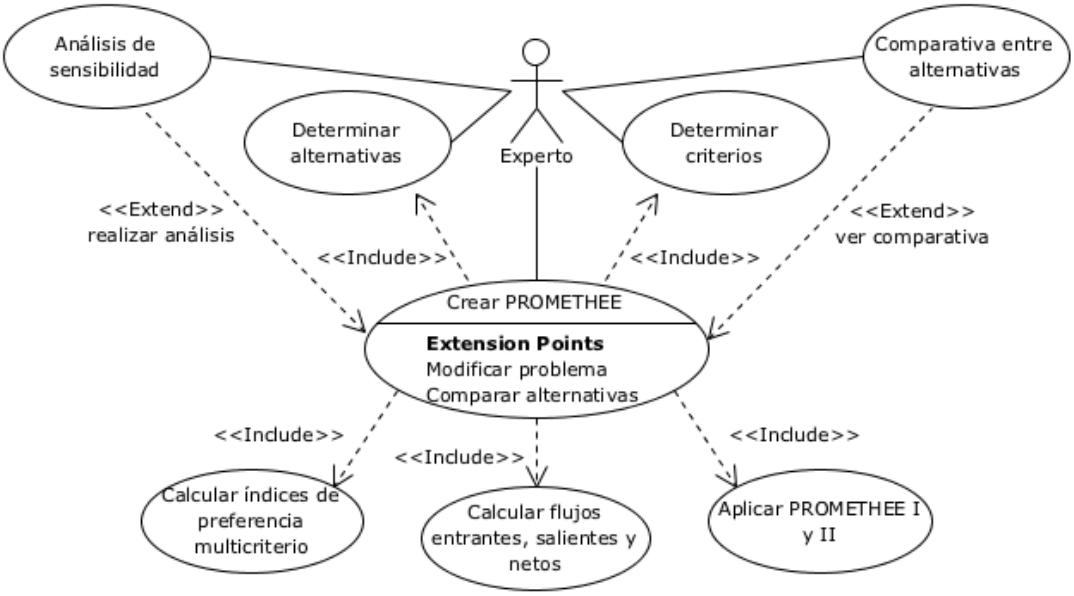


Figura 5.7. Crear PROMETHEE.

✦ ***Nombre***

Crear PROMETHEE.

✦ ***Actores***

Experto.

✦ ***Condiciones de entrada***

El experto tiene que estar identificado en el sistema. El experto debe conocer las alternativas del problema, los criterios y las valoraciones de las alternativas respecto de cada criterio.

✦ ***Flujo de eventos***

El caso de uso es iniciado por el experto.

1. El usuario ha seleccionado el tipo de problema PROMETHEE.
2. El sistema solicita las alternativas del problema.
3. El usuario introduce las alternativas y sus características.
4. El sistema pide los criterios del problema.
5. El usuario introduce cada criterio con sus respectivas características.
6. El sistema calcula los índices de preferencia multicriterio.
7. El sistema calcula los flujos entrantes, salientes y netos.
8. El sistema aplica los métodos PROMETHEE I y II y muestra los resultados obtenidos. Si el usuario realiza un análisis de sensibilidad se activa el *Subflujo 1*. Si el usuario desea ver la comparativa entre un par de alternativas se activa el *Subflujo 2*.

Subflujo 1:

1. El usuario decide modificar alguno de los datos introducidos en el problema.
2. El sistema muestra los datos actuales.
3. El usuario modifica los datos que considere pertinentes.
4. El sistema calcula los índices de preferencia multicriterio.
5. El sistema calcula los flujos entrantes, salientes y netos.
6. El sistema aplica los métodos PROMETHEE I y II y muestra los nuevos resultados obtenidos. Si el usuario realiza un análisis de sensibilidad se activa el *Subflujo 1* de nuevo. Si el usuario desea ver la comparativa entre un par de alternativas se activa el *Subflujo 2*.

Subflujo 2:

1. El usuario decide ver la comparativa entre dos alternativas.
2. El sistema muestra los datos asociados a la comparativa seleccionada.

✦ ***Condiciones de salida***

Se resuelve el problema de decisión por medio de los métodos PROMETHEE I y II obteniéndose una clasificación ordenada entre las alternativas.

5.2.8. Crear AHP

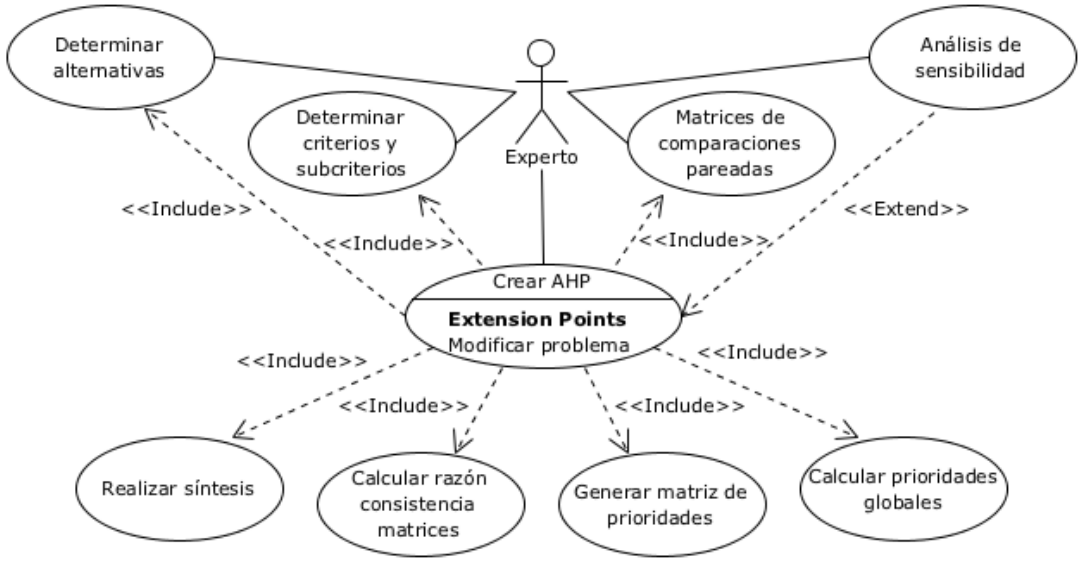


Figura 5.8. Crear AHP.

- ***Nombre***

  Crear AHP.

- ***Actores***

  Experto.

- ***Condiciones de entrada***

  El experto tiene que estar identificado en el sistema. El experto debe conocer las alternativas del problema, los criterios y las valoraciones para crear las matrices de comparaciones pareadas (matrices de comparaciones entre criterios, y de alternativas respecto de cada criterio).

- ***Flujo de eventos***

  El caso de uso es iniciado por el experto.

  1. El usuario ha seleccionado el tipo de problema AHP.
  2. El sistema solicita las alternativas del problema.
  3. El usuario introduce las alternativas y sus características.
  4. El sistema pide los criterios y posibles subcriterios del problema.

5. El usuario introduce los criterios y posibles subcriterios con sus respectivas características.

6. El sistema requiere los valores de las matrices de comparaciones pareadas.

7. El usuario introduce los valores de las comparaciones entre criterios y entre alternativas.

8. El sistema lleva a cabo el proceso de síntesis (cálculo de las prioridades de las matrices de comparaciones pareadas).

9. El sistema calcula la razón de consistencia de las matrices de comparaciones pareadas.

10. El sistema calcula la matriz de prioridades.

11. El sistema indica las matrices inconsistentes, calcula las prioridades globales de cada alternativa y muestra los resultados. Si el usuario realiza un análisis de sensibilidad se activa el *Subflujo 1*.

Subflujo 1:

1. El usuario decide modificar alguno de los datos introducidos en el problema.

2. El sistema muestra los datos actuales.

3. El usuario modifica los datos que considere pertinentes.

4. El sistema lleva a cabo el proceso de síntesis.

5. El sistema calcula la razón de consistencia de las matrices de comparaciones pareadas.

6. El sistema calcula la matriz de prioridades.

7. El sistema indica las matrices inconsistentes, calcula las prioridades globales de cada alternativa y muestra los nuevos resultados. Si el usuario realiza un análisis de sensibilidad se activa el *Subflujo 2* de nuevo.

✦ ***Condiciones de salida***

Se resuelve el problema de decisión por medio del método AHP obteniéndose una clasificación ordenada entre las alternativas.

5.3. Escenarios

Un caso de uso es una representación abstracta de una funcionalidad del sistema. La representación concreta de un caso de uso se realiza mediante la creación de uno o más escenarios que muestren todas las interacciones posibles entre el sistema y sus usuarios.

Un escenario es una posible interacción entre el sistema y algunas personas o subsistemas que suele describirse como una secuencia de mensajes [27]. Se trata de historias ficticias que describen posibles interacciones entre el usuario y la aplicación y que permiten a los diseñadores anticiparse a los problemas. Aunque son historias ficticias deben hacerse lo más detalladas posibles, así por ejemplo, los personajes deben tener nombres, motivaciones para usar la aplicación, deben encontrarse en entornos reales con las restricciones que ello conlleva, etc. De esta manera, se facilita a los diseñadores la discusión sobre la interfaz ya que a las personas nos cuesta más trabajo discutir sobre una situación abstracta.

Esta forma de proceder fuerza a los diseñadores a considerar el rango de usuarios que va a usar el sistema y el rango de actividades por las que lo van a usar. Los escenarios permiten hacer diferentes combinaciones de usuarios y actividades de forma que se tengan en cuenta todas las posibilidades.

Un escenario esta formado por los siguientes elementos:

- Un nombre único y unívoco.
- Una descripción.
- Los actores participantes.
- El flujo de eventos.

Como se ha indicado, para cada caso de uso puede haber varios escenarios. Debido al al número de casos de uso que incorpora este proyecto y a los posibles caminos alternativos que incorporan algunos, con el objetivo de reducir el número de escenarios no se van a crear todos, sino que se van a definir aquellos relacionados con el camino principal del caso de uso. De este modo se recogen las actuaciones más comunes dejando al margen las situaciones excepcionales y por tanto menos frecuentes.

5.3.1. Identificar usuario

✦ ***Nombre***

Identificar usuario.

- ***Descripción***

El experto introduce su nombre y contraseña para identificarse ante el sistema y éste los comprueba para permitirle el acceso.

- ***Actores***

Experto: Juan.

- ***Flujo de eventos***

1. Juan trata de identificarse ante el sistema.
2. El sistema solicita su nombre de usuario y su contraseña.
3. Juan introduce como nombre de usuario juan y como contraseña 123456.
4. El sistema comprueba que sus datos son válidos y le permite la entrada.

5.3.2. Crear experto

- ***Nombre***

Crear experto.

- ***Descripción***

El administrador pretende crear un nuevo experto y almacenarlo en el sistema.

- ***Actores***

Administrador: Francisco.

- ***Flujo de eventos***

1. Francisco quiere crear un nuevo experto por lo que selecciona la opción de gestionar expertos.
2. El sistema muestra los expertos almacenados en el sistema.
3. Francisco elige la opción de crear experto.
4. El sistema pide los datos del nuevo experto.
5. Francisco introduce los datos.

- Usuario: juan.
- Contraseña: 123456.
- Nombre: Juan Garrido Colmenero.
- Descripción: becario del departamento de informática.

6. El sistema comprueba que el nombre de usuario no pertenece a otro ya existente y crea el nuevo experto almacenándolo en el sistema.

5.3.3. Modificar experto

✦ ***Nombre***

Modificar experto.

✦ ***Descripción***

El administrador pretende modificar el nombre de usuario de un experto ya existente en el sistema cuyo nombre de usuario actual es juan.

✦ ***Actores***

Administrador: Francisco.

✦ ***Flujo de eventos***

1. Francisco quiere modificar el nombre de usuario de un experto por lo que selecciona la opción de gestionar expertos.

2. El sistema muestra los expertos almacenados en el sistema.

3. Francisco busca el experto que tenga como nombre de usuario juan y lo selecciona para modificarlo.

4. El sistema muestra los datos actuales del experto seleccionado, a excepción de la contraseña por razones de seguridad.

   - Usuario: juan.
   - Nombre: Juan Garrido Colmenero.
   - Descripción: becario del departamento de informática.

5. Francisco cambia el nombre de usuario por jgarrido.

   - Usuario: jgarrido.
   - Nombre: Juan Garrido Colmenero.

- Descripción: becario del departamento de informática.

6. El sistema comprueba que el nombre de usuario no pertenece a otro ya existente y actualiza los datos del experto.

5.3.4. Eliminar experto

✦ ***Nombre***

Eliminar experto.

✦ ***Descripción***

El administrador pretende eliminar del sistema a un experto ya existente cuyo nombre de usuario es jgarrido.

✦ ***Actores***

Administrador: Francisco.

✦ ***Flujo de eventos***

1. Francisco quiere eliminar a un experto del sistema por lo que selecciona la opción de gestionar expertos.
2. El sistema muestra los expertos almacenados en el sistema.
3. Francisco busca el experto que tenga como nombre de usuario jgarrido y lo elimina.
4. El sistema elimina al experto seleccionado.

5.3.5. Crear problema

✦ ***Nombre***

Crear problema.

✦ ***Descripción***

Un experto pretende crear y almacenar en el sistema un nuevo problema de decisión multicriterio para resolverlo aplicando el método PROMETHEE.

✦ ***Actores***

Experto: Juan.

- ***Flujo de eventos***

1. Juan quiere crear un nuevo problema de decisión por lo que elige la opción gestionar problemas.
2. El sistema muestra los problemas pertenecientes a Juan.
3. Juan elige la opción de crear problema.
4. El sistema solicita los datos generales del problema.
5. Juan introduce los datos generales:
   - Nombre: Estación hidroeléctrica.
   - Objetivo: Encontrar el mejor emplazamiento.
6. El sistema requiere el tipo de problema que se pretende crear.
7. Juan indica el tipo PROMETHEE.
8. De acuerdo al tipo elegido, el sistema crea el problema según el modelo PROMETHEE.
9. El sistema almacena los datos del problema y los resultados obtenidos.

5.3.6. Modificar problema

- ***Nombre***

Modificar problema.

- ***Descripción***

Un experto pretende modificar un problema ya almacenado en el sistema cuyo nombre es Estación hidroeléctrica. Quiere modificar el nombre y las valoraciones asignadas al criterio número dos.

- ***Actores***

Experto: Juan.

- ***Flujo de eventos***

1. Juan quiere modificar algunos datos de un problema de decisión, por lo que elige la opción gestionar problemas.

2. El sistema muestra los problemas pertenecientes a Juan.

3. Juan busca el problema cuyo nombre es Estación hidroeléctrica y lo selecciona para modificarlo.

4. El sistema muestra los datos actuales del problema seleccionado, entre los que se encuentran su nombre y las valoraciones de los criterios.

   - Nombre: Estación hidroeléctrica.
   - Valoraciones criterios.

	A1	A2	A3	A4	A5	A6
C1	80	65	83	40	52	94
C2	90	58	60	80	72	96
C3	6	2	4	10	6	7
C4	5,4	9,7	7,2	7,5	2,0	3,6
C5	8	1	4	7	3	5
C6	5	1	7	10	8	6

5. Juan cambia el nombre del problema por Estación hidroeléctrica MLC y las valoraciones del criterio número dos.

   - Nombre: Estación hidroeléctrica MLC.
   - Valoraciones criterios.

	A1	A2	A3	A4	A5	A6
C1	80	65	83	40	52	94
C2	45	29	30	40	36	48
C3	6	2	4	10	6	7
C4	5,4	9,7	7,2	7,5	2,0	3,6
C5	8	1	4	7	3	5
C6	5	1	7	10	8	6

6. El sistema resuelve el problema con los nuevos datos aplicando el método PROMETHEE.

7. El sistema actualiza los nuevos datos y resultados del problema.

5.3.7. Eliminar problema

- ***Nombre***

  Eliminar problema.

- ***Descripción***

  Un experto pretende eliminar del sistema un problema ya existente cuyo nombre es Estación hidroeléctrica MLC.

- ***Actores***

  Experto: Juan.

- ***Flujo de eventos***

1. Juan quiere eliminar un problema de decisión del sistema por lo que elige la opción gestionar problemas.

2. El sistema muestra los problemas pertenecientes a Juan.

3. Juan busca en la lista de problemas aquel cuyo nombre sea Estación hidroeléctrica MLC y elige eliminarlo.

4. El sistema elimina el problema.

5.3.8. Consultar problema

- ***Nombre***

  Consultar problema.

- ***Descripción***

  Un experto pretende consultar el problema de decisión denominado Ubicación de una planta industrial, visualizando sus características y la solución obtenida.

- ***Actores***

  Experto: Juan.

- ***Flujo de eventos***

  1. Juan quiere consultar los datos de un problema de decisión por lo que elige la opción gestionar problemas.

  2. El sistema muestra los problemas pertenecientes a Juan.

  3. Juan busca el problema cuyo nombre sea Ubicación de una planta industrial, lo selecciona y elige la opción consultar problema.

  4. El sistema muestra los datos del problema elegido.

     - Nombre: Ubicación de un planta industrial.
     - Objetivo: Encontrar la localización más favorable.
     - Tipo: AHP.
     - Alternativas.

Alternativas	
A1	Almería
A2	Barcelona
A3	Cádiz

     - Criterios.

Criterios	
C1	Precio del terreno
C2	Distancia a proveedores
C3	Distancia de las oficinas

     - Comparaciones entre criterios y entre alternativas.

	C1	C2	C3	Pesos
C1	1	3	6	0,667
C2	1/3	1	2	0,222
C3	1/6	1/2	1	0,111
Sumas	1,5	4,5	9	

Razón de consistencia: 0

C1	A1	A2	A3	Pesos
A1	1	3	7	0,643
A2	1/3	1	5	0,283
A3	1/7	1/5	1	0,074
Sumas	1,48	4,2	13	

Razón de consistencia: 0,082

C2	A1	A2	A3	Pesos
A1	1	1/3	3	0,272
A2	3	1	4	0,608
A3	1/3	1/4	1	0,12
Sumas	4,33	1,58	8	

Razón de consistencia: 0,086

C3	A1	A2	A3	Pesos
A1	1	1/5	1/4	0,102
A2	5	1	1/2	0,366
A3	4	2	1	0,532

C3	A1	A2	A3	Pesos
Sumas	10	3,2	1,75	

Razón de consistencia: 0,102
Matriz inconsistente.

- Matriz de prioridades.

Alternativas	C1 0,667	C2 0,222	C3 0,111
Almería	0,643	0,272	0,102
Barcelona	0,283	0,608	0,366
Cádiz	0,074	0,12	0,532

- Resultado final.

Alternativas	Resultados
Almería	0,501
Barcelona	0,364
Cádiz	0,135

5.3.9. Crear PROMETHEE

✦ ***Nombre***

Crear PROMETHEE.

✦ ***Descripción***

Un experto elige el tipo PROMETHEE para resolver un problema de decisión y poder obtener una solución al mismo.

✦ ***Actores***

Experto: Juan.

- ***Flujo de eventos***

1. Juan elige el tipo de problema PROMETHEE.
2. El sistema solicita las alternativas del problema.
3. Juan introduce las alternativas: A1, A2, A3, A4, A5 y A6.
4. El sistema pide los criterios del problema.
5. Juan introduce los criterios y sus respectivas características.

	Criterios	Pretensiones	Pesos	Tipos de criterios	Parámetros
C1	Mano de obra	Minimizar	1/6	II	q=10
C2	Potencia	Maximizar	1/6	III	p=30
C3	Coste de construcción	Minimizar	1/6	V	q=0,5 p=5
C4	Coste de mantenimiento	Minimizar	1/6	IV	q=1 p=6
C5	Ciudades a evacuar	Minimizar	1/6	I	-
C6	Nivel de seguridad	Maximizar	1/6	VI	σ=5

	A1	A2	A3	A4	A5	A6
C1	80	65	83	40	52	94
C2	90	58	60	80	72	96
C3	6	2	4	10	6	7
C4	5,4	9,7	7,2	7,5	2,0	3,6
C5	8	1	4	7	3	5
C6	5	1	7	10	8	6

6. El sistema calcula los índices de preferencia multicriterio.

Π	A1	A2	A3	A4	A5	A6
A1		0,296	0,250	0,268	0,100	0,185
A2	0,462		0,389	0,333	0,296	0,500
A3	0,236	0,180		0,333	0,056	0,429
A4	0,399	0,505	0,305		0,223	0,212
A5	0,444	0,515	0,487	0,380		0,448
A6	0,286	0,399	0,250	0,432	0,133	

7. El sistema calcula los flujos salientes, entrantes y netos.

	A1	A2	A3	A4	A5	A6
Φ^+ (x)	1,099	1,980	1,234	1,644	2,274	1,500
Φ^- (x)	1,827	1,895	1,681	1,746	0,808	1,774
Φ (x)	-0,728	0,085	-0,447	-0,102	1,466	-0,274

8. El sistema aplica los métodos PROMETHEE I y II y muestra los resultados obtenidos.

Método	Relaciones de preferencia
PROMETHEE I	- A5 es preferente sobre A2, A3 y A4. - A4 es preferente sobre A6. - A3 y A6 son preferentes sobre A1.
PROMETHEE II	- A5 es preferente sobre A2. - A2 es preferente sobre A4. - A4 es preferente sobre A6. - A6 es preferente sobre A3. - A3 es preferente sobre A1.

5.3.10. Crear AHP

- ***Nombre***

  Crear AHP.

- ***Descripción***

Un experto elige el tipo AHP para resolver un problema de decisión y poder obtener una solución al mismo.

- ***Actores***

  Experto: Juan.

- ***Flujo de eventos***

1. Juan elige el tipo de problema AHP.
2. El sistema solicita las alternativas del problema.
3. Juan introduce las alternativas.

Alternativas	
A1	Almería
A2	Barcelona
A3	Cádiz

4. El sistema pide los criterios del problema y sus posibles subcriterios.
5. Juan introduce los criterios y sus características, puesto que el problema carece de subcriterios.

Criterios	
C1	Precio del terreno
C2	Distancia a proveedores
C3	Distancia de las oficinas

6. El sistema solicita los valores de las matrices de comparaciones pareadas.

7. Juan proporciona las comparaciones entre criterios y entre alternativas.

	C1	C2	C3
C1	1	3	6
C2	1/3	1	2
C3	1/6	1/2	1

C1	A1	A2	A3
A1	1	3	7
A2	1/3	1	5
A3	1/7	1/5	1

C2	A1	A2	A3
A1	1	1/3	3
A2	3	1	4
A3	1/3	1/4	1

C3	A1	A2	A3
A1	1	1/5	1/4
A2	5	1	1/2
A3	4	2	1

8. El sistema lleva a cabo el proceso de síntesis.

	Pesos
C1	0,667
C2	0,222

	Pesos
C3	0,111

C1	Pesos
A1	0,643
A2	0,283
A3	0,074

C2	Pesos
A1	0,272
A2	0,608
A3	0,12

C3	Pesos
A1	0,102
A2	0,366
A3	0,532

9. El sistema calcula la razón de consistencia (RC) de las matrices de comparaciones pareadas.

   - RC entre criterios: 0.
   - RC entre alternativas respecto precio del terreno: 0,082.
   - RC entre alternativas respecto distancia a proveedores: 0,086.
   - RC entre alternativas respecto distancia de las oficinas: 0,102.

10. El sistema calcula la matriz de prioridades.

Alternativas	C1 0,667	C2 0,222	C3 0,111
Almería	0,643	0,272	0,102
Barcelona	0,283	0,608	0,366
Cádiz	0,074	0,12	0,532

11. El sistema calcula las prioridades globales de cada alternativa y muestra los resultados.

Alternativas	Resultados
Almería	0,501
Barcelona	0,364
Cádiz	0,135

5.4. Análisis de tareas

El análisis de tareas se define como el proceso de analizar la forma en que los usuarios realizan su trabajo [31, 32]. Incluye reconocer las cosas que hacen y también las cosas sobre las que actúan, la información que necesitan saber para llevarlo a cabo, los objetivos que les llevan a hacerlo y los posibles errores que pueden cometer al realizar ese trabajo [31, 32].

El propósito de un usuario no es interactuar con el ordenador, sino cumplir un objetivo y la interacción es el medio por el que se logra alcanzarlo. Así, el conocimiento de las tareas del usuario requiere el conocimiento de sus objetivos y motivaciones.

A la hora de realizar el análisis de tareas hay que dejar claro cuatro aspectos: lo que el usuario tiene que hacer, lo que ve, con lo que actúa y lo que el usuario podría hacer incorrectamente.

En el análisis de tareas se pone énfasis en qué se realiza y qué etapas o pasos tiene y no en los detalles concretos de cómo se realiza, ya que esa fase compete al diseño. El análisis de tareas incluye todos los aspectos de la tarea aunque sean aspectos que no se van a programar directamente en el sistema informático.

El análisis de tareas se hace a dos niveles. En un primer nivel se determina la estructura de cada tarea de interacción individual, y en un segundo nivel se determina la arquitectura de todas la tareas de interacción en su conjunto.

En análisis de tareas permite definir con precisión como se organizan las tareas de interacción que interesan a los usuarios. Además de esta utilidad obvia, existen otras razones que justifican su uso. En primer lugar, facilita enormemente la creación de manuales de usuario. En segundo lugar, el resultado que se obtiene describe de forma muy precisa la estructura y la arquitectura de la interacción al mismo tiempo que usa un lenguaje fácil de entender por el usuario. Y por último, si durante la evaluación se descubren errores u omisiones, se pueden hacer correcciones en ese mismo momento y volver a evaluar.

5.4.1. Estructura de las tareas de interacción

Para determinar la estructura de una tarea de interacción se debe tener en cuenta cual es su objetivo, qué inicia esa tarea y cuando se considera que la tarea está completa [31, 32]. Son lo que se llama respectivamente objetivo, disparador o inicializador externo y punto de parada.

En la estructura de una tarea de interacción hay que describir exactamente lo que ocurre durante el desarrollo del sistema, es decir, la comunicación que se produce entre el usuario y el sistema [31, 32].

La estructura de las tareas de interacción está formada por acciones más sencillas que incluyen también aquellas acciones que no son directamente informatizables. Para hacer un buen análisis de tareas hay que prestar atención a las situaciones que pueden formar parte de una ejecución aunque no pertenezcan al camino principal y que por tanto deben haberse previsto en la estructura de la interacción o el sistema no les podrá dar soporte. Los caminos de ejecución alternativos se suelen expresar sangrados con respecto a la acción para la que son una alternativa y utilizando una subnumeración.

En este proyecto se compone de diez tareas de interacción cuya estructura queda determinada a continuación.

- ***Identificar usuario***

  - Objetivo: identificar a un usuario ante el sistema de modo que pueda acceder a las funcionalidades del mismo.

  - Disparador: el usuario pretende acceder al sistema y para ello tiene que identificarse introduciendo su nombre de usuario y contraseña.

    1. El usuario trata de identificarse ante el sistema.

    2. El sistema solicita el nombre de usuario y la contraseña.

    3. El usuario introduce su nombre de usuario y contraseña.

    4. El sistema identifica al usuario y permite su acceso.

4.1. Si el nombre de usuario o la contraseña son inválidos se le notifica y repetir desde la acción 2.

- ***Crear experto***
  - Objetivo: crear un nuevo experto y almacenarlo en el sistema.
  - Disparador: el usuario selecciona la opción de crear experto.
    1. El usuario selecciona crear un experto.
    2. El sistema pide el nombre de usuario del experto, su contraseña, su nombre y una descripción, éste último de forma opcional.
    3. El usuario introduce los datos del experto.
    4. El sistema crea el nuevo experto y lo guarda.
       4.1. Si el nombre de usuario pertenece a otro experto ya existente en el sistema se notifica al usuario y repetir desde la acción 2.
- ***Modificar experto***
  - Objetivo: cambiar los datos pertenecientes a un experto ya existente en el sistema.
  - Disparador: el usuario selecciona el experto cuyos datos quiere modificar.
    1. El usuario selecciona al experto que quiere modificar.
    2. El sistema muestra los datos actuales del experto.
    3. El usuario cambia aquellos datos que quiera modificar.
    4. El sistema actualiza los datos del experto.
       4.1. Si modifica el nombre de usuario y pertenece a otro experto ya existente en el sistema se notifica al usuario y repetir desde la acción 2.
- ***Eliminar experto***
  - Objetivo: eliminar del sistema a un experto.
  - Disparador: el usuario selecciona el experto que quiere borrar y elige la opción de eliminarlo.
    1. El usuario selecciona al experto y elige eliminarlo.

2. El sistema elimina al experto del sistema.

✦ ***Crear problema***

- Objetivo: definir un problema de toma de decisión y almacenarlo junto a los resultados obtenidos por el método seleccionado por el usuario.
- Disparador: el usuario selecciona la opción de crear problema.

1. El usuario selecciona crear problema.
2. El sistema solicita el nombre del problema y el objetivo.
3. El usuario introduce los datos generales del problema.
4. El sistema necesita saber el tipo de problema para aplicar el método de resolución pertinente.
5. El usuario indica el tipo de problema.
6. En función del tipo seleccionado, el sistema crea el problema según el modelo PROMETHEE o AHP y almacena todos los datos y resultados.

6.1. Si antes de guardar el problema el usuario quiere realizar un análisis de sensibilidad, se le permite modificar cualquiera de los datos introducidos y se vuelve a resolver el problema.

✦ ***Modificar problema***

- Objetivo: cambiar los datos pertenecientes a un problema de decisión ya existente en el sistema y actualizar los nuevos datos y resultados obtenidos.
- Disparador: el usuario selecciona el problema cuyos datos quiere modificar.

1. El usuario selecciona el problema que quiere modificar.
2. El sistema muestra los datos actuales del problema.
3. El usuario cambia aquellos datos que quiera modificar.
4. El sistema resuelve el problema aplicando el método asociado a su tipo y actualiza los nuevos datos y resultados obtenidos.

4.1. Si antes de actualizar el problema el usuario quiere realizar un análisis de sensibilidad, se le permite modificar cualquiera de los datos actuales y se vuelve a resolver el problema.

- ***Eliminar problema***
  - Objetivo: eliminar del sistema un problema.
  - Disparador: el usuario selecciona el problema que quiere borrar y elige la opción de eliminarlo.
    1. El usuario selecciona el problema y elige eliminarlo.
    2. El sistema elimina el problema del sistema.
- ***Consultar problema***
  - Objetivo: ver los datos de un problema de decisión existente en el sistema.
  - Disparador: el usuario selecciona el problema que quiere visualizar.
    1. El usuario selecciona el problema que quiere consultar.
    2. El sistema muestra los datos del problema así como los resultados obtenidos.
- ***Crear PROMETHEE***
  - Objetivo: definir los datos específicos de un problema de decisión multicriterio y resolverlo, obteniendo así una solución de entre las alternativas disponibles por medio de la aplicación del método PROMETHEE.
  - Disparador: el usuario continúa con el desarrollo del problema e indica que es del tipo PROMETHEE.
    1. El usuario selecciona el tipo de problema PROMETHEE.
    2. El sistema solicita los nombres de las alternativas del problema.
    3. El usuario introduce el nombre de las alternativas.
    4. El sistema solicita el nombre de cada criterio, su pretensión, el peso de los mismos, el tipo generalizado de cada criterio así como sus respectivos parámetros y la evaluación de las alternativas respecto de ellos.
    5. El usuario proporciona las características de los criterios.
    6. El sistema aplica el método PROMETHEE al problema especificado y muestra los resultados obtenidos.

6.1. Si las características de los criterios son erróneas se notifica al usuario y repetir desde la acción 4.

- ***Crear AHP***

  - Objetivo: definir los datos específicos de un problema de decisión multicriterio y resolverlo, obteniendo así una solución de entre las alternativas disponibles por medio de la aplicación del método AHP.

  - Disparador: el usuario continúa con el desarrollo del problema e indica que es del tipo AHP.

    1. El usuario selecciona el tipo de problema AHP.
    2. El sistema solicita los nombres de las alternativas del problema.
    3. El usuario introduce el nombre de las alternativas.
    4. El sistema solicita el nombre de cada criterio y de los posibles subcriterios.
    5. El usuario proporciona el nombre de los criterios y posibles subcriterios.
    6. El sistema necesita que se especifiquen las matrices de comparaciones pareadas.
    7. El usuario rellena los valores de las matrices.
    8. El sistema aplica el método AHP al problema especificado y muestra los resultados obtenidos.

       8.1. Si los valores de las comparaciones pareadas son erróneos se notifica al usuario y repetir desde la acción 6.

5.4.2. Arquitectura de la interacción

Una vez determinada la estructura de cada tarea de interacción por separado, es el momento de reunirlas todas para determinar la arquitectura de la interacción del sistema completo. En esta ocasión el énfasis se pone en cómo se combinan las distintas tareas del usuario.

Hay varios enfoques para realizar esta parte del análisis de tareas, en este caso concreto se va a hacer uso del enfoque basado en descomposición de tareas. El análisis jerárquico de tareas o Hierarchical Task Analysis (HTA) es una de las formas más sencillas de análisis de tareas más conocidas y empleadas. El resultado del HTA es una jerarquía que muestra [31, 32]:

- La descomposición de la arquitectura de la interacción del sistema completo en un conjunto de tareas de interacción y en las subtareas que la forman.

- Los planes que describen en qué orden y bajo que condiciones se realizan las subtareas.

Tanto las tareas como subtareas se numeran y subnumeran respectivamente, además no hay que confundir esta nomenclatura con la que se usa para identificar los caminos alternativos en la estructura de una tarea de interacción. En cuanto a los planes, existe uno para todo el sistema y para cada tarea que tenga subtareas. Los planes también se numeran según la tarea o subtarea a la que corresponden. Las tareas y subtareas pueden realizarse siempre, opcionalmente, repetirse, llevarse a cabo al mismo tiempo, en cualquier orden o mezclando algunas de las condiciones anteriores, pero todo esto siempre queda reflejado en los planes.

La realización de un análisis jerárquico de tareas (HTA) es un proceso iterativo, se comienza a partir de los casos de uso o la estructura de cada tarea de interacción individual y se averiguan que subtareas lo forman y en qué orden se realizan.

La jerarquía de tareas puede expresarse en modo texto o también en forma de diagrama. Ambos expresan la misma información, pero en un formato distinto, haciéndola más fácil de entender según la persona que interprete el documento. Para este proyecto se ha decidido expresar la jerarquía de tareas únicamente en modo texto ya que debido a las numerosas tareas existentes es difícil dibujar el diagrama en un tamaño tan limitado.

✦ ***Listas de tareas en modo texto***

0. Sistema toma de decisión multicriterio.

    1. Identificar usuario.

        1.1. Introducir nombre de usuario.

        1.2. Introducir contraseña.

    2. Esperar comprobación.

    3. Crear experto.

        3.1. Determinar nombre de usuario.

        3.2. Determinar contraseña.

        3.3. Proporcionar nombre.

        3.4. Proporcionar descripción.

    4. Modificar experto.

        4.1. Modificar nombre de usuario.

4.2. Modificar contraseña.

4.3. Modificar nombre.

4.4. Modificar descripción.

5. Eliminar experto.

6. Crear problema.

6.1. Introducir nombre del problema.

6.2. Determinar objetivo.

6.3. Seleccionar tipo de problema.

6.4. Crear PROMETHEE.

6.4.1. Nombrar alternativas.

6.4.2. Determinar características criterios.

6.5. Crear AHP.

6.5.1. Nombrar alternativas.

6.5.2. Nombrar criterios y subcriterios.

6.5.3. Rellenar matrices de comparación pareadas.

6.6. Análisis de sensibilidad.

7. Modificar problema.

7.1. Modificar nombre del problema.

7.2. Modificar objetivo.

7.3. Modificar alternativas.

7.4. Modificar criterios.

7.5. Modificar valoraciones.

7.6. Análisis de sensibilidad.

8. Eliminar problema.

9. Consultar problema.

Ahora se detallan los planes que indican en qué orden se llevan a cabo las tareas y subtareas:

- Plan 0: Hacer 1 - 2,
  si comprobación correcta hacer 3 - 4 - 5 - 6 - 7 - 8 - 9 opcionalmente.

- Plan 1: Hacer 1.1 - 1.2 en cualquier orden.
- Plan 3: Hacer 3.1 - 3.2 - 3.3 - 3.4 en cualquier orden.
- Plan 4: Hacer 4.1 - 4.2 - 4.3 - 4.4 en cualquier orden, opcionalmente, hasta que el usuario haya hecho las modificaciones oportunas.
- Plan 6: Hacer 6.1 - 6.2 - 6.3 en cualquier orden,
  hacer 6.4 - 6.5 opcionalmente,
  hacer 6.6 opcionalmente.
- Plan 6.4: Hacer 6.4.1 - 6.4.2 en cualquier orden.
- Plan 6.5: Hacer 6.5.1 - 6.5.2 en cualquier orden,
  hacer 6.5.3.
- Plan 7: Hacer 7.1 - 7.2 - 7.3 - 7.4 - 7.5 en cualquier orden, opcionalmente, hasta que el usuario haya hecho las modificaciones oportunas,
  hacer 7.6 opcionalmente.

CAPÍTULO VI

Diseño del sistema

El diseño se describe como un proceso en el que se sintetizan representaciones de la estructura de los datos, de la estructura del programa y de las características de la interfaz del *software*, desde los requerimientos [28]. Durante esta fase se definen las estrategias a seguir para conseguir obtener un *software* que cumpla con sus objetivos, es decir, se construye un modelo que sirva como anteproyecto a la construcción de la aplicación.

Con la etapa de diseño se identifican los objetivos finales del sistema y se plantean las diversas estrategias para alcanzarlos durante la implementación. Para su desarrollo, es necesario utilizar la información generada durante la etapa previa de análisis.

En este proyecto, la fase de diseño se va a dividir en tres secciones. Por un lado, se encuentra el diseño de los objetos, que abarca aquellos diagramas relacionados con la programación orientada a objetos. Después, se detalla el diseño orientado a los datos donde se define la estructura de los mismos. Y finalmente, se define el diseño de la interfaz, donde se abarca la apariencia visual de la aplicación.

6.1. Diseño de los objetos

El cometido de esta etapa es crear un modelo orientado a objetos del sistema *software* que satisfaga los requerimientos, en donde los objetos interactúan entre sí [25]. Cada objeto mantiene su estado de forma privada, por lo que no se puede acceder a él desde fuera, y provee operaciones que son las que permiten modificar su estado.

De un modo más concreto, se dice que la etapa comprende el diseño de las clases de objetos que conforman el sistema y las relaciones entre éstas. Además, las clases definen los objetos del sistema y sus interacciones. Dentro del diseño de los objetos existen multitud de diagramas con un propósito muy bien definido. En este proyecto, se hace uso del diagrama de paquetes y del diagrama de clases.

6.1.1. Diagrama de paquetes

Como su propio nombre indica, este diagrama agrupa los paquetes que constituyen la aplicación. Un paquete se puede definir como una colección de elementos del modelo, entendiéndose por elemento del modelo, cualquier componente que pueda representarse por un elemento del diagrama [27]. En el contexto de este proyecto, los paquetes únicamente van a contener clases de diseño. Un paquete aparece en el diagrama representado como un rectángulo con otro pequeño rectángulo en el borde superior izquierdo. Los elementos contenidos en el paquete pueden dibujarse dentro de él, aunque no es necesario. De hecho, normalmente no se dibujan ya que dentro del paquete hay una gran cantidad de detalles difíciles de incluir, además, el objetivo principal del mismo es poder ocultar esos detalles.

Aunque existen varias razones para hacer uso del diagrama de paquetes, el objetivo de la Figura 6.1 es mostrar una visión global de cómo se organizan y distribuyen las clases de diseño del sistema, tratando de ocultar los detalles de agrupación y relación entre éstas.

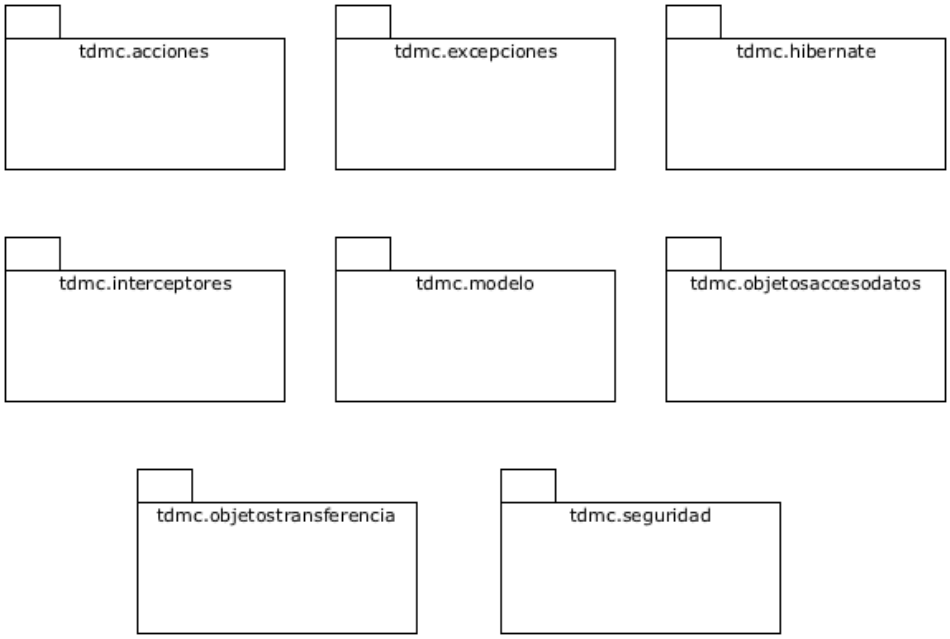


Figura 6.1. Diagrama de paquetes.

Estos ocho paquetes se describen con más detalle durante el diagrama de clases, ya que este último se va a organizar de acuerdo a los paquetes que conforman el sistema.

6.1.2. Diagrama de clases

El diagrama de clases representa las especificaciones de las clases e interfaces *software* [29]. Se utiliza para documentar y representar la estructura estática del sistema, esto es, qué clases hay y cómo están relacionadas [27]. Es importante dejar claro que el diagrama de clases no modela la interacción entre las clases para alcanzar comportamientos particulares, de esta función se encargan otro diagramas. Entre su información general se encuentran: clases, asociaciones, atributos, interfaces, con sus operaciones y constantes, métodos, navegabilidad y dependencias.

El diagrama de clases cuenta con dos elementos principales, las clases y los tipos de relaciones entre ellas.

- ***Clases***

Las clases describen un conjunto de objetos con un rol o roles equivalentes en un sistema. Se define un objeto como una instancia concreta de una clase, de forma que tome unos valores concretos para sus datos. Las clases se representan por medio de un rectángulo dividido en tres secciones. En la parte superior aparece el nombre de la clase, que normalmente se corresponde con un sustantivo. En la parte intermedia se encuentran sus atributos, que describen los datos contenidos en un objeto de la clase. Y por último, en la parte inferior se

localizan sus métodos u operaciones, que se encargan de definir las maneras en que los objetos pueden interactuar entre sí.

En el diagrama de clases del proyecto se puede distinguir tres tipos de clases que tienen ciertas particularidades:

- Interfaces: son clases que no se pueden instanciar y se distinguen porque acompañando al nombre llevan el estereotipo «interface». Este tipo de clases carecen de atributos y las operaciones que contienen carecen de implementación. Cuando una clase implementa o realiza una interfaz tiene la obligación de proporcionar una implementación a las operaciones definidas en la interfaz.

- Clases abstractas: al igual que ocurre con las interfaces, se trata de clases que no se pueden instanciar y se distinguen por el estereotipo «abstract». Este tipo de clases sí puede contener atributos y operaciones con implementación. Además, en ellas se pueden definir operaciones de tipo abstracto que carecen de implementación y obliga a las clases que la hereden a proporcionarles una. En esencia, una clase abstracta sin atributos y con todas sus operaciones abstractas es una interfaz.

- Enumerados: son un tipo de clase que sólo tiene atributos de tipo enumerado. Se representan por el estereotipo «enumeration».

✦ ***Asociaciones***

Las asociaciones expresan las relaciones entre las clases. Al igual que las clases se corresponden con sustantivos, las asociaciones se suelen corresponder con verbos. Se representan por una línea que une las clases que se relacionan.

En ocasiones, es útil proporcionar más detalles de los que se tienen inicialmente en una asociación. Suelen ir acompañadas de una flecha que indica su navegabilidad, gracias a la cual es posible identificar en qué sentido se lee una asociación entre dos clases. También puede resultar muy útil asignar multiplicidades a cada lado de la asociación. Con la multiplicidad se expresa la cantidad de objetos de una clase que va a tener la clase contenedora.

Dentro de la asociación se pueden encontrar dos subtipos que introducen algunos matices semánticos, son la agregación y la composición. En lugar de mostrar simplemente que dos clases están asociadas se puede decidir mostrar más información sobre el tipo de asociación que es. La agregación y la composición son dos formas de representar que un objeto de una clase *es parte de* un objeto de otra clase. Normalmente este tipo de asociaciones no se suelen nombrar ya que la relación «es una parte de» queda reflejada en la agregación o composición propiamente dicha.

- Agregación: se representa por medio de una línea unida a un diamante hueco que se sitúa en el lado de la clase contenedora. La agregación se produce cuando los objetos que componen las partes de la relación pueden pertenecer a más de un todo, o

dicho de otro modo, cuando la multiplicidad en el extremo de la clase contenedora es mayor que uno.

- Composición: se representa por medio de una línea unida a un diamante sólido que se sitúa en el lado de la clase contenedora. La composición se produce cuando los objetos que componen las partes de la relación únicamente pueden pertenecer a un todo, o dicho de otro modo, cuando la multiplicidad en el extremo de la clase contenedora es uno.

✦ ***Generalización***

En primer lugar hay que distinguir entre los términos generalización y especialización. Ambos hacen referencia a la misma relación entre dos clases, pero desde puntos de vista opuestos. Si se tienen dos clases, una la clase generalizada que recibe el nombre de superclase, y otra la clase especializada o subclase, la relación de generalización se produce cuando conceptualmente todos los objetos de la subclase son objetos de la superclase. Por un lado, todo lo que pueden hacer los objetos de la superclase también lo pueden hacer los objetos de la subclase, es decir, la superclase es una generalización de la subclase. En el lado opuesto, puede haber cosas que no tengan sentido para la superclase y sí para la subclase, es decir, la subclase es una especialización de la superclase. Cómo se decía al comienzo, aunque ambos términos representan la misma relación son semánticamente opuestos.

La generalización se representa por una flecha cerrada y continua que parte de la subclase en dirección a la superclase. La herencia es una manera de implementar la generalización, y aunque no es la única suele ser la más común. Se produce cuando una clase decide asimilar los atributos y métodos de otra clase como si se trataran de los suyos propios, es decir, una clase hereda las características de la otra.

✦ ***Realización***

El término realización está muy ligado a un tipo de clase particular como son las interfaces. Como ya se ha dicho antes este tipo de clases no se pueden instanciar, por lo tanto su único objetivo es determinar una serie de métodos que deben ser implementados por aquellas clases que se relacionen con ellas a través de la realización. Por tanto, la realización se define como la relación existente entre las interfaces y las clases que quieran implementar sus métodos.

La realización se representa por una flecha cerrada y discontinua que parte de la clase que realiza hacia la interfaz.

A continuación, se procede a mostrar los diagramas de clases. Debido a la complejidad del proyecto, el diagrama de clases es muy extenso, por lo que para facilitar su legibilidad se ha decidido dividirlo en varias secciones. Aprovechando que la aplicación ha quedado perfectamente dividida en ocho paquetes, se ha realizado un diagrama de clases por cada uno de ellos. Así con todo, algunos diagramas siguen siendo demasiado grandes y es por ello que

se dividen en varias partes. Cuando sea necesario, como nexo de unión entre ellas, se utilizará alguna clase que sirva de referencia entre una parte y la siguiente.

6.1.2.1. Paquete tdmc.acciones

Este paquete contiene algunas de las clases que forman el controlador de la aplicación, de acuerdo al patrón Modelo-Vista-Controlador [29, 33, 34], que se encargan de recibir y gestionar las peticiones enviadas, llamar al modelo para que realice el procesamiento y seleccionar la siguiente página que se va a mostrar.

El marco de trabajo (*framework*) Struts 2 se encarga, entre otras funciones, de facilitar buena parte de las tareas del controlador. Hay que crear las acciones, que son clases que heredan de *ActionSupport,* y en última instancia son las que reciben las peticiones que se envían y eligen la siguiente vista que se muestra. Para que las acciones puedan tener acceso a la sesión se debe implementar la clase *SessionAware*.

Algunas de las clases del paquete siguen el patrón de diseño Ayudante de la Vista [34], cuya misión es realizar las transformaciones necesarias sobre los datos introducidos en la vista para poder utilizarlos en otros niveles de la aplicación y viceversa.

Dada la extensión del paquete se ha decidido dividirlo en tres partes, representadas por las Figuras 6.3, 6.4 y 6.5.

6.1.2.2. Paquete tdmc.excepciones

Contiene las clases que representan las excepciones que la aplicación puede lanzar durante el tiempo de ejecución cuando se produce alguna situación concreta.

Dentro del paquete se pueden ver las excepciones que ocurren cuando no existe el administrador o hay más de uno, cuando un experto o problema no existen, o bien se produce algún error durante el cálculo de una función resumen.

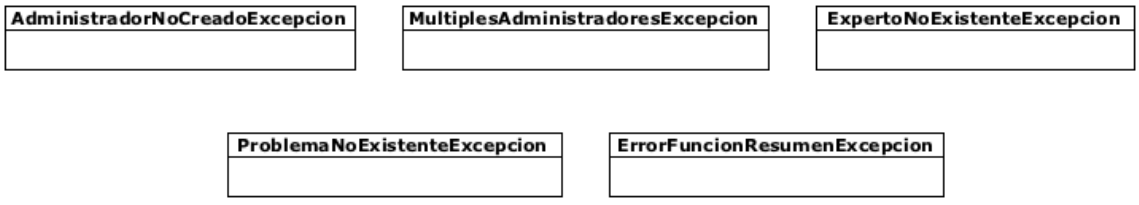


Figura 6.2. Paquete excepciones.

6.1.2.3. Paquete tdmc.hibernate

El uso del *framework* Hibernate requiere que los accesos a la base de datos se hagan a través de una sesión, que se instancia en única ocasión la primera vez que se usa. Luego, la

sesión se abre antes de iniciar una transacción y se cierra al finalizarla. La sesión es común para todos los accesos a la base de datos.

El paquete, que viene recogido por la Figura 6.6, contiene una clase que se va a encargar de instanciar la sesión de Hibernate la primera vez y de su posterior obtención en sucesivas ocasiones. Siguiendo el patrón Factoría [29, 33], se ha decidido asignar la responsabilidad de crear la sesión de Hibernate a una clase específica, que además, tiene todos sus atributos y métodos estáticos de acuerdo al patrón Singleton [29, 33], puesto que se instancia una única vez.

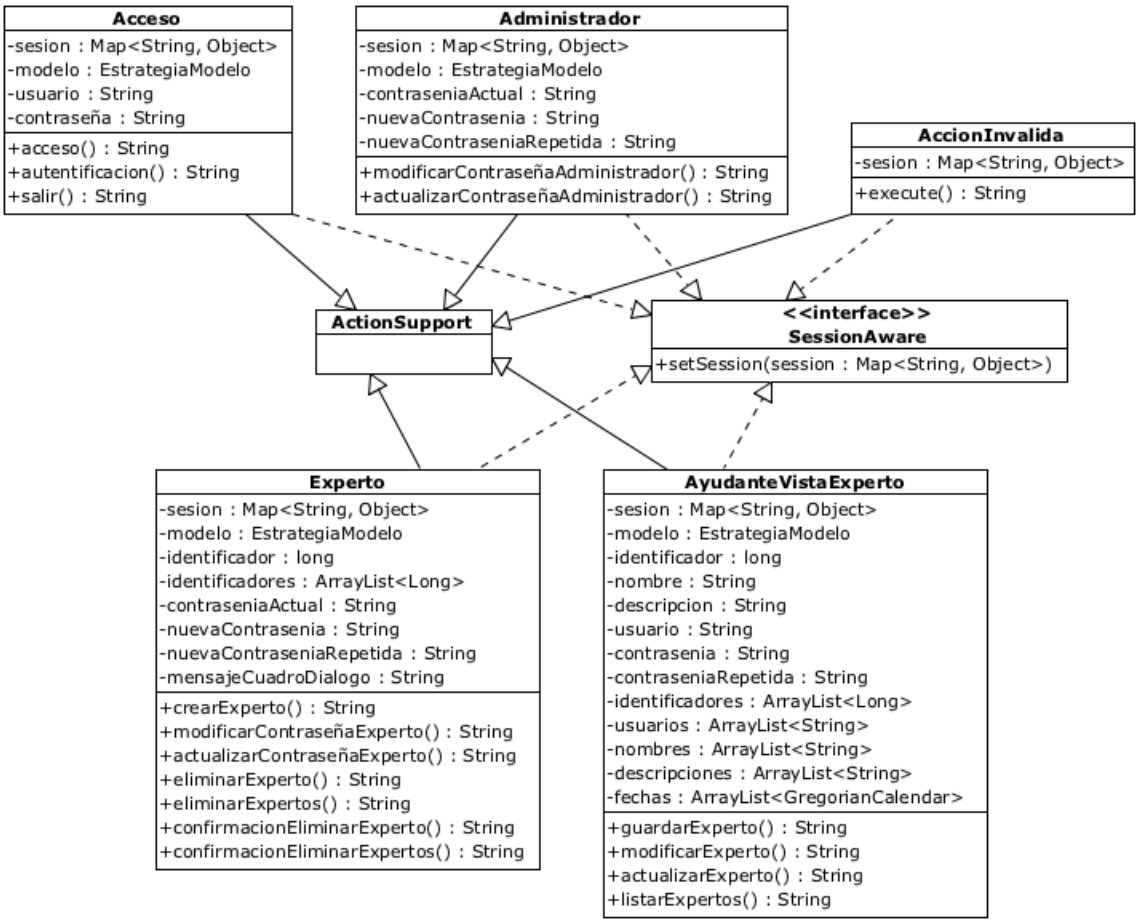


Figura 6.3. Paquete acciones. Parte I.

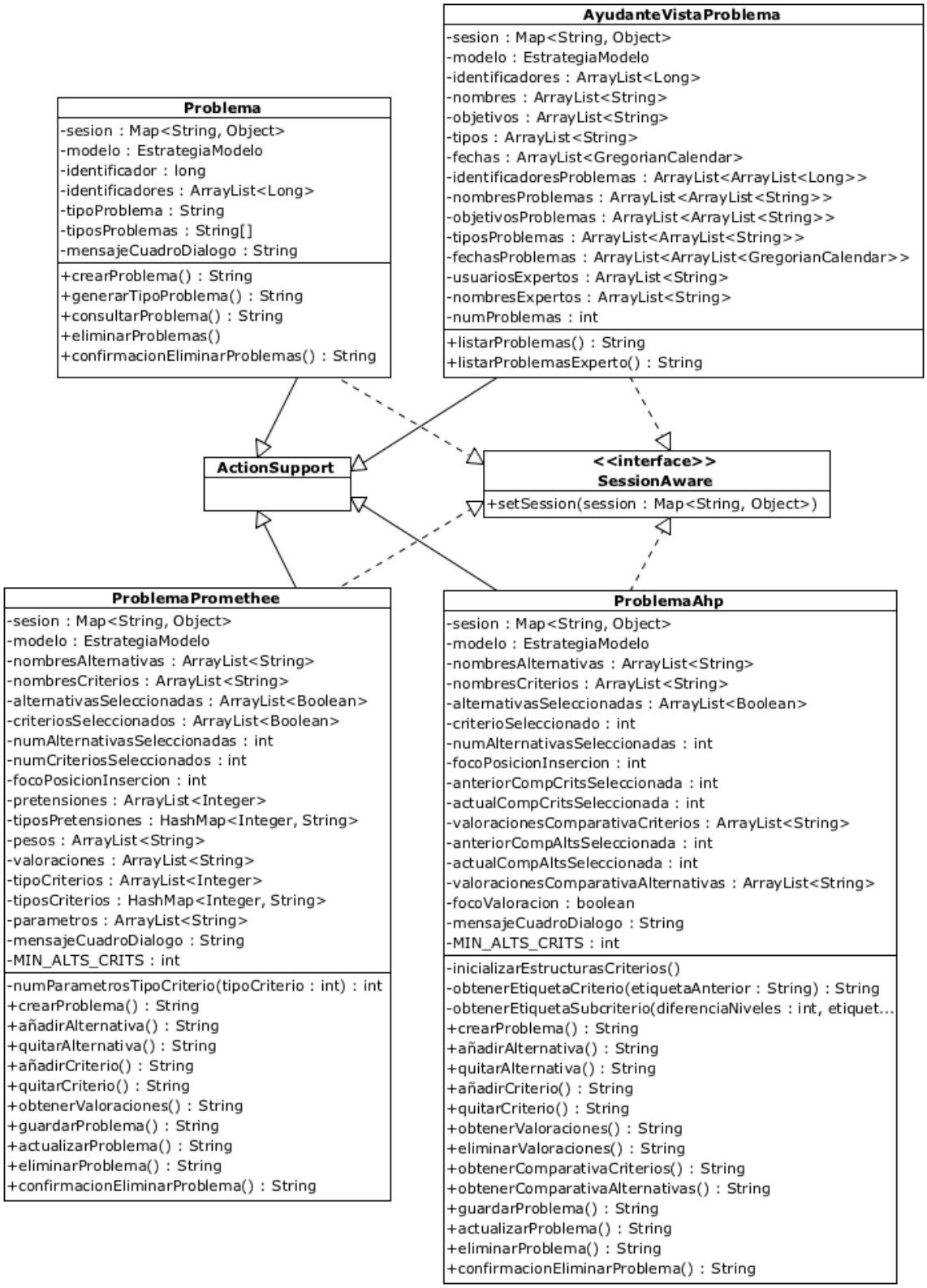


Figura 6.4. Paquete acciones. Parte II.

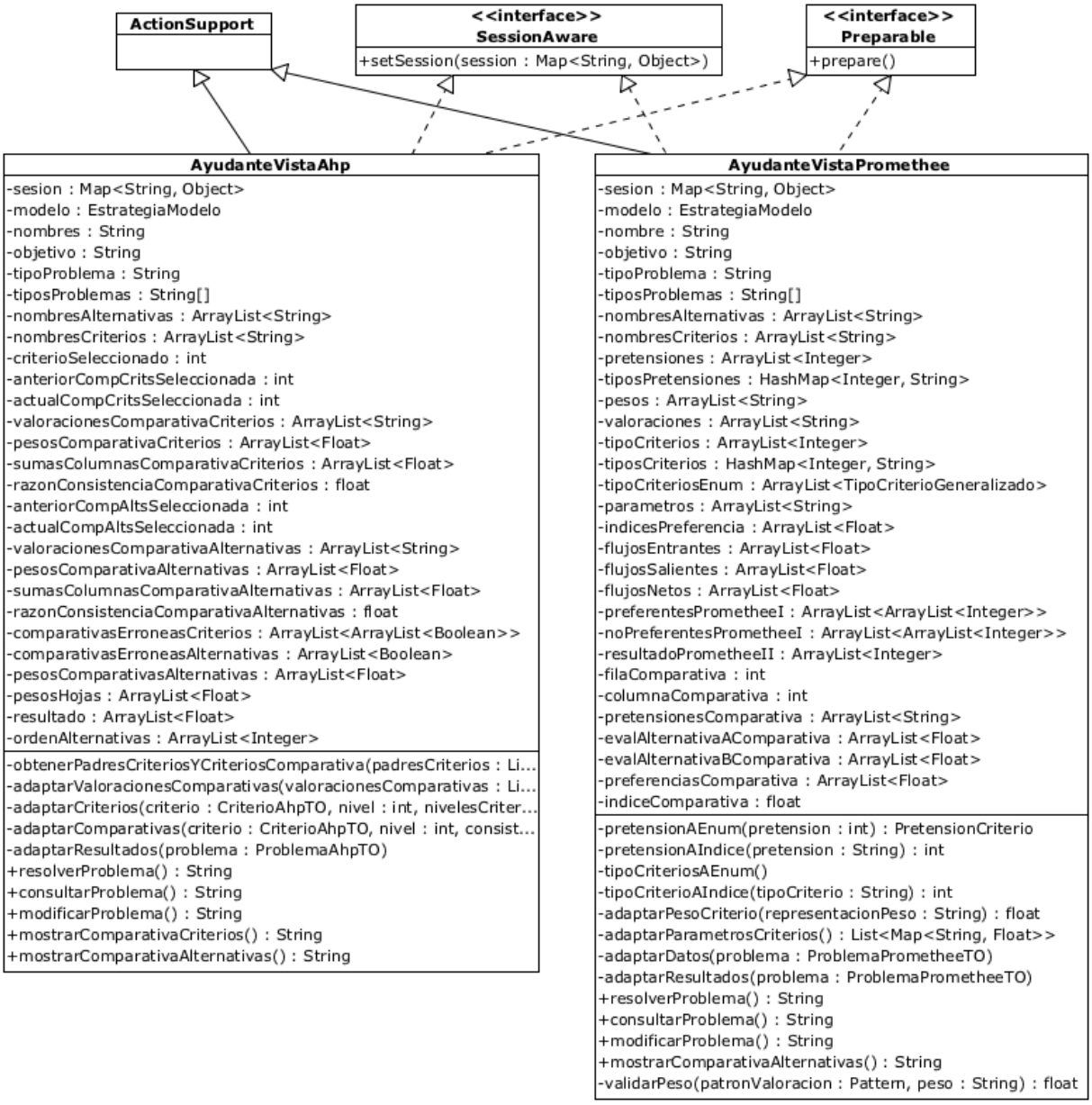


Figura 6.5. Paquete acciones. Parte III.

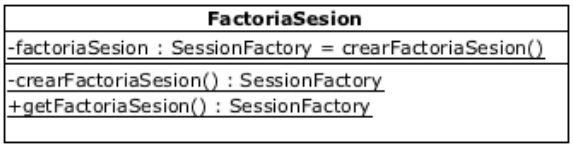


Figura 6.6. Paquete hibernate.

6.1.2.4. Paquete tdmc.interceptores

Contiene el resto de las clases que forman el controlador de la aplicación, que son los interceptores. Struts 2 proporciona el recurso de los interceptores para permitir la ejecución de código antes de que la petición llegue a la acción y justo antes de que se muestre la nueva vista.

Algunos de los interceptores que contiene el paquete están relacionados con temas de seguridad, como son comprobar que el usuario se ha autentificado antes de acceder a una determinada acción o que su rol es el adecuado. Otros se encargan de la instanciación de la fachada del modelo o de la sesión de Hibernate antes de que se ejecute una acción.

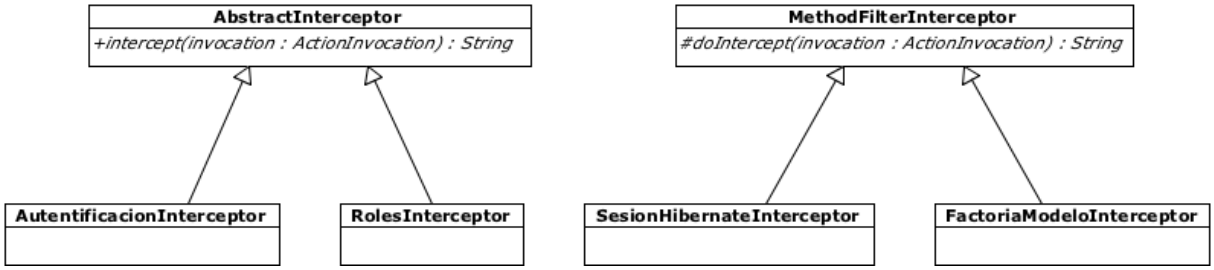


Figura 6.7. Paquete interceptores.

6.1.2.5. Paquete tdmc.modelo

Este paquete contiene las clases que forman el modelo, según el patrón Modelo-Vista-Controlador [29, 33, 34], que se encargan de desarrollar la lógica de negocio de la aplicación. Debido al gran número de clases y su tamaño, el diagrama se ha dividido en dos partes.

La primera parte, que viene representada por la Figura 6.9 alberga las clases y relaciones que definen de forma general la estructura que presentan los expertos y problemas, así como las principales métodos que permiten la creación y resolución de éstos últimos.

La segunda parte, que está representada por la Figura 6.10 contiene una pequeña sección relacionada con la clase *CriterioPromethee* que debido a la falta de espacio no se ha podido incluir en la primera parte. El diagrama también recoge, entre otras clases, los tipos enumerados y la fachada del modelo. Ésta última se ha diseñado de acuerdo a los patrones Estrategia y Fachada [29, 33], y permite al controlador acceder a la lógica de negocio de la aplicación.

6.1.2.6. Paquete tdmc.objetosaccesodatos

Este paquete contiene las clases encargadas de realizar los accesos a la base de datos. Se han diseñado de acuerdo al patrón Objetos de Acceso a Datos (DAOs) [34] que permite desacoplar el acceso a los datos de la tecnología utilizada, haciendo la aplicación más flexible ante

posibles cambios en los métodos y en el tipo de persistencia que se usa. Siguiendo el patrón Estrategia [29, 33] se han definido unas interfaces que definen los métodos de acceso a los datos. Después, en función de la persistencia utilizada basta con implementar las interfaces. También existe una factoría que se encarga de instanciar el DAO que sea necesario en cada situación. El paquete se muestra en la Figura 6.11.

6.1.2.7. Paquete tdmc.objetostransferencia

El paquete alberga las clases que encapsulan los datos que se transmiten entre los distintos niveles de la aplicación y viene representado por la Figura 6.12. Se han diseñado siguiendo el patrón Objetos de Transferencia (TOs) [34] que realiza una encapsulación de los datos de los objetos de negocio. En este caso, se tratan de representaciones simplificadas de los objetos del modelo que constan únicamente de algunos de sus atributos y los métodos de acceso a éstos.

6.1.2.8. Paquete tdmc.seguridad

Este paquete contiene una clase que se encarga de temas relacionados con la seguridad y la criptografía. Resulta evidente que las contraseñas almacenadas en la base de datos se deben encriptar para garantizar su seguridad. Del mismo modo, también es necesario proporcionar algún método para comprobar que una contraseña coincide con su versión encriptada. De estas dos funciones se encarga la clase que contiene este paquete.

Se trata de una clase que se instancia una única vez y que debe resultar accesible desde cualquier parte del código, por ello se ha diseñado con sus atributos y métodos estáticos siguiendo el patrón Singleton [29, 33].

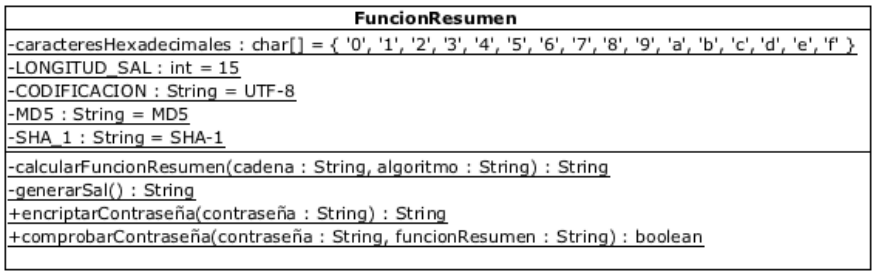


Figura 6.8. Paquete seguridad.

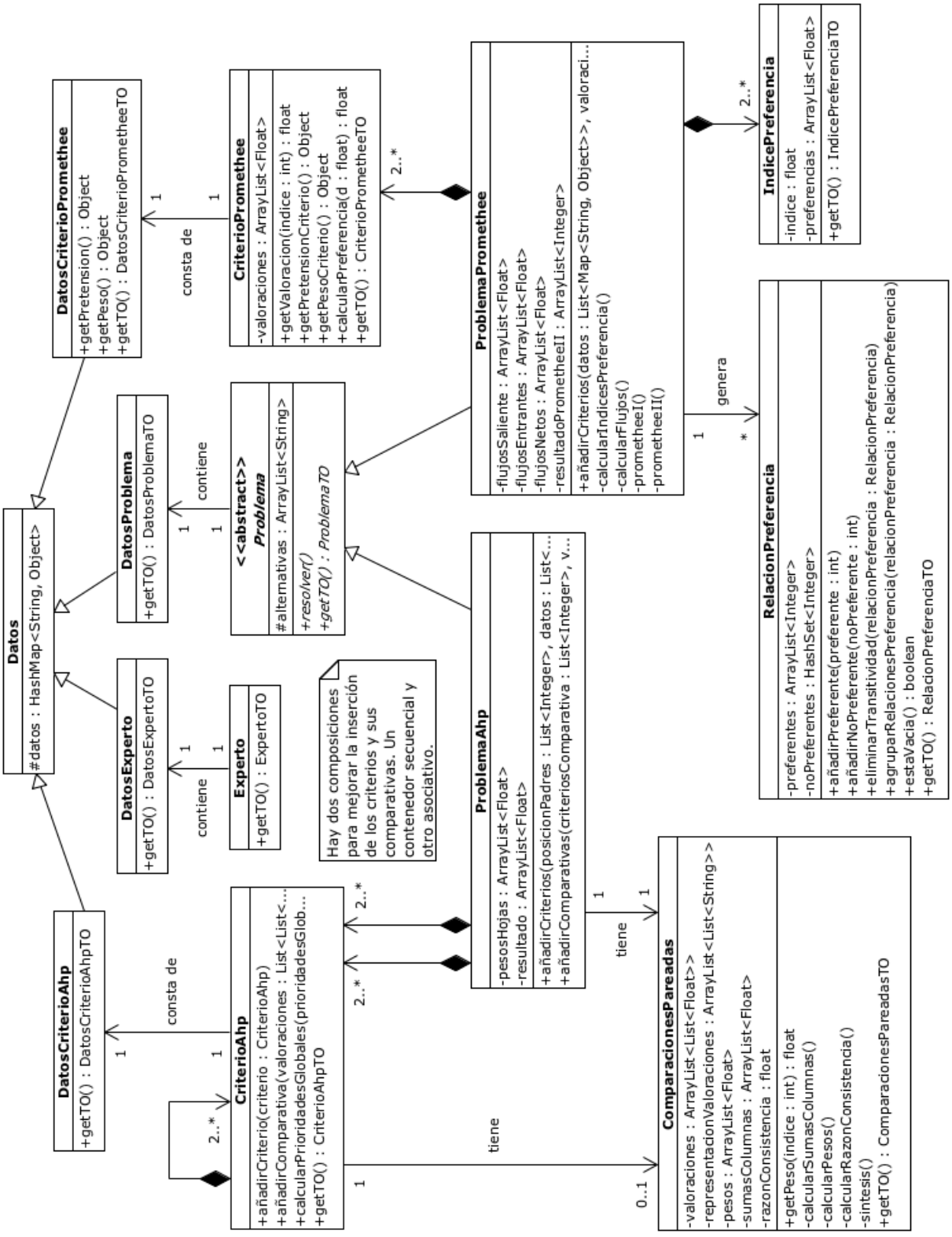


Figura 6.9. Paquete modelo. Parte I.

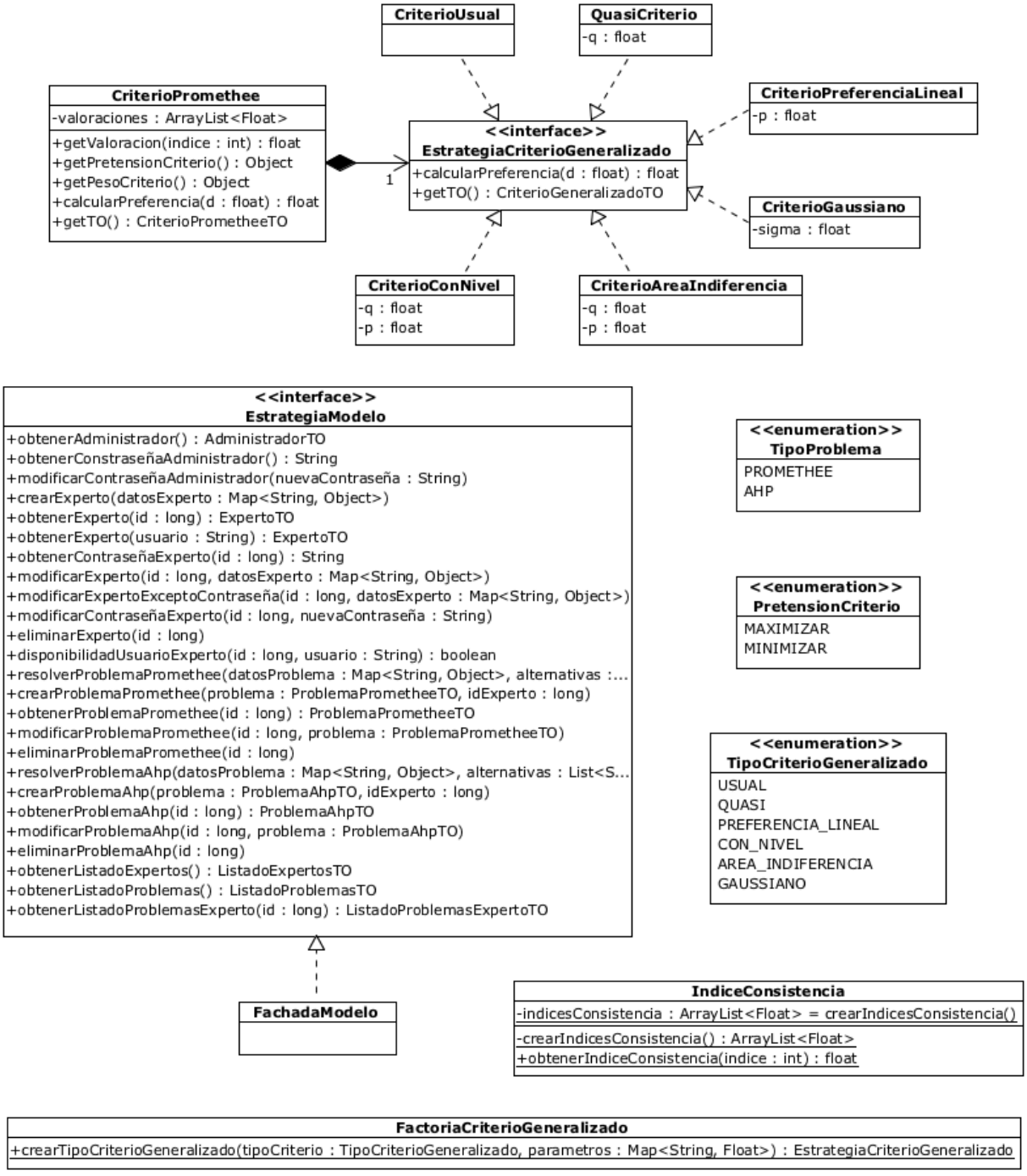


Figura 6.10. Paquete modelo. Parte II.

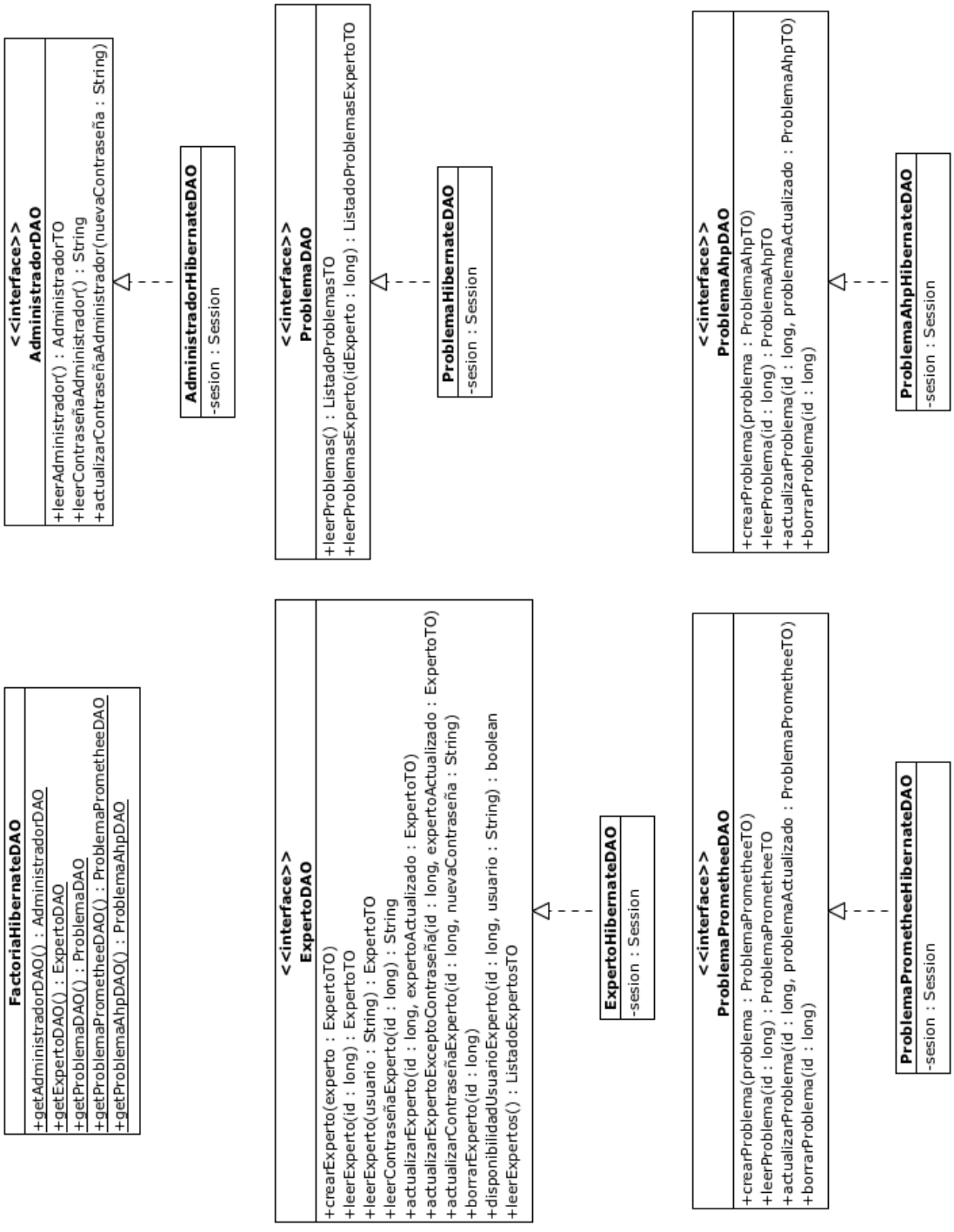


Figura 6.11. Paquete objetos acceso datos.

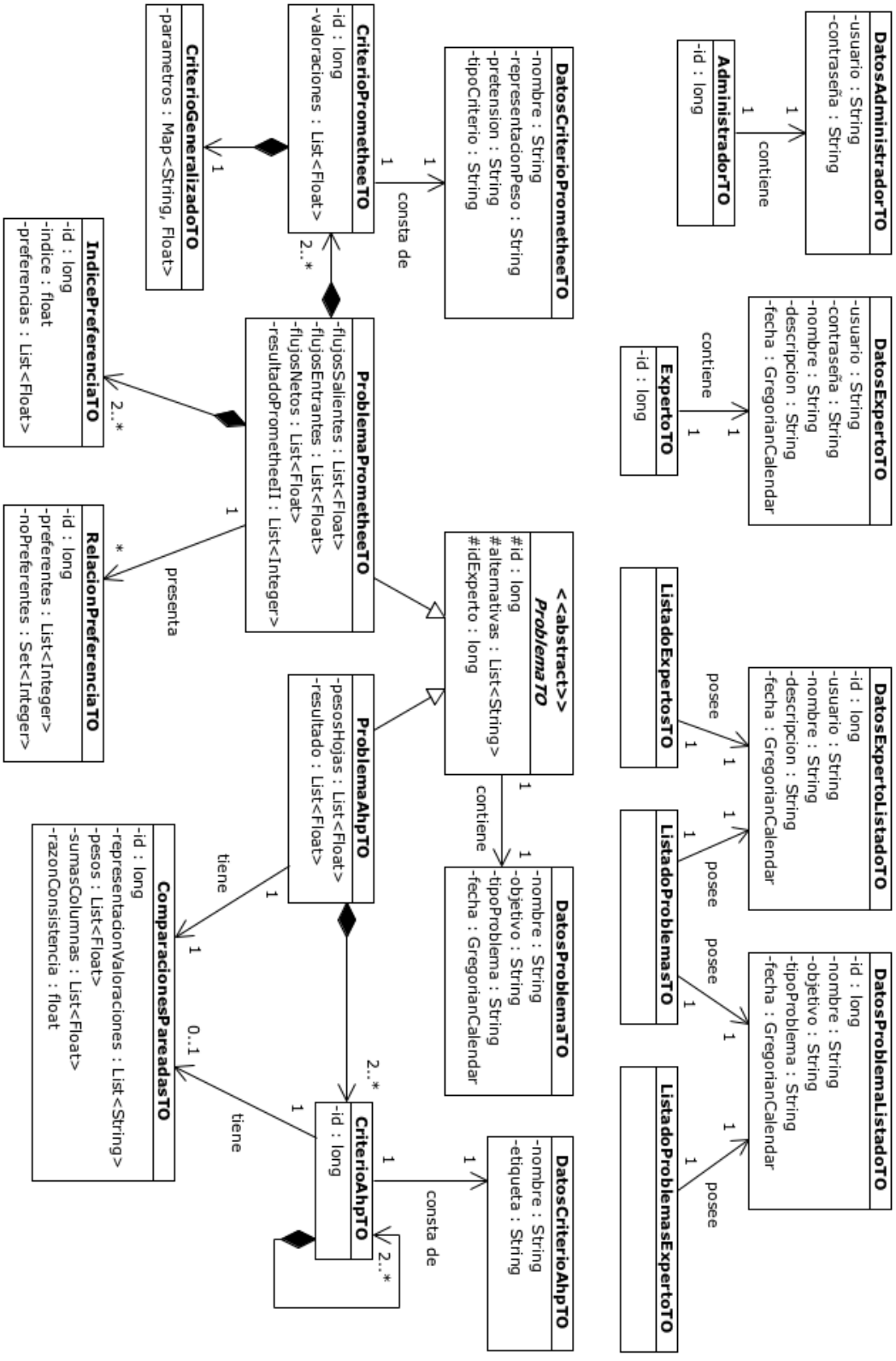


Figura 6.12. Paquete objetos transferencia.

6.2. Diseño de los datos

El diseño de los datos se encarga de crear un modelo de datos e información que se representa con un alto nivel de abstracción [28]. La intención de esta fase del diseño *software* es determinar la estructura que poseen cada uno de los elementos de información del sistema, es decir, la estructura de los datos sobre los que se va a trabajar.

Una vez determinados cuales son los elementos de información del sistema, se debe definir el modo en el que se van a hacer persistentes, o dicho de otro modo, cómo se pretende almacenarlos. Gracias a las ventajas que presentan las bases de datos para aplicaciones de cierta envergadura, como es este caso, se ha decidido hacer uso de este tipo de persistencia para el proyecto.

La estructura de los datos del sistema es la siguiente:

- El administrador, que cuenta con su identificador, usuario y contraseña.

- Los expertos, de los que se conoce su identificador, nombre, descripción, usuario, contraseña y fecha de creación.

- Los problemas PROMETHEE, que tienen su identificador, nombre, objetivo, tipo de problema, fecha de creación, criterios, índices de preferencia, flujos entrantes, flujos salientes, flujos netos, resultado PROMETHEE I y resultado PROMETHE II.

- Los criterios de PROMETHEE, de los que se conoce su identificador, nombre, peso, pretensión, tipo de criterio, valoraciones y parámetros.

- Los índices de preferencia de PROMETHEE, que cuentan con su identificador, índice de preferencia y preferencias de las alternativas que se comparan respecto de cada criterio.

- Los resultados PROMETHEE I, de los que se conoce su identificador, elementos preferentes y no preferentes.

- Los problemas AHP, que contienen su identificador, nombre, objetivo, tipo de problema, fecha de creación, criterios, comparación pareada, pesos hojas y resultado.

- Los criterios de AHP, que contienen su identificador, nombre, etiqueta, subcriterios y comparación pareada.

- Las comparaciones pareadas de AHP, que cuentan con su identificador, valoraciones, sumas de columnas, pesos y razón de consistencia.

A continuación, se presenta un esquema general con las fases del proceso seguido para el análisis y diseño de la base de datos. Para la realización de los esquemas conceptuales se ha utilizado el modelo Entidad-Relación.

1. Análisis de entidades: consiste en localizar y definir las entidades y sus atributos.
2. Análisis de relaciones: se definen las relaciones existentes entre entidades.
3. Modelar los elementos del sistema por medio de un esquema conceptual.
4. Transformar el esquema conceptual en un esquema conceptual modificado.
5. Transcribir el esquema conceptual modificado a tablas.
6. Normalizar las tablas hasta el nivel deseado. Para este proyecto concretamente se normaliza hasta la tercera forma normal.
7. Si aparecen nuevas tablas tras la normalización aplicar retrodiseño, lo que genera un nuevo esquema conceptual modificado que refleja las modificaciones.

6.2.1. Modelo Entidad-Relación

El modelo Entidad-Relación (E-R) consiste en un modelo de datos utilizado para describir esquemas conceptuales por medio de diagramas Entidad-Relación [30]. Se trata de uno de los modelos más extendidos y usados. El modelo E-R se basa en una serie de conceptos sobre los que se construyen los principios de representación de información relacionada.

Una entidad es un elemento o concepto del mundo real con características capaces de hacerlo distinguir de otros objetos. Toma como significado conceptos u objetos que juegan un papel importante en el sistema. Las entidades quedan definidas por una serie de atributos.

Se denomina superclave de una entidad a cualquier conjunto de atributos que permite identificar de forma única a una instancia de esa entidad. Si de una superclave no se puede obtener un subconjunto que sea a su vez superclave, se dice que dicha superclave en una clave candidata. De todas las claves candidatas se ha de elegir una, a la que se le denominará clave primaria. El resto de claves candidatas se denominan claves alternativas. Una clave primaria se representa con el nombre de los atributos que la forman subrayados. Si un atributo o conjunto de atributos de una entidad es clave primaria de otra entidad, a dicho conjunto se le denomina clave foránea. Una clave foránea se representa con el nombre de los atributos que la forman en cursiva.

Una vez que se conoce el término clave candidata, se pueden distinguir dos tipos de entidades. Una entidad fuerte o regular es aquella en la que se puede encontrar al menos alguna clave candidata. Se representa mediante un rectángulo con el nombre de la entidad en su interior. Por el contrario, una entidad débil es aquella en que no es posible encontrar nin-

guna clave candidata. Se representa por dos rectángulos concéntricos con el nombre de la entidad en su interior.

Una relación establece una dependencia o asociación entre dos entidades relacionadas conceptualmente. Se representa por medio de un diamante con el nombre de la relación en su interior unido a las entidades que relaciona. Una relación puede tener carácter opcional y se especifica por medio de un círculo en uno de los extremos de la misma. Ligado a una relación se encuentra el concepto de cardinalidad, que expresa cuántos elementos de un tipo pueden relacionarse con los de otro. Según la cardinalidad se pueden distinguir tres clases de relaciones.

- Uno a uno: una instancia de la entidad A se relaciona solamente con una instancia de la entidad B (1:1).
- Uno a muchos: cada instancia de la entidad A se relaciona con varias instancias de la entidad B (1:N). Aparece una flecha en el extremo de la relación que representa las múltiples instancias.
- Muchos a muchos: cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B (N:M). Aparece una flecha en ambos extremos de la relación.

Un atributo se define como un dato o característica propia de una entidad o relación. Se representan mediante un rectángulo con los bordes redondeados y el nombre del atributo en su interior.

Los conceptos básicos del modelo E-R pueden modelar la mayoría de las situaciones, pero algunos aspectos se modelan mejor con el modelo E-R extendido, el cual aporta nuevas características. En este proyecto, de hecho, aparecen dos de estas características: la generalización y la especialización.

La generalización es un caso especial de relación entre uno o varios tipos de entidades (subtipos) y otro tipo de entidad más general (supertipo), cuyos atributos son comunes a todos los subtipos. Permite modelar e incorporar la herencia de atributos de unas entidades a otras. Se representa por medio de un triángulo invertido con las letras ISA en su interior. El mecanismo de abstracción contrario se llama especialización.

6.2.2. Normalización

La normalización es un proceso consistente en imponer a las tablas ciertas restricciones mediante una serie de transformaciones consecutivas [30]. Su objetivo es asegurar que las tablas contengan los atributos necesarios y suficientes para describir la realidad de la entidad que representan, separando aquellos que pueden contener información cuya relevancia permite la creación de otra nueva tabla [30]. La normalización aporta una mayor flexibilidad al

diseño ante futuros cambios, así como la posibilidad de aplicación del cálculo y el álgebra relacional sin resultados no esperados. El proceso de basa en los conceptos de dependencia funcional, dependencia funcional completa y dependencia transitiva.

- Dependencia funcional: datos los atributos A y B, se dice que B depende funcionalmente de A si y sólo si para cada valor de A sólo puede existir un valor de B.

- Dependencia funcional completa: dados los atributos A_1, A_2 ... A_k y B, se dice que B depende funcionalmente de forma completa de A_1, A_2 ... A_k si y sólo si B depende funcionalmente del conjunto de atributos A_1, A_2 ... A_k, pero no de ninguno de sus posibles subconjuntos.

- Dependencia transitiva: dados tres atributos A, B y C, se dice que existen una dependencia funcional entre A y C si B depende funcionalmente de A y C depende funcionalmente de B.

Para asegurar la normalización se establecen tres formas normales:

- Primera forma normal (FN1): una tabla está en FN1 si todos los atributos no clave depende funcionalmente de la clave, o lo que es lo mismo, no existen grupos repetitivos para un valor de clave.

- Segunda forma normal (FN2): una tabla está en FN2 si está en FN1 y además todos los atributos que no pertenecen a la clave dependen funcionalmente de forma completa de ella. De esta definición se desprende que una tabla en FN1 y cuya clave está compuesta por un único atributo está en FN2.

- Tercera forma normal (FN3): una tabla está en FN3 si está en FN2 y además no existen atributos no clave que dependan transitivamente de la clave.

6.2.3. Esquema conceptual

Tras un minucioso estudio de los elementos del sistema para extraer las entidades, sus atributos y las relaciones, se procede a construir el esquema conceptual reflejado por las Figuras 6.13, 6.14, 6.15 y 6.16.

Debido a la complejidad y amplio tamaño del esquema conceptual, se ha dividido en cuatro partes. Para facilitar la legibilidad del diagrama y asegurar así la unión entre las distintas partes, algunas de ellas contarán con entidades que ya han aparecido con anterioridad, de modo que sirvan de referencia entre una parte y su continuación, pero en estas ocasiones únicamente aparecerá el nombre de la entidad de referencia.

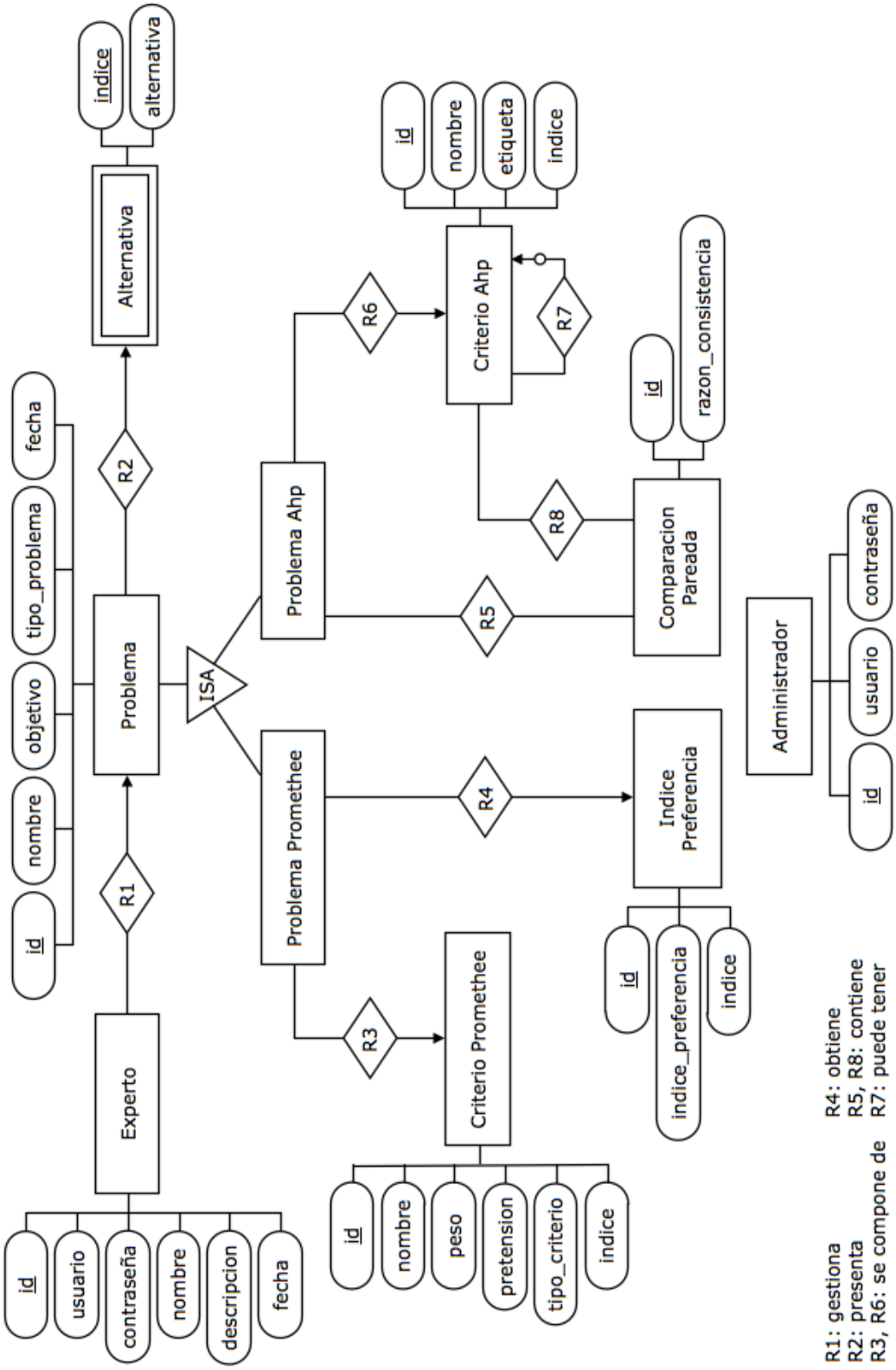


Figura 6.13. Esquema conceptual. Parte I.

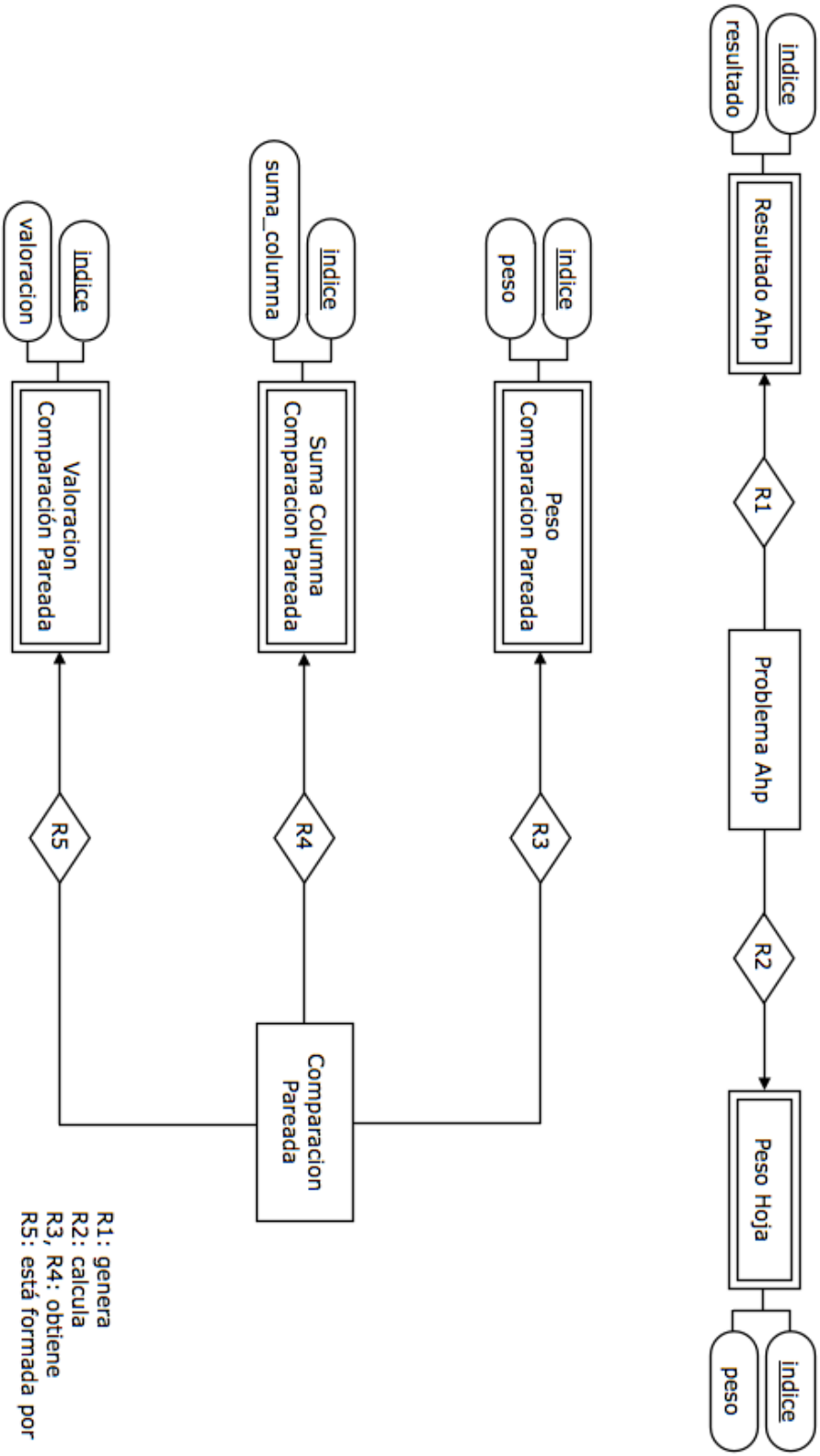


Figura 6.14. Esquema conceptual. Parte II.

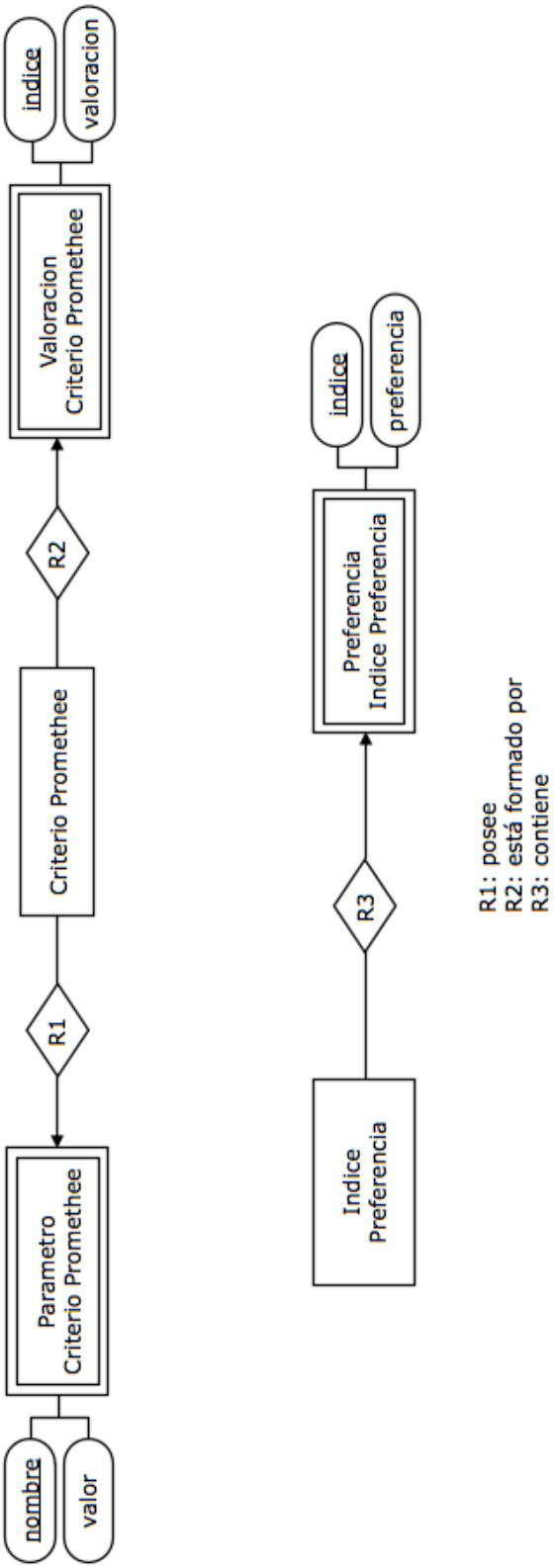


Figura 6.15. Esquema conceptual. Parte III.

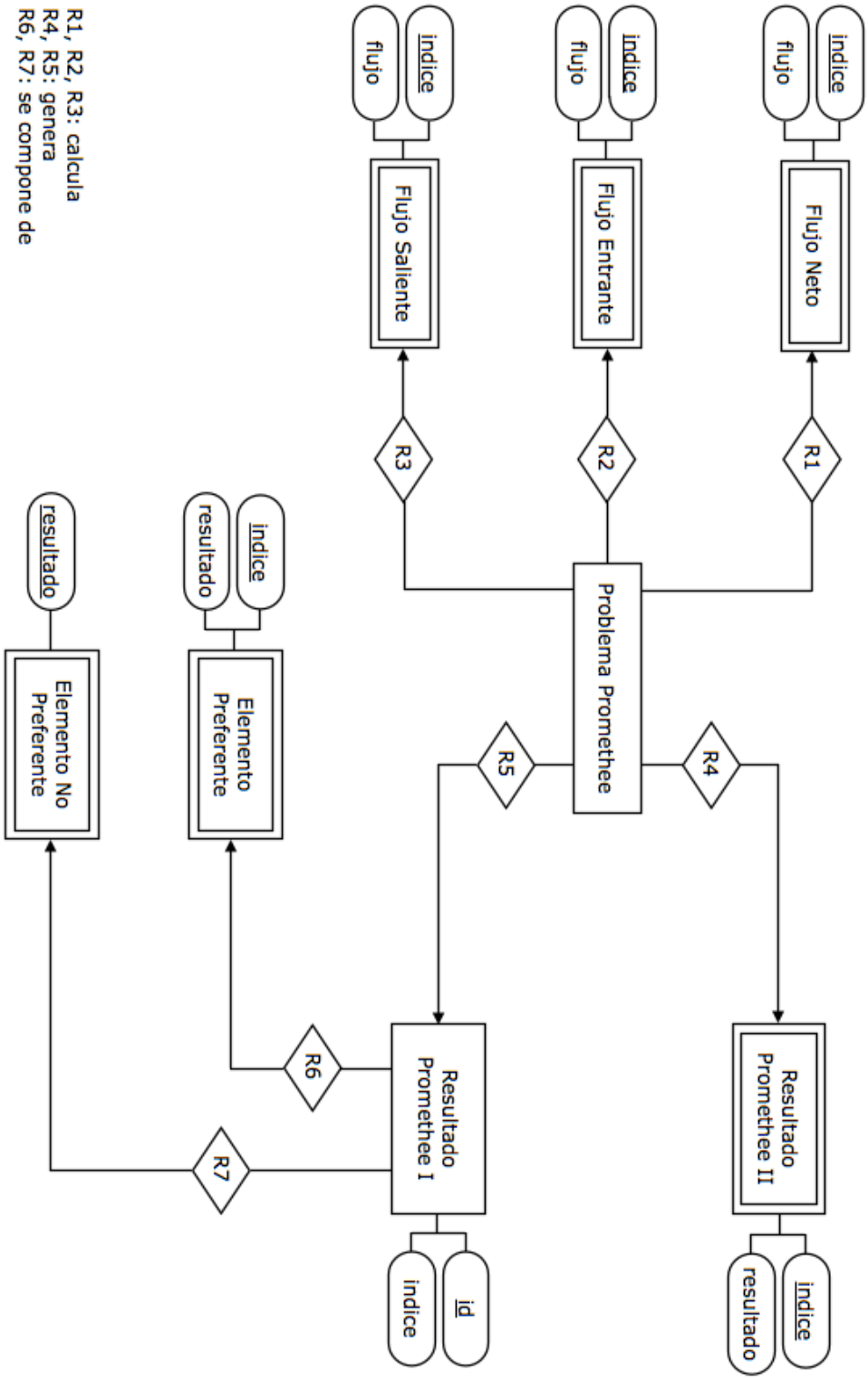


Figura 6.16. Esquema conceptual. Parte IV.

6.2.4. Esquema conceptual modificado

Una vez obtenido el esquema conceptual, es el momento de pasar al esquema conceptual modificado aplicando las siguientes transformaciones en orden riguroso.

a) Eliminar las características del modelo E-R extendido.

b) Transformación de relaciones n-arias en binarias.

c) Eliminación de relaciones cíclicas.

d) Reducción a relaciones de uno a uno y uno a muchos.

e) Conversión de entidades débiles en fuertes.

El resultado es un esquema conceptual modificado como el formado por las Figuras 6.17, 6.18, 6.19 y 6.20 que contendrá exclusivamente entidades fuertes con sus atributos y relaciones de uno a uno y uno a muchos.

Más concretamente, en este proyecto hay que eliminar la relación de generalización, la relación cíclica y las entidades débiles. En el primer caso, la relación de generalización se elimina quitando la entidad supertipo *Problema* y añadiendo sus atributos a cada uno de los subtipos, *Problema Promethee* y *Problema Ahp*, en los que se especializa. En el segundo caso, la relación cíclica de la entidad *Criterio Ahp* se transforma en dos relaciones, una de cardinalidad 1:1 y otra de 1:N que además es opcional, con una nueva entidad llamada *Criterio Criterio Ahp*, donde se almacenan las relaciones entre criterios utilizando como clave primaria los identificadores de los mismos. En el tercer caso, las entidades débiles se transforman en fuertes simplemente añadiendo a su clave primaria la clave primaria de la entidad fuerte con la que se relacionan.

El esquema conceptual modificado es similar en tamaño al esquema conceptual, por lo que se ha dividido de la misma forma y siguiendo los mismos criterios.

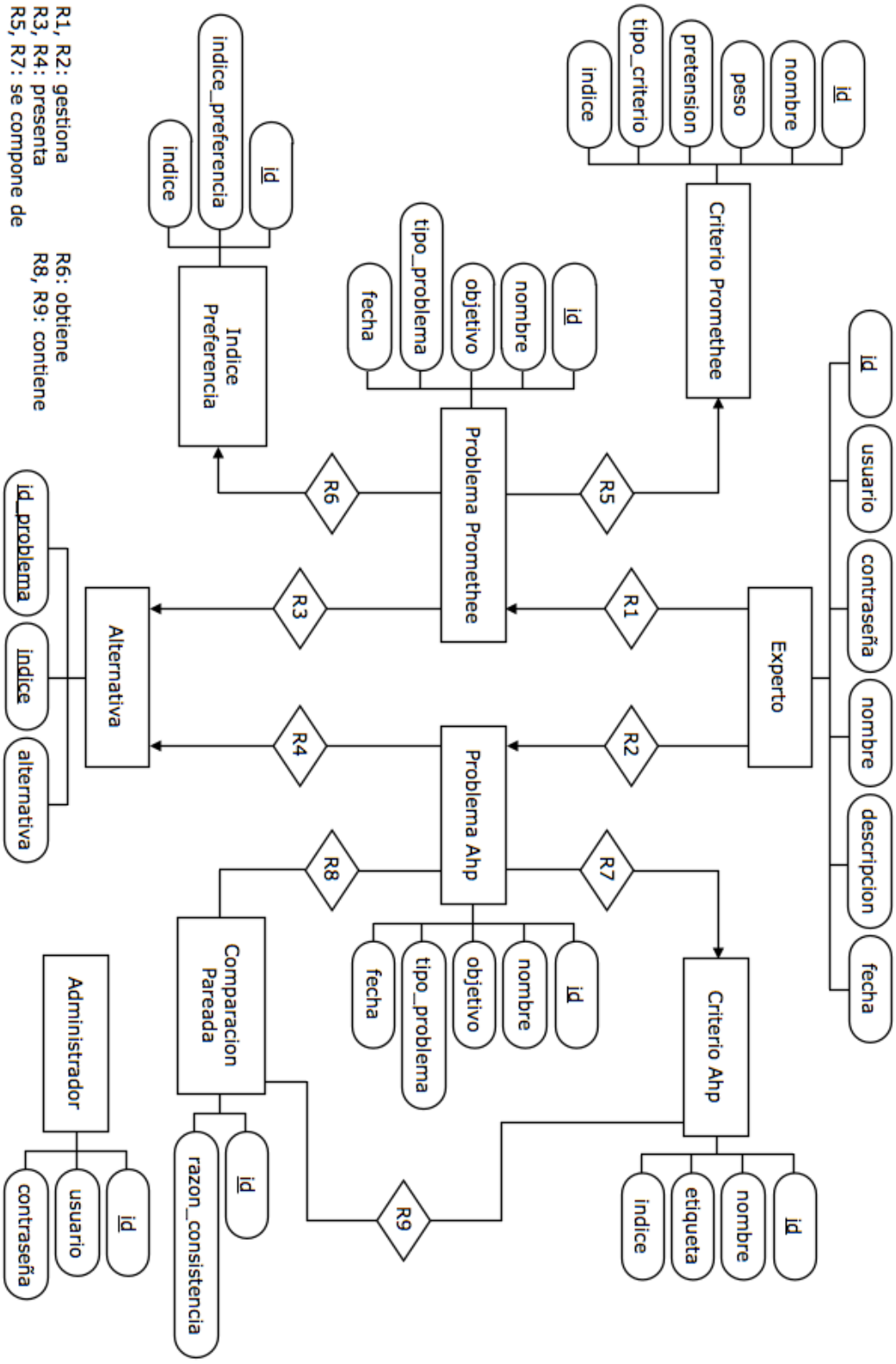


Figura 6.17. Esquema conceptual modificado. Parte I.

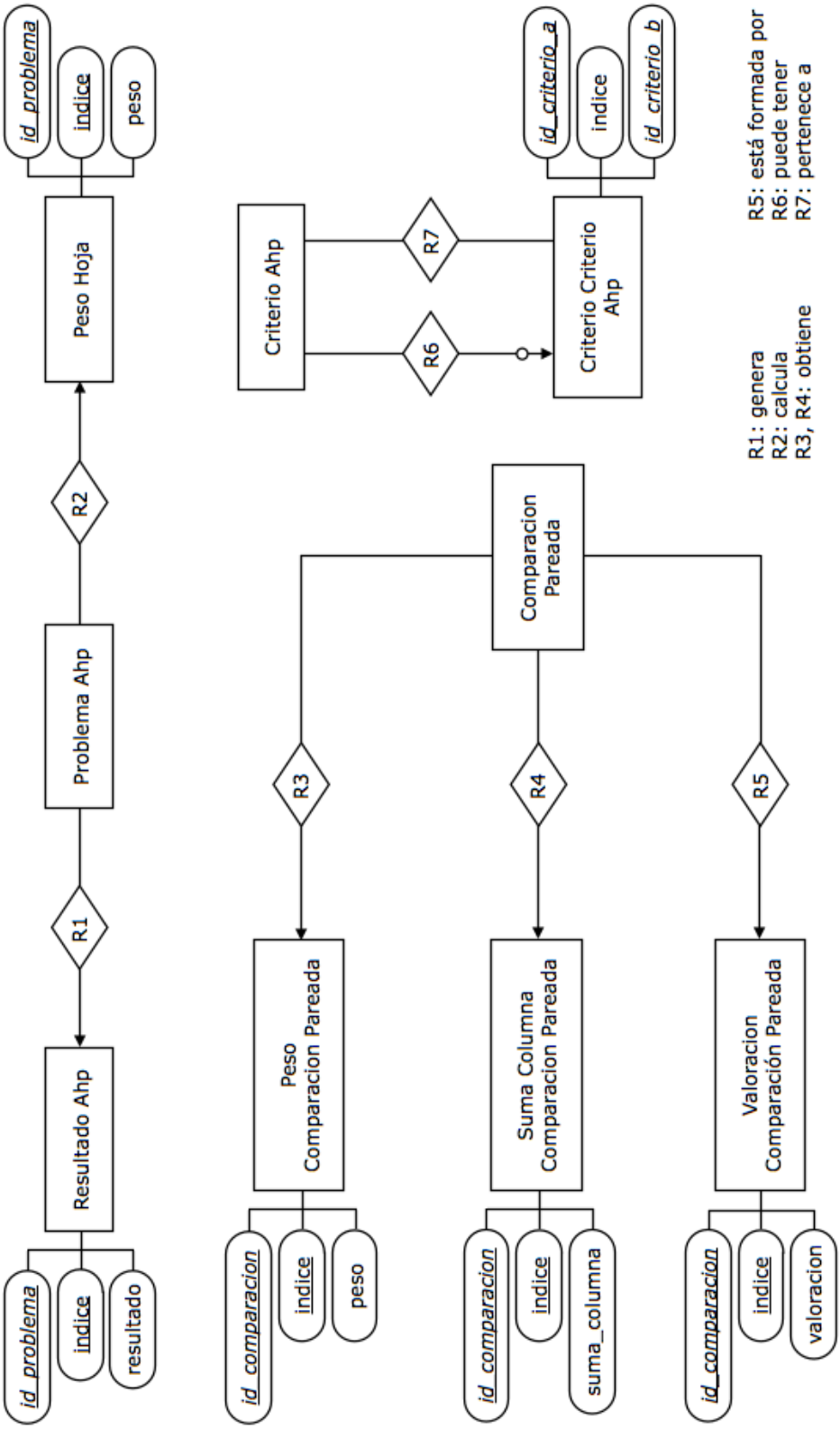


Figura 6.18. Esquema conceptual modificado. Parte II.

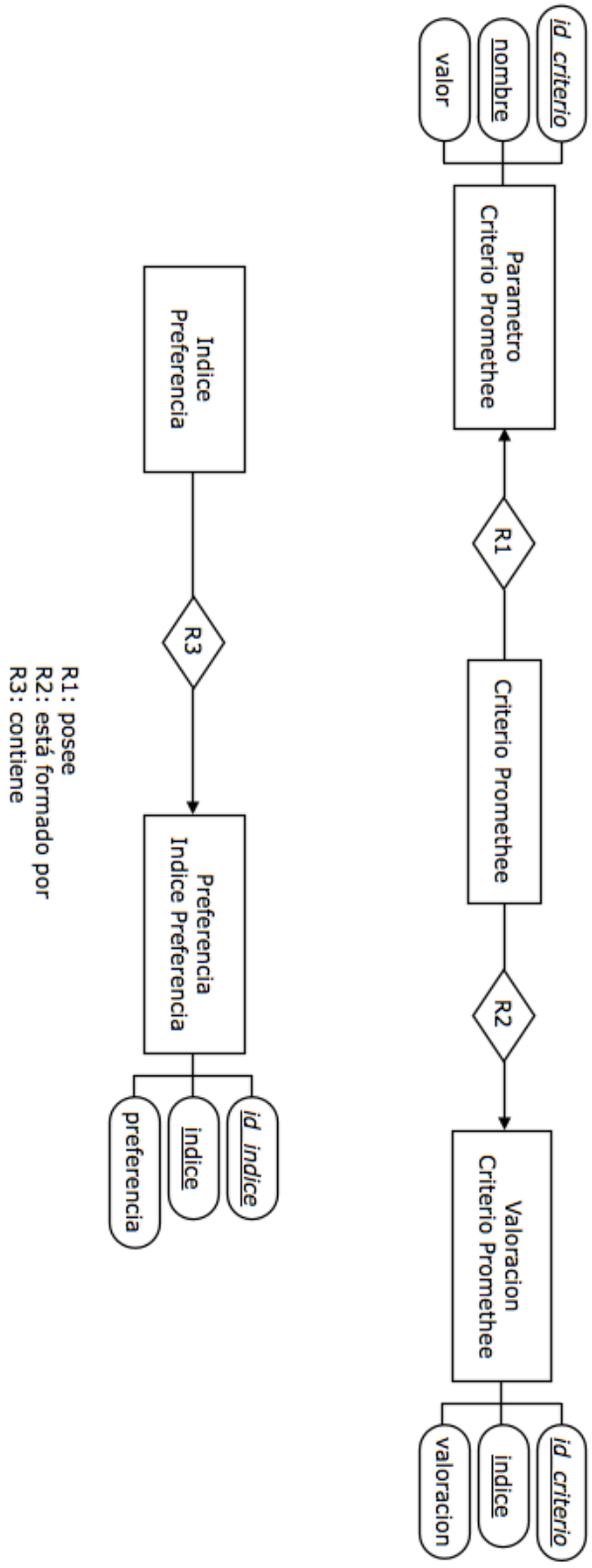


Figura 6.19. Esquema conceptual modificado. Parte III.

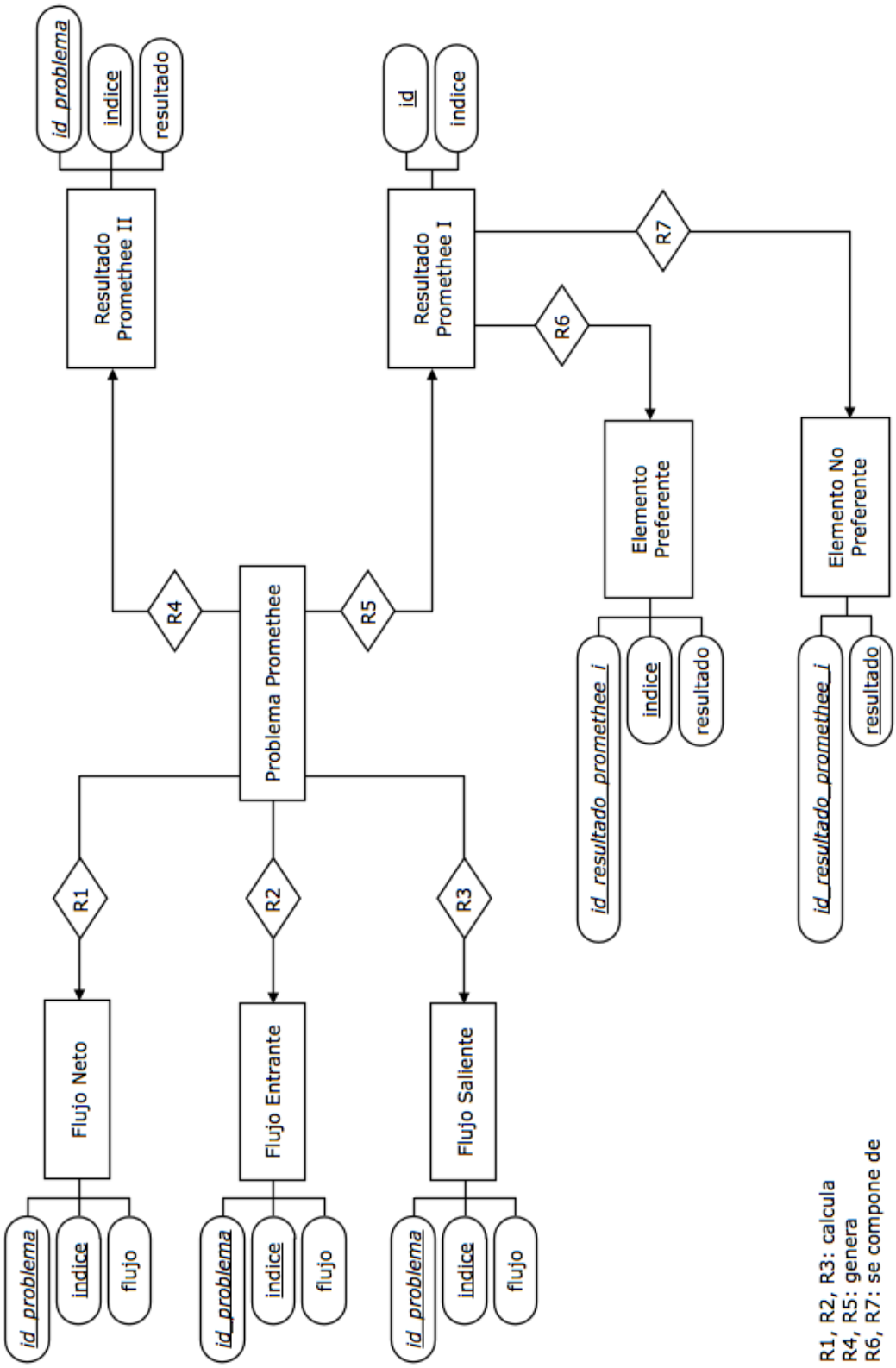


Figura 6.20. Esquema conceptual modificado. Parte IV.

6.2.5. Tablas de la aplicación

Una vez obtenido el esquema conceptual modificado la obtención de las tablas es casi automática. Para ello hay que tener en cuenta dos consideraciones:

- Cada entidad del esquema conceptual modificado genera una tabla cuyos atributos son los atributos de la entidad, y con clave su clave candidata seleccionada.

- Cada relación del esquema conceptual modificado produce la inclusión en una de las tablas de una clave foránea, siendo esta clave de la entidad propietaria de la relación.

A continuación se detallan cada una de las tablas de la aplicación con sus respectivos atributos y características asociados:

✦ ***Administrador***

Esta tabla contiene la información que requiere el administrador para acceder al sistema, es decir, su nombre de usuario y contraseña.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
usuario	Varchar(30)	No nulo y único	
contraseña	Varchar(55)	No nulo	

✦ ***Experto***

Se almacenan los expertos del sistema con sus datos correspondientes.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
usuario	Varchar(30)	No nulo y único	
contraseña	Varchar(55)	No nulo	
nombre	Varchar(100)	No nulo	
descripcion	Varchar(200)		
fecha	Date	No nulo	

- ***Alternativa***

Contiene los nombres de las alternativas pertenecientes a cada problema del sistema.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria
indice	Smallint	No nulo	Primaria
alternativa	Varchar(100)	No nulo	

- ***Problema_promethee***

La tabla almacena las datos de los problemas de tipo PROMETHEE que existen en el sistema.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
nombre	Varchar(100)	No nulo	
objetivo	Varchar(100)	No nulo	
tipo_problema	Varchar(20)	No nulo	
fecha	Date	No nulo	
id_experto	Bigint	No nulo	Foránea

- ***Criterio_promethee***

Contiene los datos de los criterios de los problemas PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
nombre	Varchar(100)	No nulo	
peso	Varchar(10)	No nulo	
pretension	Varchar(50)	No nulo	

Atributo	Tipo de dato	Restricciones	Clave
tipo_criterio	Varchar(50)	No nulo	
id_problema	Bigint	No nulo	Foránea
indice	Smallint	No nulo	

- ***Valoracion_criterio_promethee***

  Almacena las valoraciones asignadas a los criterios PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id_criterio	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
valoracion	Float	No nulo	

- ***Parametro_criterio_promethee***

La tabla contiene el nombre y el valor de los parámetros que forman parte de los criterios PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id_criterio	Bigint	No nulo	Primaria y foránea
nombre	Varchar(20)	No nulo	Primaria
valor	Float	No nulo	

- ***Indice_preferencia***

  La tabla almacena los índices de preferencia de los problemas PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
indice_preferencia	Float	No nulo	
id_problema	Bigint	No nulo	Foránea

Atributo	Tipo de dato	Restricciones	Clave
indice	Smallint	No nulo	

- ***Preferencia_indice_preferencia***

Recoge los valores de las preferencias entre alternativas respecto de cada criterio, que permiten calcular el valor del índice de preferencia.

Atributo	Tipo de dato	Restricciones	Clave
id_indice	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
preferencia	Float	No nulo	

- ***Flujo_saliente***

Almacena los flujos salientes que pertenecen a los problemas PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
flujo	Float	No nulo	

- ***Flujo_entrante***

Almacena los flujos entrantes que pertenecen a los problemas PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
flujo	Float	No nulo	

- ***Flujo_neto***

Almacena los flujos netos que pertenecen a los problemas PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
flujo	Float	No nulo	

- ***Resultado_promethee_i***

La tabla recoge los resultados obtenidos de aplicar el método PROMETHEE I a los problemas de tipo PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
id_problema	Bigint	No nulo	Foránea
indice	Smallint	No nulo	

- ***Elemento_preferente***

Almacena los elementos preferentes para cada resultado del método PROMETHEE I.

Atributo	Tipo de dato	Restricciones	Clave
id_resultado_promethee_i	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
resultado	Integer	No nulo	

- ***Elemento_no_preferente***

Almacena los elementos no preferentes para cada resultado del método PROMETHEE I.

Atributo	Tipo de dato	Restricciones	Clave
id_resultado_promethee_i	Bigint	No nulo	Primaria y foránea
resultado	Integer	No nulo	Primaria

- ***Resultado_promethee_ii***

La tabla recoge los resultados obtenidos de aplicar el método PROMETHEE II a los problemas de tipo PROMETHEE.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
resultado	Integer	No nulo	

- ***Problema_ahp***

La tabla almacena los datos de los problemas de tipo AHP que existen en el sistema.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
nombre	Varchar(100)	No nulo	
objetivo	Varchar(100)	No nulo	
tipo_problema	Varchar(20)	No nulo	
fecha	Date	No nulo	
id_experto	Bigint	No nulo	Foránea
id_comparacion	Bigint	No nulo y único	Foránea

- ***Criterio_ahp***

Contiene los datos de los criterios de los problemas AHP.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
nombre	Varchar(100)	No nulo	
etiqueta	Varchar(50)	No nulo	
id_problema	Bigint		Foránea
indice	Smallint		
id_comparacion	Bigint	No nulo y único	Foránea

- ***Criterio_criterio_ahp***

La tabla recoge las relaciones entre pares de criterios AHP, es decir, qué criterios contienen qué subcriterios.

Atributo	Tipo de dato	Restricciones	Clave
id_criterio_a	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	
id_criterio_b	Bigint	No nulo	Primaria y foránea

- ***Comparacion_pareada***

Almacena los datos de las comparaciones pareadas, es decir, las comparativas entre criterios y alternativas que conforman los juicios emitidos por los expertos para los problemas AHP.

Atributo	Tipo de dato	Restricciones	Clave
id	Bigint	No nulo	Primaria
razon_consistencia	Float	No nulo	

- ***Valoracion_comparacion_pareada***

Recoge las valoraciones que emiten los expertos para las comparaciones pareadas.

Atributo	Tipo de dato	Restricciones	Clave
id_comparacion	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
valoracion	Varchar(3)	No nulo	

- ***Suma_columna_comparacion_pareada***

  Almacena las sumas de las columnas de las comparaciones pareadas.

Atributo	Tipo de dato	Restricciones	Clave
id_comparacion	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
suma_columna	Float	No nulo	

- ***Peso_comparacion_pareada***

  Almacena el cálculo de los pesos de las comparaciones pareadas.

Atributo	Tipo de dato	Restricciones	Clave
id_comparacion	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
peso	Float	No nulo	

- ***Peso_hoja***

La tabla almacena los pesos de los criterios hoja que forman parte de la estructura de árbol de criterios que contienen los problemas AHP.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
peso	Float	No nulo	

- ***Resultado_ahp***

La tabla recoge los resultados obtenidos de aplicar el método AHP a los problemas de tipo AHP.

Atributo	Tipo de dato	Restricciones	Clave
id_problema	Bigint	No nulo	Primaria y foránea
indice	Smallint	No nulo	Primaria
resultado	Float	No nulo	

6.3. Diseño de la interfaz

El diseño se la interfaz se define como el proceso a través del cual se crea un medio de comunicación entre el usuario y la máquina [28]. En esta fase del diseño del sistema *software* se define la apariencia visual y el comportamiento de la interfaz [31, 32]. Un diseño cuidado de la interfaz de usuario es parte fundamental de la etapa de diseño, pues es vital que sea diseñada para ajustarse a las habilidades, experiencia y expectativas de los usuarios [25].

Cuando se construye una aplicación no sólo hay que centrarse en que desempeñe su cometido de forma eficaz y eficiente, pues no hay que olvidar que el usuario hace uso de ella a través de su interfaz. Es por ello que ésta debe presentarse atractiva al usuario, pero a la vez le debe de resultar fácil de entender y utilizar. No hay que perder de vista que los usuarios utilizan un *software* para llevar a cabo algún cometido y la interfaz se encuentra en medio, por lo que no deben perder tiempo utilizándola sino que debe facilitarles su misión.

Una interfaz bien o mal diseñada puede hacer cambiar drásticamente la opinión de un usuario sobre una aplicación. Una interfaz mal diseñada significa que los usuarios probablemente no podrán acceder a algunas características del sistema, cometerán errores y sentirán que el sistema les dificulta en vez de ayudarlos a conseguir su objetivo. Cuando se toman decisiones en el diseño de las interfaces de usuario deben tenerse en cuenta las capacidades físicas y mentales de las personas que utilizarán el *software*.

Tomando como entrada las conclusiones de usabilidad, los casos de uso y el análisis de tareas definidos en la fase de análisis, se da forma a las pantallas y secuencia de interacción que mejor se adaptan a las tareas del usuario. La etapa de diseño de la interfaz de usuario incluye las tareas de definir los diseños de pantallas, los caminos de navegación y los mensajes de error, éxito y confirmación de la aplicación.

6.3.1. Diseño de pantallas

Uno de los pasos fundamentales dentro del diseño de la interfaz es definir la estructura de las pantallas que van a formar parte de la aplicación. Este proceso consiste en decidir qué pantallas se van a crear, qué elementos van a aparecer en ellas y cómo se van a organizar y distribuir. La tarea del diseño de pantallas debe ejercerse de manera que la interfaz gráfica resulte fácil, cómoda e intuitiva de manejar al usuario, siempre teniendo en cuenta los principios de usabilidad.

Una buena interfaz no debe eludir el vocabulario técnico, sino más bien utilizar el que se ajusta a ese ámbito de trabajo en concreto y establecer buenas metáforas visuales y de proceso. Un factor muy importante a tener en cuenta durante el diseño de pantallas es el *excise*. El *excise* se define como el trabajo extra necesario para usar las herramientas o satisfacer restricciones de agentes externos que debemos realizar mientras intentemos conseguir nuestros objetivos [31, 32]. Por lo tanto, toda buena interfaz tratará de reducir en la medida de los posible este factor. Un último detalle a tener en cuenta a la hora de diseñar las pantallas es su comportamiento dinámico, es decir, la forma en que cambia su posición y tamaño así como su posición relativa a otro componentes.

Para comenzar a definir la estructura de las pantallas, un buen criterio consiste en hacer uso del diagrama HTA donde quedan muy claras las distintas situaciones del flujo de ejecución de las tareas, lo que permite que al diseñar las pantallas se les pueda dar soporte. En principio debe crearse una pantalla por cada tarea, aunque siempre es bueno mirar si algunas tareas pueden unificarse para simplificar la interacción y reducir el *excise*.

Algunas pantallas, pese a ser distintas, presentan casi los mismos componentes distribuidos del mismo modo, salvo algunas diferencias concretas. Es por esta razón que en estos casos se ha decidido suprimir una de ellas e indicar las diferencias entre ambas. A continuación se presentan los diseños de pantallas de la aplicación.

- ***Acceso***

El diseño de pantalla de acceso al sistema es común para el administrador y los expertos. Mediante esta pantalla se permite acceder al sistema por medio de una autentificación previa, con los credenciales correspondientes al usuario que se identifique. Viene dada por la Figura 6.21.

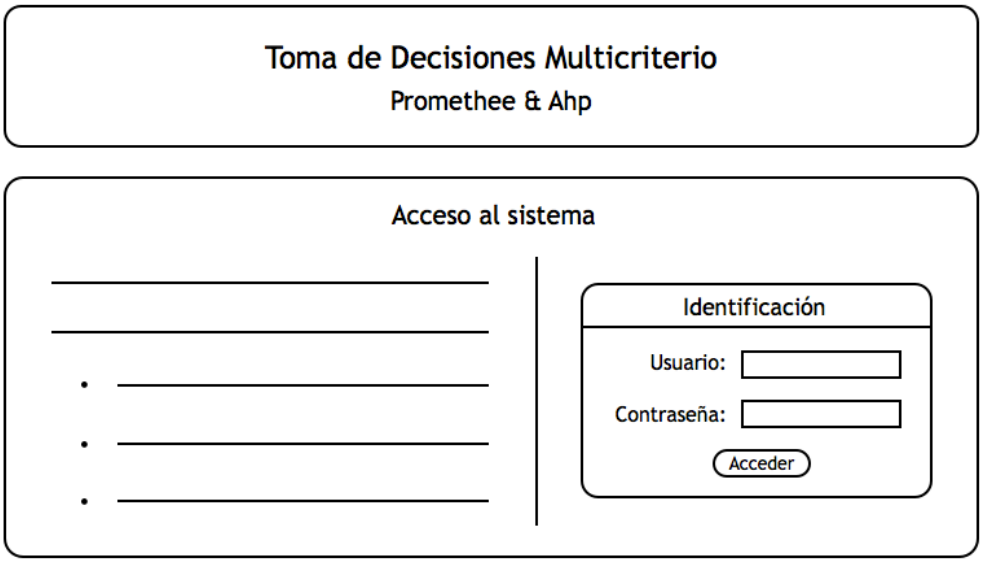


Figura 6.21. Acceso.

- ***Gestionar expertos***

La pantalla de la Figura 6.22 hace referencia a la gestión de expertos y es la primera que aparece tras la autentificación del administrador. En ella se muestra una lista de los expertos del sistema donde aparece su usuario, nombre, descripción y fecha de creación. A partir de ella se pueden crear expertos, modificarlos pulsando sobre su nombre de usuario y eliminar los expertos seleccionados. A estas funcionalidades sólo puede acceder el administrador, por lo que únicamente puede ver esta pantalla dicho usuario.

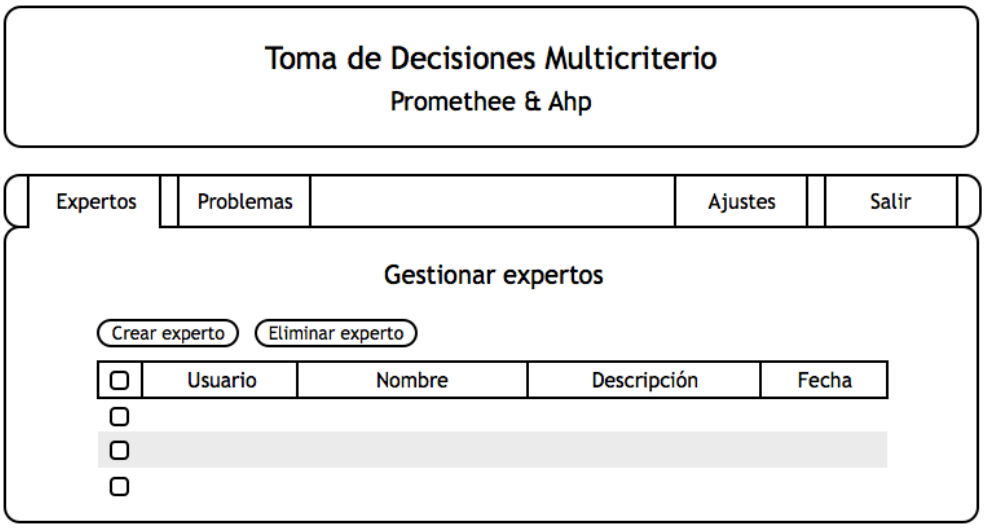


Figura 6.22. Gestionar expertos.

- ***Gestionar problemas (Administrador)***

Este diseño de pantalla hace referencia a la gestión de problemas del administrador. En la pantalla aparece un listado de los problemas del sistema clasificados de acuerdo al experto al que pertenecen. Se muestra su nombre, objetivo, tipo de problema y fecha de creación. De acuerdo a los requerimientos, el administrador solamente puede consultar problemas pulsando sobre el nombre del mismo y eliminar los problemas que seleccione.

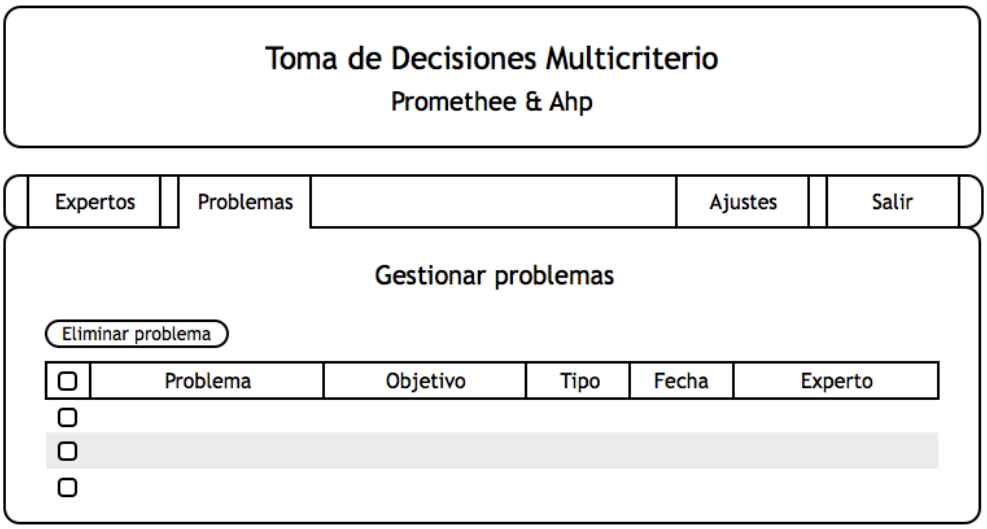


Figura 6.23. Gestionar problemas (Administrador).

- ***Modificar contraseña***

El diseño de pantalla de la Figura 6.24 está asociado al tipo de usuario administrador. Sin embargo, la pantalla es idéntica al modificar contraseña de los expertos, a diferencia de la barra de menú, que carece de la opción *Expertos*. Mediante esta pantalla se permite a un usuario cambiar su propia contraseña, siempre que proporcione la actual.

- ***Crear experto***

Esta pantalla permite acceder a la funcionalidad de crear un experto. Está reservada para el administrador y es necesario que éste proporcione los datos del experto que desea crear. El diseño de pantalla se puede apreciar en la Figura 6.25.

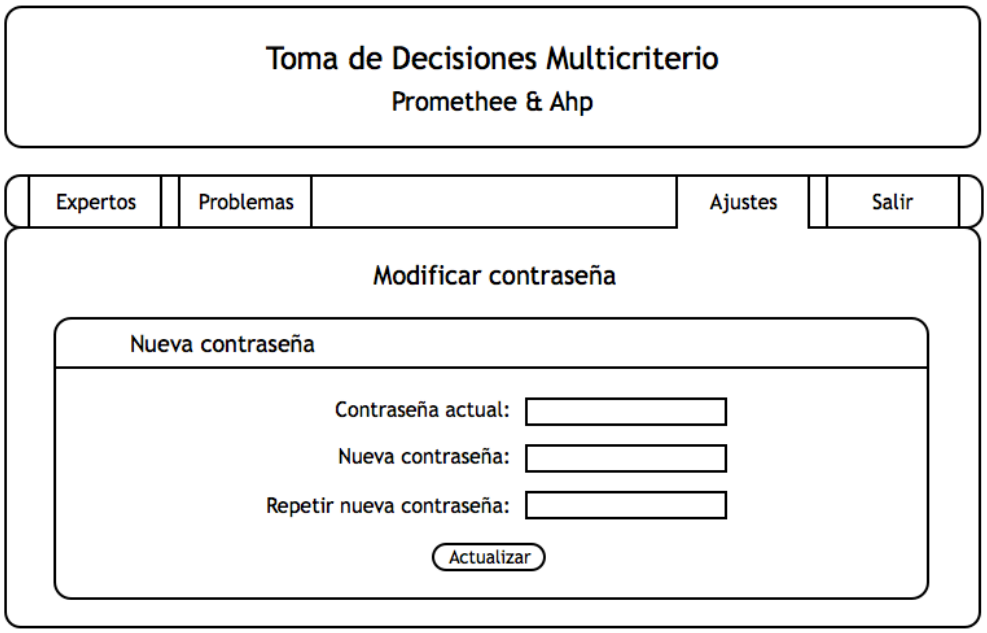


Figura 6.24. Modificar contraseña.

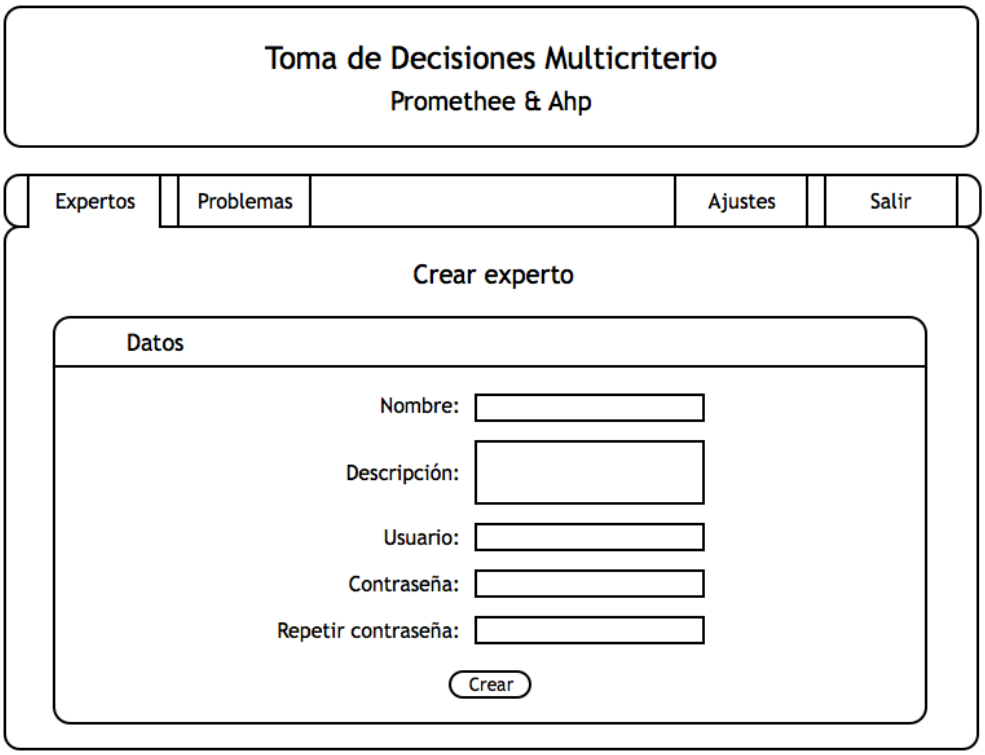


Figura 6.25. Crear experto.

- ***Modificar experto***

Otra funcionalidad del sistema es la modificación de un experto. En esta ocasión, viene representada por el diseño de pantalla de la Figura 6.26. Una vez más sólo puede acceder a ella el administrador. En esta pantalla aparecen los datos actuales del experto listos para ser modificados, a excepción de su contraseña por razones de seguridad. Además, la contraseña se modifica sólo si se proporciona una nueva, en caso contrario, permanece con la misma.

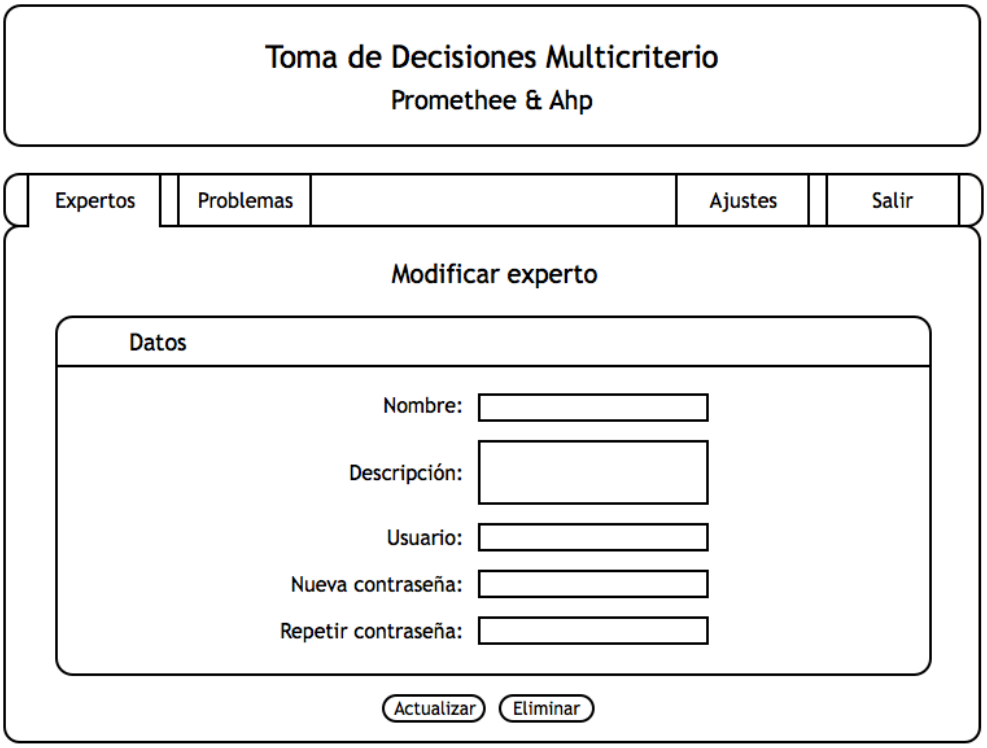


Figura 6.26. Modificar experto.

- ***Gestionar problemas (Experto)***

Este diseño de pantalla hace referencia a la gestión de problemas del experto y es la primera que aparece cuando se autentifica. Se muestra una lista de los problemas pertenecientes al experto, donde aparece su nombre, objetivo, tipo de problema y fecha de creación. Además, a partir de esta pantalla se permite a un experto crear problemas, consultarlos y modificarlos pulsando sobre su nombre, y eliminar los problemas seleccionados. Esta pantalla es única de los expertos.

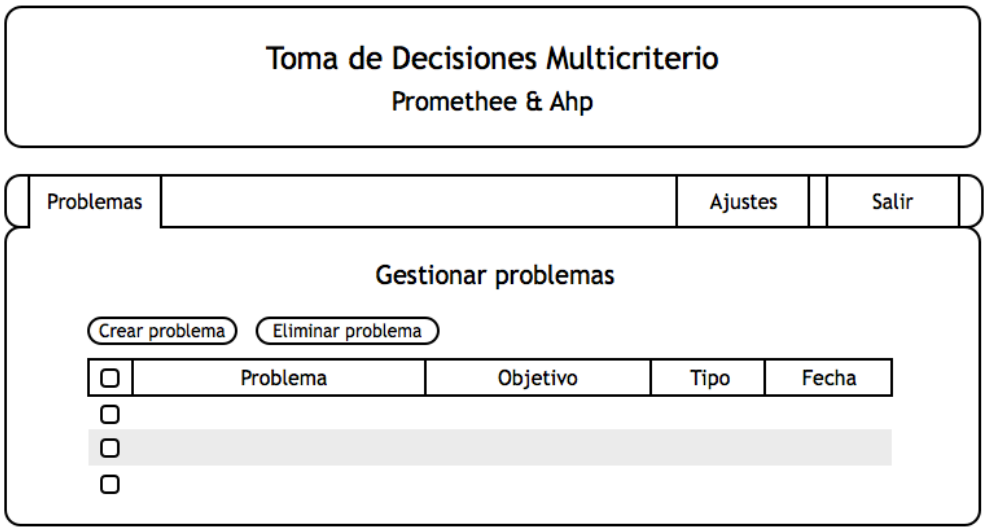


Figura 6.27. Gestionar problemas (Experto).

- ***Consultar PROMETHEE***

El diseño de pantalla de la Figura 6.28 representa la consulta de un problema de tipo PROMETHEE. En ella aparecen los datos del problema y los resultados obtenidos. Si se selecciona un índice de preferencia aparece una comparativa entre alternativas que muestra el cálculo del índice.

Aunque este diseño de pantalla pertenece a los expertos, como el administrador también puede consultar problemas PROMETHEE, el suyo es muy parecido. Se diferencian en que la barra de menú del administrador cuenta además con la opción *Expertos* y que desaparece el botón de *Modificar*, puesto que no puede modificar problemas.

- ***Consultar AHP***

En este diseño de pantalla se muestran los datos del problema unidos a los resultados obtenidos al aplicar el modelo AHP, tal y como se muestra en la Figura 6.29. Para visualizar una comparación entre criterios o alternativas diferente a la actual sólo hay que seleccionarla.

Este diseño de pantalla corresponde a los expertos, pero el administrador también puede consultar problemas AHP, por lo que cuenta con un diseño muy similar. Cómo ocurría con PROMETHEE, el menú cuenta además con la opción *Expertos* y desaparece el botón *Modificar*, ya que un administrador no debe poder modificar un problema.

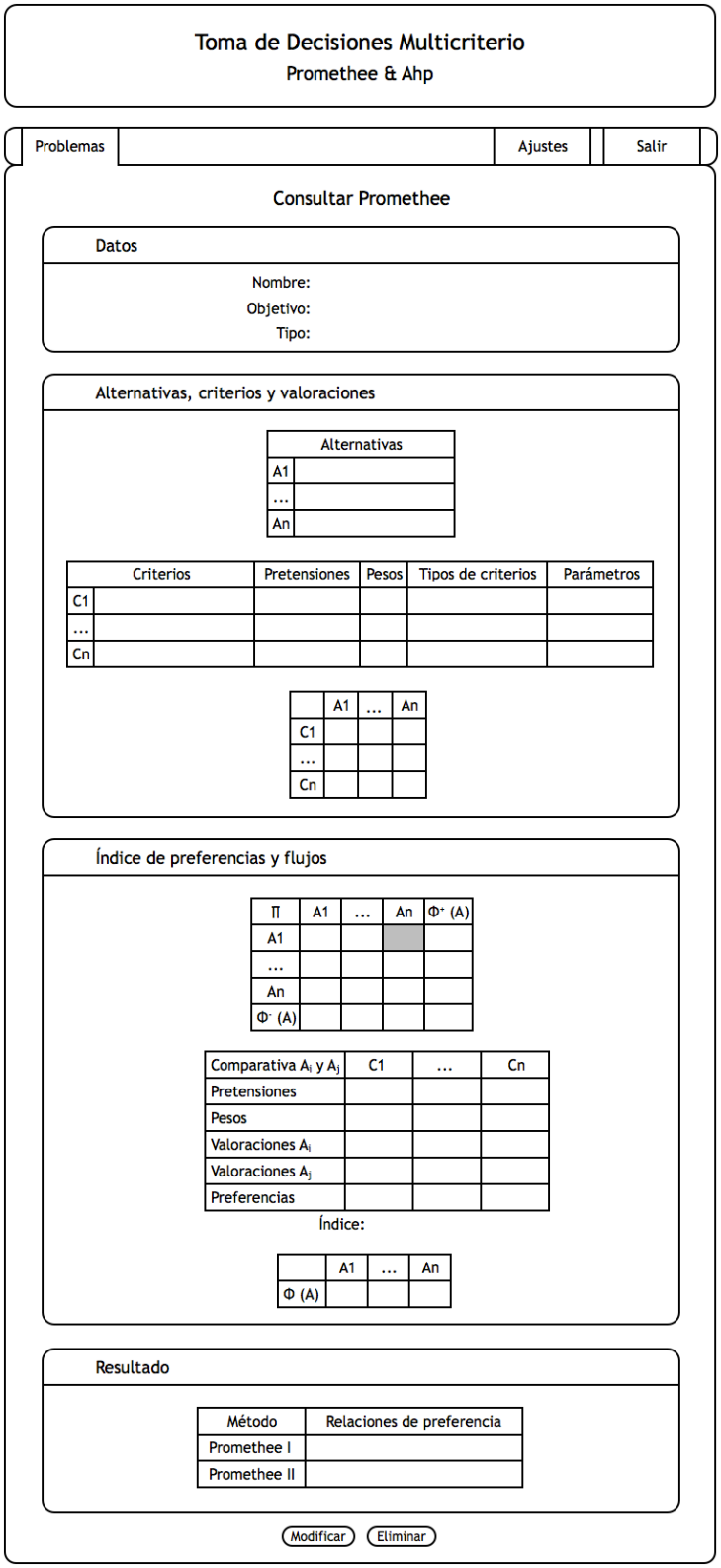


Figura 6.28. Consultar PROMETHEE.

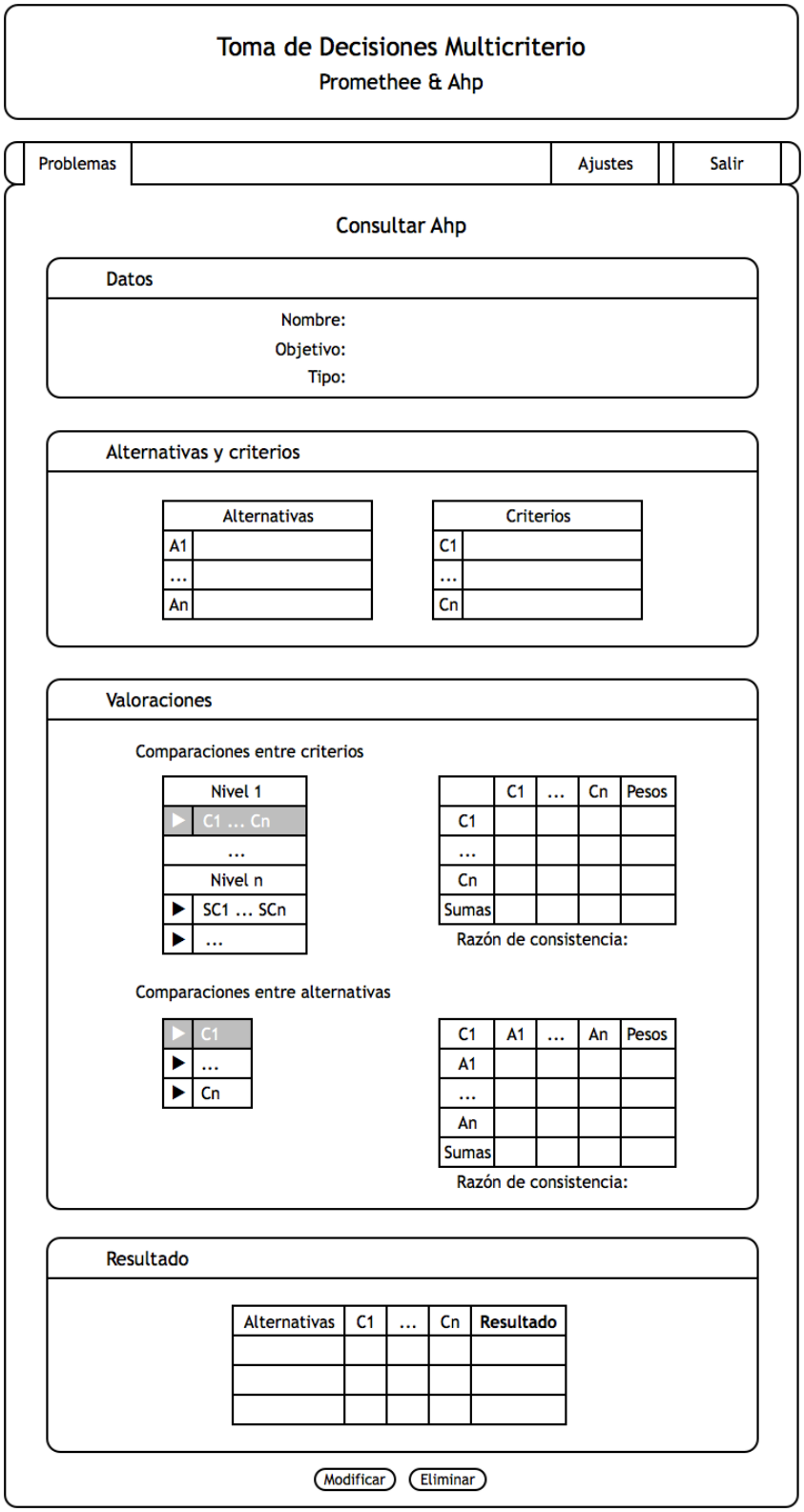


Figura 6.29. Consultar AHP.

- ***Crear PROMETHEE***

Este diseño de pantalla hace referencia a la funcionalidad de crear un problema de tipo PROMETHEE, a la que sólo pueden acceder los expertos, tal y como se indica en los requerimientos. A partir de esta pantalla, como se muestra en la Figura 6.30, se recogen todos los datos del problema que tenga que proporcionar el experto.

Esta pantalla es muy similar a la de modificar PROMETHEE. Las dos únicas diferencias radican en que esta última se titula *Modificar Promethee* y que carece del dato *Tipo*, donde se indica el tipo de problema, puesto que no tiene sentido cambiarlo.

Para ambos diseños de pantalla, las funcionalidades de crear y modificar no se pueden completar hasta que no aparece la pantalla de resultado PROMETHEE que se detalla más adelante.

- ***Resultado PROMETHEE***

Este diseño de pantalla aparece tras el de crear y modificar PROMETHEE y permite completar la creación y modificación de un problema de este tipo. En él se muestran los datos y resultados del problema en cuestión. Cada vez que se selecciona un índice de preferencia se muestra una comparativa entre alternativas con los datos asociados al cálculo del índice.

La Figura 6.31 muestra el diseño de pantalla asociado a la creación de un problema PROMETHEE, pero puede sufrir alguna variación en cuanto a la modificación, y es que un problema se puede modificar atendiendo a dos modos. Un problema que se crea se puede modificar si se decide realizar un análisis de sensibilidad antes de almacenarlo. En este caso el diseño de pantalla es éste mismo. Sin embargo, un problema que ya existe en el sistema también se puede modificar, y es en esta ocasión cuando el diseño varía, cambiándose el botón *Guardar* por el de *Actualizar*.

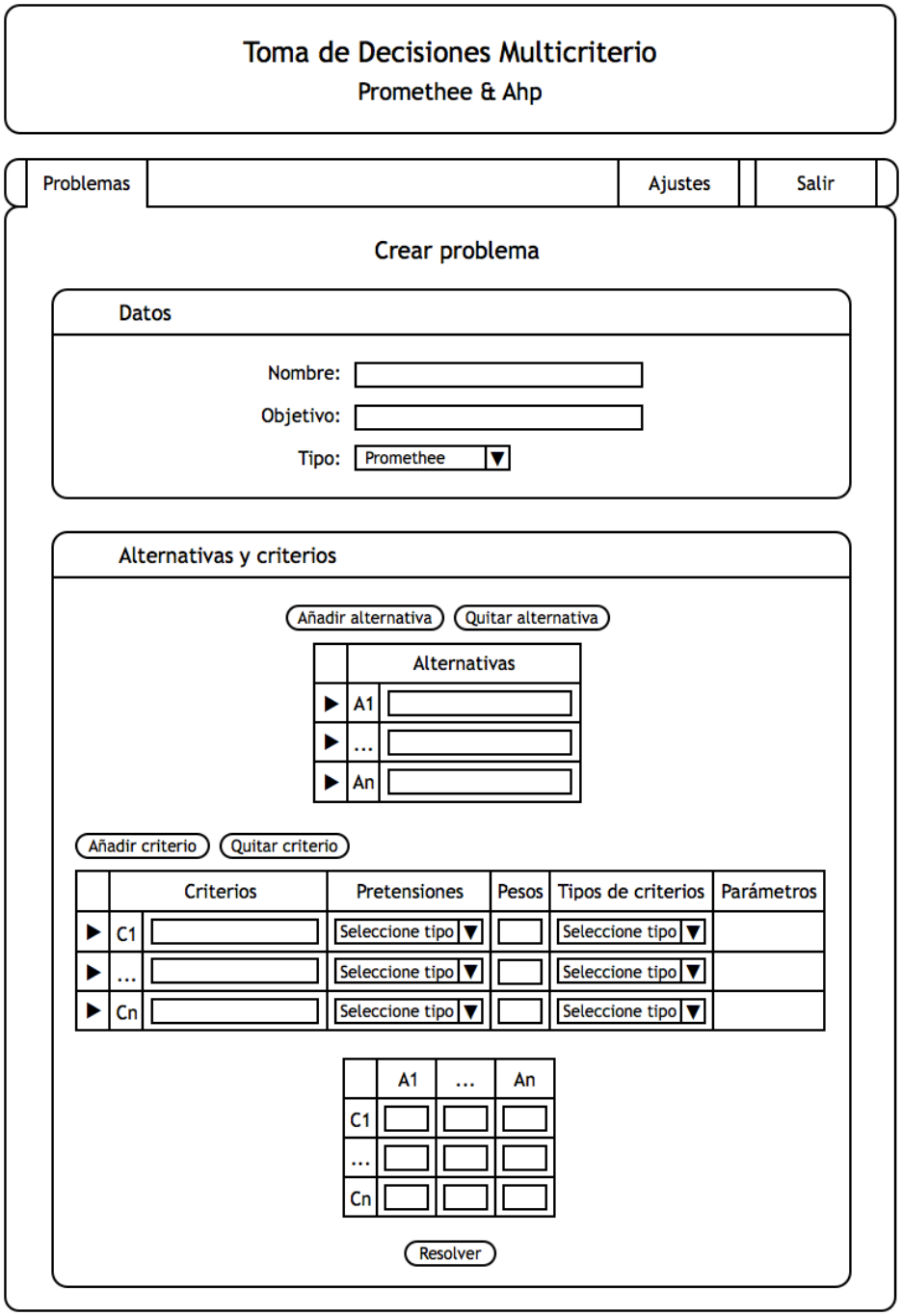


Figura 6.30. Crear PROMETHEE.

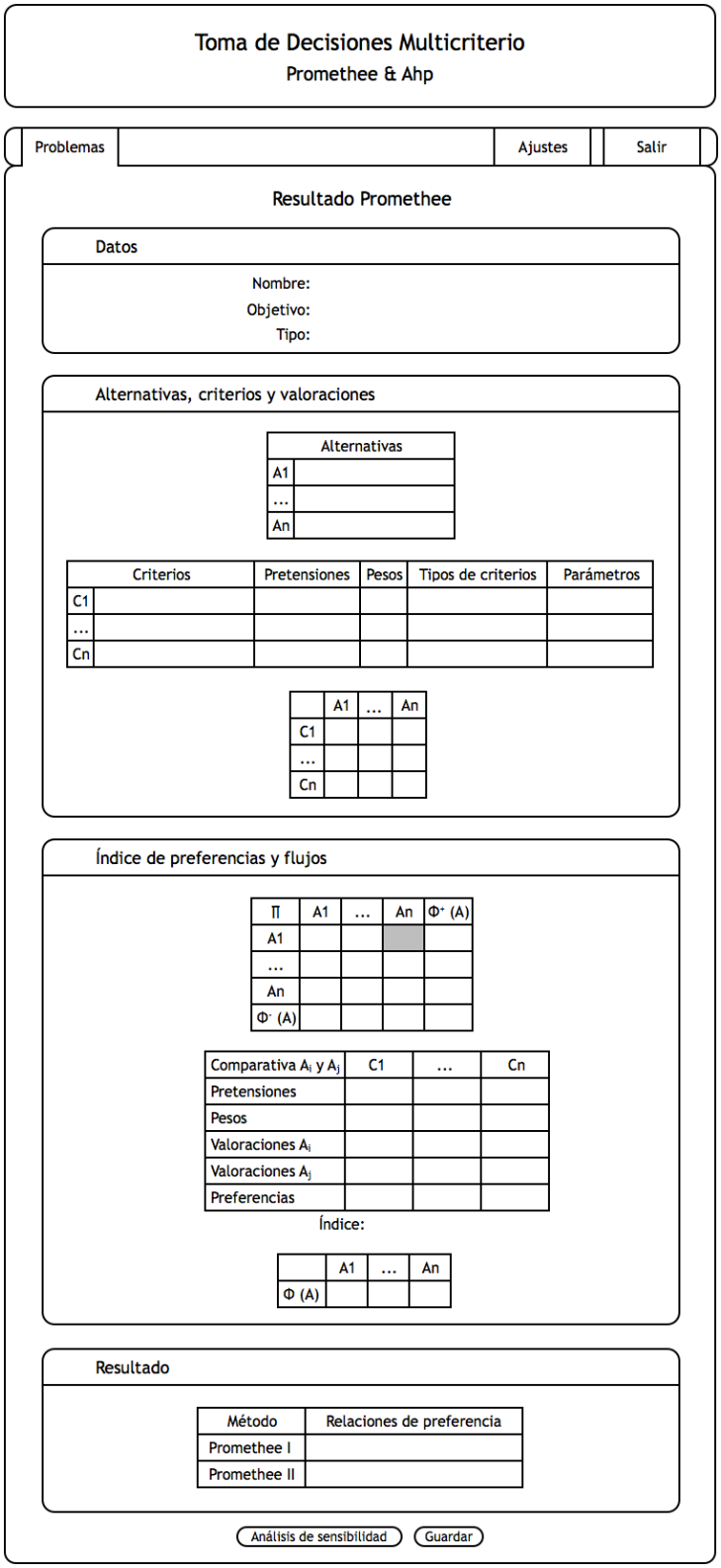


Figura 6.31. Resultado PROMETHEE.

- ***Crear AHP***

La creación de un problema AHP sólo está al alcance de los expertos y viene representada por el diseño de pantalla de la Figura 6.32. Se encarga de recoger los datos del problema necesarios para poder resolverlo. Para alternar entre una comparación entre criterios o alternativas y otra sólo hay que pulsar sobre el nombre de ésta.

El diseño de pantalla de modificar AHP es muy parecido al de crear. Existen dos diferencias principales. Una es que el título cambia por el de *Modificar Ahp*. Y la otra es que desaparece el dato *Tipo* del problema, puesto que carece de sentido cambiarlo.

La creación y modificación de problemas AHP no se puede finalizar hasta que no aparece la pantalla de resultado AHP que se detalla más adelante.

- ***Resultado AHP***

El diseño de pantalla resultado AHP se muestra a continuación del de crear y modificar AHP. Con él se finaliza la creación y modificación de un problema de este tipo. En la pantalla se recogen los datos del problema y los resultados obtenidos al resolverlo. Para navegar por las comparaciones entre criterios y alternativas hay que pulsar sobre sus nombres.

Aunque el diseño de la Figura 6.33 está asociado a la creación de un problema AHP, puede sufrir alguna variación en la modificación, puesto que un problema se puede modificar atendiendo a dos ámbitos. Por una parte, un problema que se está creando puede modificarse antes de que se almacene, si se decide realizar un análisis de sensibilidad, con lo que aparece este mismo diseño. Sin embargo, también es posible modificar un problema creado con anterioridad, pero en esta ocasión el botón de *Guardar* se sustituye por el de *Actualizar*.

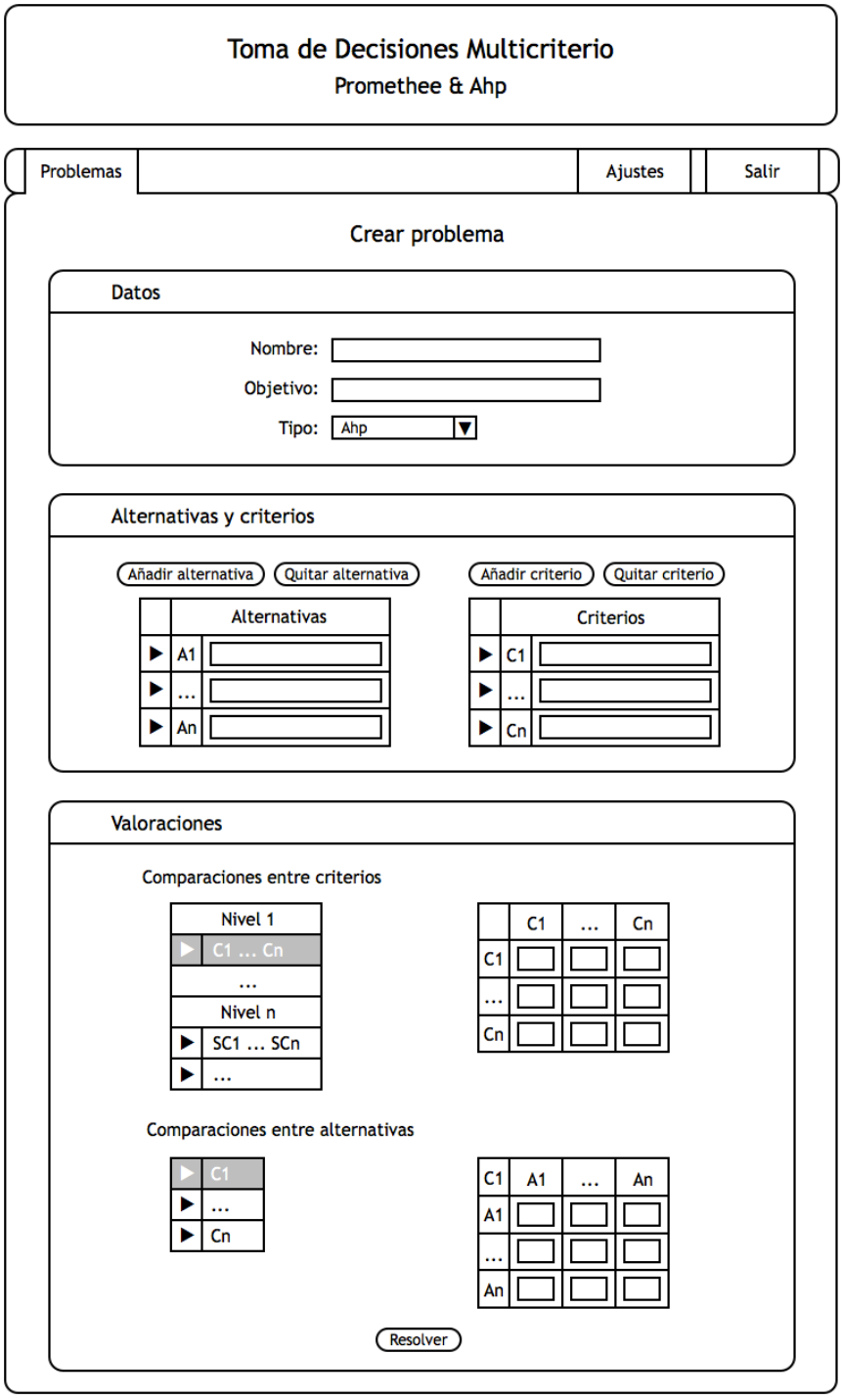


Figura 6.32. Crear AHP.

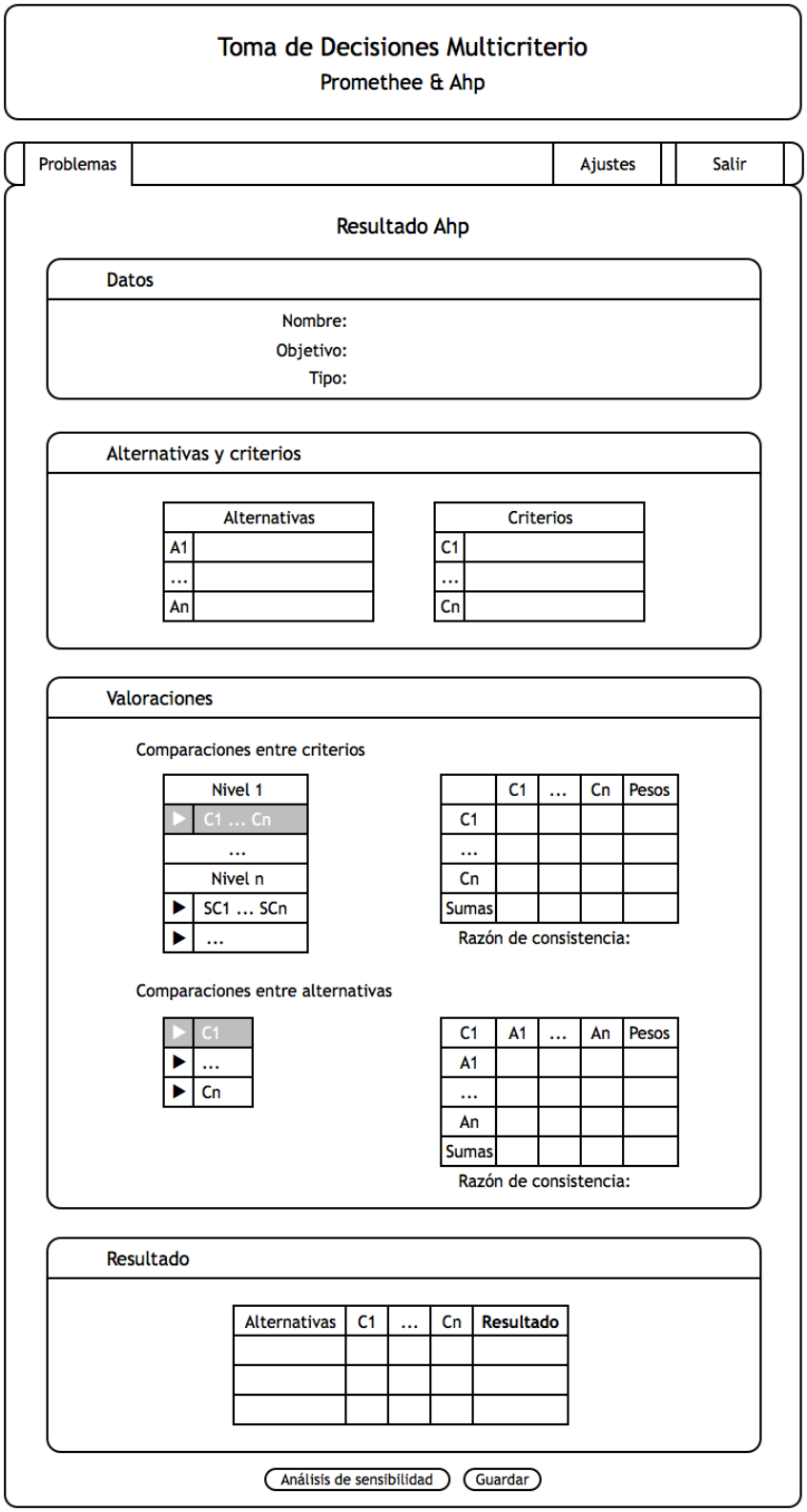


Figura 6.33. Resultado AHP.

6.3.2. Caminos de navegación

Hasta el momento se tiene un diseño visual estático de la interfaz, es decir, cada pantalla individualmente diseñada, pero no se tiene una idea de si en el conjunto de la interacción la acción va a transcurrir de forma fluida y comprensible para el usuario. Por lo tanto, es el momento de diseñar la interfaz en movimiento y comprobar que es usable.

Para abordar los caminos de navegación existentes en la aplicación se va a hacer uso de una herramienta llamada *storyboard*. Mediante un *storyboard* se muestra, a modo de secuencia, como se realiza una interacción concreta del usuario con la aplicación, haciendo uso de flechas para reflejar las distintas pantallas que van apareciendo mientras el usuario lleva a cabo la tarea. Se apoyan en los diseños de pantallas realizados con anterioridad, sin embargo, a diferencia de éstos, los *storyboards* muestran situaciones reales de uso, por lo tanto los campos no deben aparecer en blanco sino rellenos con los valores correspondientes a una interacción concreta posible.

Esta técnica es muy útil para describir los escenarios de situaciones concretas que ayuden a entender las partes del sistema. Además, se pueden analizar las pantallas del escenario para ver si los pasos que se dan son los correctos y si la acción se entiende y se modela correctamente.

En algunas ocasiones los *storyboards* son demasiado grandes, por lo que se han dividido en varias partes para facilitar su visualización. Asimismo, al tener el administrador y los expertos acceso a funcionalidades comunes, se ha decidido mostrar estas interacciones bajo el mismo *storyboard* para reducir el número de diagramas.

✦ *Acceso*

Se muestran dos *storyboards* relacionados con la entrada en el sistema. Por un lado el acceso por parte del administrador, como se muestra en la Figura 6.34, que da lugar a la pantalla de gestión de expertos, y por otro lado el del experto, representado por la Figura 6.35, que abre paso a la pantalla de gestión de problemas.

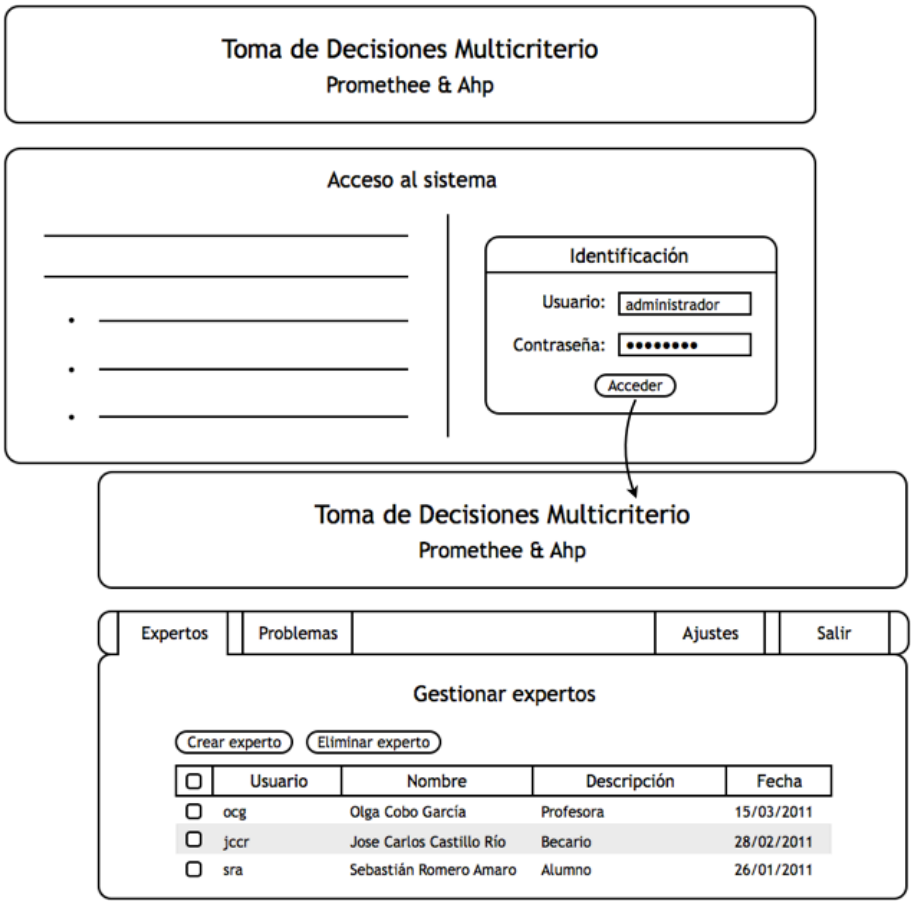


Figura 6.34. Acceso (Administrador).

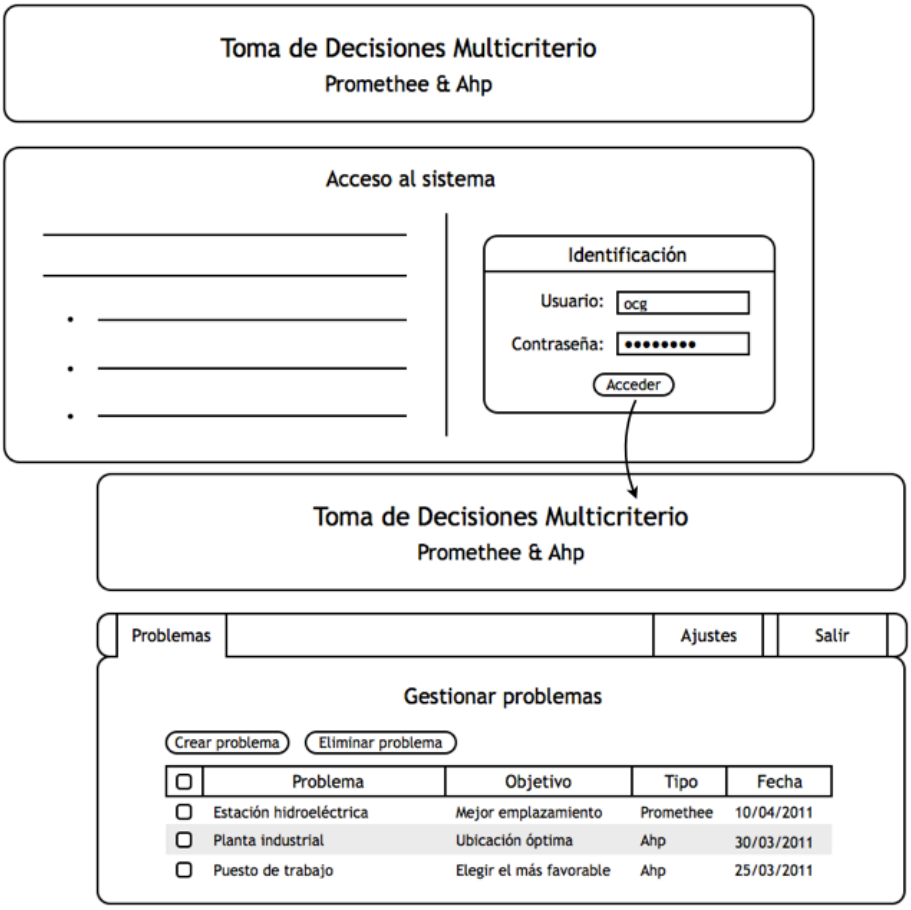


Figura 6.35. Acceso (Experto).

✦ ***Modificar contraseña***

La acción de modificar contraseña está disponible tanto para el administrador como los expertos. Es por ello que el *storyboard* es idéntico para ambos usuarios. La interacción viene representada por la Figura 6.36.

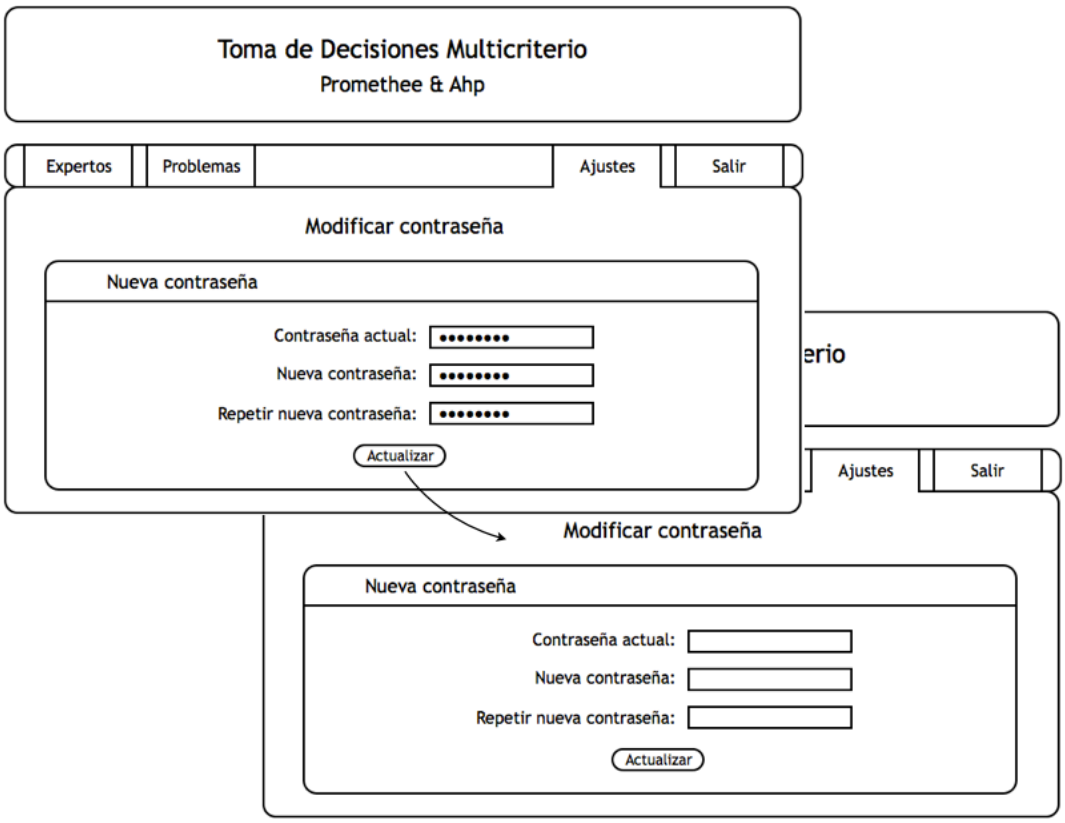


Figura 6.36. Modificar contraseña.

✦ *Crear experto*

La tarea de crear expertos únicamente puede ser llevada a cabo por el administrador. El *storyboard* viene determinado por la Figura 6.37.

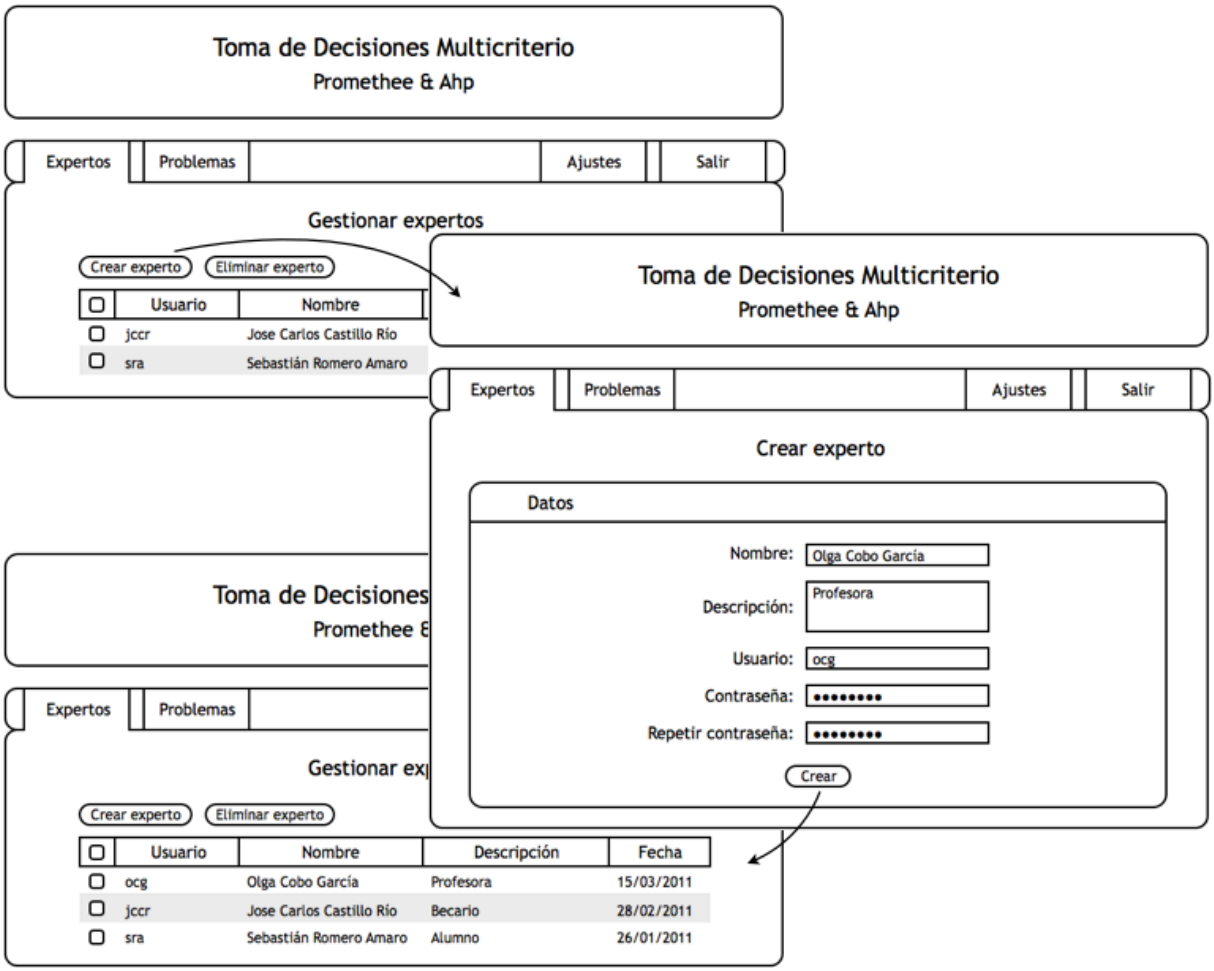


Figura 6.37. Crear experto.

✦ *Modificar experto*

La acción de modificar experto viene representada por la Figura 6.38 y al igual que ocurre con la creación de expertos sólo el administrador puede modificarlos.

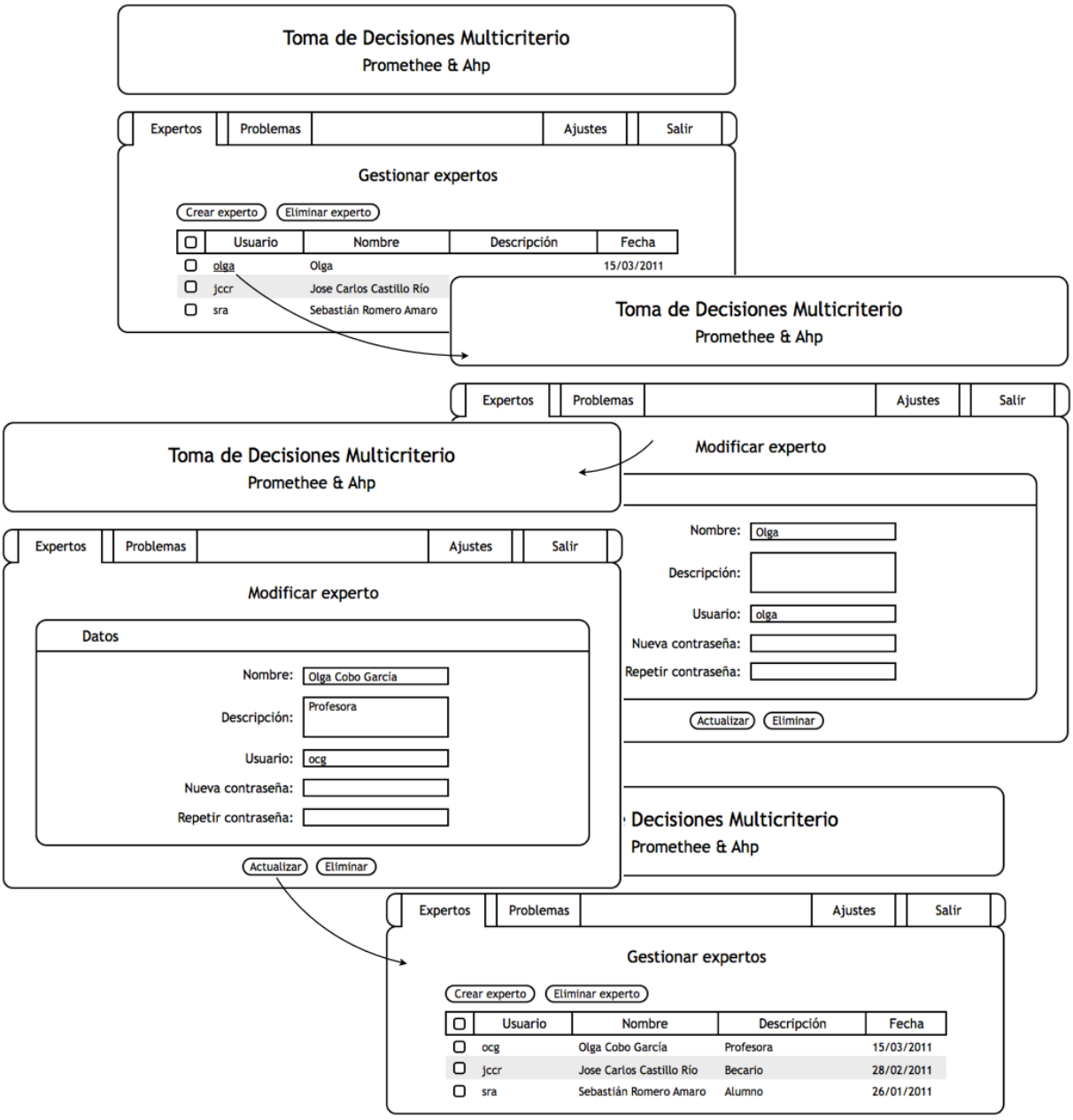


Figura 6.38. Modificar experto.

✦ *Eliminar experto*

La eliminación de un experto se puede llevar a cabo de dos modos. Se puede consultar el experto y posteriormente eliminarlo, interacción reflejada por la Figura 6.39. O también es posible seleccionar uno o varios expertos desde la pantalla de gestión y eliminarlos, como se muestra en la Figura 6.40. En ambos casos es necesario confirmar la acción.

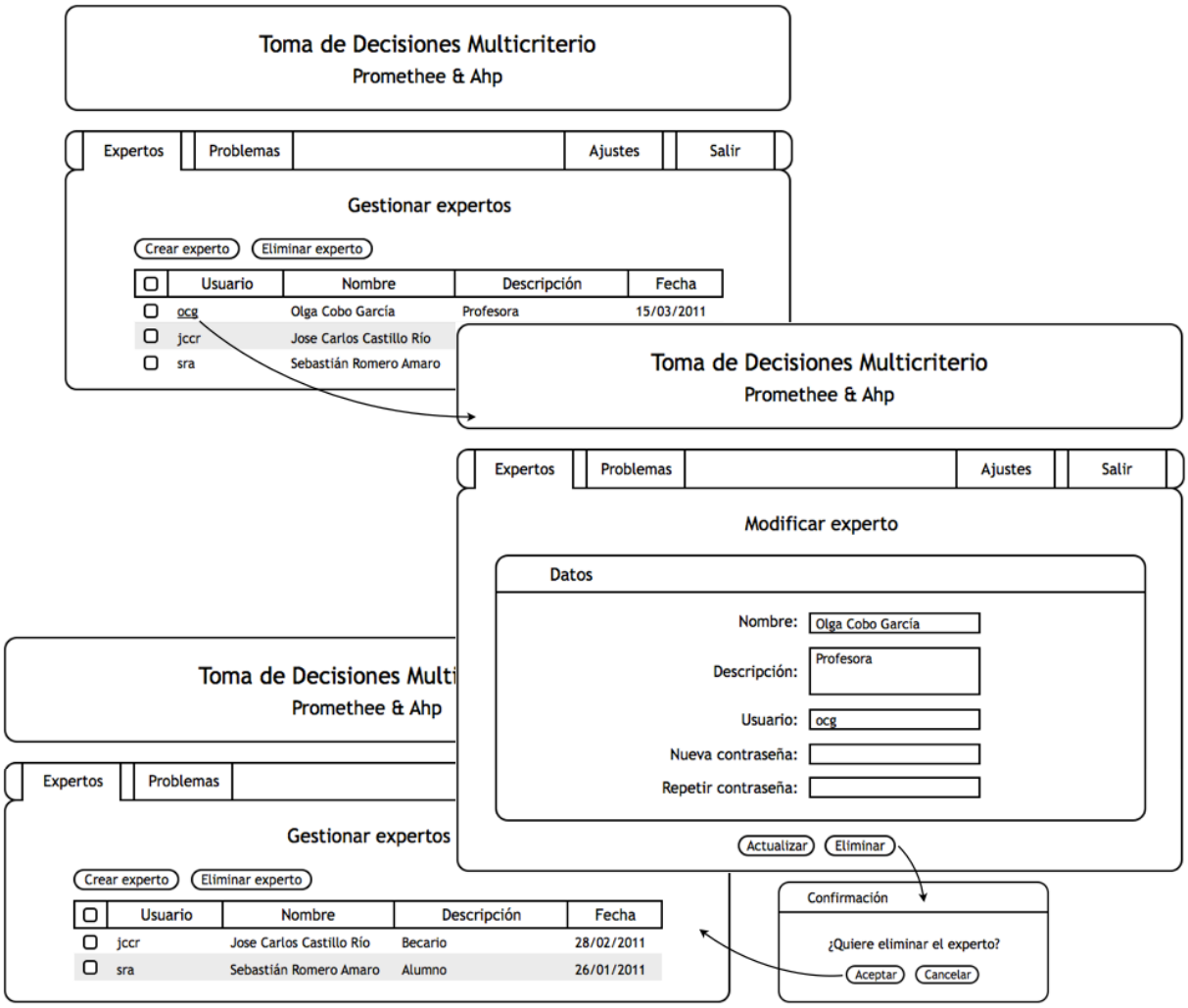


Figura 6.39. Eliminar experto.

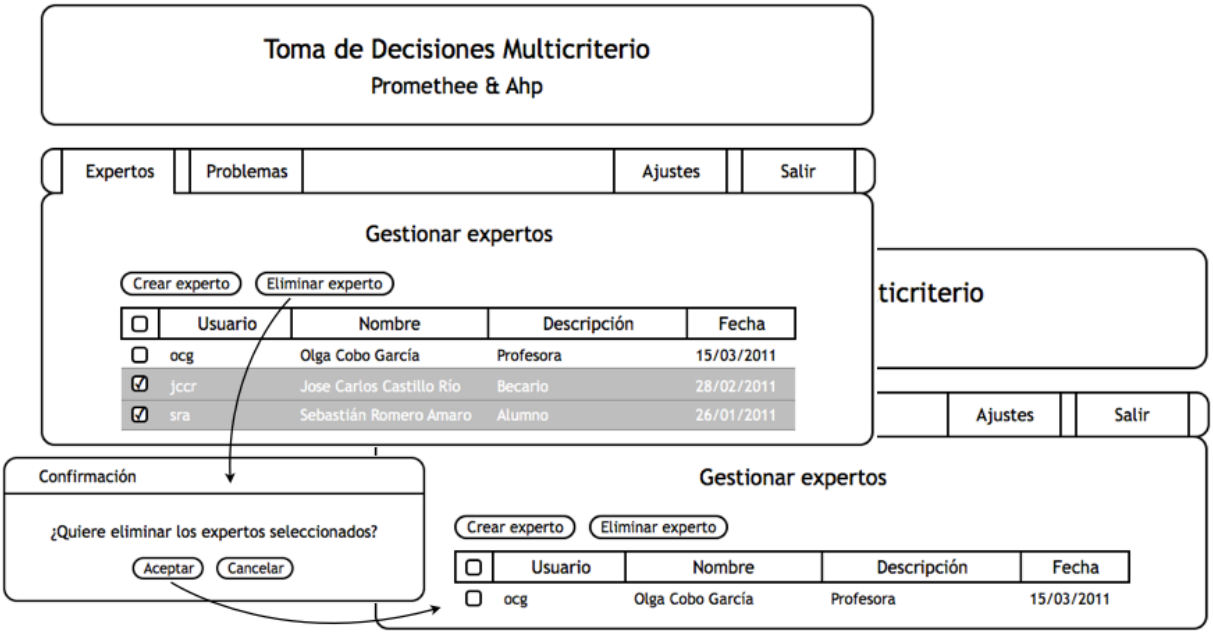


Figura 6.40. Eliminar expertos.

- ***Consultar PROMETHEE***

La acción de consultar un problema de tipo PROMETHEE puede realizarla tanto el administrador como los expertos. Por ello, aunque el *storyboard* de la Figura 6.41 representa la interacción seguida por un experto al consultar el problema, los pasos a seguir son exactamente los mismos para el administrador.

- ***Consultar AHP***

La Figura 6.42 muestra como un experto consulta un problema de tipo AHP. Pero los administradores, al igual que los expertos, también pueden consultar problemas de este tipo, por lo que la interacción seguida es la misma.

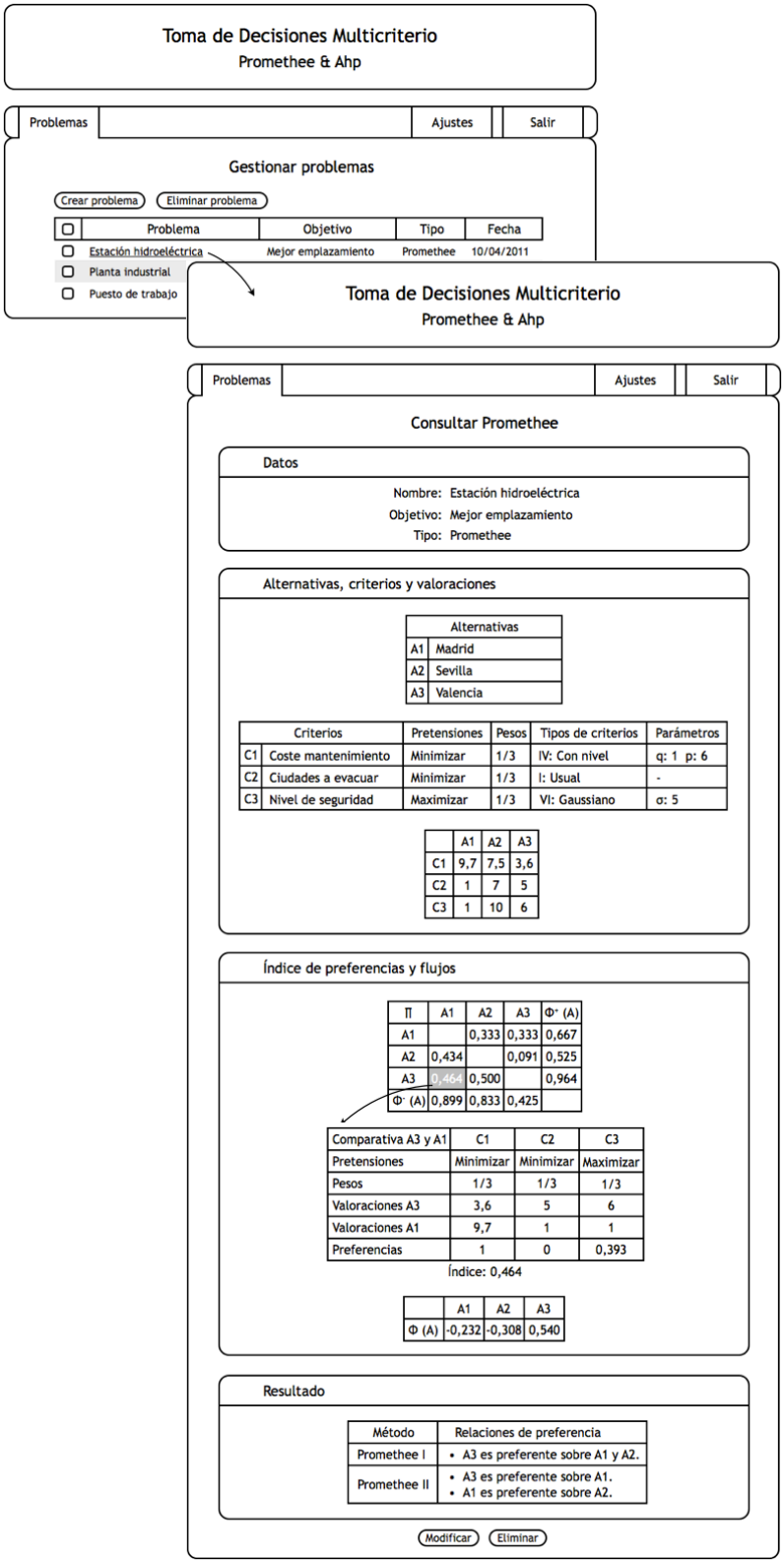


Figura 6.41. Consultar PROMETHEE.

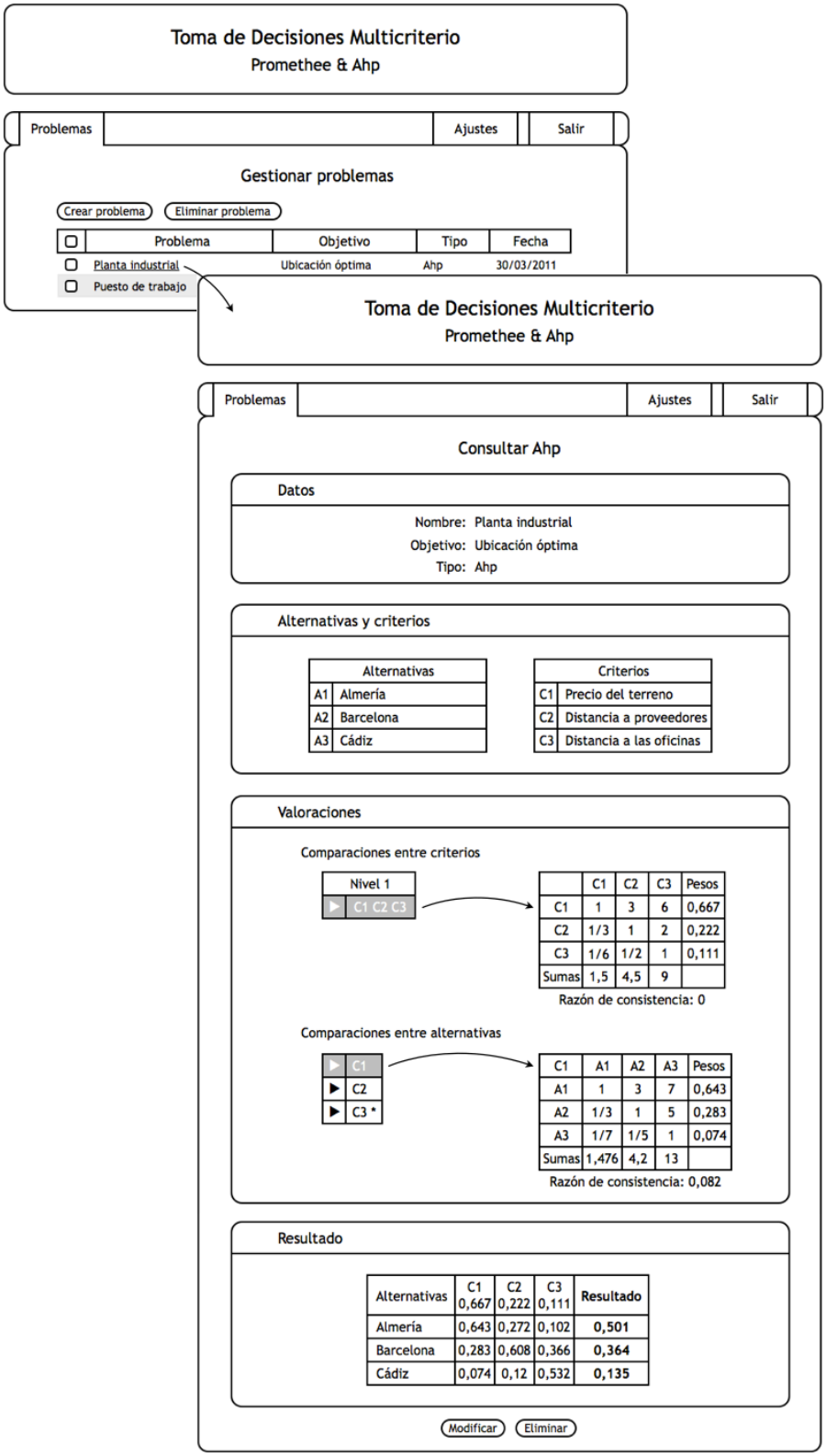


Figura 6.42. Consultar AHP.

- ***Crear PROMETHEE***

El *storyboard* que representa la interacción seguida para crear un problema de tipo PROMETHEE es demasiado extenso, por esta razón se ha divido en dos partes. La primera parte, reflejada por la Figura 6.43, abarca el inicio de la acción y la introducción de los datos que definen el problema. La segunda parte, reflejada por la Figura 6.44, muestra los resultados obtenidos al resolver el problema y su posterior almacenamiento.

Con el objetivo de reducir el número de *storyboards* y dada la gran similitud entre las interacciones para crear y modificar un problema PROMETHEE, se ha decidido no incluir un *storyboard* específico para ésta última. La modificación de un problema PROMETHEE se inicia del mismo modo que en la acción *Consultar PROMETHEE*, reflejada por la Figura 6.41, y tras pulsar el botón *Modificar,* la interacción continúa del mismo modo que cuando se crea.

- ***Crear AHP***

El *storyboard* que abarca la creación de un problema de tipo AHP es demasiado grande, por ello se ha dividido en dos partes. La primera está representada por la Figura 6.45, y muestra el comienzo de la acción seguido de la introducción de los datos del problema. La presentación de los resultados y el posterior almacenamiento del problema se recoge en la Figura 6.46.

Como la creación y modificación de un problema AHP son muy parecidas, se ha tomado la determinación de no crear un *storyboard* específico para la última. La modificación de un problema AHP comienza igual que *Consultar AHP*, tal y como puede observarse en la Figura 6.42, y una vez se pulsa el botón Modificar, la interacción continúa del mismo modo que cuando se crea.

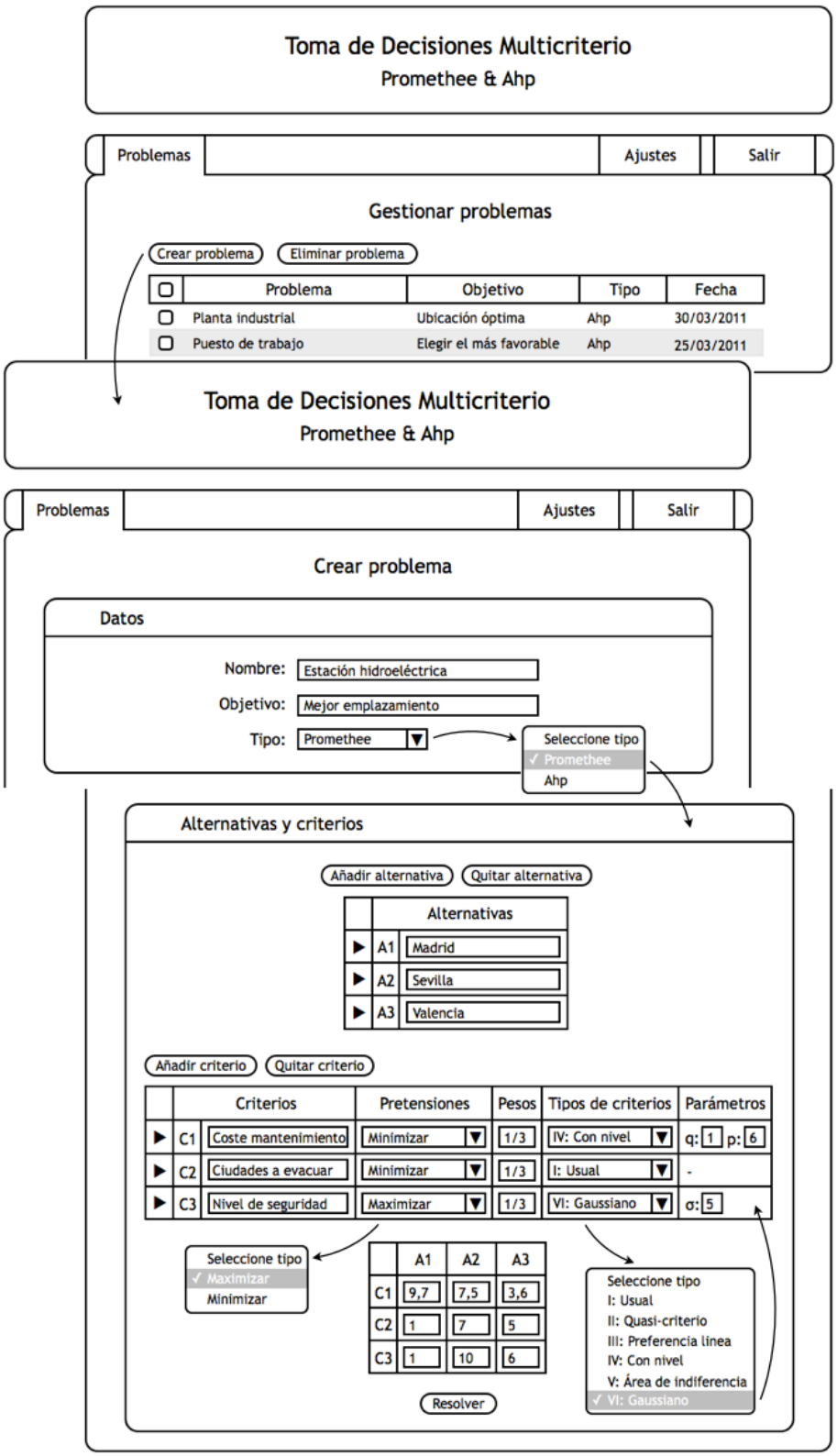


Figura 6.43. Crear PROMETHEE. Parte I.

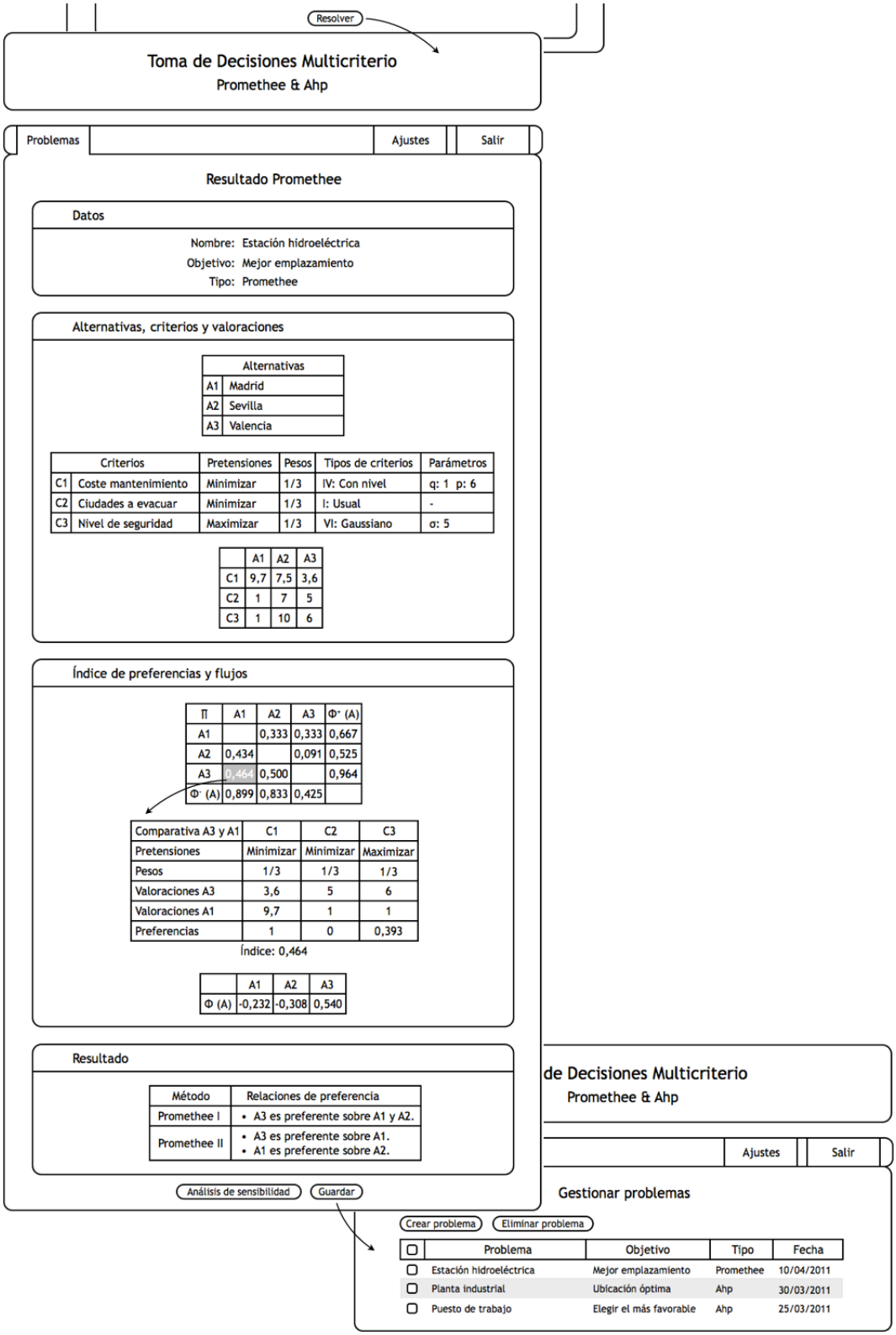


Figura 6.44. Crear PROMETHEE. Parte II.

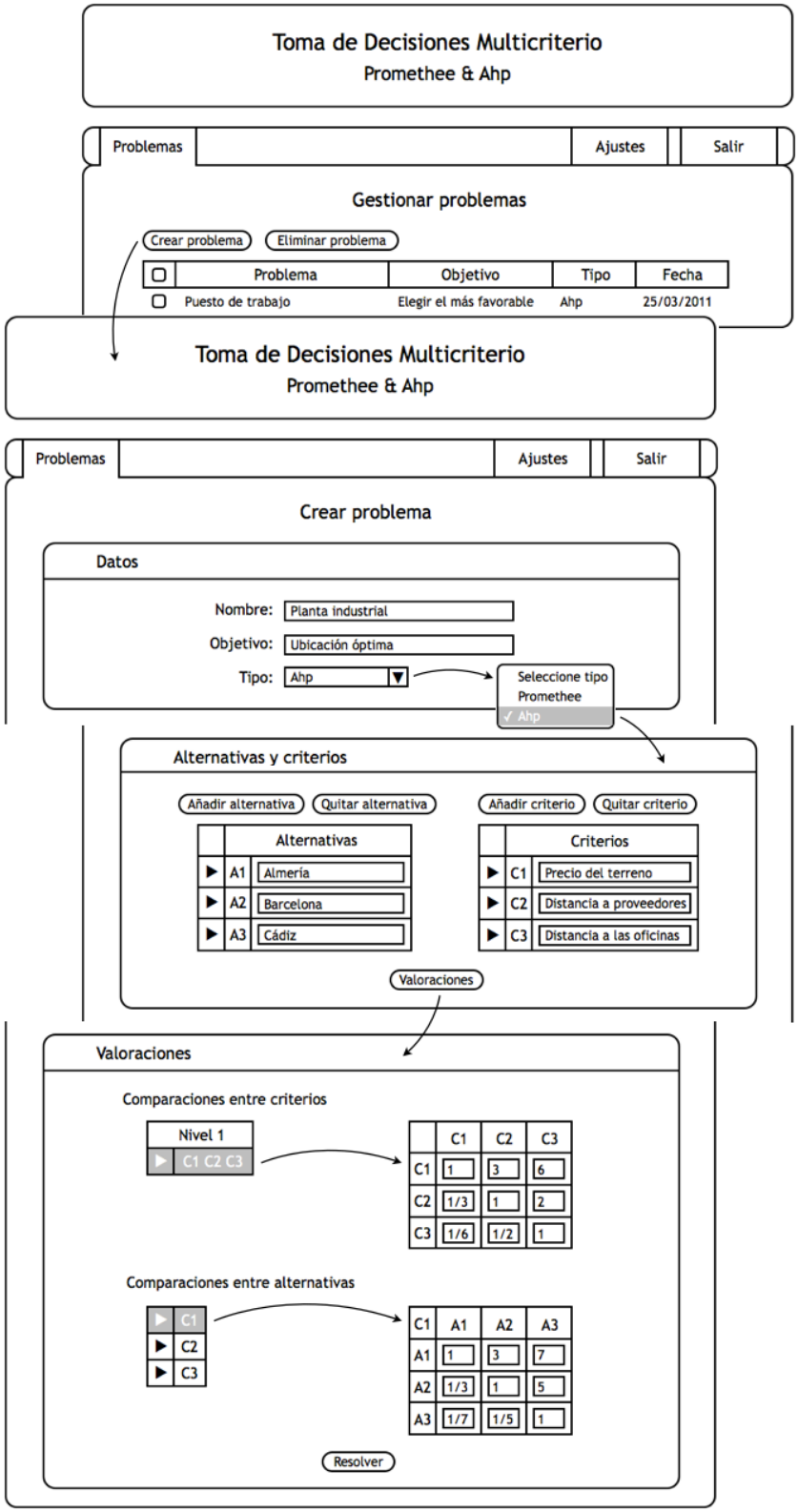


Figura 6.45. Crear AHP. Parte I.

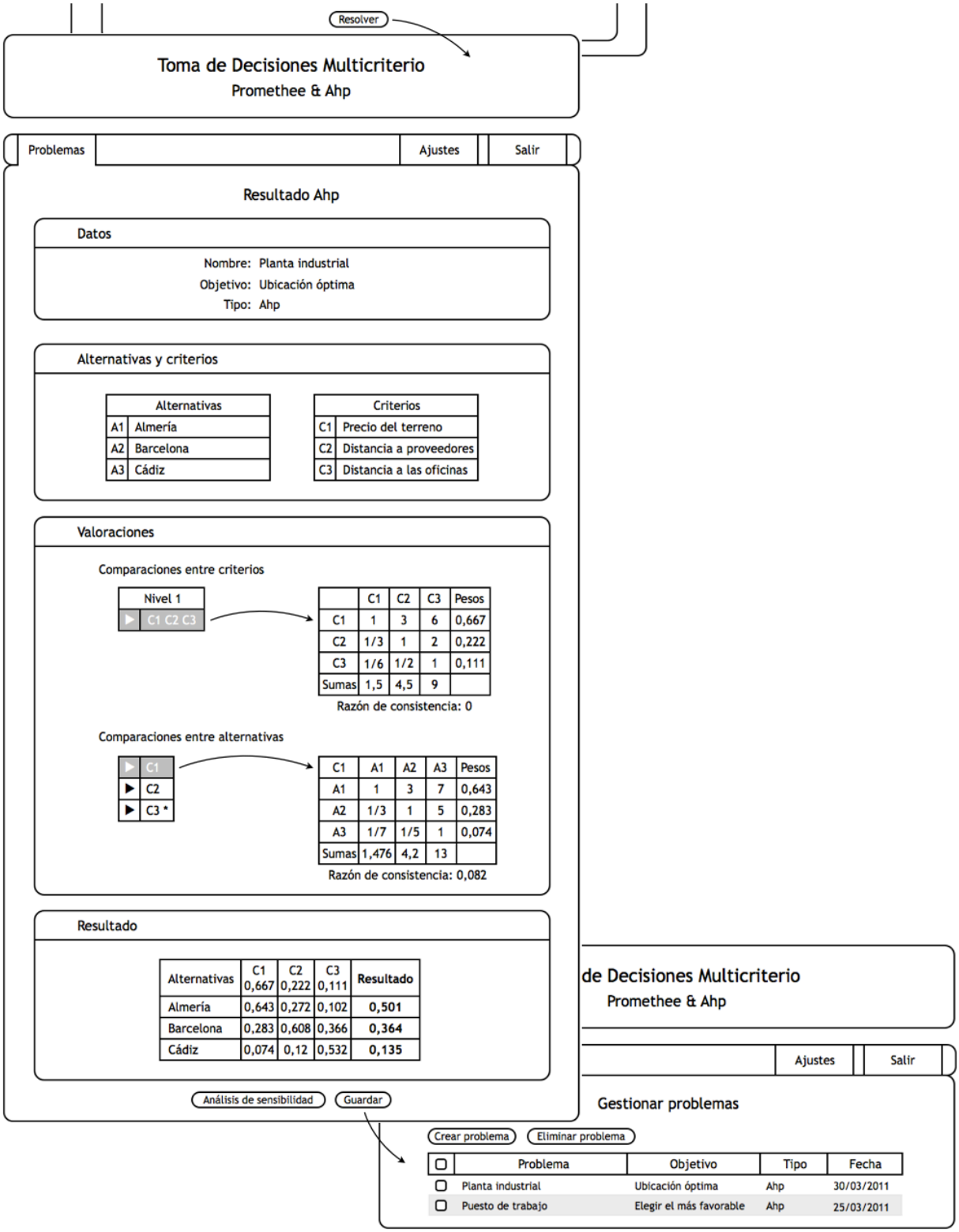


Figura 6.46. Crear AHP. Parte II.

✦ *Eliminar problema*

La eliminación de un problema se puede llevar a cabo de dos modos. Se puede consultar el problema y posteriormente eliminarlo, interacción reflejada por la Figura 6.48. O también es posible seleccionar uno o varios problemas desde la pantalla de gestión y eliminarlos, como se muestra en la Figura 6.47. En ambos casos es necesario confirmar la acción.

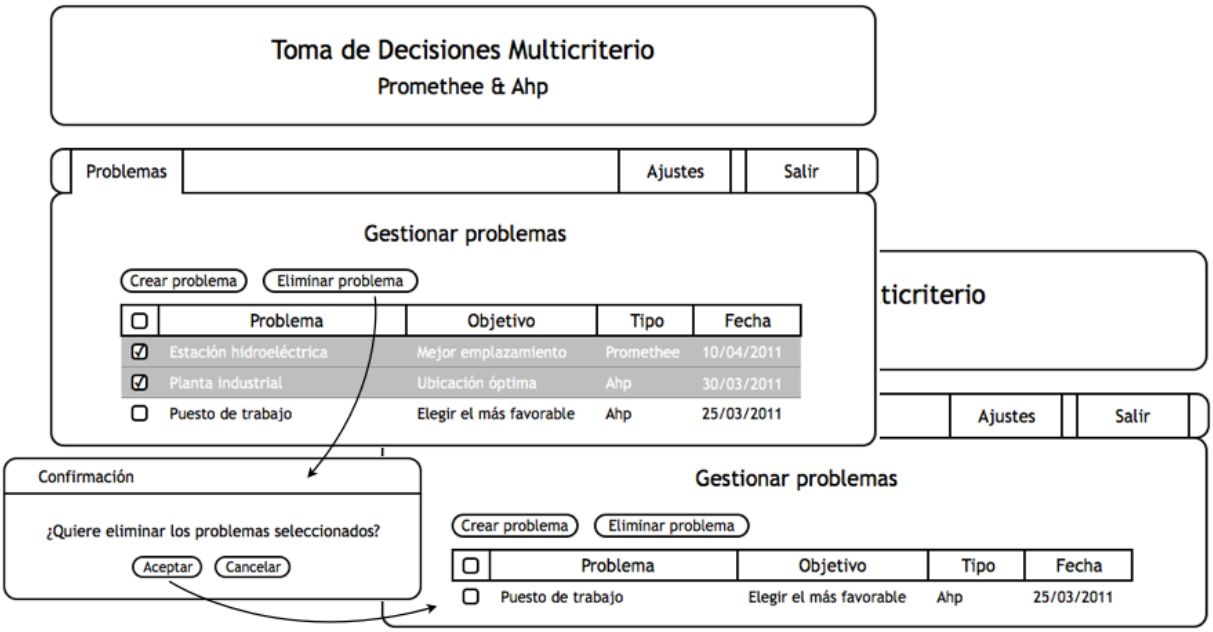


Figura 6.47. Eliminar problemas.

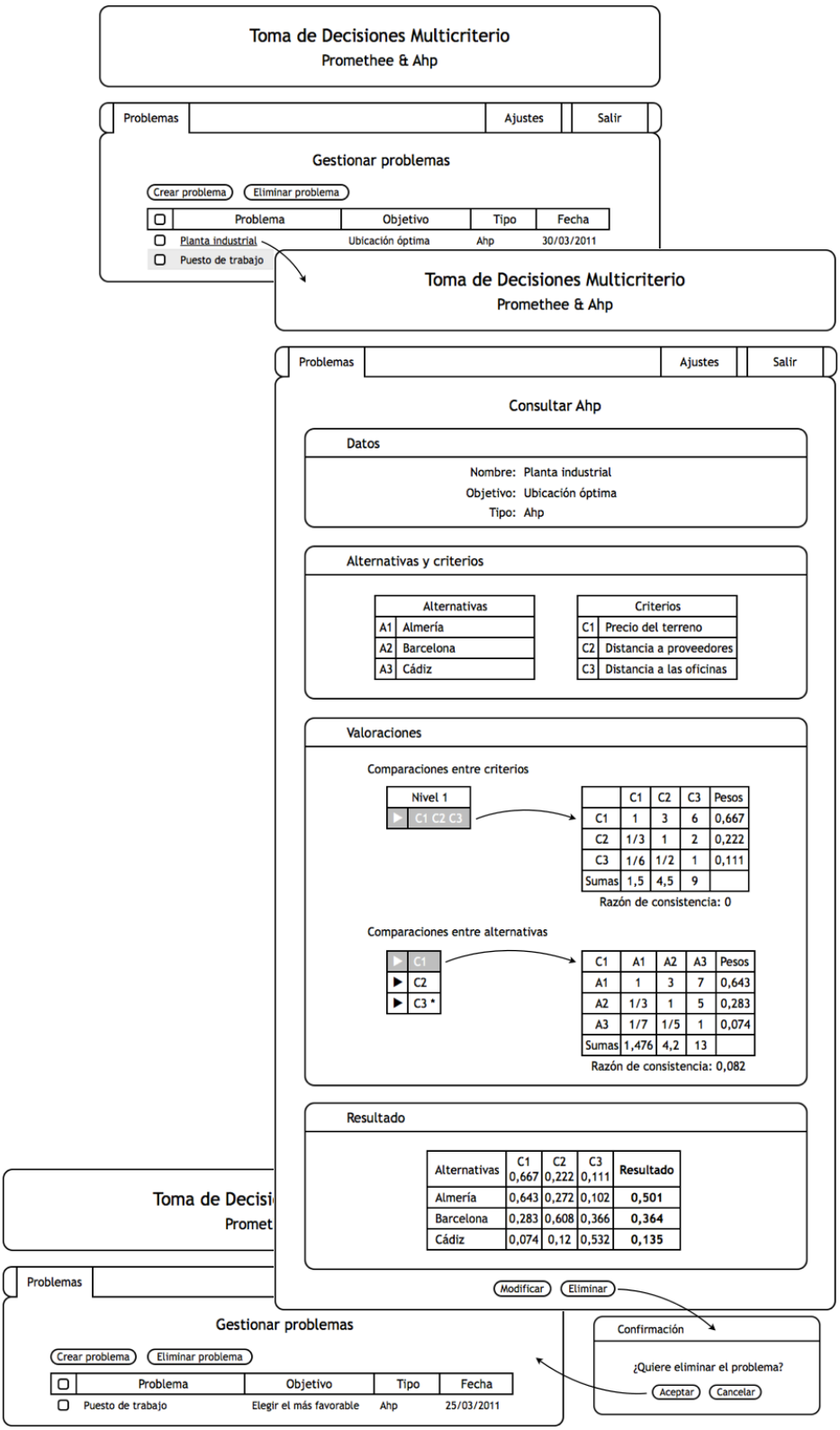


Figura 6.48. Eliminar problema.

6.3.3. Mensajes de error

Una vez que se tiene el análisis de tareas, que refleja las secuencias de diálogo entre el usuario y la interfaz, y una vez diseñadas las pantallas y los caminos de navegación, se tiene una idea muy clara de cómo se va a desarrollar físicamente la interacción entre usuario y ordenador y por tanto de las posibles situaciones de error que pueden ocurrir durante esa interacción. Por ello ahora es el momento de diseñar los mensajes de error.

Los mensajes de error son el medio por el que el sistema comunica al usuario que se ha producido un error en la interacción. Los errores se producen por falta de comunicación sobre la interfaz porque no se ha entendido correctamente el estado del sistema, se está tratando de hacer alguna acción no permitida o bien inadvertidamente.

Cuando un usuario comete un error se siente confuso y aumenta su ansiedad, lo que puede producir estrés. Esta situación debe evitarse, ya que a menudo, una interfaz se recuerda más por lo que pasó en una situación de error que por lo que pasó cuando las cosas fueron bien. Estas situaciones pueden llevar a una mala valoración de la calidad de la interfaz por los usuarios.

Los mensajes de error son muy importante de cara a la usabilidad, ya que bien diseñados, permiten aumentar la confianza del usuario en el uso de la interfaz y que pueda seguir con su tarea. Es por ello que hay que diseñarlos con cuidado. A la hora de diseñar mensajes de error se deben seguir las siguiente reglas:

- Ser breves: el usuario es una persona ocupada tratando de llevar a cabo una tarea. Si se le presenta un mensaje de error muy largo, lo normal es que no lo lea ya que no tendrá tiempo. Por ello hay que diseñar mensajes de error claros pero breves.

- Ser específicos: los mensajes demasiado generales no son adecuados ya que dicen al usuario que algo ha ido mal, pero no le indican claramente qué. Por lo tanto el usuario no sabrá qué hacer para impedir que se produzca el error y su sensación de frustración aumentará.

- Usar un tono positivo y constructivo: nunca recriminar al usuario qué ha hecho mal, tampoco informar sin más de lo que ocurre, en su lugar siempre que sea posible hay que indicarle qué debe hacer para eliminar el error.

- Usar un formato físico apropiado: la mayoría de usuarios encuentran más fácil leer un mensaje donde se mezclan mayúsculas y minúsculas de la forma habitual, por tanto este formato es siempre preferible. Los mensajes escritos únicamente con mayúsculas deberían reservarse para avisos breves y graves. Los mensajes de error deberían aparecer siempre en la misma posición de la pantalla.

A continuación se detallan los mensajes de error que pueden aparecer durante el uso de la aplicación. Se han dividido en dos categorías.

✦ ***Situaciones de error***

Ocurren cuando sucede algún problema debido a causas ajenas al usuario o bien porque éste en lugar de seguir el flujo natural de acciones que marca la aplicación utiliza el historial de navegación del navegador web. Estos tipo de mensajes ocurren rara vez y se clasifican atendiendo a la naturaleza del error.

- Servicio no disponible: puede ocurrir en el momento de la autentificación y debido a tres causas diferentes. Bien porque los datos del administrador no están almacenados en la base de datos, luego se desconoce su nombre de usuario y contraseña. También puede ocurrir que exista más de un administrador en la base de datos. Y por último, si se produce algún problema durante la comprobación de la contraseña. *"El servicio no está disponible temporalmente"*.

- Experto o problema no existente: se produce cuando se trata de consultar, actualizar o eliminar un experto o problema que ya no existe en el sistema. Requiere un uso inapropiado del historial. *"El experto ya no existe"* y *"El problema ya no existe"*.

- Problema ya guardado: se produce cuando se intenta guardar de nuevo un problema recién almacenado. Requiere un uso inapropiado del historial. *"El problema ya ha sido guardado"*.

- Guardar o modificar experto: si al crear o modificar un experto se produce algún error con la encriptación de su contraseña. *"Se ha producido un error al guardar el experto"* y *"Se ha producido un error al modificar el experto"*.

- Modificar contraseña: se produce cuando un administrador o experto tratan de modificar su contraseña. Puede ocurrir en dos ocasiones durante esta tarea: en el momento de comprobar la contraseña actual o en el de actualizarla. Además puede tener varios motivos según el usuario. En el caso del administrador puede deberse a que no exista en la base de datos, haya múltiples administradores o bien se haya producido algún error al encriptar la contraseña. En el caso del experto, bien porque no exista o bien por un error con la encriptación de la contraseña. *"Se ha producido un error al comprobar la contraseña actual"* y *"Se ha producido un error al modificar la contraseña"*.

✦ ***Errores de validación***

Se producen cuando los datos que introduce el usuario durante el uso de la aplicación son incompletos o incorrectos. Este tipo de mensajes suelen ser los más comunes y se clasifican de acuerdo al campo al que están asociados y al tipo de error.

- Usuario o contraseña: se produce durante el proceso de autentificación si se introduce un nombre de usuario o una contraseña inválidos. *"El usuario o la contraseña son incorrectos"*.

- Nombre: cuando al crear o modificar un experto o problema no se ha proporcionado un nombre. *"Introduzca el nombre"*.

- Descripción: se produce si la descripción del experto que se crea o modifica supera los 200 caracteres. *"Longitud máxima de 200 caracteres"*.

- Usuario: se produce durante la autentificación, o bien al crear o modificar un experto si no se introduce un nombre de usuario. *"Introduzca el usuario"*.

- Usuario ocupado: si al crear o modificar un experto su nombre de usuario pertenece a otro experto ya existente. *"Nombre de usuario ocupado"*.

- Contraseña: se produce durante la autentificación o bien al crear un experto si no se introduce una contraseña. También puede aparecer al modificar un experto si no se rellena el campo contraseña y sí el de repetirla. *"Introduzca la contraseña"*.

- Repetir contraseña: se produce al crear un experto si no se introduce una contraseña o al modificarlo si no se rellena el campo de repetir contraseña y sí el de contraseña. *"Repita la contraseña"*.

- Contraseña actual: cuando al modificar su contraseña el administrador o un experto no se proporciona la contraseña actual. *"Introduzca la contraseña actual"*.

- Contraseña actual no coincidente: si al modificar su contraseña el administrador o un experto la contraseña actual proporcionada no coincide. *"La contraseña no coincide"*.

- Nueva contraseña: si al modificar su contraseña el administrador o un experto no se introduce una nueva contraseña. *"Introduzca la nueva contraseña"*.

- Repetir nueva contraseña: si al modificar su contraseña el administrador o un experto no se repite la nueva contraseña. *"Repita la nueva contraseña"*.

- Contraseñas no coincidentes: se produce cuando la contraseña y su repetición no son iguales al crear o modificar un experto, o bien cuando el administrador o experto modifican su contraseña. *"Las contraseñas no coinciden"*.

- Objetivo: si el objetivo de un problema no se rellena. *"Introduzca el objetivo"*.

- Alternativas: si no se introduce el nombre de las alternativas. *"Introduzca el nombre de las alternativas"*.

- Criterios: si no se introduce los datos de los criterios PROMETHEE de forma adecuada. *"Introduzca los datos de los criterios correctamente"*. O cuando no se introduce el nombre de los criterios AHP. *"Introduzca el nombre de los criterios"*.

- Valoraciones: se produce cuando no se rellenan las valoraciones PROMETHEE correctamente. *"Introduzca las valoraciones correctamente"*. O cuando no se completan todas las comparativas AHP. *"Complete todas las comparaciones"*.

- Pesos: si la suma de los pesos de los criterios PROMETHEE no suma exactamente uno. *"La suma de los pesos debe ser uno"*.

6.3.4. Mensajes de éxito

Los mensajes de éxito son aquellos que aparecen cuando el usuario finaliza una tarea con éxito. Su principal objetivo es informar al usuario de que la acción que estaba realizando ha finalizado y se ha completado satisfactoriamente. De cara a la usabilidad son muy importantes ya que bien diseñados permiten aumentar la confianza del usuario en el uso de la aplicación.

A la hora de diseñarlos hay que prestar especial interés en retroalimentar al usuario de la forma más adecuada posible. Se busca que el mensaje capte la atención del usuario durante un breve periodo de tiempo, que no se rompa la interacción con la interfaz en ningún momento y que sea lo menos invasivo posible, de modo que el usuario lo olvide una vez lo haya leído. En cuanto al contenido, se busca que el mensaje sea breve y conciso para que ocupe poco espacio y sea rápido de leer, pero que al mismo tiempo sea muy claro explicativo.

Se clasifican de acuerdo a la acción que se ha completado con éxito.

- Guardar experto: aparece cuando se almacena con éxito un experto en el sistema. *"Experto guardado"*.

- Actualizar experto: cuando se modifica un experto. *"Experto actualizado"*.

- Eliminar experto: cuando se elimina uno o varios expertos. *"Experto eliminado"* y *"Expertos eliminados"*.

- Guardar problema: aparece cuando se almacena un problema con éxito. *"Problema guardado"*.

- Actualizar problema: cuando se modifica un problema. *"Problema actualizado"*.

- Eliminar problema: cuando se elimina uno o varios problemas. *"Problema eliminado"* y *"Problemas eliminados"*.

- Actualizar contraseña: aparece cuando el administrador o un experto modifica su contraseña con éxito. *"Contraseña actualizada"*.

6.3.5. Mensajes de confirmación

Los mensajes de confirmación son aquellos que aparecen cuando el sistema va a realizar un acción importante y requiere una confirmación por parte del usuario. Normalmente este tipo de mensajes ocurren porque el sistema va a realizar una tarea que no puede deshacerse y hay datos que no se van a poder recuperar, por lo que necesita de una confirmación por parte del usuario para asegurarse que quiere llevar a cabo dicha tarea. Más concretamente, en este proyecto únicamente aparecerán cuando el usuario pretenda eliminar datos del sistema.

Durante su diseño, se pretende que estos mensajes llamen la atención del usuario en todo momento, rompiendo el flujo de eventos con la interfaz, de modo que no se puedan hacer otras acciones hasta que se haya tomado una decisión. Ante todo se busca que su contenido sea muy claro y nada complicado, que se entienda perfectamente la acción que se pretende realizar y las consecuencias de llevarla a cabo.

Se clasifican de acuerdo a la acción que se pretende realizar.

- Eliminar experto del listado: aparece cuando se pretende eliminar uno o varios expertos seleccionados en el listado de gestión de expertos. *"¿Quiere eliminar el experto seleccionado?"* y *"¿Quiere eliminar los expertos seleccionados?"*.

- Eliminar experto: cuando después de consultar un experto se pretende eliminarlo. *"¿Quiere eliminar el experto?"*.

- Eliminar problema del listado: aparece cuando se pretende eliminar uno o varios problemas seleccionados en el listado de gestión de problemas. *"¿Quiere eliminar el problema seleccionado?"* y *"¿Quiere eliminar los problemas seleccionados?"*.

- Eliminar problema: cuando después de consultar un problema se pretende eliminarlo. *"¿Quiere eliminar el problema?"*.

CAPÍTULO VII

Implementación y prueba del sistema

7.1. Implementación

La etapa de implementación del sistema es el proceso por el que se codifica la aplicación, siempre basándose y siguiendo la información y diagramas obtenidos durante la etapa de diseño, que son los que marcan las pautas y directrices a seguir a la hora de implementar.

Antes de comenzar con la implementación de una aplicación existen muchas decisiones a tomar, ya que existen multitud de arquitecturas, tecnologías y lenguajes de programación que marcarán el desarrollo del *software*. Algunas de estas decisiones vienen marcadas por la naturaleza y requerimientos del proyecto. Otras sin embargo, se dejan a elección del ingeniero, que a buen seguro toma las decisiones basándose en unos criterios que justifican su elección.

En este capítulo se pretende dar una visión de las tecnologías elegidas para la implementación del *software*, explicando tanto su funcionamiento como la motivación por la que se han elegido, así como de las herramientas necesarias para su desarrollo.

7.1.1. Otras tecnologías empleadas

Algunas de las arquitecturas y tecnologías más importantes sobre las que se fundamenta la aplicación quedaron reflejadas en el capítulo cuarto de este proyecto. Se detalló el funcionamiento y motivación de la arquitectura cliente-servidor y del patrón arquitectónico Modelo-Vista-Controlador, ambas seguidos en la implementación del *software*. También se explicó detalladamente el funcionamiento de los *frameworks* Struts 2 e Hibernate utilizados en el desarrollo.

Sin embargo, en el desarrollo de la aplicación se hace uso de otras tecnologías menos estructurales que están directamente relacionadas con la vista y su presentación.

Al tratarse de una plataforma web, resulta evidente que para su construcción se requiere el uso de páginas web. Las páginas web se escriben en el HyperText Markup Language (HTML), que consiste en un lenguaje de marcado que el navegador sabe cómo interpretar para mostrar las páginas. Se compone de un conjunto de etiquetas de marcas que se usan para describir las páginas web. HTML sólo permite crear páginas con contenido estático, por lo que para dotarlas de dinamismo e interactividad hay que combinar este lenguaje con otras tecnologías.

Una de estas tecnologías y el principal componente que compone la vista son las Java Server Pages (JSP), cuyo uso viene claramente marcado por el desarrollo de la aplicación web en Java. Se trata de una tecnología que permite crear páginas web con contenido dinámico. Son páginas HTML en las que se incluye procesamiento mediante el uso de *scripts* en código Java o a través de librerías de etiquetas. Se prefiere el uso de ésta últimas, ya que utilizan un formato idéntico al de las etiquetas de HTML, de forma que el código de la página es homo-

géneo y fácil de leer, en lugar de incluir código Java mezclado con HTML. Las páginas JSP se interpretan en el servidor, de forma que al final se envía una respuesta al cliente en HTML.

Otra tecnología que permite la creación de páginas web dinámicas e interactivas es JavaScript. Se trata de un lenguaje de programación interpretado que generalmente se ejecuta en el lado del cliente. La potencia de esta tecnología reside en que permite acceder a cualquier elemento de la página web a través del Document Object Model (DOM), con lo que se pueden añadir nuevos elementos o modificar y eliminar los existentes. Otra característica de JavaScript es que puede reaccionar a los eventos que inicia el usuario, de forma que es posible asociar código en JavaScript para un evento determinado que el usuario realiza sobre un elemento específico. También permite acceder a las propiedades del navegador y a los eventos del usuario. El código en JavaScript se puede insertar en las páginas de dos modos: dentro de las etiquetas *script* en la misma página o en un fichero externo que se enlaza desde la cabecera de la página web. Este último método suele ser el más usado.

Hasta el momento, cada vez que se envía una petición al servidor, éste devuelve como respuesta una página web completa en código HTML que el navegador interpreta. Sin embargo, hay ocasiones en las que una petición no requiere una página entera como respuesta, sino una pequeña parte de ella, de modo que se pueden actualizar secciones de una página sin necesidad de recargarla por completo. Para conseguirlo hay que recurrir al Asynchronous JavaScript And XML (AJAX), que consiste en una tecnología que permite el intercambio de información con el servidor de forma asíncrona y en segundo plano, para actualizar partes de una página web sin necesidad de recargarla por completo. Por medio de JavaScript, AJAX accede al objeto XMLHttpRequest del navegador, que es el encargado de realizar la comunicación asíncrona con el servidor y de recibir la respuesta en XML, aunque también puede ser en texto plano, HTML u otros formatos. Finalmente, a través de JavaScript y DOM se actualiza alguna zona de la página con la información recibida.

A la hora de crear páginas web es muy importante separar el contenido de las páginas de su presentación. Con esta división de responsabilidades es posible realizar cambios en la presentación sin necesidad de modificar el contenido y viceversa. Además tener ambas partes separadas facilita la modificación y el mantenimiento de las páginas web. Para definir el estilo y diseño de las páginas web se usan las hojas de estilo en cascada. El Cascade Style Sheets (CSS) es un lenguaje formal que permite definir la presentación de una página web separándola de su estructura interna. Los estilos se pueden aplicar a las páginas de tres modos: en el atributo *style* de las etiquetas HTML, dentro de las etiquetas *style* en la cabecera de la página o en un fichero externo que se enlaza desde la cabecera de la página web. Esta última forma suele ser la más utilizada.

7.1.2. Herramientas de desarrollo

Dentro de este apartado se describen todas aquellas herramientas que se han utilizado y han permitido el desarrollo de la aplicación que abarca este proyecto. Atendiendo a su naturaleza se clasifican en: aplicaciones y librerías.

- ***Aplicaciones***

Para el desarrollo de la aplicación se ha utilizado el entorno de desarrollo integrado Netbeans IDE en su versión 6.9.1 con soporte para desarrollar aplicaciones Java Enterprise Edition (Java EE). Netbeans incluye el Software Development Kit (SDK) de Java que es necesario para poder crear y ejecutar aplicaciones Java.

Para acceder a la aplicación, ésta debe desplegarse en un servidor de aplicaciones web. Se ha usado el servidor GlassFish 3.0.1, ya que viene incluido en la versión de Netbeans utilizada.

Para dar soporte a la persistencia se requiere el uso de un Sistema de Gestión de Base de Datos (SGBD). El utilizado ha sido Apache Derby 10.5.3.0, que de nuevo viene incluido por defecto en Netbeans.

Por último, para utilizar la aplicación es necesario un navegador web que siga los estándares de XHTML 1.0 y CSS 3, soportados por la mayoría de los navegadores actuales. Se ha usado principalmente Safari en su versión 5, aunque la aplicación también se ha utilizado en otros navegadores como Firefox 4, Google Chrome 12, Opera 11 e Internet Explorer 8, siendo éste último el que no cumple con todos los estándares de CSS 3 utilizados. También se han utilizado una serie de herramientas para desarrolladores que proporciona Safari, que permiten inspeccionar los elementos HTML y depurar código en JavaScript.

- ***Librerías***

Para poder hacer uso de los *frameworks* Struts 2 e Hibernate se han utilizado las librerías incluidas en la versión de Struts 2.2.3 e Hibernate 3.6.5.

Para la gestión del *pool* de conexiones de la base de datos se ha hecho uso de la librería c3p0 en su versión 0.9.1.2.

Finalmente, para facilitar y simplificar el uso de AJAX desde Struts 2 se ha utilizado el *plugin* struts2-jquery en su versión 3.0.2, que a su vez hace uso de la librería jQuery 1.5.2.

Para poder usar las librerías sólo hay que incluirlas en el apartado destinado para tal fin dentro del proyecto de Netbeans.

7.2. Verificación, validación y prueba

En esta fase del proceso de Ingeniería del *Software* se pretende garantizar que se ha construido un sistema de calidad que representa una revisión final de las especificaciones, del diseño y de la codificación [19]. Para ello se necesitan tanto técnicas que se centren en la calidad del producto, como técnicas que se centren en la calidad del proceso [27].

Durante y después del proceso de implementación, el programa que se está desarrollando debe ser comprobado para asegurar que satisface la especificación y funcionalidad que busca el usuario. La verificación y validación es el nombre dado a estos procesos de análisis y pruebas [25]. Tienen lugar durante todas las etapas del proceso del *software*. Comienzan con revisiones de los requerimientos y continúa con revisiones del diseño e inspecciones de código hasta la prueba del producto.

Aunque a veces tienden a confundirse, los términos verificación y validación no están dentro de la misma categoría [25, 27]:

- Verificación: es el proceso de asegurar que se ha construido el producto correctamente, eso es, que cumple con su especificación. Debería comprobarse que cumple con los requerimientos funcionales y no funcionales.

- Validación: es el proceso de asegurar que se ha construido el producto correcto, es decir, que el producto cumple su propósito. Va más allá de la especificación, hay que demostrar que el producto hace lo que quiere el usuario.

La manera fundamental de asegurar ambas propiedades es probando el sistema y buscando problemas de cualquier tipo.

Dentro del proceso de verificación y validación, existen dos aproximaciones complementarias para el análisis y comprobación de los sistemas [25]:

1. Las inspecciones de *software* analizan y comprueban los requerimientos, diagramas de análisis, diseño y el código fuente. Pueden usarse en todas las etapas del proceso. Se consideran técnicas estáticas, ya que no requieren ejecutar el programa en un ordenador. Este tipo de técnicas se han venido realizando de un modo informal y no documentado, debido a su extensión, durante el desarrollo de todo el proyecto, por lo que no se va a hacer más referencias a ellas.

2. Las pruebas del *software* implican ejecutar una implementación del programa con datos de prueba. Se examinan las salidas del *software* y su entorno operacional para probar que funciona tal y como se requiere. Las pruebas se consideran técnicas dinámicas de verificación y validación. Este tipo de técnicas son en las que se va a centrar esta fase del proceso.

Una prueba se define como el proceso de ejecución de un programa con la intención de descubrir errores [19]. Por lo tanto, se considera que un caso de prueba es bueno si tiene una alta probabilidad de mostrar errores no descubiertos hasta entonces. Se considera que una prueba tiene éxito si descubre un error no detectado hasta ahora [19, 27]. Las pruebas tienen tres objetivos principales [27]:

- Ayudar a encontrar errores.

- Convencer al usuario de que no hay errores.

- Proporcionar información que ayudará en la evolución del sistema.

Es fácil hacer que un sistema pase todas las pruebas; simplemente hay que utilizar un conjunto de pruebas inadecuado, lo verdaderamente difícil es seleccionar las pruebas adecuadas que van a permitir eliminar los errores. Aunque las pruebas se pueden clasificar en múltiples categorías, a grandes rasgos, se pueden distinguir pruebas de caja blanca y pruebas de caja negra:

- Pruebas de caja blanca: están destinadas a comprobar el funcionamiento interno del sistema. Se planifican ejecuciones de los métodos de la aplicación, buscando clases representativas de los valores que pueden adoptar las distintas variables, de manera que se exploren todos los caminos posibles que puede llevar a cabo una ejecución.

- Pruebas de caja negra: conciben el sistema como un elemento cerrado a cuyo funcionamiento interno no se tiene acceso. Se estudian las entradas y salidas de éste, comprobando que las respuestas dadas a las entradas introducidas son correctas. Se centran en qué hace el sistema y no cómo lo hace.

Debido a la extensión del proyecto, tratar de llevar a cabo pruebas de caja blanca sería un proceso demasiado largo y tedioso que se sale del ámbito del mismo, además de poco útil puesto que durante la implementación se han ido realizando las pruebas oportunas para garantizar la calidad del código. Por lo tanto, se va a centrar la atención en diseñar y utilizar las pruebas de caja negra para comprobar que el *software* cumple con sus objetivos y está libre de errores.

7.2.1. Pruebas de caja negra

Tal y como se explicó en el epígrafe anterior, en este apartado se pretende comprobar el comportamiento del *software* ante una serie de entradas concretas. Este tipo de pruebas se centran en los requisitos funcionales, es decir, las pruebas de caja negra permiten obtener conjuntos de condiciones de entrada que ejerciten todos los requerimientos funcionales del programa.

✦ ***Acceso al sistema***

El sistema debe proporcionar un método de autentificación que permita identificar al usuario que trata de acceder al mismo, para lo cual debe proporcionar un nombre de usuario y una contraseña. Sólo se puede acceder al sistema si el usuario es el administrador o un experto existente en el mismo y proporciona unos credenciales válidos.

Prueba realizada	Resultado obtenido
Se introduce un nombre de usuario y contraseña válidos.	Se permite su acceso al sistema.
Se introduce un nombre de usuario o una contraseña inválidos.	Se informa de que el nombre de usuario o la contraseña son incorrectos.
No se introduce el nombre de usuario.	Se informa de que falta el nombre de usuario.
No se introduce la contraseña.	Se notifica que falta la contraseña.
Se trata de acceder al sistema pero los datos del administrador no están en la base datos.	Se notifica que el servicio no está disponible.
Se trata de acceder al sistema pero existe más de un administrador en la base de datos.	Se notifica que el servicio no está disponible.
Se trata de acceder al sistema pero se produce algún problema durante la comprobación de la contraseña.	Se notifica que el servicio no está disponible.

- ***Crear experto***

El sistema debe permitir crear nuevos expertos. A esta funcionalidad sólo puede acceder el administrador.

Prueba realizada	Resultado obtenido
Se introducen los datos del experto correctamente.	Se crea el experto.
No se introduce el nombre del experto.	Se informa de que falta el nombre del experto.
Se introduce una descripción superior a 200 caracteres.	Se notifica que la descripción no puede superar esa longitud.
No se inserta el nombre de usuario.	Se notifica que falta el nombre de usuario.
Se introduce un nombre de usuario que pertenece a otro experto.	Se informa de que el nombre de usuario está ocupado.

Prueba realizada	Resultado obtenido
No se introduce una contraseña.	Se informa de que falta la contraseña.
No se repite la contraseña.	Se notifica que no se ha repetido la contraseña.
Se inserta una contraseña y su repetición pero no coinciden.	Se informa de que las contraseñas no coinciden.
Se trata de crear el experto pero se produce algún error durante la encriptación de la contraseña.	Se notifica que se ha producido un error al crear el experto.

- ***Modificar experto***

Mediante esta funcionalidad se permite cambiar los datos asociados a un experto. Únicamente puede hacer uso de ella el administrador.

Prueba realizada	Resultado obtenido
Se modifican los datos del experto que se pretendan cambiar correctamente.	Se actualizan los datos del experto.
Se deja vacío el nombre.	Se informa de que falta el nombre del experto.
Se introduce una descripción superior a 200 caracteres.	Se notifica que la descripción no puede superar esa longitud.
Se deja vacío el nombre de usuario.	Se notifica que falta el nombre de usuario.
Se modifica el nombre de usuario por otro que pertenece a otro experto.	Se informa de que el nombre de usuario está ocupado.
No se introduce una contraseña pero sí su repetición.	Se informa de que falta la contraseña.
Se introduce una contraseña pero no su repetición.	Se notifica que no se ha repetido la contraseña.
Se inserta una contraseña y su repetición pero no coinciden.	Se informa de que las contraseñas no coinciden.

Prueba realizada	Resultado obtenido
Se trata de modificar el experto pero se produce algún error durante la encriptación de la contraseña.	Se notifica que se ha producido un error al modificar el experto.
Se trata de modificar un experto que ya no existe en el sistema. Se debe a un uso inapropiado del historial.	Se informa de que el experto ya no existe.

- ***Eliminar experto***

Al igual que el administrador puede crear y modificar expertos, también debe poder eliminarlos.

Prueba realizada	Resultado obtenido
Se seleccionan del listado los expertos que se pretenden eliminar.	Se eliminan los expertos seleccionados.
Se elimina el experto que se está mostrando.	Se elimina el experto.
Se trata de eliminar el experto que se está mostrando pero éste ya no existe en el sistema. Se debe a un uso inapropiado del historial.	Se informa de que el experto ya no existe.

- ***Crear y modificar problema***

El sistema debe permitir crear nuevos problemas y su posterior modificación. A éstas dos funcionalidades sólo pueden acceder los expertos. Los expertos únicamente pueden crear y modificar sus propios problemas.

En esta ocasión, se ha decidido unir las pruebas realizadas durante la creación y la modificación de un problema puesto que son prácticamente las mismas, a excepción de algunas variaciones las cuales se indican en las tablas. Asimismo, se ha tomado la decisión de dividir las pruebas en tres tablas, de acuerdo a si son comunes a los dos tipos problemas o bien son específicas de cada uno.

PROMETHEE y AHP	
Prueba realizada	Resultado obtenido
Se introducen los datos del problema correctamente.	Se resuelve el problema y se muestran sus datos junto con los resultados obtenidos.
Se realiza un análisis de sensibilidad.	Se muestran los datos del problema para que se puedan modificar.
Se guarda el problema resuelto (crear problema).	Se almacena el problema.
Se actualiza el problema resuelto (modificar problema).	Se actualiza el problema.
Se selecciona el tipo de problema PROMETHEE (crear problema).	Se muestran los datos específicos por rellenar asociados a PROMETHEE.
Se selecciona el tipo de problema AHP (crear problema).	Se muestran los datos específicos por rellenar asociados a AHP.
Se añade una alternativa.	Se agrega una nueva alternativa con su nombre vacío.
Se quitan las alternativas seleccionadas.	Se eliminan las alternativas seleccionadas. Siempre van a quedar un mínimo de dos.
No se introduce el nombre del problema.	Se informa de que falta el nombre del problema.
No se inserta el objetivo del problema.	Se notifica que falta el objetivo del problema.
No se introducen los nombres de las alternativas.	Se notifica que falta el nombre de las alternativas.
Se trata de guardar de nuevo un problema que ha sido recién almacenado. Se debe a un uso inapropiado del historial.	Se informa de que el problema ya ha sido guardado.

PROMETHEE	
Prueba realizada	Resultado obtenido
Se añade un criterio.	Se agrega un nuevo criterio con sus datos vacíos.
Se quitan los criterios seleccionados.	Se eliminan los criterios seleccionados. Siempre van a quedar un mínimo de dos criterios.
No se introducen los datos de los criterios.	Se notifica que hay que introducir los datos de los criterios correctamente.
Se introducen pesos de los criterios con formato inválido.	Se notifica que hay que introducir los datos de los criterios correctamente.
La suma de los pesos de los criterios es distinta de uno.	Se informa de que la suma de los pesos debe ser uno.
Se introducen parámetros de los criterios con formato inválido.	Se notifica que hay que introducir los datos de los criterios correctamente.
No se introducen las valoraciones.	Se informa de que hay que introducir las valoraciones correctamente.
Se introducen valoraciones con formato inválido.	Se informa de que hay que introducir las valoraciones correctamente.

AHP	
Prueba realizada	Resultado obtenido
Se añade un criterio.	Se agrega un nuevo criterio con su nombre vacío.
Se selecciona un criterio que carece de subcriterios y se añade uno.	Se agregan al criterio seleccionado dos subcriterios con sus nombres vacíos.
Se selecciona un criterio que posee subcriterios y se añade uno.	Se agrega al criterio seleccionado un subcriterio con su nombre vacío.
Se quitan los criterios y subcriterios seleccionados.	Se eliminan los criterios y subcriterios seleccionados. Siempre van a quedar un mínimo de dos criterios.
Se generan las comparaciones pareadas entre criterios y alternativas.	Se muestran las comparativas en función de las alternativas y la estructura de criterios definidas.
Se selecciona una comparación entre criterios distinta de la ya seleccionada.	Se muestra la matriz con los datos de la comparativa.
Se selecciona una comparación entre alternativas distinta de la ya seleccionada.	Se muestra la matriz con los datos de la comparativa.
No se introducen los nombres de los criterios y subcriterios.	Se informa de que faltan los nombres de los criterios.

- ***Eliminar problema***

Tal y como ocurre con los expertos, el sistema debe dar soporte a la eliminación de problemas. Tanto el administrador como los expertos pueden eliminar problemas pero con ciertas salvedades. El administrador puede eliminar cualquier problema del sistema, mientras que un experto sólo puede eliminar los problema que le pertenezcan.

Prueba realizada	Resultado obtenido
Se seleccionan del listado los problemas que se pretenden eliminar.	Se eliminan los problemas seleccionados.
Se elimina el problema que se está consultando.	Se elimina el problema.

Prueba realizada	Resultado obtenido
Se trata de eliminar el problema que se está consultando pero éste ya no existe en el sistema. Se debe a un uso inapropiado del historial.	Se informa de que el problema ya no existe.

- ***Consultar problema***

El sistema debe permitir que el administrador y los expertos puedan consultar problemas. Análogamente a la eliminación, el administrador puede consultar cualquier problema, mientras que un experto únicamente puede consultar los suyos propios.

Prueba realizada	Resultado obtenido
Se selecciona del listado el problema que se quiere consultar.	Se muestran los datos y resultados del problema.
Se selecciona un índice de preferencia distinto del ya seleccionado (PROMETHEE).	Se muestra la comparativa entre alternativas, con los datos a partir de los que se calcula el índice.
Se selecciona una comparación entre criterios distinta de la ya seleccionada (AHP).	Se muestra la matriz con los datos de la comparativa.
Se selecciona una comparación entre alternativas distinta de la ya seleccionada (AHP).	Se muestra la matriz con los datos de la comparativa.
Se trata de consultar un problema que ya no existe en el sistema. Se debe a un uso inapropiado del historial.	Se informa de que el problema ya no existe.

- ***Modificar contraseña***

El sistema debe soportar que el administrador y los expertos puedan modificar su contraseña de acceso al sistema cuando lo consideren oportuno, siempre que proporcionen la contraseña actual.

Prueba realizada	Resultado obtenido
Se introducen los datos necesarios para modificar la contraseña del usuario correctamente.	Se modifica la contraseña del usuario.
No se inserta la contraseña actual.	Se informa de que falta la contraseña actual.
Se introduce una contraseña actual errónea.	Se notifica que la contraseña actual no coincide.
No se introduce la nueva contraseña.	Se notifica que falta la nueva contraseña.
No se repite la nueva contraseña.	Se informa de que no se ha repetido la nueva contraseña.
Se inserta una nueva contraseña y su repetición pero no coinciden.	Se informa de que las contraseñas no coinciden.
Se trata de modificar la contraseña pero los datos del administrador no están en la base de datos (Administrador).	Se notifica que se ha producido un error al comprobar la contraseña actual.
Se trata de modificar la contraseña pero existe más de un administrador en la base de datos (Administrador).	Se notifica que se ha producido un error al comprobar la contraseña actual.
Se trata de modificar la contraseña pero el experto ya no existe en el sistema (Experto).	Se notifica que se ha producido un error al comprobar la contraseña actual.
Se trata de modificar la contraseña pero se produce algún problema durante su encriptación.	Se notifica que se ha producido un error al modificar la contraseña.

CAPÍTULO VIII

Conclusiones

8.1. Conclusiones

Desde que apareciera la primera computadora hace décadas, los ordenadores han evolucionado mucho en tecnología y diseño. Con el paso del tiempo, los ordenadores han ido aumentando sus capacidades de cómputo, almacenamiento e intercomunicación, al mismo tiempo que se ha reducido su tamaño y abaratado su coste, permitiendo ampliar sus funcionalidades. Sin embargo, hay una directriz que ha permanecido inalterable durante todo este tiempo, y es que las personas hacen uso de los ordenadores para satisfacer sus necesidades y alcanzar sus objetivos de forma más eficaz y eficiente.

Por ello, el principal propósito del proyecto consistía en hacer uso de las múltiples capacidades de los ordenadores actuales para permitir resolver problemas de toma de decisión multicriterio de forma rápida, sencilla e intuitiva, por medio de la aplicación de dos modelos matemáticos ampliamente conocidos y utilizados como son PROMETHEE y AHP. Se buscaba eliminar la necesidad de realizar tediosos cálculos de forma manual y reducir por tanto la posibilidad de cometer errores. Pero además, no sólo se pretendía crear un sistema que únicamente resolviera este tipo de problemas, sino que también se buscaba poder gestionarlos y almacenarlos. Por último, en un mundo donde en gran parte gracias a la la proliferación de Internet los ordenadores están cada vez más interconectados, se hacía necesario aprovechar esta característica para permitir un acceso a la aplicación desde cualquier punto con acceso a Internet.

Para conseguirlo, se optó por desarrollar una plataforma web que permitiera resolver problemas de toma de decisión multicriterio aplicando los modelos PROMETHEE y AHP, que además permitiera gestionar los problemas por medio de expertos. De hecho, esta última característica hace la aplicación perfecta para usarla con fines docentes.

Tras una etapa de localización y estudio de información bibliográfica acerca de este tipo de problemas y de los modelos de resolución, se llevó a cabo un serio trabajo de Ingeniería del *Software*, que por medio de las etapas de análisis, diseño e implementación, dio como resultado una aplicación que logra una simulación representativa y fiable de los modelos de resolución antes mencionados. Finalmente se sometió al *software* a una etapa de verificación y validación en la que se constató el correcto y buen funcionamiento del mismo.

El desarrollo de la plataforma se llevó a cabo siguiendo técnicas basadas en patrones de diseño y que mediante el uso de ciertas tecnologías a la hora de la implementación permitieron crear una aplicación de calidad.

Durante la fase de diseño intervinieron diversos patrones, de entre los que destaca el patrón arquitectónico Modelo-Vista-Controlador, que permitieron crear una aplicación modular, con una alta cohesión y bajo acomplamiento, características que aportan una alta posibilidad de reutilización de sus componentes y una aplicación fácil de modificar y mantener. También se dedicó especial atención a la interfaz, tratando de maximizar la usabilidad y hacerla intuitiva, para mejorar en lo posible la experiencia del usuario.

El paso de la etapa de diseño a la implementación se tradujo en la utilización de algunas tecnologías muy útiles que permitieron cumplir con los objetivos planteados. En primer lugar, la plataforma se desarrolló siguiendo la arquitectura cliente-servidor sobre la que se fundamenta Internet, lo que hace posible acceder a la aplicación desde cualquier sitio donde exista un ordenador con un navegador web y conexión a Internet. Después, el uso del *framework* Struts 2 facilitó la aplicación del patrón MVC, gracias al cual es posible realizar cambios sobre la vista sin necesidad de cambiar el modelo y viceversa. Finalmente, el *framework* Hibernate mitigó la tediosa tarea del mapeo objeto-relacional, además de aportar independencia del sistema de gestión de base de datos utilizado, permitiendo cambiarlo cuando se desee únicamente modificando su fichero de configuración.

Desde un punto de vista personal, la realización de este proyecto fin de carrera me ha permitido adentrarme y conocer más extensamente el ámbito de la toma de decisión multicriterio, del mismo modo que he adquirido competencias teóricas y prácticas en la resolución de problemas por medio de los modelos PROMETHEE y AHP. El desarrollo de la aplicación siguiendo el proceso de Ingeniería del *Software* me ha permitido afianzar y ampliar muchos de los conocimientos adquiridos durante el transcurso de la Ingeniería Técnica y Superior, al mismo tiempo que he aprendido otros nuevos. El aprendizaje y la experiencia en el uso de los *frameworks* Struts 2 e Hibernate ha resultado ser muy beneficioso para mi formación, puesto que son dos tecnologías de desarrollo ampliamente utilizadas en la actualidad en la creación de aplicaciones web Java de ámbito empresarial.

8.2. Ampliaciones del proyecto

Al tratarse de un proyecto fin de carrera y disponer de un tiempo de desarrollo limitado, hay ciertas características cuya inclusión en la aplicación hubieran resultado útiles e interesantes, por lo que se dejan propuestas como futuras ampliaciones del proyecto.

- La aplicación se puede escalar fácilmente para permitir otros modelos de resolución de problemas.

- Actualmente la aplicación resuelve problemas con información precisa, luego se podría ampliar su funcionalidad permitiendo expresar las valoraciones de los expertos con valores intervalares, términos lingüísticos y conjuntos difusos.

- Aunque la aplicación se ha construido con soporte para la internacionalización, únicamente está en español, por lo que se puede traducir a otros idiomas fácilmente con sólo incluir un fichero de texto con las traducciones.

- Durante el desarrollo de la aplicación se ha sido especialmente cuidadoso en todos los temas relacionados con la seguridad. Sin embargo, es interesante incluir algún método que permita establecer una comunicación segura entre el cliente y servidor a través del protocolo HTTPS (HyperText Transfer Protocol Secure). De esta manera

se garantiza la identidad del emisor y receptor al mismo tiempo que se encriptan los datos que se intercambian durante la comunicación.

ANEXO I

Manual de instalación

Este anexo recoge el manual con las indicaciones necesarias para la instalación de la aplicación y del resto de infraestructuras requeridas para su funcionamiento. Para su ejecución es necesario un servidor de aplicaciones y una base de datos, de forma que el manual de instalación está dividido en estas dos partes.

En la actualidad hay multitud de servidores de aplicaciones y de bases de datos, por lo que la instalación de la aplicación varía en función de la elección realizada. Este anexo recoge algunas directrices generales para una instalación en cualquier servidor y base de datos, a la vez que también incluye los pasos detallados para desplegar la aplicación en un servidor y base de datos concretos.

1.1. Servidor de aplicaciones

En primer lugar, para instalar el *software* es necesario un servidor donde se despliegue la aplicación y así poder acceder a ella. A la hora de elegirlo hay que tener en cuenta algunos detalles, como que la aplicación se ha desarrollado en Java, por lo que el servidor debe soportar este tipo de aplicaciones. También debe tener soporte para las librerías Java Persitence API (JPA) y JavaServer Pages Standard Tag Library (JSTL) versión 1.2. Por si se eligiera un servidor que no tuviera estas librerías, se ha decidido incluirlas en el directorio ******** del CD.

De forma general, para instalar la aplicación en el servidor únicamente hay que incluir la carpeta *TomaDecisionesMulticriterio* del directorio ****** del CD en el servidor, proceso que dependerá de este último. Esta carpeta es el resultado de descomprimir el fichero *TomaDecisionesMulticriterio.war*, que también se proporciona en el CD dentro del mismo directorio. Si el servidor no trajera soporte para las mencionadas librerías JPA y JSTL 1.2, como es el caso de Tomcat, hay que incluirlas en el directorio WEB_INF/lib de la aplicación. Se desconoce que otro tipo de carencias pueden presentar esta clase de servidores.

El servidor de aplicaciones elegido es GlassFish Server 3.1 Open Source Edition, que cuenta con todos los recursos necesarios para soportar la ejecución del *software*. Dentro de la edición *open source*, GlassFish cuenta con dos distribuciones: Full Platform y Web Profile, con la diferencia de que esta última tiene algunas funcionalidades menos. Se ha elegido la distribución Web Profile ya que permite ejecutar la aplicación sin problemas y además es más ligera. A su vez, esta distribución cuenta con tres tipos de instaladores, en función de la plataforma donde se vaya a instalar el servidor. Se ha seleccionado la versión ZIP, ya que es multiplataforma y su instalación es muy poco intrusiva. La versión ZIP de GlassFish se encuentra alojada en el directorio ***** del CD. En la página web de GlassFish (http://glassfish.java.net/) se pueden adquirir diferentes versiones del servidor para plataformas concretas, además de incorporar toda la documentación sobre la instalación y uso del producto.

La instalación se ha realizado en la misma plataforma donde se ha desarrollado todo el proyecto, es decir Mac OS X, que se trata de un sistema UNIX. Por tanto, el resto del epígrafe abarca los pasos para instalar GlassFish y desplegar la aplicación en sistemas de este tipo, aunque la única diferencia con plataformas Windows radica en los comandos que se usan en el terminal, ya que la barra que indica el directorio en UNIX (/) es a la inversa en Windows (\).

Para instalar GlassFish sólo hay que descomprimir el fichero ZIP en la localización donde se quiere que resida el servidor. Una vez instalado, para iniciar y detener GlassFish se usan los siguientes comandos en un terminal:

- Iniciar servidor: <directorio_instalación_glassfish>/bin/asadmin start-domain

- Detener servidor: <directorio_instalación_glassfish>/bin/asadmin stop-domain

Tras iniciar el servidor, el despliegue de la aplicación se puede realizar de dos modos: de forma gráfica o por comandos en un terminal.

Para el método gráfico en primer lugar hay que iniciar la consola de administración, a la que se accede desde un navegador web por medio de la URL: *http://<dirección_del_servidor>:4848*. Después, hay que ir a la sección *Aplicaciones* del menú lateral y pulsar en *Implementar*, como muestra la Figura I.1.

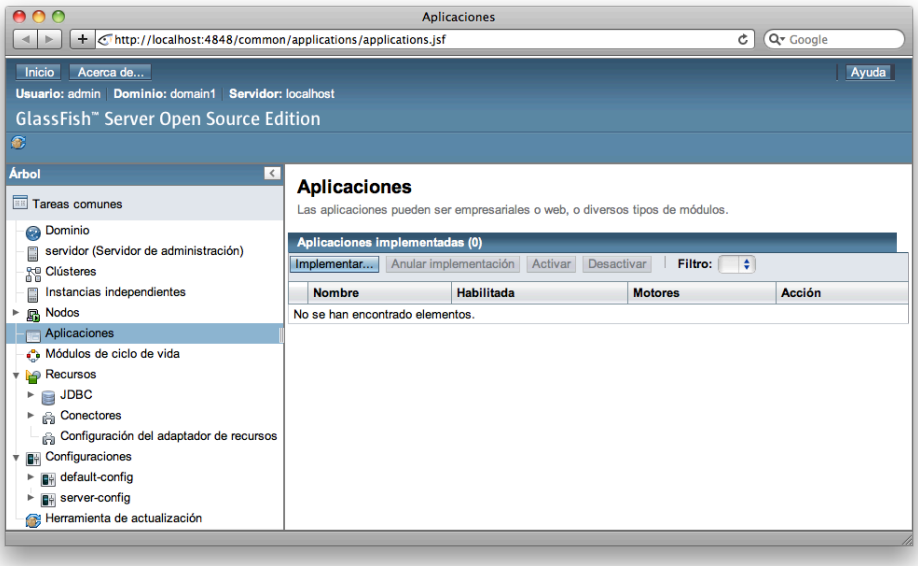


Figura I.1. Lista de aplicaciones desplegadas.

Luego se elige la opción *Archivo empaquetado local o directorio accesible desde GlassFish Server*, se selecciona la carpeta *TomaDecisionesMulticriterio* del directorio ***** del CD del proyecto y se pulsa *Aceptar*. Puede interesar seleccionar en el campo desplegable *Tipo* la opción *Aplicación Web*, ya que permite personalizar algunos parámetros del despliegue.

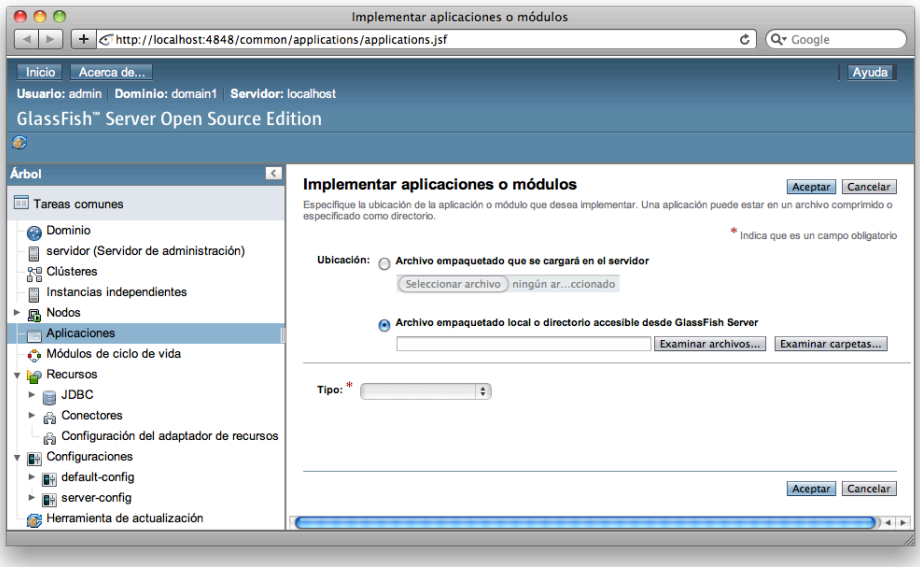


Figura I.2. Implementación de aplicaciones.

Finalmente la aplicación queda desplegada y habilitada en el servidor.

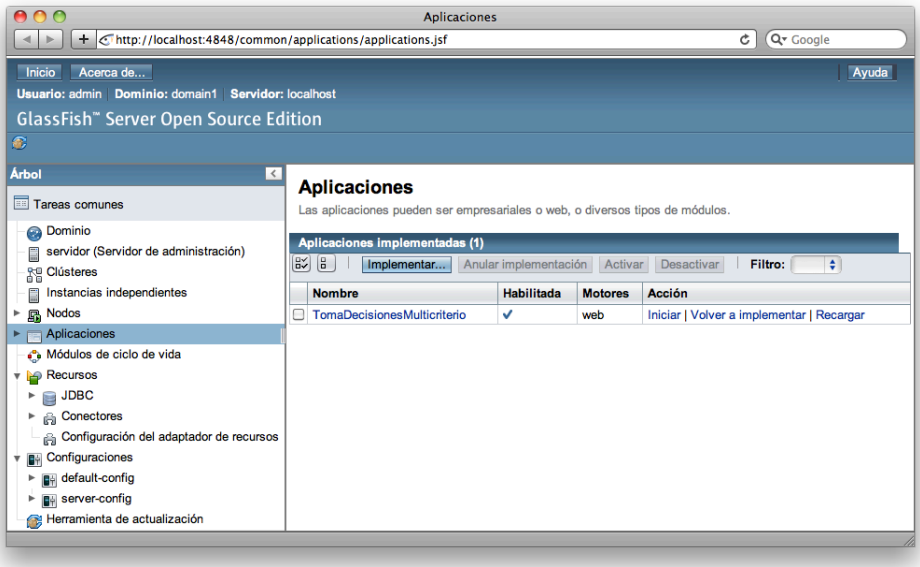


Figura I.3. Aplicación desplegada.

Si se decidiera anular el despliegue de la aplicación bastaría con seleccionarla del listado y pulsar en *Anular Implementación*.

Para desplegar y anular el despliegue de la aplicación a través del terminal se usan los siguientes comandos:

- Desplegar aplicación: <directorio_instalación_glassfish>/bin/asadmin deploy <directorio_aplicación>

- Anular despliegue aplicación: <directorio_instalación_glassfish>/bin/asadmin undeploy <nombre_aplicación>

1.2. Base de datos

Finalmente, para instalar el *software* es necesario un sistema de gestión de bases de datos (SGBD) que permita hacer persistentes los datos que gestiona la aplicación. El uso de Hibernate proporciona una independencia total del SGBD, de forma que gracias a este *framework* se puede elegir la base de datos que más se adapte a nuestros intereses o cambiarla cuando se desee sin necesidad de modificar el código de la aplicación, siempre que la base de datos esté incluida dentro de la amplia variedad que soporta Hibernate.

Una vez hecha la elección, hay que tener en cuenta algunos detalles. En función del SGBD elegido hay que utilizar un *driver* de conexión concreto para realizar la comunicación con la base de datos, que debe incluirse en el directorio WEB_INF/lib de la aplicación. Además, el código SQL que se incluye para crear y eliminar la base de datos, o para crear y eliminar las tablas, puede que necesite alterarse para adaptarse al dialecto concreto que use el SGBD, aunque en la mayoría de los casos no será necesario. Finalmente, según el SGBD elegido, la base de datos creada y el usuario que tiene los permisos adecuados para gestionarla, hay que modificar algunos parámetros en el fichero de configuración de Hibernate.

De forma general, para terminar con la instalación de la aplicación, en primer lugar hay que acceder al SGBD para crear la base de datos, por medio del *script Crear base de datos.sql* que se encuentra en el directorio ***** del CD del proyecto. Después, es necesario crear las tablas de la base de datos a través del *script Crear tablas.sql* localizado en el mismo directorio. Finalmente, hay que cambiar cinco parámetros en el fichero de configuración de Hibernate, hibernate.cfg.xml, localizado en el directorio WEB_INF/classes de la aplicación. Estos parámetros se recogen en la Figura I.4 y son: el driver de conexión con la base de datos, la URL a través de la que se accede, el nombre de usuario y contraseña del usuario que se va a utilizar y el dialecto de SQL que debe usar Hibernate para comunicarse con ella.

```
<!-- Configuración de la conexión con la base de datos -->
<property name="connection.driver_class"></property>
<property name="connection.url"></property>
<property name="connection.username"></property>
<property name="connection.password"></property>

<!-- Dialecto SQL -->
<property name="dialect"></property>
```

Figura I.4. Parámetros de configuración de Hibernate.

El sistema de gestión de base de datos elegido es MySQL Community Server 5.5.13, puesto que es uno de los SGBD gratuitos más extendidos y utilizados. Al igual que ocurre

con el servidor de aplicaciones, se ha instalado sobre la plataforma Mac OS X, por lo que los pasos que se muestran a continuación recogen la instalación de MySQL y la creación de la base de datos para este sistema operativo. Dentro de la versión de MySQL para Mac OS X existen diferentes versiones, siendo la que más se adapta a la plataforma de desarrollo la versión DMG para Mac OS X 10.6 con arquitectura de 64 bits, que se puede encontrar en el directorio ***** del CD del proyecto. En la página web de MySQL (http://www.mysql.com/) se puede encontrar el resto de versiones para otras plataformas y arquitecturas.

Para instalar MySQL hay que montar el fichero DMG y ejecutar el archivo mysql-5.5.13-osx10.6-x86_64.pkg. Una vez finaliza la instalación, MySQL puede iniciarse y detenerse con los siguiente comandos en un terminal:

- Iniciar MySQL: sudo /usr/local/mysql/bin/mysqld_safe

- Detener MySQL: sudo /usr/local/mysql/bin/mysqladmin -u root -p shutdown

Tras iniciar MySQL, hay que iniciar una sesión para crear la base de datos. Por defecto, MySQL trae un superusuario *root* que carece de contraseña y posee todos los privilegios. Para este ejemplo concreto, al tratarse de una demostración, se va a hacer uso de este usuario *root*, aunque para una utilización en producción de la aplicación se recomienda establecer una contraseña a este usuario. Además, también es preferible crear un nuevo usuario para acceder a la base de datos, pero no hay que olvidar darle permisos de lectura y escritura sobre ella.

Para iniciar una sesión en MySQL y posteriormente cerrarla se utilizan dos comandos:

- Iniciar sesión: sudo /usr/local/mysql/bin/mysql --default-character-set=utf8 -u root -p

- Cerrar sesión: exit

Para acceder con un usuario diferente sólo hay que cambiar *root* por el nombre del otro usuario. Una vez se inicia sesión, se crea la base de datos y sus tablas ejecutando los *scripts* *Crear base de datos.sql* y *Crear tablas.sql* del directorio ****** del CD, por medio del comando *source* de MySQL, que permiter cargar ficheros con sentencias SQL:

- Cargar fichero: source <directorio_fichero>/<nombre_fichero.sql>

Dentro del mismo directorio se proporcionan dos *scripts* más, *Eliminar base de datos.sql* y *Eliminar tablas.sql*, para eliminar la base de datos y sus tablas respectivamente.

Finalmente para concluir, habría que incluir el driver de conexión con MySQL en la aplicación y cambiar los parámetros de configuración de Hibernate para ajustarlos a MySQL. Sin embargo, la versión de la aplicación que se incluye en el directorio ***** del CD ya viene con el driver incorporado y los parámetros modificados para facilitar la instalación. Aún así,

se ha considerado interesante incluir la Figura I.5 para mostrar como se detallan los parámetros de configuración de Hibernate para MySQL.

```
<!-- Configuración de la conexión con la base de datos -->
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="connection.url">jdbc:mysql://localhost/tdmc</property>
<property name="connection.username">root</property>
<property name="connection.password"></property>

<!-- Dialecto SQL -->
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
```

Figura I.5. Parámetros de configuración de Hibernate para MySQL.

ANEXO II

Manual de usuario

Dentro de este anexo se presenta una guía de usuario que explica el funcionamiento de la aplicación desarrollada en el proyecto y cómo hacer uso de las funcionalidades que presenta.

Para acceder a la aplicación hay que iniciar un navegador web e introducir en su barra de direcciones la siguiente URL:

http://<dirección_del _servidor>/TomaDecisionesMulticriterio

Una vez introducida la dirección aparece la pantalla de acceso al sistema, que permite la identificación del usuario a través de un nombre de usuario y una contraseña.

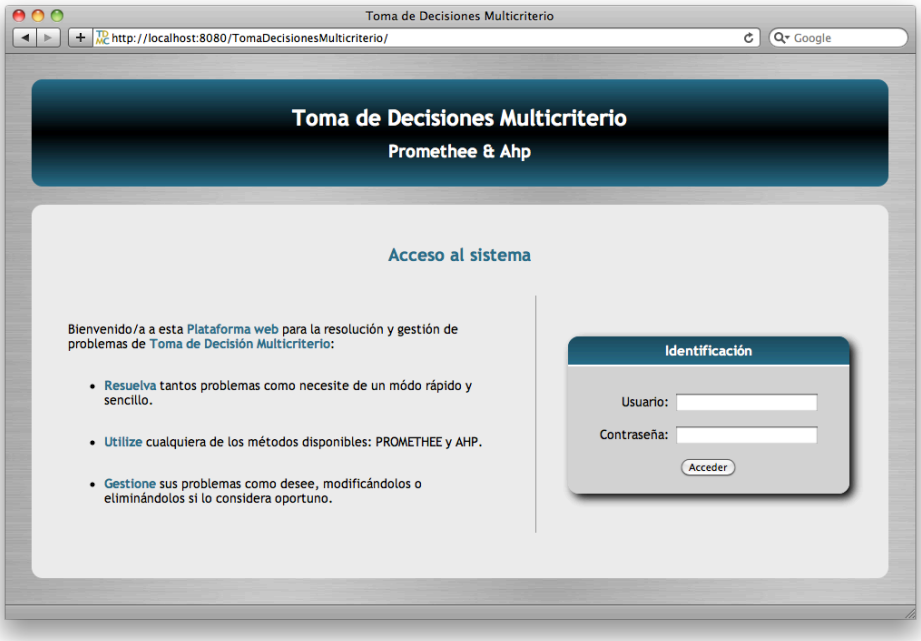


Figura II.1. Acceso al sistema.

Como la aplicación puede ser utilizada por dos tipos de usuarios, el manual se ha dividido en dos partes en función de ellos. La primera parte hace referencia al administrador y la segunda al experto.

2.1. Administrador

Los datos por defecto para la identificación del administrador son:

- Usuario: administrador.
- Contraseña: administrador.

La contraseña puede modificarse una vez se accede a la aplicación. Se recomienda por motivos de seguridad cambiar la contraseña que viene por defecto.

Al identificarse el usuario como administrador aparece su página principal, que es la de gestionar expertos. Todas las páginas cuentan en su parte superior con un menú constante que el administrador usa para navegar por las distintas secciones de la aplicación. Para salir de la aplicación basta con seleccionar la opción *Salir*.

2.1.1. Gestionar expertos

A esta sección también se puede acceder a través de la opción *Expertos* del menú. Dentro de este apartado aparece un listado con los expertos almacenados en el sistema y se pueden realizar algunas tareas de gestión sobre ellos, como crear expertos, modificarlos o eliminarlos.

Figura II.2. Gestionar expertos.

2.1.1.1. Crear experto

Al pulsar el botón *Crear experto* se da paso a la página que recoge sus datos. Éstos son su nombre, una descripción opcional, su nombre de usuario y contraseña. Tras rellenar el formulario se pulsa *Crear*.

Figura II.3. Crear experto.

2.1.1.2. Modificar experto

Para modificar un experto hay que pulsar en su nombre de usuario para que se muestre una página con sus datos actuales. Una vez modificados los datos que se consideran pertinentes basta con pulsar *Actualizar*.

Figura II.4. Modificar experto.

2.1.1.3. Eliminar experto

Hay dos formas de eliminar un experto. Una manera es pulsando sobre su nombre de usuario, del mismo modo que si se fuera a modificar, por lo que aparece la pantalla de la Figura II.4, pero en esta ocasión se pulsa *Eliminar*. Antes de eliminarlo es necesario confirmar la acción.

Figura II.5. Confirmación eliminar experto.

El otro modo permite eliminar más de un experto a la vez y se realiza desde la sección de gestionar expertos. Primero se seleccionan del listado los expertos que se quieren eliminar y después se pulsa *Eliminar experto*.

Figura II.6. Eliminar expertos.

De nuevo aparece un cuadro de diálogo modal para confirmar la acción.

Figura II.7. Confirmación eliminar expertos.

2.1.2. Gestionar problemas

El acceso al apartado de gestión de problemas se realiza a través de la opción *Problemas* del menú. Esta sección muestra un listado de los problemas que poseen los experto del sistema y en ella se puede consultar el problema de un experto o bien eliminarlo, de acuerdo a los requerimientos.

Figura II.8. Gestionar problemas (Administrador).

2.1.2.1. Consultar problema

Cuando se quiere consultar uno de los problemas del listado sólo hay que pulsar sobre su nombre. A continuación se muestra la consulta de los dos tipos de problemas que existen en la aplicación.

- ***PROMETHEE***

Al pulsar sobre el nombre del problema aparecen sus datos y resultados, tal y como refleja la Figura II.9. Si se quiere mostrar una comparativa entre dos alternativas sólo hay que pulsar sobre alguno de los índices de preferencia que forman parte del resultado obtenido, como se puede observar en la Figura II.10. En la comparativa aparecen algunos de los cálculos realizados para obtener dicho índice.

- ***AHP***

Los datos y resultados del problema que se consulta se muestran como en la Figura II.11. Para navegar por las comparaciones entre criterios y alternativas basta con pulsar sobre su nombre, como muestra la Figura II.12, de forma que la matriz se actualiza con los datos referentes a la comparación seleccionada. Si la matriz es inconsistente se muestra un asterisco junto al nombre de la comparación.

Figura II.9. Consultar PROMETHEE (Administrador).

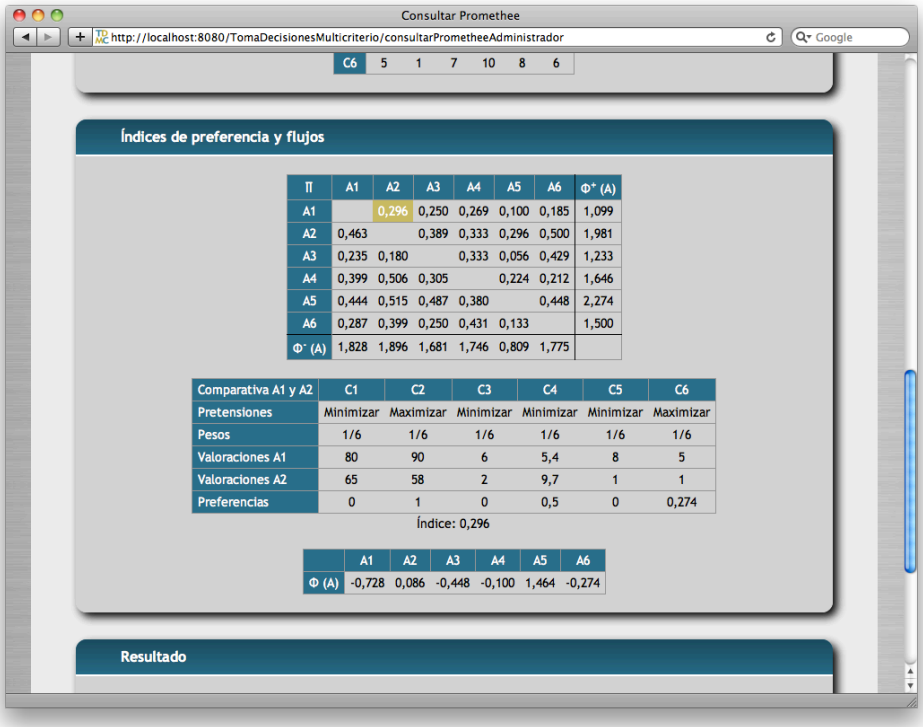


Figura II.10. Consultar PROMETHEE: comparativa alternativas (Administrador).

Figura II.11. Consultar AHP (Administrador).

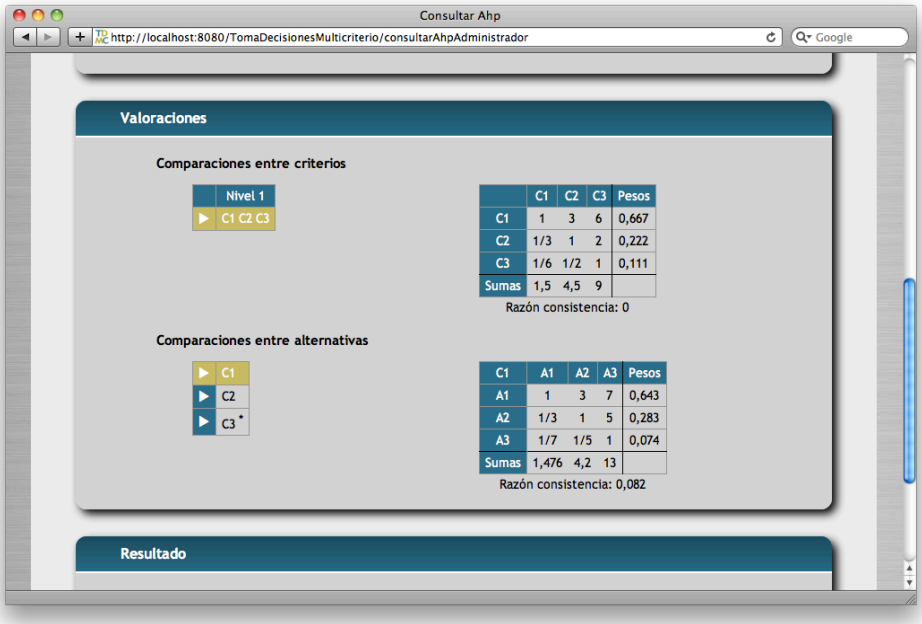


Figura II.12. Consultar AHP: comparaciones criterios y alternativas (Administrador).

2.1.2.2. Eliminar problema

Para eliminar un problema existen dos métodos posibles. Un primer modo es consultando el problema, pulsando en su nombre desde el listado de la sección gestionar problemas, de forma que según su tipo aparece la Figura II.9 o II.11. Después hay que pulsar sobre el botón *Eliminar* y finalmente, confirmar la acción.

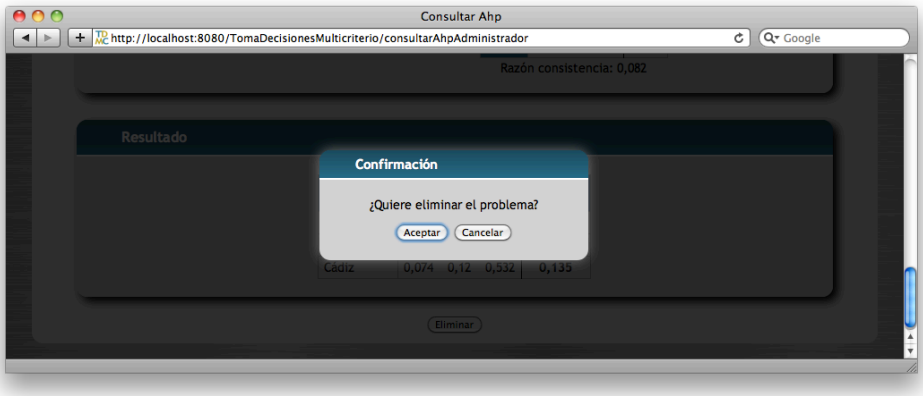


Figura II.13. Confirmación eliminar problema (Administrador).

El segundo modo permite eliminar varios problemas de forma simultánea y sin necesidad de consultarlos con anterioridad. Desde la sección de gestionar problemas, sólo hay seleccionarlos del listado y pulsar en *Eliminar problema*.

Figura II.14. Eliminar problemas (Administrador).

Por último, aparece un cuadro de diálogo modal para confirmar la acción.

Figura II.15. Confirmación eliminar problemas (Administrador).

2.1.3. Modificar contraseña

A este apartado se accede a través de la opción *Ajustes* del menú. Para que el administrador modifique su contraseña es necesario que proporcione la actual y una nueva contraseña. Finalmente hay que pulsar *Actualizar*.

Figura II.16. Modificar contraseña (Administrador).

2.2. Experto

Si el usuario se identifica como un experto entonces aparece la pantalla de gestionar problemas, que se corresponde con su página principal. Al igual que ocurría con el administrador, mientras el experto navega por la aplicación, cuenta con un menú constante en la par-

te superior de las páginas que se utiliza para acceder a las diferentes secciones. Para salir sólo hay que pulsar la opción *Salir*.

Algunas de las tareas que realiza un experto son comunes a las del administrador, por lo que en estas ocasiones se hace referencia a las figuras ya mostradas de éste último, con el objetivo de reducir la extensión del manual de usuario. Las diferencias entre ellas residen principalmente en el menú.

2.2.1. Gestionar problemas

A la gestión de problemas también se puede acceder a través de la opción *Problemas* del menú. En este apartado se muestra un listado de los problemas que pertenecen al experto. Desde aquí el experto puede realizar una serie de tareas de gestión sobre sus problemas, como consultarlos, crear nuevos, modificarlos o eliminarlos.

Figura II.17. Gestionar problemas (Experto).

2.2.1.1. Consultar problema

Para consultar uno de los problema del listado hay que pulsar sobre su nombre. Luego, se muestra una página donde aparecen sus datos y el resultado obtenido. A continuación, se recoge la consulta de un problema de tipo PROMETHEE y AHP.

- ***PROMETHEE***

El problema se muestra en una página como la de la Figura II.9. Al pulsar sobre un índice de preferencia aparece una comparativa entre dos alternativas, que muestra parte de los cálculos para obtener el índice, tal y como recoge la Figura II.10.

- ***AHP***

Se visualizan los datos y resultados del problema de igual forma que en la Figura II.11. Para pasar de una comparación a otra sólo hay que pulsar sobre su nombre, así la matriz se actualiza con sus datos correspondientes, como se puede ver en la Figura II.12. Si la matriz es inconsistente se muestra un asterisco a continuación del nombre de la comparación.

2.2.1.2. Crear problema

Partiendo de la sección gestionar problemas, en primer lugar se pulsa el botón *Crear problema*. A continuación, aparece la página donde se introducen sus datos, que inicialmente son el nombre, objetivo y tipo de problema que se pretende crear.

Figura II.18. Crear problema: datos.

En función del tipo elegido, PROMETHEE o AHP, la aplicación requiere los datos asociados a uno u otro modelo.

- ***PROMETHEE***

Al seleccionar el tipo de problema PROMETHEE, se despliega un nuevo apartado donde el experto puede añadir y quitar las alternativas y criterios que necesite. Tiene que introducir los nombres de las alternativas, los datos asociados a los criterios y las valoraciones. Los datos inválidos se muestran de color rojo.

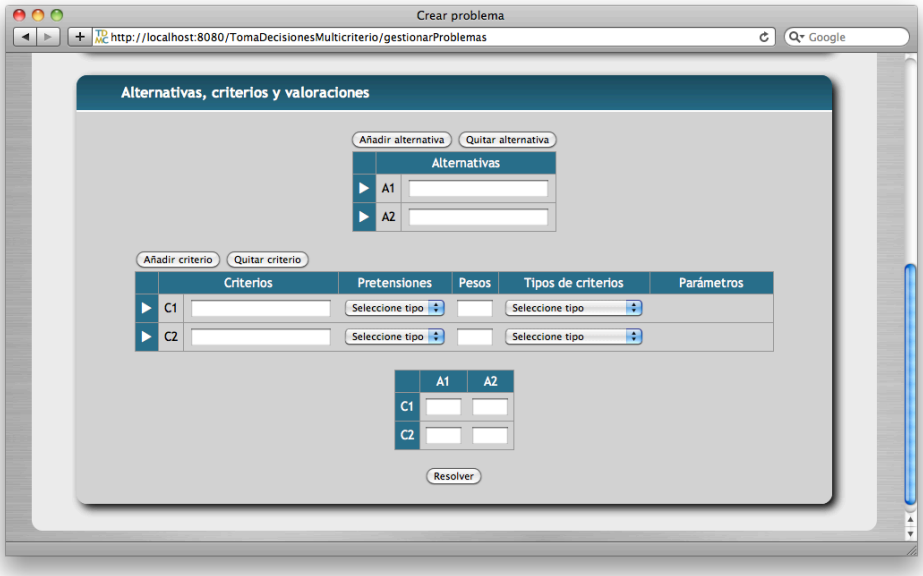


Figura II.19. Crear PROMETHEE: alternativas, criterios y valoraciones.

Una vez se introducen los datos del problema, se pulsa en *Resolver* para aplicar el modelo PROMETHEE. Seguidamente, aparece una página que recoge los datos del problema unidos al resultado obtenido.

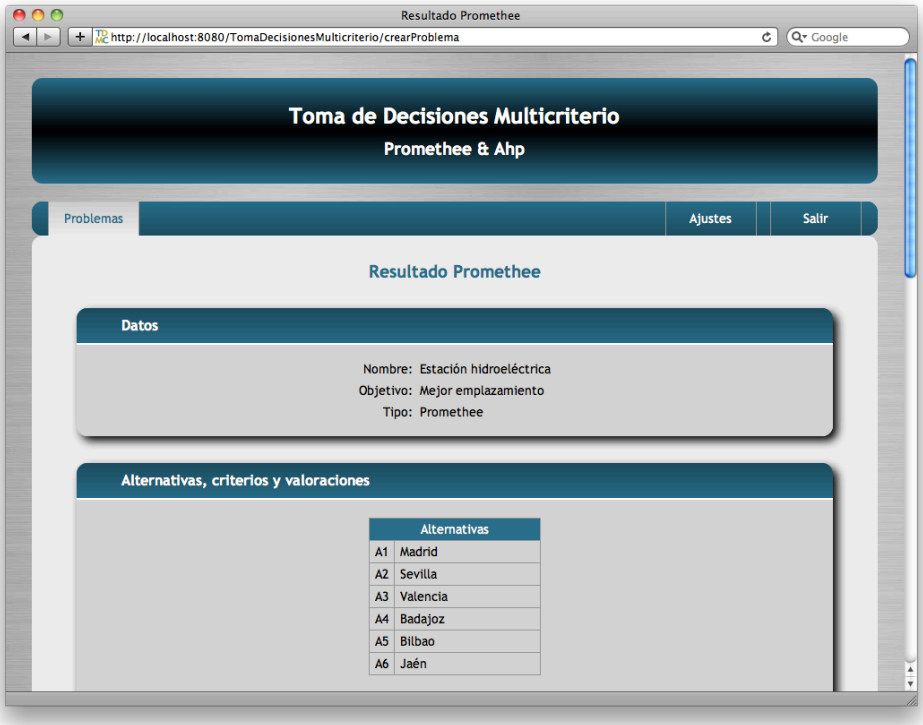


Figura II.20. Resultado PROMETHEE.

Dentro de la página de resultados, el experto puede visualizar la comparativa entre dos alternativas seleccionando alguno de los índices de preferencia. Las comparativas muestran de un modo más detallado el proceso seguido para calcular el índice.

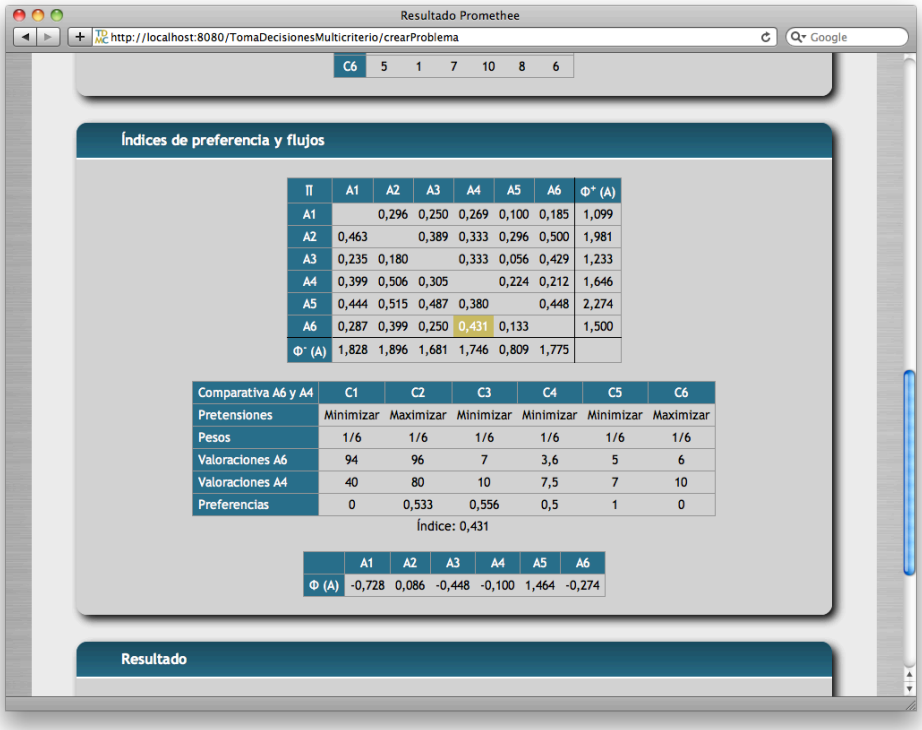


Figura II.21. Resultado PROMETHEE: comparativa alternativas.

Finalmente, si es necesario modificar alguno de los datos introducidos, se puede realizar un análisis de sensibilidad antes de almacenar el problema, pulsando el botón destinado a tal fin, apareciendo una página como la de la Figura II.26. Si por el contrario se quiere almacenar el problema, se pulsa en *Guardar*.

- ***AHP***

Análogamente a PROMETHEE, si el experto selecciona el tipo de problema AHP, se despliega un nuevo apartado donde se determinan las alternativas y la estructura de los criterios. Es necesario introducir los nombres de ambos. Como la aplicación soporta la creación de subcriterios, éstos se pueden introducir seleccionando el criterio padre y después pulsando en *Añadir criterio*.

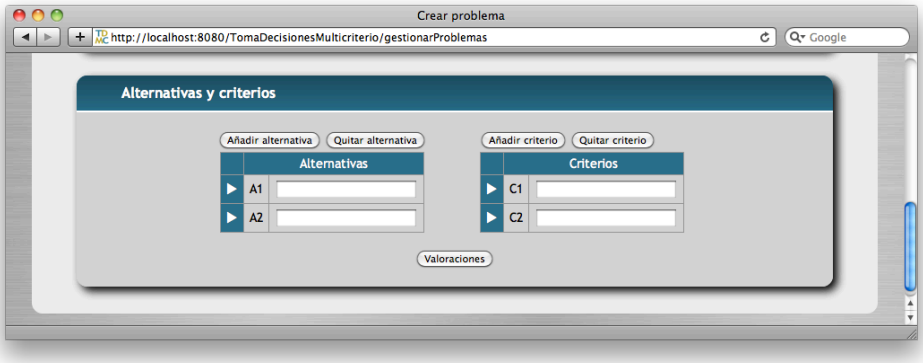


Figura II.22. Crear AHP: alternativas y criterios.

Una vez definidas las alternativas y criterios, se pulsa en *Valoraciones* para generar las matrices de comparaciones pareadas entre criterios y alternativas. Cada comparación se identifica con un nombre, de modo que para pasar de una a otra sólo hay que pulsar en él, con lo que la matriz se actualiza con sus datos correspondientes. Si el nombre aparece en gris, significa que la matriz no está completa o algún valor es erróneo. Las valoraciones inválidas se muestran en rojo.

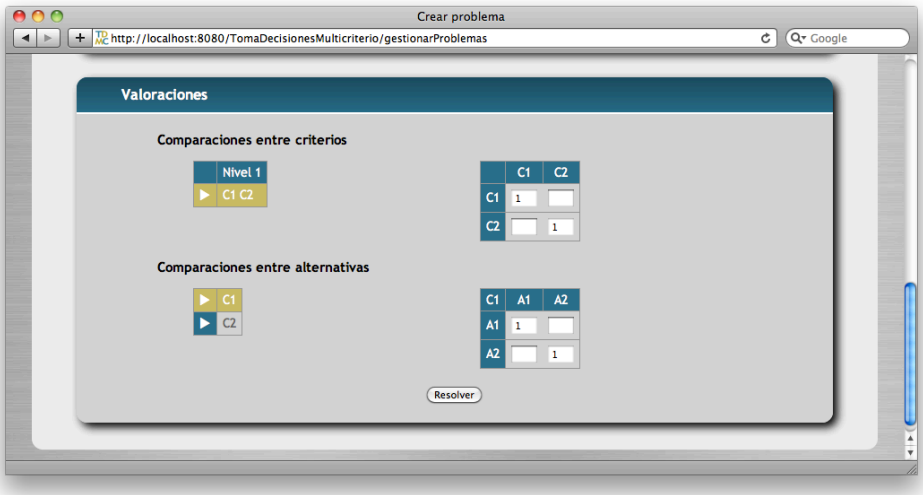


Figura II.23. Crear AHP: valoraciones.

Cuando se definen las alternativas, criterios y valoraciones del problema, se pulsa el botón *Resolver*, que aplica el modelo AHP para obtener una solución. Después, se muestra una página donde aparecen los datos del problema junto con el resultado obtenido.

Figura II.24. Resultado AHP.

De nuevo, para navegar a través de las comparaciones sólo hay que seleccionar su nombre. Además, si la matriz es inconsistente se muestra un asterisco al lado del nombre de la comparación.

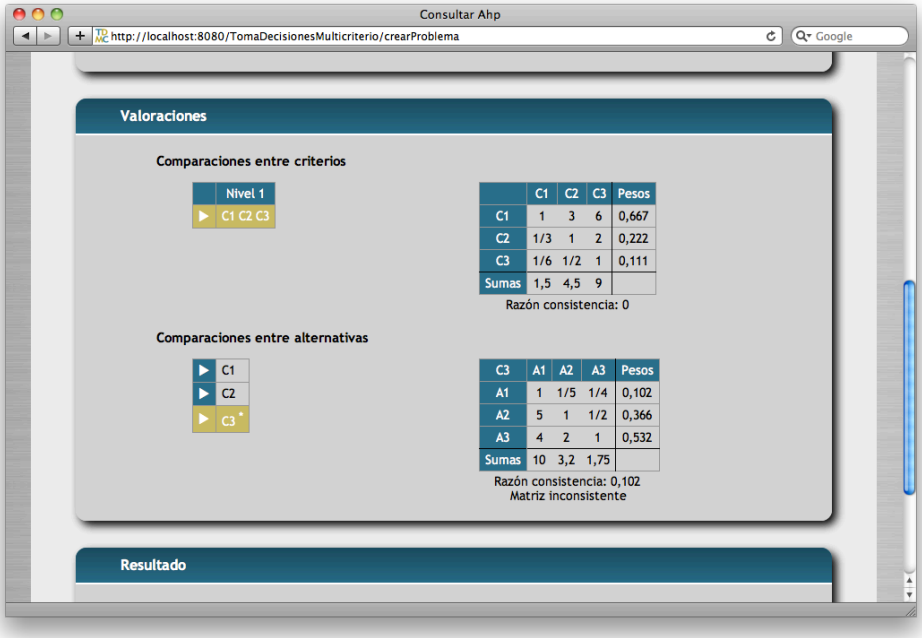


Figura II.25. Resultado AHP: comparaciones criterios y alternativas.

Por último, si antes de almacenar el problema se quiere modificar algún dato, se puede realizar un análisis de sensibilidad pulsando el botón con este nombre, con lo que se muestra una página como la de la Figura II.27. Si por el contrario se desea almacenar el problema, únicamente hay que pulsar en *Guardar*.

2.2.1.3. Modificar problema

Para modificar un problema en primer lugar hay que consultarlo, pulsando sobre su nombre en la sección de gestionar problemas. Según el tipo de problema que sea, aparece una página como la de la Figura II.9 o II.11, que muestra sus datos y resultados. A continuación, se pulsa en *Modificar* y se hace una distinción según el tipo de problema.

- ***PROMETHEE***

Si el problema es de tipo PROMETHEE, aparece una página como la de la Figura II.26, que se encarga de mostrar los datos actuales del problema y permite su modificación.

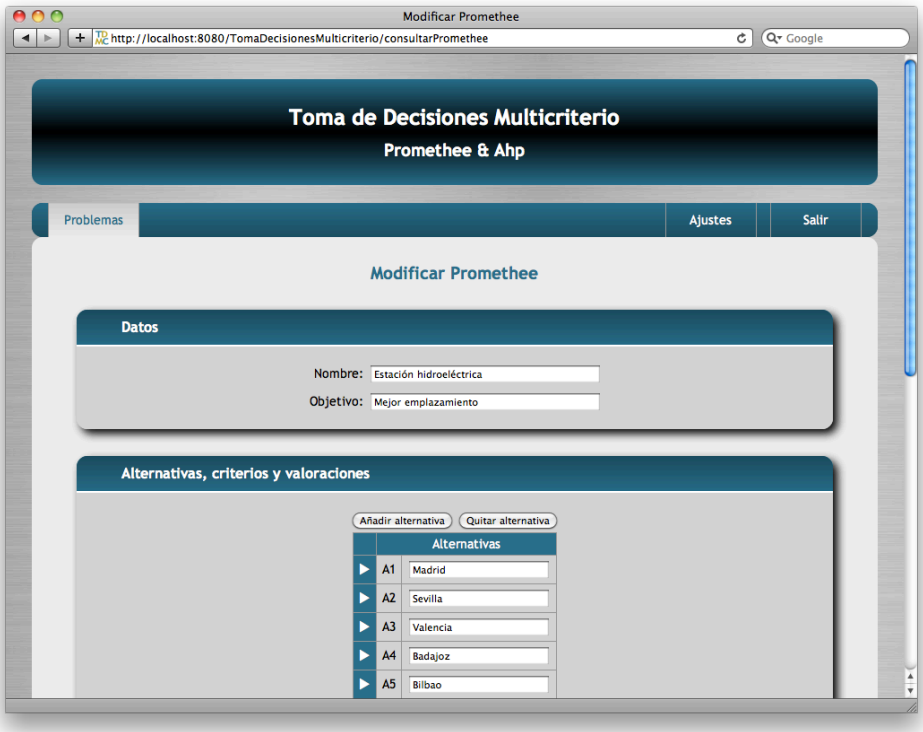


Figura II.26. Modificar PROMETHEE.

Una vez se modifican los datos necesarios se resuelve de nuevo el problema, pulsando en *Resolver*, para dar paso a una página como la de la Figura II.20 que recoge los resultados obtenidos. Por último, hay que pulsar en *Actualizar* para hacer definitiva la modificación.

✦ *AHP*

Si por el contrario el problema es de tipo AHP, se da paso a una página como la de la Figura II.27, que muestra los datos que presenta el problema y permite que se puedan modificar.

Figura II.27. Modificar AHP.

Una vez se modifican los datos oportunos, se pulsa en *Resolver* y se muestra una página como la de la Figura II.24 que recoge los resultados obtenidos. Finalmente, para terminar con la modificación se pulsa en *Actualizar*.

2.2.1.4. Eliminar problema

Un experto puede eliminar sus problema de dos maneras. Una forma es pulsando sobre el nombre del problema que quiere eliminar, partiendo del listado de gestionar problemas, de la misma forma que cuando se consulta uno. En función del tipo de problema, aparece una página como la de Figura II.9 o II.11. Finalmente, sólo hay que pulsar el botón *Eliminar* y confirmar la acción como en la Figura II.13.

El segundo método permite eliminar más de un problema a la vez y sin necesidad de consultarlos primero. Desde la sección de gestionar problemas se seleccionan del listado los problemas que se quieren eliminar. Finalmente, se pulsa el botón *Eliminar problema* y aparece un cuadro de diálogo modal como el de la Figura II.15 para confirmar la acción.

Figura II.28. Eliminar problemas (Experto).

2.2.2. Modificar contraseña

Para modificar la contraseña, en primer lugar hay que seleccionar la opción *Ajustes* del menú. Después, aparece una página como la de la Figura II.16 donde el experto debe introducir la contraseña actual y proporcionar una nueva. Finalmente se pulsa *Actualizar*.

Bibliografía

(1) J.P. Brans y PH. Vincke. A preference ranking organisation method - (The PROMETHEE method for multiple criteria decision-making). Management Science (6): 647-656 1985.

(2) J.P. Brans, PH. Vincke y B. Mareschal. How to select and how to rank projects: The PROMETHEE method. European Journal of Operational Research (2): 228-238 FEB 1986.

(3) Wim De Keyser y Peter Peeters. A note on the use of PROMETHEE multicriteria methods. European Journal of Operational Research 89 (3): 457-461 MAR 22 1996.

(4) Yoram Wind y Thomas L. Saaty. Marketing applications of the analytic hierarchy process. Management Science 26 (7): 641-658 1980.

(5) R.W. Saaty. The analytic hierarchy process – What it is and how it is used. Mathematical Modelling 9 (3-5): 161-176 1987.

(6) Thomas L. Saaty. How to make a decision: The analytic hierarchy process. European Journal of Operational Research 48 (1): 9-26 SEP 5 1990.

(7) Rafael Infante Macías. Teoría de la Decisión. Universidad Nacional de Educación a Distancia, 1978.

(8) Daniel Cohen. Sistemas de Información para la Toma de Decisiones. McGraw Hill, 1999.

(9) Carlos Romero. Teoría de la decisión multicriterio: conceptos, técnicas y aplicaciones. Alianza Editorial, 1993.

(10) Sergio Barba-Romero y Jean-Charles Pomerol. Decisiones Multicriterio - Fundamentos Teóricos y Utilización Práctica. Servicio de publicaciones de la Universidad de Alcalá de Henares, 1997.

(11) Gabriela M. Fernández Barberis. Los Métodos PROMETHEE: Una metodología de Ayuda a la Toma de Decisiones Multicriterio Discreta. Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA, 2002.

(12) José María Moreno Jiménez. El Proceso Analítico Jerárquico (AHP). Fundamentos, Metodología y Aplicaciones. Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA, 2002.

(13) Ángel M. Gento, Raúl García y Alberto Toribio. WINELECTRE: Ayuda a la Decisión Mediante los Métodos ELECTRE. Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA, 2002.

(14) Mª Carmen Escribano Ródenas y Mª Carmen García Centeno. Dificultades de la Puesta en Práctica de los Métodos de Decisión Multicriterio Discretos. Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA, 2002.

(15) María Amparo León Sánchez y Fernando L. Domínguez Goizueta. Modelo de Decisión Multicriterio Discreto en la Selección de Alternativas de Aprovechamiento de la Madera. Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA, 2002.

(16) Tomas L. Saaty. Toma de Decisiones para Líderes: El proceso analítico jerárquico la toma de decisiones en un mundo complejo. RWS Publications, 1997.

(17) Toskano Hurtado y Gérard Bruno. El Proceso de Análisis Jerárquico (AHP) como Herramienta para la Toma de Decisiones en la Selección de Proveedores. Tesis Digitales UNMSM.

(18) Ruth Maritza Avila Mogollón. El AHP (Proceso Analítico Jerárquico) y su aplicación para determinar los usos de las tierras.

(19) J. Aguarón y J. M. Moreno-Jiménez. Local Stability Intervals in the Analytic Hierarchy Process. European Journal of Operational Research 125(1), 114-133.

(20) H. A. Simon. Theories of Decision-Making in Economics and Behavioral-Science. American Economic Review 49 (3): 253-283 1959.

(21) Mª Socorro García y Mª Teresa Lamata. Un problema de mantenimiento basado en el proceso analítico jerárquico.

(22) Jie Lu, Guangquan Zhang, Da Ruan y Fengjie Wu. Multi-Objective Group Decision Making - Methods, Software and Applications with Fuzzy Set Techniques. Imperial College Press, 2003.

(23) L. Mandow, J. L Pérez de la Cruz, F. Triguero y R. Conejo. Diseño Mediante Satisfacción de Metas. Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga.

(24) J. Doyle. Prospects for preferences. Computational Intelligence, 20(2):111-136, 2004.

(25) Ian Sommerville. Ingeniería del Software. PEARSON Addison Weasly, 2006.

(26) Shari Lawrence Pfleeger. Ingeniería del software - Teoría y Práctica. Prentice Hall, 2002.

(27) Perdita Stevens y Rob Pooley. Utilización de UML en Ingeniería del Software con Objetos y Componentes. Addison Wesley, 2002.

(28) Roger S. Pressman. Ingeniería del Software - Un enfoque práctico. Mc Graw Hill, 2003.

(29) Craig Larman. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. PEARSON Prentice Hall, 2003.

(30) Francisco Feito Higueruela, Andrés Molina Aguilar y Juan Ruiz de Miras. Análisis y gestión de datos. Universidad de Jaén, 1996.

(31) Alan Dix, Janet Finlay, Gregory D. Abowd y Rusell Beale. Human-Computer Interaction. Pearson education Ltd, 2004.

(32) Ben Shneiderman y Catherine Plaisant. Designing the user interface. Addison Wesley, 2004.

(33) Eric Freeman, Elisabeth Freeman, Kathy Sierra y Bert Bates. Head First, design patterns. O'Reilly, 2004.

(34) Deepak Alur, John Crupi y Dan Malks. Core J2EE Patterns: Best Practices and Design Strategies. Prentice Hall / Sun Microsystems Press, 2001.

(35) Documentación de Struts 2: http://struts.apache.org/2.x/docs/home.html

(36) Documentación de Hibernate: http://www.hibernate.org/docs

(37) Documentación del *plugin* struts2-jquery: http://code.google.com/p/struts2-jquery/w/list

(38) Patrón de diseño *Open Session in View*: http://community.jboss.org/wiki/OpenSessionInView