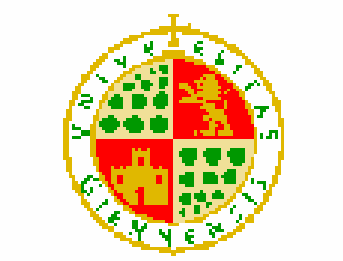
COMPARATIVA DE MODELOS DE AGREGACIÓN EN UN SISTEMA DE APOYO A LA DECISIÓN

Alumna:	Virginia Rus Martínez
Tutores:	Luis Martínez López Pedro J. Sánchez Sánchez
Departamento:	Informática



Universidad de Jaén

Índice General

CAPÍTULO 1

PREFACIO

CAPÍTULO 1

PREFACIO

Contenidos

1.1. Introducción al proyecto

La Toma de Decisiones (TD) es un proceso al que los seres humanos se enfrentan casi continuamente en su devenir habitual. Tomar una buena decisión consiste en fijar el objetivo que se quiere conseguir, reunir toda la información relevante y tener en cuenta las preferencias y características de las alternativas sobre las que se tiene que tomar dicha decisión. Si queremos hacerlo correctamente, debemos ser conscientes de que una buena decisión es un proceso que necesita tiempo y planificación. En la toma de decisiones se deben seleccionar entre varias opciones la/s más adecuada/s atendiendo a una serie de objetivos que se persiguen a la hora de tomar una decisión, dando unos valores de utilidad o valoraciones a las distintas acciones que se presentan.

Dicho formalmente, la toma de decisiones se define como la selección de un curso de acciones entre alternativas, es decir, que existe un plan, un compromiso de recursos de dirección o reputación. En ocasiones los ingenieros consideran la toma de decisiones como su trabajo principal ya que tienen que seleccionar constantemente qué se hace, quien lo hace y cuando, dónde e incluso cómo. Sin embargo, la toma de decisiones es sólo un paso de la planificación ya que forma la parte esencial de los procesos que se siguen para la elaboración de los objetivos o metas trazadas a seguir. Rara vez se puede juzgar sólo un curso de acción, porque prácticamente cada decisión tiene que estar engranada con otros planes.

La Teoría de Decisión clasifica los problemas de decisión en distintos modelos atendiendo a diferentes criterios, como por ejemplo:

- Número de Expertos. En caso de que en el problema de decisión sólo participe un único experto hablamos de Toma de Decisiones Individual, y en caso de participar varios expertos hablamos de *Toma de Decisiones Multiexperto.*

- Número de Criterios o Atributos. Existen problemas de decisión que implican una "optimización simple" de las alternativas de acuerdo a un único criterio, mientras que otros problemas implican una "optimización múltiple"

de acuerdo a varios criterios, hablándose de *Toma de Decisiones Multicriterio*.

En problemas de TD diferenciamos dos tipos de procesos: la *Selección de Alternativas y el Consenso*.

1. *Selección de alternativas:* aquí se busca el conjunto solución de alternativa/s que mejor se adecue a los objetivos perseguidos en el problema de Decisión. En este proceso distinguimos dos fases:

   (i) Una *fase de Agregación* en la que se combinan las preferencias individuales de criterios y/o expertos para obtener una preferencia colectiva sobre las alternativas del problema.

   (ii) Una *fase de Explotación* en la que a partir de las preferencias colectivas se aplicará un grado de selección para generar un conjunto solución de alternativas para el problema.

2. *Consenso*: El proceso de selección de alternativas en problemas de decisión con múltiples expertos puede dar lugar a soluciones que no son aceptadas como buenas por parte de algunos expertos, por lo que el estudio del *Consenso* se ha convertido en un campo de investigación de gran importancia dentro de la TD. Podemos decir que el Consenso es un proceso de discusión en grupo e iterativo que es coordinado por un *moderador* que ayuda a los expertos del problema a acercar sus opiniones. El consenso se ha definido clásicamente como el acuerdo total y unánime de todos los expertos que participan en un problema, debido a esta definición primero se utilizaron medidas de consenso absolutas para posteriormente evolucionar hacia medidas difusas (Soft Consensus) que ofrecen una mayor flexibilidad para expresar una medida vaga *"per se"* como es el consenso.

En este proyecto nos vamos a centrar en la fase de *Selección*, y dentro de ésta, en la de *agregación*, como se observa en la *Figura 1*, y atendiendo al tipo de problema, nos centramos en problemas de decisión con múltiples expertos.

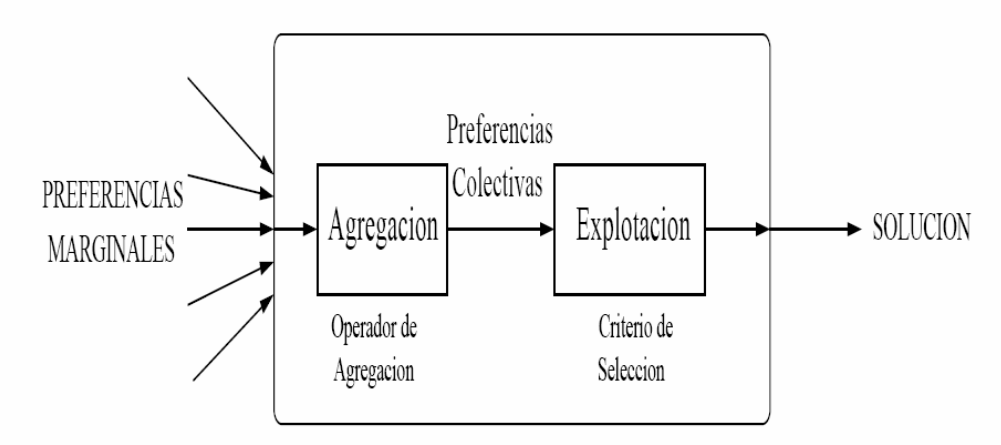


Figura 1. Selección de alternativas

1.2 Propósito

El propósito de este proyecto es implementar un prototipo de un Sistema de Ayuda a la Decisión (SAD), utilizando el modelo de decisión de la *Figura* 1, que sea capaz de abordar problemas Multi-Experto y que nos permita utilizar diferentes operadores de agregación fundamentalmente basados en la familia de operadores OWA (con distintos cuantificadores lingüísticos difusos) y comparar los resultados que obtendría el proceso de selección de alternativas en base a estos operadores, así podremos ver cuál es la importancia de las condiciones de agregación en distintos problemas de Toma de Decisión.

1.3. Objetivos

- Búsqueda y revisión bibliográfica:
    - de la temática de Sistemas de Soporte a la Toma de Decisiones
    - de los operadores de agregación, centrándonos fundamentalmente en la familia de operadores de la *media ponderada ordenada* (OWA)

- Implementar un sistema de ayuda a la toma de decisión de acuerdo al presentado en la *Figura1* con diferentes operadores de agregación, algunos de ellos basados en la familia de operadores OWA, para la resolución de problemas MultiExperto.

- Visualización de la importancia de los operadores de agregación en los procesos de decisión para la extracción de distintas conclusiones.

1.4. Resumen

Los objetivos planteados en esta memoria se desarrollan a lo largo de la misma como sigue:

Para conseguir el primero de los objetivos desarrollamos el segundo capítulo, en el que veremos con más detalle lo que es la toma de decisiones, y clasificaremos los problemas de toma de decisión en dos grupos, según el contexto y según un número de elementos que toman parte en la decisión, como pueden ser el número de expertos y de criterios que caracterizan las alternativas. También en este capítulo revisaremos formalmente los operadores de agregación que vamos a utilizar a lo largo del proyecto, así como la representación de los datos que utilizarán los expertos para dar sus valoraciones a las alternativas.

En el tercer capítulo queda plasmado el segundo objetivo. En este capítulo describiremos el proyecto en sí y veremos que se trata de un proyecto de desarrollo software y, como tal, deben seguirse para su desarrollo las actividades de la Ingeniería del Software, como son: la especificación de requerimientos funcionales y no funcionales, el análisis y diseño del sistema y cuando tengamos todo esto claro y bien documentado pasaremos a implementación de nuestro Sistema de Ayuda a la Decisión y a su posterior validación.

Para el tercer y último objetivo de nuestro proyecto, escribiremos el capítulo cuarto, en el que expondremos las conclusiones de este trabajo tras haber realizado diferentes ejecuciones con los distintos operadores implementados.

En el apartado de bibliografía enumeraremos las referencias bibliográficas hechas durante la memoria de este proyecto.

Por último, esta memoria también consta de cuatro anexos, el primero de ellos para hablar con más detalle de los cuantificadores lingüísticos que se utilizan en este proyecto para uno de los operadores que implementamos, y los otros tres anexos para elaborar un manual de instalación del servidor, un manual del usuario Administrador del sistema y otro manual del usuario Experto.

CAPÍTULO 2

ANTECEDENTES

CAPÍTULO 2

ANTECEDENTES

Contenidos

2.1. Introducción a la Toma de Decisión

En los posteriores apartados daremos una visión detallada de lo que es la Toma de Decisión, las etapas que la componen y los diversos tipos en los que se clasifica.

2.1.1. Descripción

La Toma de Decisiones es sin duda una de las actividades fundamentales de los humanos, ya que constantemente nos estamos enfrentando a situaciones en las que existen varias alternativas y, al menos en algunas ocasiones, tenemos que decidir cuál es mejor, o cuál llevar a cabo.

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, que han dado lugar a la Teoría de Decisión.

Un problema clásico de decisión tiene los siguientes elementos básicos:

1. Un conjunto de alternativas o decisiones posibles.
2. Un conjunto de estados de la naturaleza que definen el contexto de definición del problema.
3. Un conjunto de valores de utilidad, cada uno de los cuales está asociado a un par formado por una alternativa y un estado de la naturaleza.
4. Una función que establece las preferencias del experto sobre los posibles resultados.

2.1.2. Etapas

La Toma De Decisión está formada por las etapas que se enumeran a continuación:

1. Identificación y diagnostico del problema
2. Generación de soluciones alternativas
3. Evaluación de alternativas
4. Selección de la mejor alternativa
5. Implementación de la decisión
6. Evaluación de la decisión

1. Identificación y diagnóstico del problema:

   Reconocemos en la fase inicial el problema que deseamos solucionar, teniendo en cuenta el estado actual con respecto al estado deseado. Una vez que el problema es identificado se debe realizar el diagnóstico y después de esto podremos desarrollar las medidas correctivas.

2. Generación de soluciones alternativas:

   La solución de los problemas puede lograrse por varios caminos y no sólo seleccionar entre dos alternativas, se pueden formular hipótesis ya que con la alternativa hay incertidumbre.

3. Evaluación de alternativas:

   La tercera etapa implica la determinación del valor o la adecuación de las alternativas que se generaron. Los expertos en la toma de decisiones deben considerar distintos tipos de consecuencias. Por supuesto que deben intentar predecir los efectos sobre las medidas financieras u otras medidas de desarrollo. Pero también existen otras consecuencias menos definidas que hay que atender. Las decisiones establecen un precedente y hay que determinar si este será una ayuda o un obstáculo en el futuro.

Por supuesto, no es posible predecir los resultados con toda precisión. Entonces pueden generar planes de contingencia, esto es, curso alternativo de acción que se pueden implantar con base en el desarrollo de los acontecimientos.

Una vez encontrada la alternativa apropiada, el siguiente paso es evaluar y seleccionar aquellas que contribuirán mejor al logro de la meta.

a) FACTORES CUANTITATIVOS

Son factores que se pueden medir en términos numéricos, como es el tiempo, o los diversos costos fijos o de operación.

b) FACTORES CUALITATIVOS

Son difíciles de medir numéricamente. Como la calidad de las relaciones de trabajo, el riesgo del cambio tecnológico o el clima político internacional.

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

4. Selección de la mejor alternativa:

Cuando el experto ha considerado las posibles consecuencias de sus opciones, ya está en condiciones de tomar la decisión. Debe considerar tres términos muy importantes. Estos son: maximizar, satisfacer y optimizar.

- *Maximizar*: 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 delegan dicha responsabilidad en otras personas. Debe existir la comprensión total sobre la elección de la toma de decisión en sí, las razones que la motivan y sobre todo debe existir el compromiso de su implementación exitosa. Para tal fin, las personas que participan en esta fase del proceso, deberían estar involucradas desde las primeras etapas que anteriormente se han mencionado.

Podemos estar seguros de que cuando una toma de 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 estos pueden representar.

6. Evaluación de la decisión:

"Evaluar la decisión", forma parte de la etapa final de este proceso. Se recopila toda la información que nos indique la forma como funciona una decisión, es decir, es un proceso de retroalimentación que podría ser positiva o negativa. Si la retroalimentación es positiva, entonces nos indica que podemos 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
- nos puede indicar que la decisión fue equivocada, para lo cual debemos volver al principio del proceso (re)definición del problema. Si esto ocurriera, sin duda tendríamos más información y probablemente sugerencias que nos ayudarían a evitar los errores cometidos en el primer intento.

El arte de tomar decisiones está basado en cinco ingredientes básicos:

a. *Información*:

Ésta se recoge tanto para los aspectos que están a favor como en contra de la alternativa para resolver el problema, con el fin de definir sus limitaciones. 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.

b. *Conocimientos*:

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

c. *Experiencia*:

Cuando un individuo aporta información para solucionar un problema en 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 carecemos de experiencia entonces tendremos que experimentar; pero sólo en el caso en que las consecuencias de un mal experimento no sean desastrosas. Por lo tanto, los problemas más importantes no pueden solucionarse con experimentos.)

d. *Análisis*:

No puede hablarse de un método en particular para analizar un problema, debe existir un complemento, pero no un reemplazo de los otros ingredientes. 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 un camino que tomar, entonces ésta es la única opción disponible.

e. *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.

2.1.3. Criterios de clasificación

Los problemas de Toma de Decisión se clasifican según diferentes criterios. En este proyecto, la vamos a clasificar según el tipo de problema, y según el ambiente en el que se toma la decisión. La clasificación hecha según el tipo de problema la veremos con más detalle en un apartado posterior ya que nuestro proyecto se centra en este tipo de clasificación. En este apartado solo nos vamos a centrar en el segundo tipo de clasificación.

Atendiendo al segundo tipo de clasificación, los problemas de toma de decisiones pueden clasificarse según el ambiente en el que se toma la decisión. Las situaciones, ambientes o contextos en los cuales se toman las decisiones, se pueden clasificar según el conocimiento y control que se tenga sobre las variables que intervienen o influencian el problema, ya que la decisión final o la solución que se tome va a estar condicionada por dichas variables.

Dependiendo de esto tenemos tres tipos de problemas:

- Ambiente de certidumbre
- Ambiente de riesgo
- Ambiente de incertidumbre

Ambiente de certidumbre

Se tiene conocimiento total sobre el problema, las alternativas de solución que se planteen van a causar siempre resultados conocidos e invariables. Al tomar la decisión solo se debe pensar en la alternativa que genere mayor beneficio.

Ambiente de riesgo

La información con la que se cuenta para solucionar el problema es incompleta, es decir, se conoce el problema, se conocen las posibles soluciones, pero no se conoce con certeza los resultados que pueden arrojar.

En este tipo de decisiones, las posibles alternativas de solución tienen cierta probabilidad conocida de generar un resultado. En estos casos se pueden usar modelos matemáticos o también el decisor puede hacer uso de la probabilidad objetiva o subjetiva para estimar el posible resultado.

La probabilidad objetiva es la posibilidad de que ocurra un resultado basándose en hechos concretos, puede ser cifras de años anteriores o estudios realizados para este fin. En la probabilidad subjetiva se determina el resultado basándose en opiniones y juicios personales.

Ambiente de incertidumbre

Se posee información deficiente para tomar la decisión, no se tienen ningún control sobre la situación, no se conoce como puede variar o la interacción de la variables del problema, se pueden plantear diferentes alternativas de solución pero no se le puede asignar probabilidad a los resultados que arrojen.

Con base en lo anterior hay dos clases de incertidumbre:

- *Estructurada*: No se sabe que puede pasar entre diferentes alternativas, pero sí se conoce que puede ocurrir entre varias posibilidades.
- *No estructurada*: No se sabe que puede ocurrir ni las probabilidades para las posibles soluciones, es decir no se tienen idea de qué pueda pasar.

La Teoría Clásica de Decisión proporciona una gran cantidad de modelos sobre las distintas situaciones de decisión, aunque no es capaz de dar solución a todos los problemas de decisión. Los métodos clásicos se adecuan fácilmente a problemas desarrollados en ambiente de certidumbre y de riesgo, sin embargo no son adecuados en situaciones de incertidumbre, es decir, en problemas que presentan información vaga e imprecisa. En estas situaciones hablamos de problemas de decisión en contexto difuso o de "Toma de Decisiones Difusa".

2.1.4. Modelado de preferencias

El modelado de preferencias es un elemento fundamental en áreas, tales como, la Toma de Decisiones, la Economía, la Psicología, etc. En el modelado de preferencias debemos decidir la representación de las valoraciones de las preferencias dadas por los expertos que participan en el problema. Al hablar de representación en este caso, lo hacemos desde dos puntos de vista claramente diferenciados e igualmente transcendentes, como son:

1. Decidir la naturaleza y el dominio de la información que se va a utilizar para expresar los valores de las preferencias sobre las distintas alternativas del problema. Dependiendo de la opción que se tome aquí podemos hablar de:

    a) *Modelado binario de preferencias*. Se utilizan dos valores, normalmente {0,1}, para expresar la preferencia sobre cada par de alternativas.

    b) *Modelado de preferencias valorado en [0, 1].* La información utilizada para expresar las preferencias son valores numéricos en el intervalo [0,1], que modelan la incertidumbre existente en la preferencia de cada alternativa.

    c) *Modelado difuso de preferencias*. Cuando nos encontramos con problemas que presentan alternativas sobre las que tenemos un conocimiento incierto o impreciso, y se utilizan números difusos para expresar la preferencia sobre las distintas alternativas del problema.

d) *Modelado lingüístico de preferencias*. Los valores utilizados para expresar la preferencia sobre las alternativas del problema son etiquetas lingüísticas.

Nota: distintos autores consideran (b) y (c) con el mismo nombre, modelado difuso de preferencias.

2. Seleccionar la estructura que soportará y dará significado a las preferencias con las que trabajará el problema.

   a) *Vector de utilidad*. Los valores de preferencia son expresados en un vector, donde cada valor expresa la preferencia para cada una de las alternativas que se están calificando en el problema.

   b) *Relación de preferencia*. Un experto proporciona una relación binaria sobre el conjunto de alternativas, es decir una función que asocia a cada par de alternativas un número real que refleja en cierto sentido el grado de preferencia de una alternativa sobre otra cualquiera.

Vector de Utilidad

En este caso, un experto, $e_k \in E$, proporciona sus preferencias sobre X a través de un conjunto de n valores de utilidad, $U^k = \{u^k_1,,..., u^k_n\}$, $u^k_i \in [0,1]$, es decir, el experto asocia a cada alternativa una valuación de utilidad que representa el grado de cumplimiento de su punto de vista por parte de dicha alternativa. Para cada conjunto de valores de utilidad, supondremos, sin perdida de generalidad, que cuanto mayor es la valuación de una alternativa mejor satisface dicha alternativa el objetivo del experto.

Relaciones de Preferencia

En teoría matemática clásica, las preferencias sobre un conjunto de alternativas se pueden modelar a través de una relación binaria R definida como sigue:

$$x_i \, R \, x_j \iff \text{“}x_i \text{ no es peor que } x_j\text{”}$$

Esta definición considera una relación binaria como una relación de preferencia débil, e implica que dicha relación R es reflexiva. Con esta definición, es natural asociar un número real, llamado valuación y denotado $R(x_i, x_j) \in R$, el cual representa el grado de verdad de la afirmación *“*x_i no es peor que x_j*”*, o grado de preferencia de la alternativa x_i sobre la alternativa x_j . Cuando el conjunto de alternativas es finito, podemos asociar una matriz P_R a la relación R, tomando como elemento ij-ésimo el valor $R(x_i, x_j)$.

En problemas de decisión con varios individuos, como es nuestro caso, cada uno de ellos tiene su propio conocimiento sobre las alternativas del problema y cada uno de ellos según su área, su conocimiento sobre el problema o a la naturaleza de los aspectos valorados puede expresar sus preferencias en un dominio de expresión diferente. Esto implica que la información que maneja el problema podría ser no homogénea. Nosotros en esta memoria asumimos que la naturaleza de la información es numérica, utilizando un modelado de preferencias valorado en el intervalo [0, 1] y utilizando como estructura de representación el vector de utilidad.

2.2. Tipos de problemas

La Teoría de Decisión clasifica los problemas de decisión en distintos modelos atendiendo a diferentes criterios. Nosotros nos vamos a fijar únicamente en los dos siguientes:

- Número de Criterios. Existen problemas de decisión que implican una "optimización simple" de las alternativas de acuerdo a un único criterio, mientras que otros problemas implican una "optimización múltiple" de acuerdo a varios criterios, hablándose de *Toma de Decisiones Multicriterio*.

- Número de Expertos. En caso de que en el problema de decisión sólo participe un único experto hablamos de Toma de Decisiones Individual, y en caso de participar varios expertos hablamos de *Toma de Decisiones Multiexperto.*

En el modelo de Toma de Decisiones en el que existe un único experto, éste se encarga de valorar la/s alternativa/s de acuerdo a sus preferencias suministrando un vector de utilidad en el que hay un valor de preferencia para cada alternativa/s.

En cuanto al número de criterios, los procesos de toma de decisiones se han venido analizando tradicionalmente en base a un paradigma que puede esquematizarse de la siguiente forma:

- Se selecciona el criterio bajo el cual se desea decidir la mejor solución.
- Se define el conjunto de restricciones que limitan la solución del problema.

Seguidamente utilizando técnicas más o menos sofisticadas, se procede a buscar entre las soluciones aquella que obtenga un mejor valor del criterio seleccionado, a esto se le denomina solución óptima.

Las soluciones posibles de acuerdo a esta estructura son aquellas que den cumplimiento al conjunto de restricciones del problema y que representen los mejores valores del criterio seleccionado por el decisor. Este modelo posee una gran solidez desde el punto de vista lógico, sin embargo posee importantes debilidades que lo

desvían considerablemente de los procesos reales de toma de decisiones empresariales. Dado por que en la realidad, los decisores no están interesados en buscar la solución con respecto a un único criterio, sino que desean efectuar esta tarea con arreglo a diferentes criterios que reflejen sus preferencias.

El principal objetivo de nuestro proyecto es el de implementar un sistema de toma de decisión para la resolución de problemas con múltiples expertos. En los siguientes apartados comentaremos en detalle este problema, y los problemas multicriterio.

2.2.1. Problemas Multicriterio

En un problema de decisión multicriterio, las alternativas se evalúan de acuerdo a un conjunto de criterios, cada criterio induce un orden particular de alternativas, por lo que se necesita un procedimiento para construir un orden global de preferencias. Vemos la similitud entre este tipo de problemas y los problemas de toma de decisión multiexperto. En ambos casos, existen múltiples valoraciones en torno a las alternativas y hay que integrarlos en un único orden global de preferencia. La diferencia consiste en que en un tipo de problemas las valoraciones representan las preferencias de distintas personas y en otros la importancia de distintos criterios.

El número de criterios en problemas de decisión multicriterio se asume que es finito. Sean $X = \{x_1, x_2, \ldots, x_n\}$ y $C = \{c_1, c_2, \ldots, c_k\}$, un conjunto de alternativas y un conjunto de criterios de cada alternativa caracterizando una situación de decisión, respectivamente. Entonces, la información básica relativa al problema de TDMC puede expresarse mediante la siguiente tabla:

Alternativas	*Criterios*			
(x_i)	c_1	c_2	...	c_k
x_1	y_{11}	y_{12}	...	y_{1k}
...	...	...	...	...
x_n	y_{n1}	y_{n2}	...	y_{nk}

Figura 2. Tabla que representa las alternativas en un problema TDMC.

Cada entrada de la tabla y_{ij} indicará la importancia de la alternativa x_j respecto del criterio c_i, y según el modelado de preferencias que utilicemos estará valorada en un dominio determinado.

2.2.2. Problemas Multiexperto

En el modelo de Toma de Decisiones Multiexperto con el que vamos a trabajar, hay un conjunto de expertos en el que cada experto valora las alternativas de acuerdo a sus preferencias suministrando un vector de utilidad en el que hay un valor de preferencia para cada una de las alternativas con el que establece un orden sobre ellas. A continuación habrá que integrar los diferentes vectores de utilidad en un vector global de preferencia, para ello se utilizará un proceso de agregación.

El esquema general de un problema TDME podemos definirlo como sigue. Sea $X = \{x_1, x_2, ..., x_n\}$ ($n \geq 2$) un conjunto finito de alternativas que serán calificadas de acuerdo a la preferencia de un conjunto finito de expertos $P = \{p_1, p_2, ..., p_m\}$ ($m \geq 2$). Cada experto p_i suministra una preferencia y_{ij} para cada alternativa x_i, es decir:

Alternativas	*Expertos*			
(x_i)	p_1	p_2	...	p_m
x_1	y_{11}	y_{12}	...	y_{1m}
...	...	...	...	...
x_n	y_{n1}	y_{n2}	...	y_{nm}

Figura 3. Tabla que representa las alternativas en un problema TDME.

Vamos analizar las ventajas y desventajas del trabajo en grupo o comités:

Ventajas:

- Información y conocimiento más completos: Lógicamente un grupo logra recopilar más información, teniendo acceso a más fuentes informativas que un solo individuo, independiente de la educación y de la experiencia de este. Por lo tanto los grupos pueden ofrecer mayores aportes, tanto en la cantidad como en la diversidad para la Toma de decisiones.
- Incrementar la aceptación de una solución o bien la variedad de puntos de vista: Muchas decisiones fracasan después de elegida una opinión, debido a que un sector de gente no la acepta como una solución posible, cada uno de sus integrantes tiene un punto de vista propio que difiere, en cierta medida, del de los demás, como resultado, la cantidad y tipos de opciones son mayores que los del individuo que trabaja solo. La participación en grupo facilita una amplia discusión y una aceptación más participativa, es posible que haya divergencias en los acuerdos, pero se plantea y permite su discusión para cuando ya sea aceptada, sea un compromiso de todo un conjunto.

Es difícil que los asistentes al grupo de discusión ataquen o dificulten una decisión que ellos ayudaron a desarrollar. Las decisiones grupales incrementan la aceptación de la solución final y facilitan su instrumentación.

- Incrementan la Legitimidad: Los métodos democráticos son aceptados por todos los componentes de la sociedad. Cuando el proceso es grupal, intervienen todos los aditamentos de los ideales democráticos. Si el tomador de decisiones no consulta a

otros antes de tomar una de ellas, el hecho del poder que tiene no le exime de quedar como una persona autoritaria y arbitraria.

Las decisiones grupales no tienen la varita mágica de la perfección, pero sin lugar a dudas son las menos peligrosas y por lo tanto las que tienen un menor nivel de error.

- Reducción de los problemas de comunicación: Puesto que el grupo participa en la toma de decisión, todos sus integrantes están conscientes de la situación, por lo general la puesta en marcha de la solución se realiza sin tropiezos. Las preguntas, las objeciones y los obstáculos a los que normalmente se enfrenta la implantación de una decisión, con frecuencia desaparecen, cuando esta última es resultado de la participación del grupo.

Desventajas:

- Requieren mucho tiempo: El reunir al grupo toma su tiempo, pero con una buena organización, las reuniones estarán programadas de antemano en un espacio de tiempo oportuno (varía de acuerdo a la organización y no debe ser menor de dos semanas). El resultado es que los grupos consumen más tiempo en alcanzar una decisión a diferencia de un solo individuo.
- Presiones de aceptación: Si bien se supone que todos los miembros del grupo deben sentirse libre para expresar sus opiniones, sugerencias y recomendaciones, no deja de ser cierto que a veces existe cierta presión para que todo el mundo se reúna y acate el consenso general, llamado con frecuencia "Pensamiento grupal". Esta presión puede provocar que el grupo pase por alto un consejo o sugerencia positiva de algunos de los presentes.

Se presiona a los inconformes para que se ajusten y adhieran a la opinión de la mayoría.

En los grupos existen presiones sociales. El deseo de los miembros del grupo de ser aceptados y por lo tanto ser protagonistas, puede resultar en un intercambio de pareceres condicionado a deseos de una demostración de un liderazgo. Finalmente se llegará a un mismo resultado que necesariamente debe ser aceptado por todos para tener validez.

- Responsabilidad ambigua: Los miembros de un grupo tienen que compartir la responsabilidad, por lo tanto la individualidad se diluye, dándole un gran valor a los resultados.
- El Compromiso: En ciertas ocasiones el grupo se estanca y se muestra incapaz de llegar a un acuerdo sobre qué soluciones recomendar. Obligados a tomar una decisión, se alienta a los miembros a llegar a un compromiso o a darse por vencidos, aceptando una versión diferente de su solución. Este inconveniente es muy usual cuando el grupo se subdivide en grupos más pequeños, cada uno de los cuales apoya una solución diferente.

La toma de decisiones en grupo puede utilizarse con mucha eficiencia si el supervisor maneja la situación como debe ser. Uno de los factores más importantes consiste en ganarse el apoyo de los miembros del grupo; señalándoles el valor de sus aportes en la solución del problema. Un segundo enfoque muy útil consiste en dar a cada integrante del grupo elementos específicos en que pensar y trabajar, para que pueda reconocer sus aportes; también crear un entorno donde las personas puedan expresarse abierta y francamente y que estimule tanto los aportes creativos como las discusiones sobre las fallas o los errores en que podría incurrirse. Esto último es de especial importancia para evitar el surgimiento del Pensamiento Grupal.

En nuestro proyecto, los expertos no van a trabajar en grupo presencialmente, sino que trabajan de forma aislada, de manera que cada experto aporta sus valoraciones a las distintas alternativas del problema planteado y cuando se tengan todas estas valoraciones de todos los expertos que forman parte de dicho problema, éstas se agregan para formar el vector de utilidad que será la solución al problema.

2.3. Problemas TDME

En este apartado nos vamos a centrar en el tipo de problema que ocupa nuestro proyecto y lo detallaremos con más precisión.

2.3.1. Definición

Un problema de TDME está formado por un conjunto finito de alternativas $X = \{x_1, \ldots, x_n\}$ con n ≥ 2, así como un conjunto finito de expertos $E = \{e_1, \ldots, e_m\}$ con m ≥ 2, que tienen como objetivo alcanzar una solución al problema teniendo en cuenta todas las opiniones.

Para lograr este objetivo, cada experto ha de valorar cada una de las alternativas en base a su conocimiento sobre las mismas para que, posteriormente, todas las soluciones de los expertos sean agregadas en un único vector de utilidad que será la solución al problema planteado.

2.3.2. Modelo de Decisión

El proceso de resolución del problema de TDME consiste en la obtención de un conjunto solución de alternativas, $X_{sol} \subset X$, a partir de las preferencias proporcionadas por los expertos. El proceso de resolución se desarrolla generalmente aplicando dos procesos, que veremos en más detalle en los siguientes apartados:

(1) *Proceso de selección*

(2) *Proceso de consenso*.

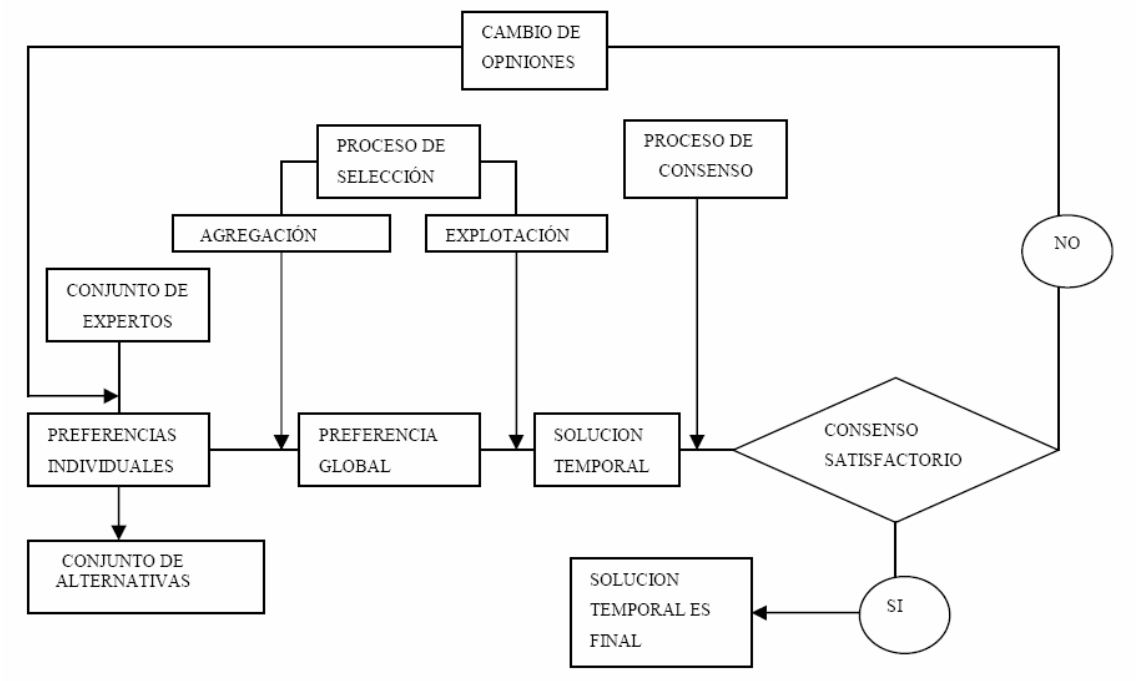


Figura 4. Diagrama del Proceso de resolución en un problema TDME-MC

2.3.2.1. Selección de alternativas

En este proceso se busca el conjunto solución de alternativa/s que mejor se adecue a los objetivos perseguidos en el problema de Decisión.

El proceso de selección se subdivide a su vez en dos etapas antes de elegir la(s) alternativa(s) solución(es): *Agregación* y *Explotación*.

En el siguiente apartado se hablará con más detalle de estos procesos, centrándonos en el de *Agregación*, que es el que ocupa nuestro proyecto.

2.3.2.2. Consenso

El proceso de consenso hace referencia a cómo alcanzar el mayor grado de acuerdo o coincidencia entre los individuos sobre el conjunto solución de alternativas.

Normalmente, al inicio de todo problema de TDME las opiniones de los expertos suelen diferir sustancialmente, en cuyo caso es necesario desarrollar un proceso de consenso en un intento de obtener una solución al problema sobre la que dicho conjunto de expertos muestren cierto grado de aceptación. Clásicamente, consenso se define como acuerdo unánime y total de todos los expertos sobre todas las posibles alternativas. Sin embargo, esta definición de consenso es inconveniente, ya que, en primer lugar, nos permite diferenciar tan sólo entre dos posibles estados, la existencia y la ausencia de consenso, y, en segundo lugar, las posibilidades de alcanzar tal acuerdo total, en caso de ser necesario, son prácticamente nulas.

Todo esto ha conducido a la definición y uso de un nuevo concepto de grado de consenso, el cual ha sido llamado grado de consenso suave (soft). Lo que todos los procesos de consenso intentan es alcanzar el máximo grado de consenso posible entre los individuos, aunque éste no sea ideal. En este sentido, un proceso de consenso se entiende como un proceso dinámico e iterativo, coordinado por un moderador, el cual ayuda a los individuos a acercar sus posiciones. Se asume, pues, que la sesión con los expertos consta de dos fases: *i)* la expresión de opiniones y *ii)* discusión en grupo. En cada paso del proceso de consenso, el moderador, una vez las opiniones de los expertos han sido expresadas, conoce el estado de consenso existente entre los individuos, a través de una medida de consenso, que establece la distancia al estado ideal de consenso. Si considera que el nivel de consenso no es aceptable, es decir, si aún existe mucha discrepancia entre los individuos, lo cual puede implicar un rechazo por parte de éstos de la solución final del problema de TDME, urgirá a los mismos a discutir con profundidad sus opiniones en un intento de acercar posiciones y de esta forma aumentar el acuerdo y en consecuencia el consenso entre ellos. Por el contrario, cuando el nivel de consenso es satisfactorio, el moderador aplicará el proceso de selección descrito para obtener la solución final de consenso del problema de TDME. El proceso de consenso se aplica tantas veces como sea necesario hasta que se logre alcanzar el nivel de consenso satisfactorio.

2.3.3. Selección de alternativas

En este apartado nos vamos a centrar solamente en la parte del proceso de resolución de problemas de TDME que ocupa nuestro proyecto, haciendo más hincapié en la importancia de la agregación de la información.

2.3.3.1. Introducción

Por proceso de selección se entiende el proceso mediante el cual se obtiene el conjunto de alternativas solución a partir de las preferencias individuales, sobre el conjunto de alternativas, de cada uno de los expertos implicados en el proceso de toma de decisión.

Para conseguir este objetivo, se ha de tener claro el criterio global (o de conjunto) a aplicar en la elección de la alternativas que formarán parte del conjunto solución. Este criterio global suele reducirse a una comparación de las alternativas entre sí para lo que normalmente se suele utilizar una función, llamada de selección, para asociar a cada alternativa un valor, llamado grado de selección, que se utilizará obviamente para producir un orden parcial de las alternativas.

En este proceso distinguimos dos fases, que se observan en la *Figura 3*:

a) *Agregación*. En todo problema de TDME, sobre la base de la información que proporcionan o de que disponen un conjunto de expertos, un paso necesario es el que consiste en la forma de agregar dicha información. Este es un problema muy importante y por ello muy estudiado.

   En este tipo de problemas, la agregación se realiza sobre las preferencias individuales que los expertos proporcionan sobre el conjunto de alternativas, de forma que la información obtenida tras el proceso de agregación, llamada preferencia global o de conjunto, sea resumen y reflejo de las propiedades contenidas en ellas.

b) *Explotación*. La fase de explotación actúa sobre la anterior información global de comparación para clarificar la decisión mediante la obtención de un orden parcial de las alternativas a partir del cual tomar la decisión. Este proceso puede realizarse de diversas formas, siendo el más habitual aquel que asocia un valor de utilidad global a cada alternativa, obteniendo de esta forma un orden natural de alternativas de mejor a peor.

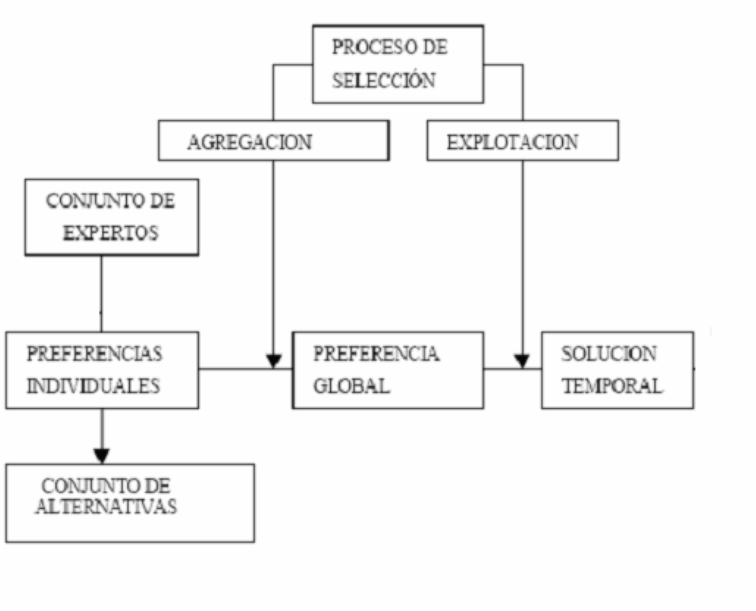


Figura 5. Proceso de Selección de alternativas

2.3.3.2. Importancia de la Agregación

El problema de la agregación de información ha sido ampliamente estudiado, existiendo una gran cantidad de publicaciones al respecto, entre los que cabe citar [7, 16, 17, 18]. La agregación de información de manera eficiente y flexible se ha convertido en la principal tarea de los problemas de acceso de información y otros problemas de decisión multicriterio, puesto que precisan procesar una gran cantidad de información cuya calidad y precisión es muy variada. En particular, resultan muy útiles las agregaciones entre el operador mínimo y el operador máximo a través de los operadores de medias.

Por otra parte, la toma de decisiones en las empresas suele tener un componente difuso que proporciona flexibilidad. Evidentemente no existe un único criterio para seleccionar los operadores de agregación, y esto ha hecho que se propongan algunas condiciones que se deben tener en cuenta para elegirlos:

1. *Fuerza axiomática*. En igualdad de condiciones, un operador es mejor cuanto menos limitado esté por los axiomas que satisface.

2. *Ajuste empírico*. Además de que los operadores satisfagan ciertos axiomas o tengan ciertas cualidades formales, deben reflejar adecuadamente la realidad, lo que normalmente sólo puede ser testado con pruebas empíricas.

3. *Adaptabilidad*. Los operadores tienen que adaptarse al contexto específico en el que se encuentran, esencialmente, mediante la parametrización. Por ejemplo, los operadores mín y máx no son nada flexibles, mientras que los operadores OWA pueden ser adaptados a ciertos contextos eligiendo parámetros adecuados.

4. *Eficiencia numérica*. El esfuerzo computacional de cálculo es especialmente importante cuando se tienen que resolver problemas grandes.

5. *Compensación y rango de compensación*. Cuanto mayor sea el grado en que se contrarrestan las funciones de pertenencia de los conjuntos difusos

agregados, el operador de agregación representará mejor las situaciones en las que unos atributos son compensados por otros.

6. *Comportamiento agrega*do. El grado de pertenencia de un conjunto difuso en el conjunto agregado depende muy frecuentemente del número de conjuntos combinados. De este modo, cada conjunto adicional añadido normalmente disminuirá los grados agregados de pertenencia resultantes, y este hecho debe ser tenido en cuenta cuando se elige el operador.

7. *Nivel de escala requerido de las funciones de pertenencia*. Diferentes operadores pueden requerir diferentes niveles de escala (nominal, intervalo, ratio o absoluto) de información de pertenencia para ser admisibles. En igualdad de condiciones, se prefiere el operador que requiere el nivel de escala más bajo.

2.4. Operadores de Agregación

En este apartado vamos a introducir los operadores que vamos a implementar en nuestro proyecto y más adelante los definiremos en detalle.

2.4.1 Introducción

Como ya hemos definido en apartados anteriores, entendemos por *agregación* la operación consistente en transformar un conjunto de elementos (conjuntos difusos, opiniones individuales sobre un conjunto de alternativas expresadas cardinalmente o lingüísticamente, etc.) en un único elemento representativo del mismo. En los problemas de TDME la agregación se realiza sobre las preferencias individuales que los expertos proporcionan sobre el conjunto de alternativas, de forma que la información obtenida tras el proceso de agregación, llamada preferencia global o de conjunto, sea resumen y reflejo de las propiedades contenidas en ellas.

Las preferencias sobre un conjunto de alternativas expresadas de forma numérica pueden considerarse como conjuntos difusos del conjunto de alternativas, por lo que el problema de agregación de preferencias se ha estudiado desde la perspectiva de la agregación de conjunto difusos, estudio que se convirtió en un tema de gran importancia desde las primeras publicaciones sobre conjuntos difusos debidas a Zadeh [1] y Goguen [2,3].

Una de las aproximaciones más directas para agregar preferencias es la utilización de procedimientos de agregación utilizados con frecuencia en la teoría de utilidad. Según Bonissone y Decker [4], existen tres clases básicas de operaciones de agregación: conjuntivas, disyuntivas y promedios. Las dos primeras formas de agregación se estudian utilizando t-normas y t-conormas, mientras que la tercera se hace con los llamados operadores de promedio. Estos operadores no cuentan con unos homólogos en la teoría clásica de conjuntos, y se localizan entre el operador mínimo y el operador máximo. Estos límites tienen una explicación muy intuitiva, ya que si se permiten compensaciones en presencia de criterios conflictivos, el promedio

resultante debería situarse entre la cota inferior más optimista y la cota superior más pesimista, es decir, entre la mejor y la peor estimación local [5].

Operadores tales como la media aritmética, media geométrica, ponderadas y no ponderadas, son ejemplos de operadores de agregación no paramétricos. Dentro de los operadores de agregación paramétricos destacan el operador "y compensatorio" de Zimmermann y Zysno [6], cuyo parámetro indica la localización entre el "y" lógico y el "o" lógico.

Otros de los operadores que siguen en la línea de compensación paramétrica son los obtenidos como combinaciones lineales convexas de cada una de las funciones de pertenencia de cada uno de los conjuntos difusos a agregar. Estos operadores, llamados operadores OWA fueron propuestos por Yager en [7] y más recientemente fueron caracterizados en [8], y constituyen una transición continua entre el operador mínimo y máximo. El problema añadido con este tipo de operadores es el del cálculo de los parámetros de la citada combinación lineal convexa, así como el de establecer un orden entre ellos. Yager soluciona el segundo de los problemas asociando no un agregado a un parámetro, sino al contrario, asociando un parámetro con un agregado, los cuales previamente han debido ser ordenados de mayor a menor. En Yager [7,8] y Zadeh [9] se proporcionan procedimientos para el cálculo de los parámetros de estos operadores, dando de esta forma posibles soluciones al primero de los problemas mencionados. Uno de los procedimientos es aquel que hace uso de un cuantificador lingüístico para el cálculo de los parámetros [9,10,11].

2.4.2. Operadores

En este apartado vamos a definir cada uno de los operadores con los que vamos a trabajar en nuestro proyecto:

- *Media aritmética*
- *Media ponderada*
- *Familia de operadores OWA*

2.4.2.1. MEDIA ARITMÉTICA

Definición 4. [14]. *Sea $x = \{x_1,...,x_n\}$ un conjunto de valores numéricos para una variable x. La media aritmética $\bar{x}$ se obtiene dividiendo la suma de todos los valores por* su cardinalidad, i.e.,

$$\bar{x}(x_1,...,x_n) = 1/n \sum_i x_i$$

Este operador simboliza el concepto intuitivo de punto de equilibrio o centro del conjunto de valores.

2.4.2.2. MEDIA PONDERADA

La media ponderada permite que diferentes valores x_i tengan diferente importancia. Esto se realiza asignando a cada valor x_i un peso asociado w_i, que indica cuál es la importancia de ese valor.

Definición 5.[15]. *Sea $x = \{x_1,...,x_n\}$ un conjunto de valores numéricos y $W = (w_1,...,w_n)$ un vector numérico con los pesos asociados a cada x_i, tal que, w_1 corresponde a x_1 y así sucesivamente. La media ponderada se obtiene como sigue:*

$$\bar{x} = \sum_i x_i w_i / \sum_i w_i$$

2.4.2.1. FAMILIA OWA

Dentro de esta familia de operadores, los que vamos utilizar en nuestro proyecto, y que pasaremos a definir en este apartado, son los siguientes:

- *OWA*
- *Or Like S-OWA*
- *And Like S-OWA*

OWA (Ordered Weighted Aggregation)

Este operador introducido por Yager en [12] es un operador de agregación ponderado, en el cuál, los pesos no están asociados a un valor predeterminado sino que están asociados a una posición determinada.

Un operador OWA se define como sigue [7]:

Definición 1. [12] *Un operador OWA de dimensión n es una función Φ,*

$$\Phi: [0,1]^n \rightarrow [0,1]$$

que tiene asociado un conjunto de pesos. Así, sea $\{a_1,...,a_n\}$ un conjunto de valores que deseamos agregar, entonces el operador OWA Φ se define como

$$\Phi(a_1,...,a_n) = W \cdot B^T = \sum_i w_i.b_i$$

donde $W = \{w_1,...,w_n\}$ es un vector de pesos verificando $w_i \in [0,1]$ y $\sum_i w_i = 1$; y B es el vector de valores ordenados, siendo $b_i \in B$ el i-ésimo mayor valor del conjunto $\{a_1,...,a_n\}$.

Una forma para obtener los pesos es utilizar la siguiente función:

$$w_i = Q(i/m) - Q(i-1/m), \qquad i=1,..., m$$

donde *m* es el número de valores que vamos a agregar y *Q* es el cuantificador lingüístico "*at least half*":

$$Q(x) = \begin{cases} 0 & si\ x < a \\ x-a/b-a & si\ a \le x \le b \\ 1 & si\ x>b \end{cases}$$

Se puede verificar de forma inmediata que los operadores OWA son conmutativos, crecientes e idempotentes, pero en general no son asociativos. Por otro lado, como hemos dicho, una cuestión natural en la definición del operador OWA es cómo obtener el vector de pesos asociados. Yager propuso dos formas distintas de obtenerlo. El primero consiste en la utilización de un mecanismo de aprendizaje de los parámetros por medio de algunos datos muestrales; mientras que el segundo método consiste en proporcionar a los pesos algún tipo de sentido semántico. Este último caso ha permitido múltiples aplicaciones en áreas tales como lógica difusa y lógica multivaluada, teoría de la evidencia, diseño de controladores difusos, así como en agregación de información guiada por un cuantificador.

Una de las principales ventajas que presenta el operador OWA es su gran flexibilidad a la hora de seleccionar el tipo de regla de agregación de las que hay disponibles. Esta flexibilidad se basa en la forma de determinar el vector de pesos que se va a utilizar en una determinada aplicación. Yager y Filev en [12] presentaron dos familias de operadores OWA llamadas S-OWA, la primera familia es la "orlike" y la segunda la "andlike".

OR-LIKE S-OWA

El operador "orlike" S-OWA, denotado como F_{SO}, es definido por una familia de pesos OWA, tales que con $\zeta \in [0, 1]$.

$$w_i = \begin{cases} \frac{1}{n}(1-\zeta)+\zeta, & i=1 \\ \frac{1}{n}(1-\zeta), & i=2,...,n. \end{cases}$$

Definición 2. [13] *Sea A = { a_1,...,a_n} un conjunto de valores numéricos para agregar, y los pesos a usar en la agregación calculados tal y como acabamos de indicar, un operador "orlike" S-OWA será definido como:*

$$F_{SO}(a_1,...,a_n) = \zeta \cdot \max_i\{a_i\} + \frac{1}{n}(1-\zeta)\cdot\sum_i a_i.$$

AND-LIKE S-OWA

El operador "andlike" S-OWA, denotado como F_{SA}, es definido por la familia de pesos OWA tal que con $\vartheta \in [0, 1]$

$$w_i = \begin{cases} \frac{1}{n}(1-\vartheta), & i = n \\ \frac{1}{n}(1-\vartheta)+\vartheta, & i = 1, ..., n-1. \end{cases}$$

Definición 3. [13] *Sea A = { $a_1,...,a_n$ } un conjunto de valores numéricos a ser agregados, utilizando los pesos anteriores, obtenemos un operador "andlike" S-OWA*:

$$F_{SA}(a_1, ..., a_n) = \vartheta \cdot \min_i\{a_i\} + \frac{1}{n}(1-\vartheta) \cdot \sum_i a_i.$$

CAPÍTULO 3

PROYECTO

CAPÍTULO 3

PROYECTO

Contenidos

3.1. Descripción

Una vez presentado el propósito y los objetivos del proyecto y realizada la introducción teórica a los Modelos de Toma de Decisiones y a la familia de operadores OWA llega el momento de pasar a detallar el desarrollo del proyecto que se va a realizar.

Este proyecto consta de dos partes:

1. La implementación de un sistema de ayuda a la Decisión con diferentes operadores de agregación para resolver problemas multiexperto.

2. Comparativa de los operadores de agregación utilizados para ver los resultados que ofrece cada uno de ellos.

La primera parte es un proyecto de desarrollo software y, como tal, para su desarrollo deben seguirse las actividades de la Ingeniería del Software. No existe una definición única y estandarizada para la Ingeniería del Software pero las dos que se presentan a continuación pueden resultar perfectamente válidas para este cometido:

- Ingeniería del Software es la construcción de software de calidad con un presupuesto limitado y un plazo de entrega en contextos de cambio continuo.

- Ingeniería del Software es el establecimiento y uso de principios y métodos firmes de ingeniería para obtener software económico que sea fiable y funcione de manera eficiente en máquinas reales.

Las actividades que conforman la Ingeniería del Software son las siguientes:

- **Especificación de Requerimientos**: se obtienen el propósito del sistema y las propiedades y restricciones del mismo.

- **Análisis del Sistema**: se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.

- **Diseño del Sistema**: se definen los objetivos del proyecto y las estrategias a seguir para conseguirlos.

- **Implementación**: se traduce el modelo a código fuente.

- **Prueba**: verificar y validar el sistema.

En los puntos siguientes se profundizará en cada una de estas actividades y en como se han llevado a cabo en el ámbito nuestro proyecto.

3.2. Especificación de Requerimientos

El primer paso en la Ingeniería del Software debe ser determinar el propósito último del proyecto, las propiedades que debe satisfacer y las restricciones a las que está sometido.

Este es, sin duda, una paso de vital importancia dentro del desarrollo de un proyecto software ya que, sin conocer el propósito del proyecto y todas las limitaciones de diversa índole a las que debe hacer frente, difícilmente se podrá realizar una aplicación software que cumpla dicho propósito.

En nuestro proyecto de tipo académico el propósito es conocido desde el mismo momento de la concepción del mismo:

> *Vamos a implementar un prototipo de un Sistema de Ayuda a la Decisión (SAD), utilizando el modelo de decisión de la Figura 1, que sea capaz de abordar problemas de decisión Multiexperto, y que nos permita utilizar diferentes operadores de agregación en su proceso de resolución.*

Habiendo determinado el propósito último del proyecto, el siguiente paso consiste en especificar los requerimientos del mismo. Los requerimientos de un proyecto software son el conjunto de propiedades o restricciones definidas con total precisión, que dicho proyecto software debe satisfacer. Existen dos tipos bien diferenciados de tales requerimientos:

I. *Requerimientos funcionales*: aquellos que se refieren específicamente al funcionamiento de la aplicación o sistema.

II. *Requerimientos no funcionales*: aquellos no referidos al funcionamiento estricto sino a otros factores externos.

En los dos siguientes subapartados definiremos cuales son estos requerimientos (tanto funcionales como no funcionales) para el proyecto del que se ocupa esta memoria. Sin embargo, estas definiciones sólo serán previas ya que en la actividad de análisis del sistema, donde se crearán los casos de uso y sus escenarios,

se descubrirán nuevas necesidades que no son observables en esta primera actividad y que permitirán refinar completamente estos requerimientos.

3.2.1. Requerimientos Funcionales

Los requerimientos funcionales de un sistema software son aquellos que se encargan de describir las funcionalidades que el sistema debe proporcionar a los usuarios del mismo para cumplir sus expectativas.

Las funcionalidades que se esperan de un sistema como el nuestro son las siguientes:

Para ambos usuarios (*Administrador* y *Experto):*

a) Identificación y validación cuando un usuario entra en el sistema mediante una base de datos.

b) Salir del sistema.

Para el usuario *Administrador:*

a) Crear un nuevo problema.

b) Asignar problemas.

c) Consultar las soluciones de un problema.

d) Resolver problemas.

e) Crear una nueva alternativa.

f) Dar de alta a un nuevo *Experto*.

Para el usuario *Experto:*

a) Valorar un problema.

b) Consultar las valoraciones.

Una vez definidas cuales son las funcionalidades que los distintos usuarios pueden reclamar a nuestro sistema, se hace necesario caracterizar de una manera más formal y concreta como va a responder a estas funcionalidades nuestro sistema. Además, para distinguir cada uno de los requisitos utilizaremos este identificador*: RF-Número_del_Requisito:*

1) **RF-01: Identificación y validación cuando un usuario entra en el sistema mediante una base de datos.**
   Para poder entrar en nuestra aplicación, el sistema debe proporcionar a un usuario un formulario en el que tiene que introducir su nombre de usuario y su contraseña. El sistema valida estos datos mediante una base de datos, y si son correctas el usuario accede a la aplicación.

2) **RF-02: Salir del sistema.**
   En los menús de usuario, ya sea éste *Administrador* o *Experto*, el sistema proporciona esta opción para salir de la aplicación, mostrando de nuevo el formulario de entrada para volver a entrar en el sistema.

3) **RF-03: Crear un nuevo problema.**
   El sistema debe proporcionar al *Administrador* la posibilidad de crear nuevos problemas, e informar si un problema determinado no puede crearse y el por qué de esto.

4) **RF-04: Asignar problemas.**

El sistema debe proporcionar al *Administrador* la posibilidad de asignar a cada problema un conjunto de expertos y de alternativas, todos ellos previamente creados y guardados en la base de datos. Por defecto se selecciona el primer problema creado, el nombre del primer experto creado y de la primera alternativa creada para que no haya errores si se pulsa accidentalmente el botón de *Aceptar* antes de que un problema sea asignado.

5) **RF-05: Consultar las soluciones de un problema.**

El sistema debe proporcionar al *Administrador* la posibilidad de consultar las soluciones de un problema, una vez que éstas han sido calculadas, y en caso de que aún no hayan sido calculadas, el sistema deberá informar de este suceso.

6) **RF-06: Resolver problemas.**

El sistema debe proporcionar al *Administrador* la posibilidad de resolver un problema asignado mediante diferentes operadores, siempre y cuando sus alternativas hayan sido previamente valoradas por los expertos que componen dicho problema. Por defecto aparece seleccionado el primer operador para que no se produzca ningún error si se pulsa el botón de *Aceptar* sin darse cuenta. Si el problema escogido para ser resuelto no ha sido valorado aún, el sistema deberá informar de ello.

7) **RF-07: Crear una nueva alternativa.**

El sistema debe proporcionar al *Administrador* la posibilidad de crear nuevas alternativas, e informar si una alternativa determinada no puede crearse y el por qué de esto.

8) **RF-08: Dar de alta a un nuevo *Experto*.**

El sistema debe proporcionar al *Administrador* la posibilidad de dar de alta a un nuevo *Experto*, e informar si un *Experto* determinado no puede crearse y el por qué de esto.

9) **RF-09: Valorar un problema.**

El sistema debe proporcionar al *Experto* la posibilidad de poder valorar los problemas que tenga asignados.

10) RF-10: Consultar las valoraciones.

El sistema debe proporcionar al *Experto* la posibilidad de consultar las valoraciones que ha hecho previamente de un problema que tenga asignado. Si dicho problema no ha sido valorado aún, el sistema deberá informar al *Experto* de este suceso.

3.2.2. Requerimientos no Funcionales

Los requerimientos no funcionales son aquellos que restringen los requerimientos funcionales. Son tan importantes como los propios requerimientos funcionales y pueden incluso a llegar a ser críticos para la aceptación del sistema.

Estos requerimientos normalmente especifican propiedades del sistema o del producto en si (plataforma, velocidad, rendimiento...) y del diseño de la interfaz gráfica con el usuario además de todas las restricciones impuestas por la organización (políticas de empresa, estándares, legalidad vigente...).

Los requerimientos no funcionales que se deben obtener y analizar para este proyecto son los referentes a las necesidades hardware y software de los equipos informáticos para que éstos proporcionen al usuario las funcionalidades requeridas de forma eficiente y los referentes a la interfaz gráfica entre la aplicación y el usuario.

A. Requerimientos del equipo informático

Al hablar de los requerimientos del equipo informático y debido a que el marco del desarrollo de la aplicación es una arquitectura cliente/servidor, debemos diferenciar los requerimientos de equipo que necesita el servidor y los que necesita el cliente.

Las necesidades de equipo informático del cliente son muy simples ya que tan solo le hace falta un computador conectado a Internet (preferiblemente

de banda ancha) y tener instalado un navegador capacitado para visualizar de forma correcta la aplicación (se recomienda Internet Explorer pero podría ser válido cualquier otro).

Los requerimientos del equipo informático del servidor, el cual se aconseja que sea un equipo dedicado, son más amplios y se dividen en dos tipos: los requerimientos de hardware y los requerimientos software.

1. Hardware

- *Velocidad*: el equipo debe ser lo suficientemente rápido como para ejecutar la aplicación en el menor tiempo posible y con la mayor fiabilidad. Cualquier microprocesador actual es capaz de cumplir con esta labor.

- *Memoria*: el equipo debe disponer de la suficiente memoria RAM libre para realizar las operaciones que se soliciten entre la aplicación y la base de datos.

- *Almacenamiento*: el equipo que haga la labor de servidor debe tener una capacidad de almacenamiento suficiente para almacenar la base de datos con la que trabaja la aplicación y permitir con holgura las transacciones entre ambas entidades.

- *Tarjeta gráfica*: las tarjetas gráficas de las que disponen los equipos informáticos actuales son de gran potencia por lo que es inútil establecer ningún requerimiento en este aspecto.

- *Monitor*: el monitor debe soportar una resolución de 1024x768 y superiores.

- *Conexión a Internet*: el servidor debe encargarse de que la aplicación sea accesible a través de Internet para todos sus usuarios por lo que es indispensable que se encuentre conectado a Internet a través de banda ancha las 24 horas del día.

2. Software

- *Sistema Operativo*: el servidor de la aplicación trabaja sobre un sistema operativo Windows XP.

- *Navegador*: la aplicación debe poder ser visualizada desde cualquier navegador Web actual aunque se recomienda el uso de Internet Explorer en sus últimas versiones.

- *Sistema Gestor de Bases de Datos*: la aplicación trabaja con la base de datos *phpMyAdmin Database Manager Version 2.6.4-pl4.*

- El resto del software necesario será proporcionado al administrador de la aplicación, el cual dispone de un manual para su instalación en el Anexo II.

B. Requerimientos de la interfaz

Los requerimientos de la interfaz gráfica entre la aplicación y el usuario están íntimamente ligados a la *usabilidad* y sus principios. La usabilidad se puede definir de varias formas [17]:

- Usabilidad se define coloquialmente como facilidad de uso, ya sea de una página Web, una aplicación informática o cualquier otro sistema que interactúe con un usuario.

- Usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

- Usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico.

A partir de estas tres definiciones se pueden obtener los principios básicos de la usabilidad, los cuales se asociarán a los requerimientos no funcionales que deberá cumplir la interfaz gráfica:

- ***Facilidad de aprendizaje***: se refiere a la facilidad con la que nuevos usuarios pueden tener una interacción efectiva. Depende de los siguientes factores:
  - *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.
  - *Síntesis*: los cambios de estado tras una acción deben ser fácilmente captados.
  - *Generalización*: las tareas semejantes se resuelven de modo parecido.
  - *Familiaridad*: el aspecto de la interfaz tiene que resultar conocido y familiar para el usuario.
  - *Consistencia*: siempre se han de seguir una misma serie de pasos para realizar una tarea determinada.

- ***Flexibilidad***: relativa 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.

- ***Robustez***: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos o, también, la capacidad del sistema para tolerar fallos. Esta relacionada con los siguiente factores:
  - *Navegable*: el usuario debe poder observar el estado del sistema sin que esta observación repercuta de forma negativa en él.
  - *Recuperación de información*: la aplicación permitir volver a un estado anterior.
  - *Tiempo de respuesta*: es el tiempo necesario para que el sistema pueda mostrar los cambios realizados por el usuario.

3.3. Análisis del Sistema

Una vez conocido el propósito del proyecto software, las propiedades que debe cumplir y las restricciones a las que debe someterse, llega el momento de analizar el sistema y crear un modelo del mismo que sea correcto, completo, consistente, claro y verificable. Para conseguir esto se estudiarán los perfiles de usuario y se crearán y definirán casos de uso en base a los requerimientos previamente obtenidos. Por último se describirán ciertos escenarios de acción de dichos casos de uso.

3.3.1. Perfil de Usuario

En esta fase el primer paso es determinar quienes son los usuarios potenciales de la aplicación, para a partir de esto, obtener las características generales que nos permitan caracterizar los requisitos de usabilidad que posteriormente habrá que tener en cuenta en el diseño de la aplicación y de su interfaz gráfica. Nuestro sistema, cuenta con dos tipos de usuarios: administrador y experto.

Pasamos a comentar las características de cada uno de los usuarios de nuestro sistema:

Administrador:

- *Conocimientos del dominio del problema*: debe tener un buen conocimiento del dominio para saber resolver un problema (saber cómo funcionan los operadores y qué cuantificador lingüístico, peso o valor elegir en un determinado momento).

- *Sobre uso de equipos/programas informáticos*: es importante que tenga conocimientos sobre programas informáticos, ya que será el encargado del mantenimiento de la base de datos. También deberá tener conocimientos de servidores Tomcat, que es el utilizado en la aplicación. En cuanto al uso de equipos informáticos debe tener conocimientos a nivel experto del manejo de un servidor.

- *Entorno de trabajo*: el lugar de trabajo será una oficina, estará situado en un punto concreto, por lo cual no necesitará el uso de ningún software o hardware especial.

- *Nivel cultural*: se presupone un nivel cultural medio-alto, necesario para la comprensión de sus resultados asociados.

- *Habilidades sociales*: el administrador debe tener habilidades de trato con los expertos, y también será importante que sepa trabajar en grupo y tener habilidades de interrelación personal con sus compañeros de trabajo.

Experto:

- *Conocimientos del dominio del problema*: debe tener un conocimiento experto específico del dominio del problema como para saber valorar las alternativas de un problema de una forma acertada.

- *Sobre uso de equipos/programas informáticos:* no es necesario que tenga conocimientos sobre otros programas informáticos, sólo sobre nuestra aplicación y el navegador que utilice, ya que no será necesario el uso de ningún otro programa específico. En cuanto al manejo de equipos si deberá tener unos conocimientos básicos a nivel de usuario como son: encendido y apagado del equipo, manejo del ratón, teclado, etc.

- *Entorno de trabajo:* el lugar de trabajo estará situado en un punto concreto, por lo cual no necesitará el uso de ningún software o hardware especial.

- *Nivel cultural:* se presupone un nivel cultural medio-alto, dependiendo del problema.

- *Habilidades sociales:* puesto que no es una ocupación de cara al público, el evaluado no tiene porque tener habilidades específicas de trato hacia el público.

3.3.2. Casos de uso

Un caso de uso representa una clase de funcionalidad dada por el sistema como un flujo de eventos. También se puede definir como la representación de una situación o tarea de interacción de un usuario con la aplicación.

Los casos de uso son tareas con significado, coherentes y relativamente independientes, que los actores realizan en su trabajo cotidiano. En un caso de uso concreto puede participar más de un actor.

Los casos de uso describen como se realiza una tarea de manera exacta y constan de los siguientes elementos:

- Nombre único e unívoco
- Actores participantes
- Condiciones de entrada
- Flujo de eventos
- Condiciones de salida
- Requerimientos especiales

Por lo tanto, es necesario determinar cuales son los actores participantes en cada no de los casos de uso. Un actor modela una entidad externa que se comunica con el sistema, es decir, es un tipo de usuario del sistema. Un actor, al igual que un caso de uso, debe tener un nombre único y puede tener una descripción asociada.

En nuestro sistema contamos con los tres actores siguientes:

- *Administrador*: es el responsable de la aplicación, el que se encarga de definir los problemas y calcular las soluciones en base a los operadores de agregación implementados.

- *Experto*: es cada una de las personas (expertas) que da sus valoraciones sobre las alternativas que forman parte de cada problema.

- *BBDD*: es la base de datos que proporciona los datos a la aplicación (para la validación del administrador y expertos, para relacionar cada problema con su conjunto de alternativas y de expertos, para controlar que cada experto valore el conjunto de alternativas de un problema solamente una vez, etc.).

Una vez definidos cuales van a ser los actores del sistema, es el momento de crear los distintos casos de uso. A la hora de realizar esta acción es importante que cada uno de los requerimientos funcionales ya definidos aparezca en al menos uno de los casos de uso aunque, por otra parte, puede haber casos de uso nuevos, en los que no aparezca ninguno de los requerimientos, ya que estamos en una fase de refinamiento del sistema donde queremos construir un modelo detallado del mismo.

Un paso previo a la creación y descripción de los distintos casos de uso es la obtención de los diversos diagramas de casos de uso de nuestro sistema. El primero es un diagrama frontera, es decir, un diagrama que describe completamente la funcionalidad de un sistema:

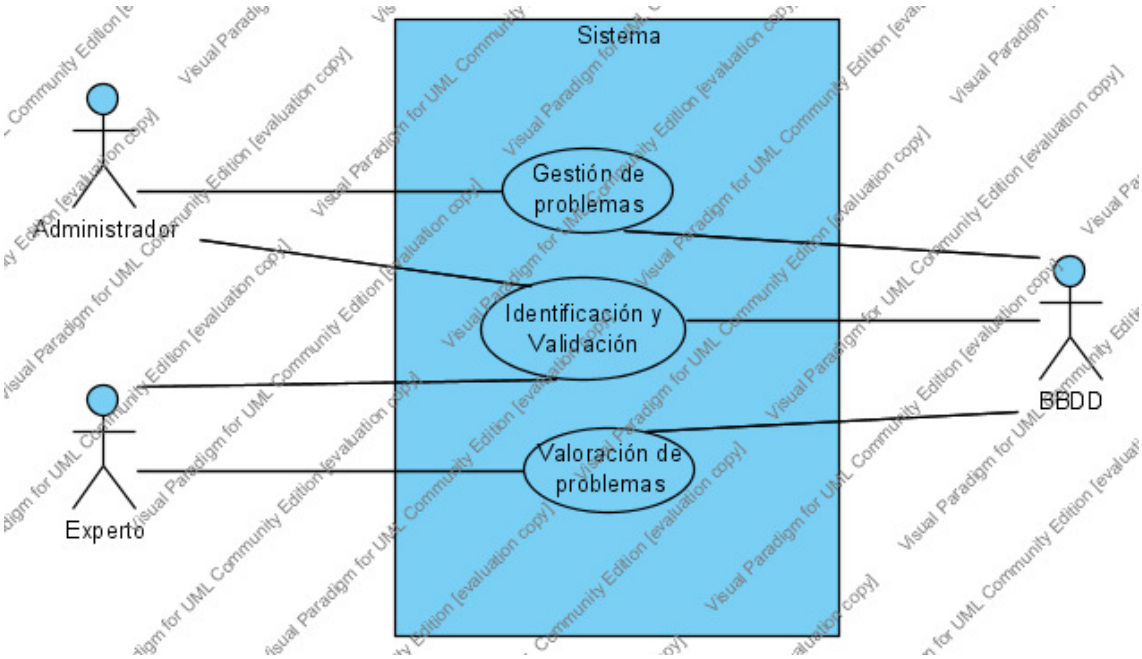


Figura 6. Diagrama frontera del Sistema

Los casos de uso mostrados en un diagrama frontera pueden ser lo suficientemente exactos o, por el contrario, pueden ser concretados con un mayor detalle. A la hora de detallar un caso de uso se pueden emplear dos tipos de relaciones:

- *<<extend>>:* es una relación cuya dirección es hacia el caso de uso a detallar que representa comportamientos excepcionales del caso de uso.

- *<<include>>:* es una relación cuya dirección es contraria a la de la relación *<<extend>>* que representa un comportamiento común del caso de uso

En nuestro caso nos encontramos con que los casos de uso *"Gestión de problemas*" y "*Valoración de problemas*" requieren ser detallados en más profundidad. En las figuras *Figura 7* y *Figura 8* mostramos los diagramas de casos de uso de estos dos casos:

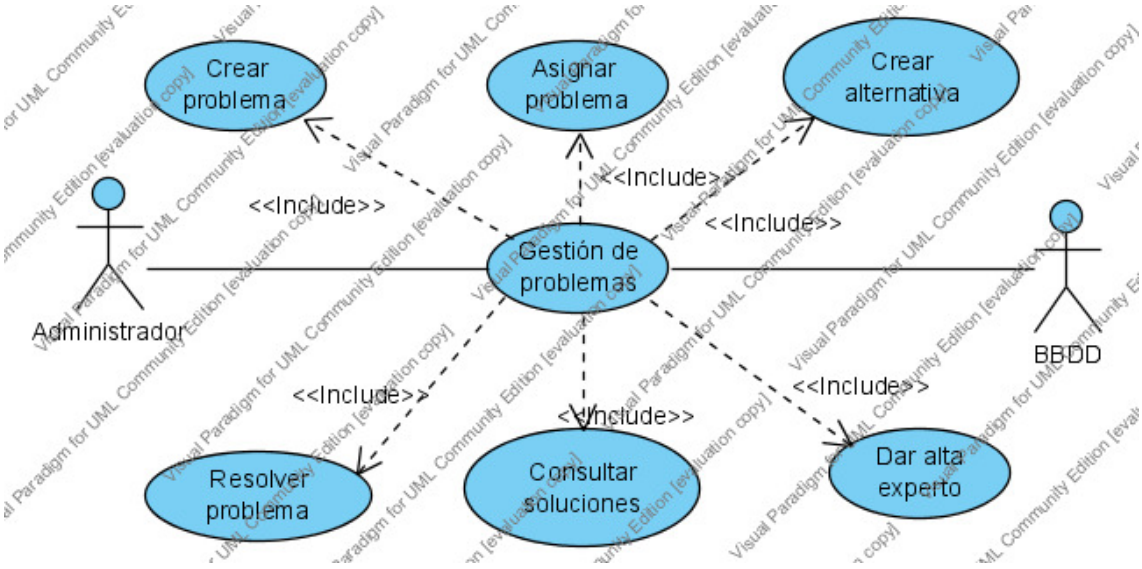


Figura 7. Diagrama del caso de uso "Gestión de problemas".

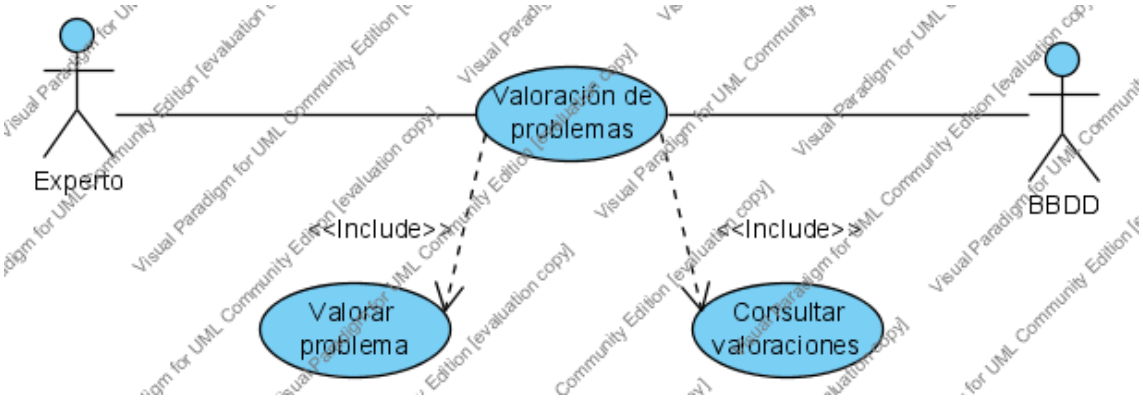


Figura 8. Diagrama del caso de uso "Valoración de problemas".

A continuación, se describen detalladamente cada uno de los casos de uso mostrados en las figuras anteriores.

- **Caso de Uso 1: Identificación y Validación**

Actores participantes: *Administrador*, *Experto* y *BBDD*.

Condiciones de entrada: Que existan cuentas de usuario en la aplicación.

Flujo de eventos:

1. El sistema muestra un formulario de entrada.
2. El usuario introduce su nombre de usuario y contraseña y pulsa el botón *Entrar*.
3. El sistema comprueba que ambos datos son válidos (E-1).
4. Según el identificador sea de:
    4.1. *Administrador*, entonces el usuario entra al sistema y este le muestra el menú de administrador.
    4.2. *Experto*, entonces el usuario entra al sistema y este le muestra el menú de experto.

Condiciones de salida: El nombre de usuario y la contraseña han sido comprobadas.

Excepciones:

E-1: El nombre de usuario y/o la contraseña introducido/s por el usuario no es/son válido/s. El sistema informa al usuario de dicha situación mostrando un mensaje de error.

- Caso de Uso 2: Crear Problema

Actores participantes: *Administrador* y *BBDD*.

Flujo de eventos:

1. El sistema muestra un formulario para la creación del problema.
2. El *Administrador* introduce el código y la descripción del problema que quiere crear y pulsa el botón *Crear*.
3. El sistema introduce los datos del nuevo problema en la base de datos y muestra una notificación confirmando que los cambios han sido guardados correctamente (E-1).

Condiciones de salida: El *Administrador* ha creado un nuevo problema.

Excepciones:

E-1: Si el código del problema que el *Administrador* intenta crear ya existe en la base de datos porque está siendo utilizada por otro problema, el sistema muestra una notificación de error.

- **Caso de Uso 3: Asignar Problema**

Actores participantes: *Administrador* y *BBDD*.

Condiciones de entrada: Que existan problemas, expertos y alternativas.

Flujo de eventos:

1. El sistema obtiene de la base de datos los nombres del problema, *Expertos* y alternativas (E-1).
2. El *Administrador* elige el problema al que le quiere asignar un conjunto de *Expertos* y de alternativas.
3. El sistema marca ese problema.
4. El *Administrador* selecciona el conjunto de expertos para ese problema.
5. El sistema deja marcados ese conjunto de expertos.
6. El *Administrador* selecciona el conjunto de alternativas para el problema.
7. El sistema deja marcados ese conjunto de alternativas.
8. El *Administrador* pulsa el botón *Aceptar*.
9. El sistema muestra una notificación confirmando que los cambios han sido guardados correctamente.

Condiciones de salida: El *Administrador* ha asignado a un problema un conjunto de *Expertos* y de alternativas.

Excepciones:

E-1: Estos datos (problema, *Expertos* y alternativas) se obtienen accediendo a la base de datos, y puede ser que se produzca un error cuando el sistema intenta comunicarse con dicha base de datos.

- Caso de Uso 4: Consultar problema

Actores participantes: *Administrador* y *BBDD*.

Condiciones de entrada: Que exista al menos un problema resuelto para que pueda/n ser consultada/s su/s solución/es.

Flujo de eventos:

1. El sistema obtiene de la base de datos las descripciones de los problemas que ya están resueltos (E-1).
2. El *Administrador* selecciona el problema que desea consultar.
3. El sistema muestra una tabla con los operadores utilizados para resolver el problema y sus soluciones (E-2).

Condiciones de salida: El *Administrador* observa una tabla con los operadores que utilizó para resolver el problema, junto con sus soluciones.

Excepciones:

E-1: Esos datos se obtienen accediendo a la base de datos, y puede ser que se produzca un error cuando el sistema intenta comunicarse con dicha base de datos.

E-2: Esos datos se obtienen accediendo a un fichero, y puede ser que se produzca un error cuando el sistema abrir dicho fichero.

- **Caso de Uso 5: Resolver problema**

Actores participantes: *Administrador* y *BBDD*.

Condiciones de entrada: Que exista al menos un problema cuyas alternativas hayan sido valoradas por todos los expertos que forman parte de ese problema.

Flujo de eventos:

1. El sistema obtiene de la base de datos las descripciones de los problemas que aún no están resueltos (E-1).
2. El *Administrador* selecciona el problema que desea resolver.
3. El sistema muestra una lista de operadores para resolver el problema.
4. El *Administrador* elige un operador y pulsa el botón *Aceptar.*
5. El sistema muestra el resultado del problema.

Condiciones de salida: El *Administrador* calcula la solución a un problema.

Excepciones:

E-1: Esos datos se obtienen accediendo a la base de datos, y puede ser que se produzca un error cuando el sistema intenta comunicarse con dicha base de datos.

- Caso de Uso 6: Crear alternativa

Actores participantes: *Administrador* y *BBDD*.

Flujo de eventos:

1. El sistema muestra un formulario para la creación de la alternativa.
2. El *Administrador* introduce el código y la descripción de la alternativa que quiere crear y pulsa el botón *Crear*.
3. El sistema introduce los datos de la nueva alternativa en la base de datos y muestra una notificación confirmando que los cambios han sido guardados correctamente (E-1).

Condiciones de salida: El *Administrador* ha creado una nueva alternativa.

Excepciones:

E-1: Si el código de la alternativa que el *Administrador* intenta crear ya existe en la base de datos porque está siendo utilizada por otra alternativa, el sistema muestra una notificación de error.

- **Caso de Uso 7: Dar alta experto**

Actores participantes: *Administrador* y *BBDD*.

Flujo de eventos:

1. El sistema muestra un formulario para dar el alta a un nuevo *Experto*.
2. El *Administrador* introduce el código y la descripción del *Experto* al que quiere dar el alta y pulsa el botón *Alta*.
3. El sistema introduce los datos del nuevo *Experto* en la base de datos y muestra una notificación confirmando que los cambios han sido guardados correctamente (E-1).

Condiciones de salida: El *Administrador* ha dado de alta a un nuevo *Experto*.

Excepciones:

E-1: Si el código del *Experto* que el *Administrador* intenta dar de alta ya existe en la base de datos porque está siendo utilizado por otro *Experto*, el sistema muestra una notificación de error.

- **Caso de Uso 8: Valorar problema**

Actores participantes: *Experto* y *BBDD*.

Condiciones de entrada: Que exista al menos un problema que el *Administrador* haya asignado previamente para que el *Experto* pueda valorar sus alternativas.

Flujo de eventos:

1. El sistema obtiene de la base de datos las descripciones de los problemas que el *Administrador* ha asignado al *Experto* (E-1).
2. El *Experto* selecciona el problema que desea valorar.
3. El sistema muestra la/s alternativa/s que debe/n ser valorada/s.
4. El *Experto* elige un valor para cada alternativa dentro de un rango dado y pulsa *Aceptar.*
5. El sistema muestra una notificación para comunicarle al *Experto* que sus valoraciones han sido guardadas correctamente.

Condiciones de salida: El *Experto* ha valorado la/s alternativa/s correspondiente/s a un problema.

Excepciones:

E-1: Estos datos se obtienen accediendo a la base de datos, y puede ser que se produzca un error cuando el sistema intenta comunicarse con dicha base de datos.

- **Caso de Uso 9: Consultar valoraciones**

Actores participantes: *Experto* y *BBDD*.

Condiciones de entrada: Que exista al menos un problema que el *Experto* haya valorado previamente para poder consultar sus alternativas.

Flujo de eventos:

1. El sistema obtiene de la base de datos las descripciones de los problemas que ya ha valorado el *Experto* (E-1).
2. El *Experto* selecciona el problema que desea valorar.
3. El sistema muestra una tabla con la/s descripción/es de la/s alternativa/s y su/s valoración/es, y ofrece la opción de volver a valorar las alternativas (E-2).

Condiciones de salida: El *Experto* ha podido ver las valoraciones correspondientes a un problema.

Excepciones:

E-1: Estos datos se obtienen accediendo a la base de datos, y puede ser que se produzca un error cuando el sistema intenta comunicarse con dicha base de datos.

E-2: Esos datos (valoraciones) se obtienen accediendo a un fichero, y puede ser que se produzca un error cuando el sistema abrir dicho fichero.

3.3.3. Escenarios

Un caso de uso es una representación abstracta, una abstracción, de una funcionalidad del sistema a realizar. 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.

Los escenarios son historias ficticias que describen posibles interacciones con una interfaz. Permiten a los diseñadores anticiparse a los problemas. Aunque son historias ficticias deben hacerse lo más detalladas posible, así por ejemplo, los personajes deben tener nombres, motivaciones para usar la interfaz, 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. Nosotros vamos a definir un solo escenario para cada caso de uso, así se tendrá un ejemplo de las principales funcionalidades del sistema:

- Escenario *IdentificaciónDelAdministrador.*
- Escenario *AsignarProblema'CochesConMayorPotencia'.*
- Escenario *ResolverProblema'CochesConMayorPotencia'.*
- Escenario *ValorarProblema'CochesConMayorPotencia'.*

Nombre: *IdentificaciónDelAdministrador*.

Descripción: El administrador del sistema *Isidro Cuevas* quiere entrar al sistema y para ello debe identificarse.

Actores participantes: *Isidro Cuevas* y *BBDD*.

Flujo de eventos:

1. El sistema muestra el formulario de entrada para la identificación del usuario.
2. *Isidro Cuevas* introduce su nombre de usuario y su contraseña.
3. El sistema valida correctamente al usuario mediante la BBDD y éste entra en la aplicación como administrador del sistema.

Figura 9. Escenario IdentificaciónDelAdministrador.

Nombre: *AsignarProblema'CochesConMayorPotencia'.*

Descripción: El administrador del sistema *Isidro Cuevas* quiere asignar al problema '*Coches con mayor potencia'*, previamente creado, un conjunto de expertos y alternativas.

Actores participantes: *Administrador* y *BBDD*.

Flujo de eventos:

1. *Isidro Cuevas* se identifica y entra a la aplicación.
2. El sistema muestra el menú principal del administrador.
3. *Isidro Cuevas* elige la opción de *Asignar un problema*.
4. El sistema se comunica con la BBDD y obtiene los nombres del problema, *Expertos* y alternativas
5. *Isidro Cuevas* elige el problema '*Coches con mayor potencia'*.
6. El sistema marca ese problema como seleccionado.
7. *Isidro Cuevas* selecciona el conjunto de expertos para ese problema.
8. El sistema deja marcados ese conjunto de expertos.
9. *Isidro Cuevas* selecciona el conjunto de alternativas para el problema.
10. El sistema deja marcados ese conjunto de alternativas.
11. *Isidro Cuevas* pulsa el botón de *Aceptar*.
12. El sistema muestra una notificación confirmando que los cambios han sido guardados correctamente.

Figura 10. Escenario AsignarProblema'CochesConMayorPotencia'.

Nombre: *ResolverProblema'CochesConMayorPotencia'*.

Descripción: El administrador del sistema *Isidro Cuevas* quiere calcular la solución al problema '*Coches con mayor potencia'* usando el operador de agregación *OWA* y el cuantificador lingüístico *Most*.

Actores participantes: *Isidro Cuevas y BBDD*.

Flujo de eventos:

1. *Isidro Cuevas* se identifica y entra a la aplicación.
2. El sistema muestra el menú principal del administrador.
3. *Isidro Cuevas* elige la opción *Resolver un problema*.
4. El sistema se comunica con la BBDD y obtiene las descripciones de los problemas y los muestra.
5. *Isidro Cuevas* elige el problema '*Coches con mayor potencia'*.
6. El sistema muestra una lista de operadores de agregación.
7. *Isidro Cuevas* selecciona el operador *OWA*.
8. El sistema muestra una lista de cuantificadores lingüísticos.
9. *Isidro Cuevas* selecciona el cuantificador lingüístico *Most* y pulsa *Aceptar*.
10. El sistema muestra una tabla con el resultado del problema.

Figura 11. Escenario Resolver'CochesConMayorPotencia'.

Nombre: *ValorarProblema'CochesConMayorPotencia'*.

Descripción: El experto *Antonio Ruiz Salcedo* quiere entrar al sistema para valorar el problema '*Coches con mayor potencia'*.

Actores participantes: *Antonio Ruiz Salcedo y BBDD*.

Flujo de eventos:

1. *Antonio Ruiz Salcedo* identifica y entra a la aplicación.
2. El sistema muestra el menú principal del experto.
3. *Antonio Ruiz Salcedo* elige la opción *Valorar un problema*.
4. El sistema muestra los problemas que tiene asignados el experto.
5. *Antonio Ruiz Salcedo* elige el problema '*Coches con mayor potencia'*.
6. El sistema muestra las alternativas para que sean valoradas.
7. *Antonio Ruiz Salcedo* valora cada una de las alternativas y pulsa *Aceptar*.
8. El sistema muestra una notificación confirmando que las valoraciones han sido guardadas correctamente.

Figura 12. Escenario Valora'CochesConMayorPotencia'.

3.4. Diseño del Sistema

Sin duda, realizar de manera adecuada cada una de las actividades que conlleva la Ingeniería del Software es indispensable para la realización de un proyecto software de calidad. Por lo tanto, no se puede decir que ninguna de estas actividades sea más importante que otra. Sin embargo, si podemos decir que la actividad de diseño es la más delicada y la más laboriosa de llevar a cabo.

- Es delicada porque si no se lleva a cabo correctamente se hace imposible el codificar, de manera correcta, en la actividad de implementación el modelo obtenido en el análisis del sistema, lo que puede repercutir en el desperdicio de todo el esfuerzo realizado durante las primeras actividades de la Ingeniería del Software.

- Y es laboriosa porque las estrategias realizadas para conseguir que esta traducción entre modelo y código se lleve a cabo correctamente son muy diversas y bastante complejas.

Se puede decir, por tanto, que el diseño del sistema es la actividad de la Ingeniería del Software en la que se identifican los objetivos finales del sistema y se plantean las diversas estrategias para alcanzarlos en la actividad de implementación.

Sin embargo, el sistema no se suele diseñar de una sola vez sino que hay que diferenciar entre el diseño y estructura de los datos que se van a manejar y el diseño de la interfaz entre la aplicación y el usuario. Estas dos fases del diseño no se realizan de forma consecutiva una detrás de la otra sino que lo normal es realizarlas de manera concurrente y finalizarlas a la vez.

Las distintas fases de diseño que veremos a continuación son:

- **Diagrama de clases**: muestran la estructura del sistema.
- **Diseño de los datos**: estructura de los datos.
- **Diseño de la interfaz**: se define la apariencia visual de la aplicación.

3.4.1. Diagrama de clases

Los diagramas de clases se utilizan para mostrar la estructura estática del sistema modelado. Pueden estar contenidos por clases, interfaces, paquetes, relaciones e incluso instancias, como objetos o enlaces.

Son una potente herramienta de diseño, ayudando a los desarrolladores a planificar y establecer la estructura del sistema y subsistema antes de escribir ningún código. Esto permite asegurar que el sistema está bien diseñado desde el principio.

Son utilizados en la fase de diseño prácticamente en la totalidad de los sistemas que utilizan UML para su modelado.

Los diagramas de clases tienen los siguientes componentes:

- Clases: son los componentes fundamentales de los diagramas de clase. Su notación general es un rectángulo dividido en tres secciones, mostrando la primera el nombre de la clase, la siguiente los atributos y la última las operaciones.

- Relaciones: es una conexión semántica entre elementos. Existen cuatro tipos principales de relaciones:

    - Generalización: es una relación de especialización.
    - Asociación: es una relación estructural. Existen dos subtipos, la agregación y la composición.
    - Realización: es una relación contractual, en la cual una clase especifica un contrato que otra clase garantiza que cumplirá (por ejemplo, una interface).
    - Dependencia: Es una relación de uso.

Una vez descritos brevemente los componentes de un diagrama de clases, vamos a proceder a analizar los propios del sistema.

3.4.1.1 Diagrama completo de clases

Antes de mostrar el diagrama de clases, vemos apropiado explicar como va a ser el funcionamiento o la lógica de negocio del sistema para ver más claramente el paso de información o comunicación entre las distintas capas diseñadas. A la hora de diseñar una aplicación con una interfaz gráfica de usuario es importante seguir el esquema de diseño *Modelo-Vista-Controlador* (MVC). En este esquema se definen tres roles bien diferenciados:

- **Modelo**: el modelo es la representación de la información de un problema.

- **Vista**: una vista es una posible visualización de la información contenida en un modelo. Un modelo puede tener varias vistas definidas.

- **Controlador**: el controlador se encarga de coordinar la interacción entre las vistas y los modelos. Cada vez que un modelo cambia internamente, actualiza sus vistas.

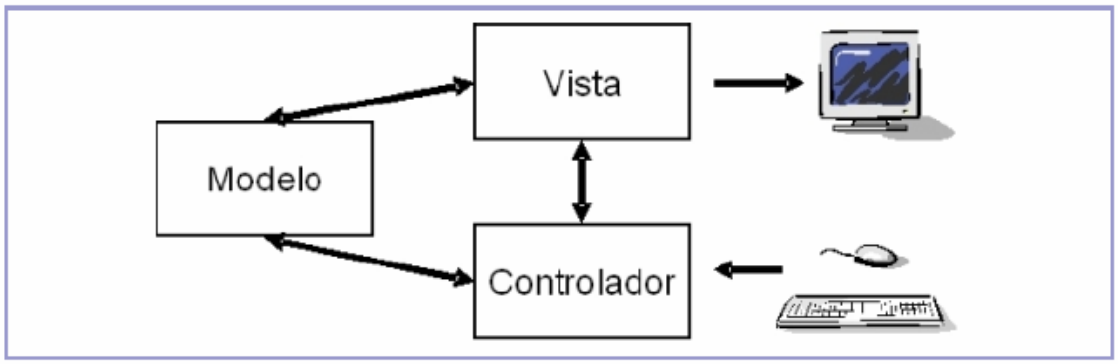


Imagen 1. Esquema de la arquitectura MVC

El mensaje más importante que proporciona este esquema de diseño es que siempre hay que separar la representación del problema con la interfaz de usuario u otra forma de visualización de la información del modelo (informe impreso, gráfico, etc.).

En nuestra aplicación, tendremos una capa modelo de la información que representará todos los datos del sistema, que será la encargada de obtener los datos necesarios de la BBDD. Esta información será gestionada por una serie de servlets que harán la función de controlar el sistema y realizar cualquier operación necesaria. Finalmente, cualquier representación gráfica de datos en la pantalla será lanzada por otros Servlets que harán la labor de la Vista de la aplicación.

El diagrama completo de clases es el siguiente:

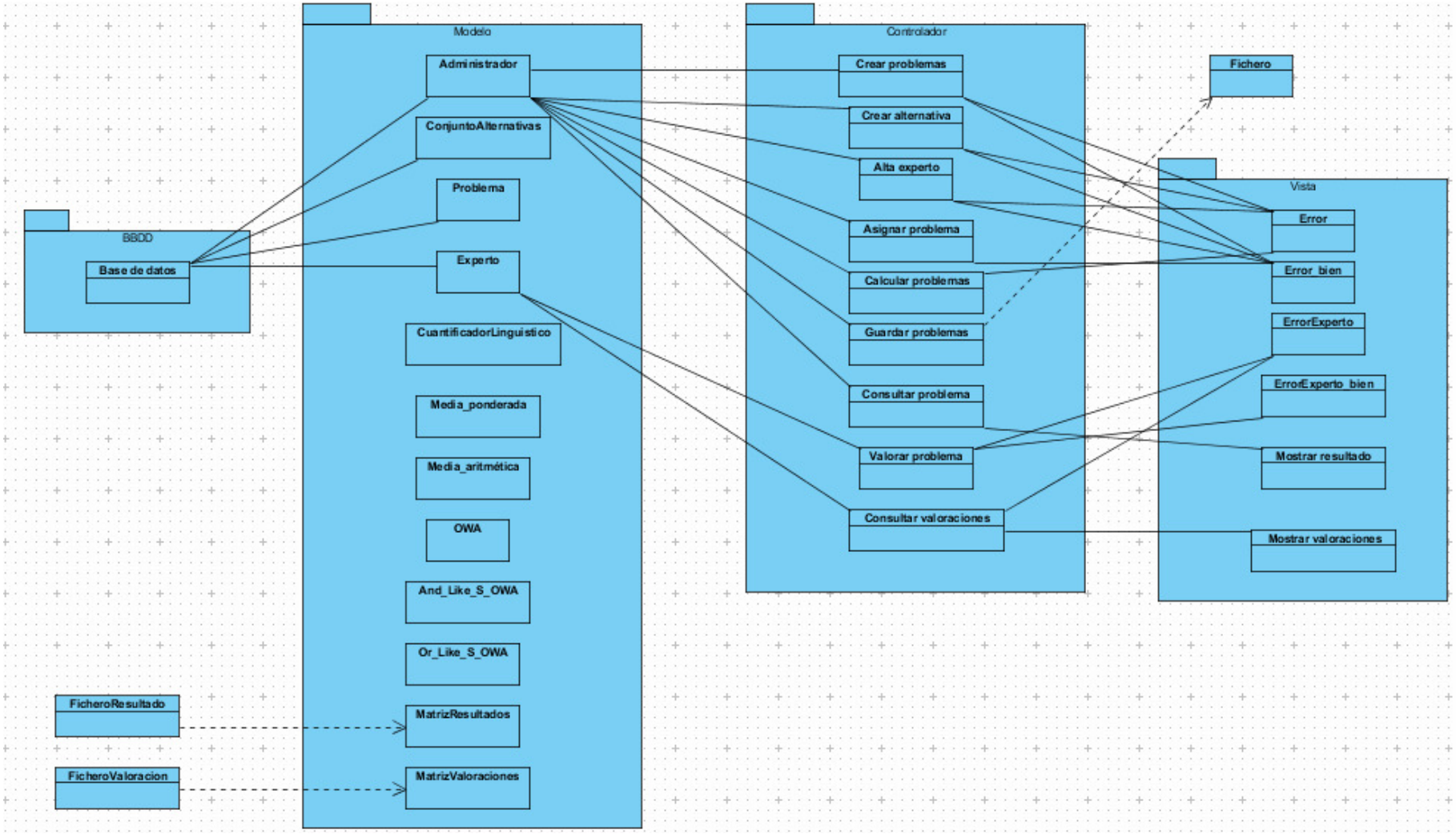


Figura 13. Diagrama de clases.

Analizamos cada uno de los paquetes de nuestro diagrama de clases:

- **Paquete Base de Datos**: es el paquete al que hacen referencia algunas de las clases pertenecientes al modelo, para obtener la conexión a la BBDD.

- **Paquete Modelo**: Representa la capa de modelo de la información de la aplicación y, como tal, debe ser gestionado por otros paquetes controladores para administrar de forma adecuada toda la información.

- **Paquete Controlador**: la capa controladora, que en nuestro caso está formada por servlets, llevará a cabo todo el control del sistema y permitirá al usuario realizar cualquier operación sobre éste. Como capa de negocio, utilizará constantemente clases del paquete Modelo para obtener y guardar información, y será la encargada de operar con la información para mostrarla en la Vista.

- **Paquete Vista**: la capa vista también está formada por servlets y es la encargada de mostrar la información de la aplicación.

3.4.1.2. Diagrama por paquete

En los siguientes apartados vamos a ver con más detalle los paquetes más importantes de nuestra aplicación.

3.4.1.2.1. Paquete Modelo

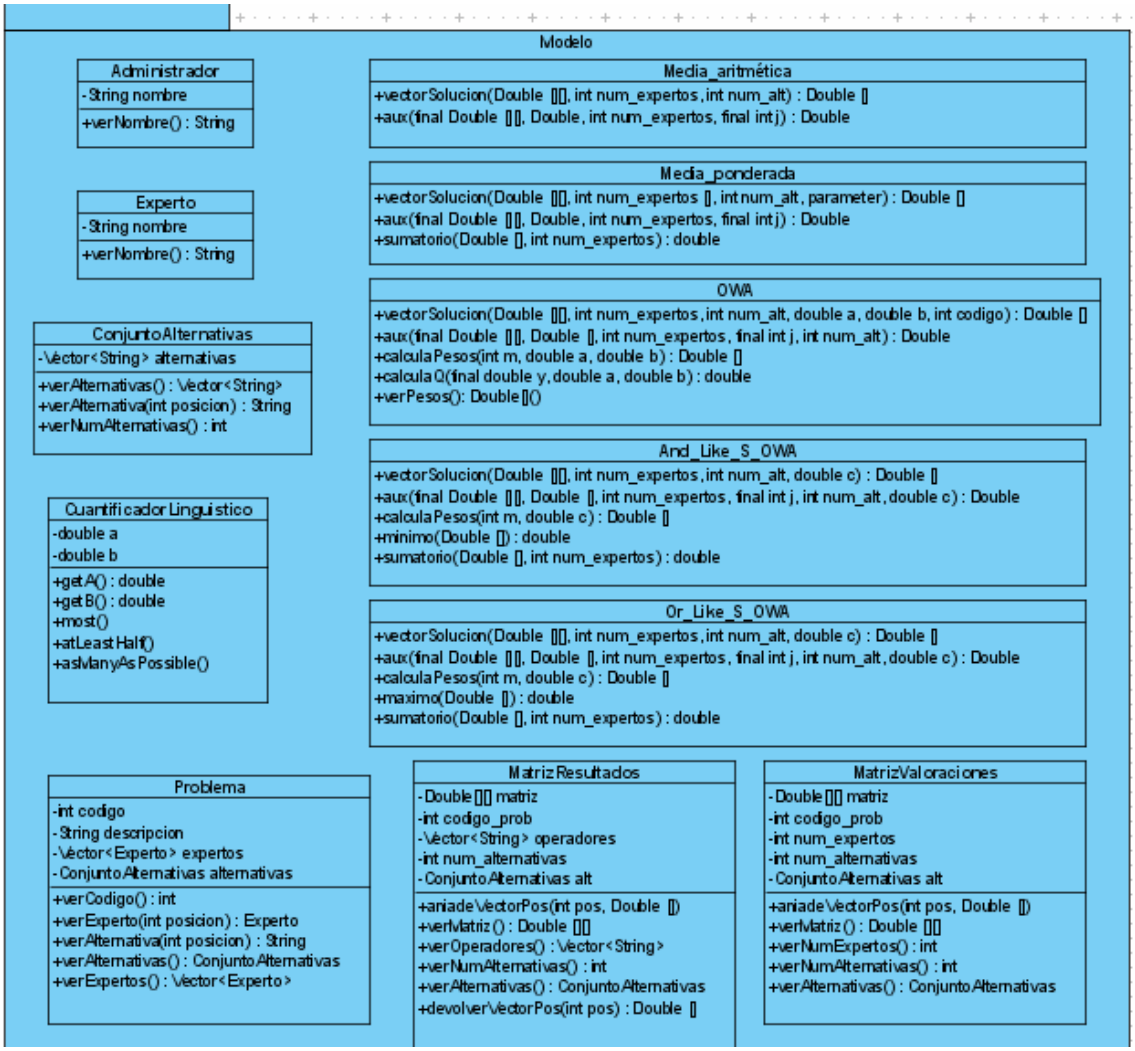


Figura 14. Paquete Modelo.

3.4.1.2.2. Paquete Vista

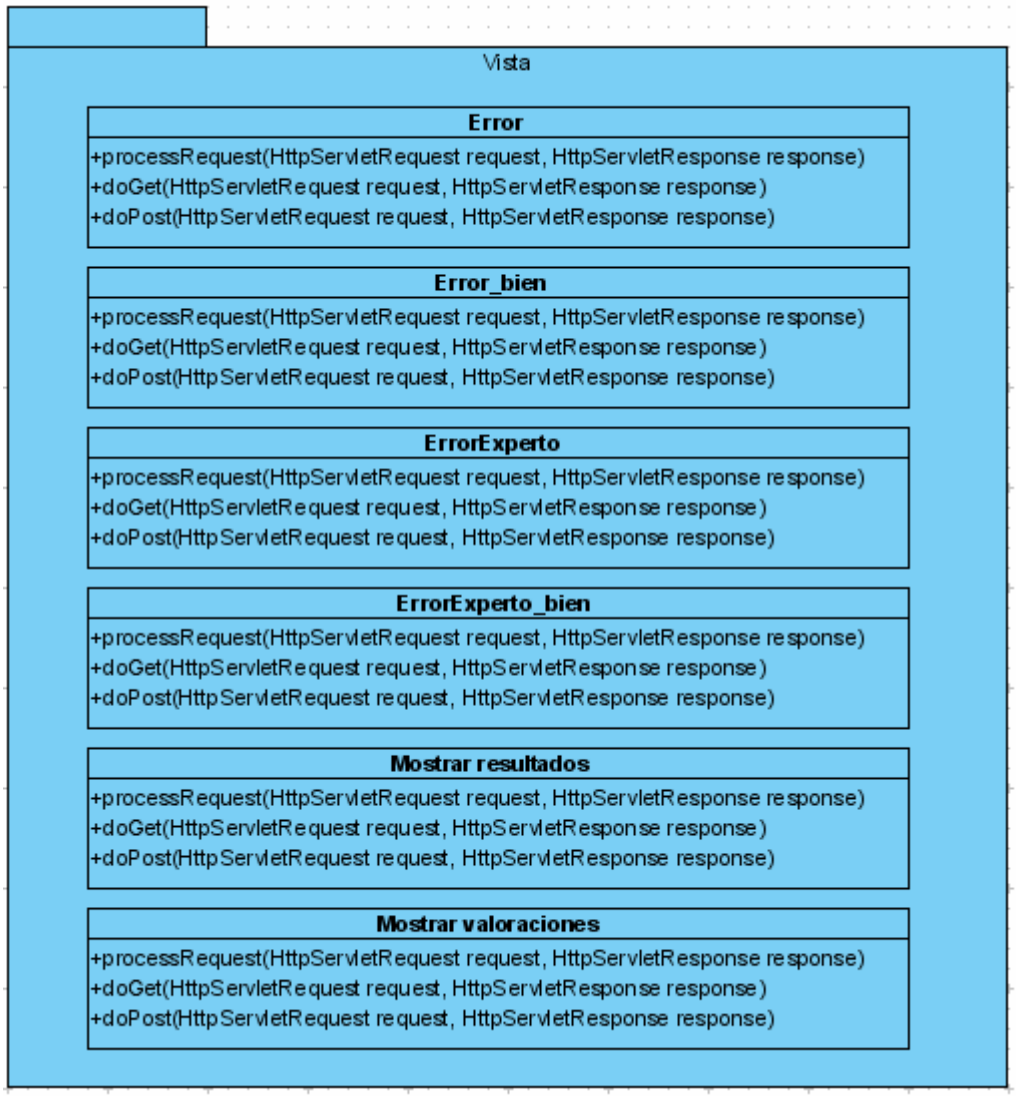


Figura 15. Paquete Vista.

3.4.1.2.3. Paquete Controlador

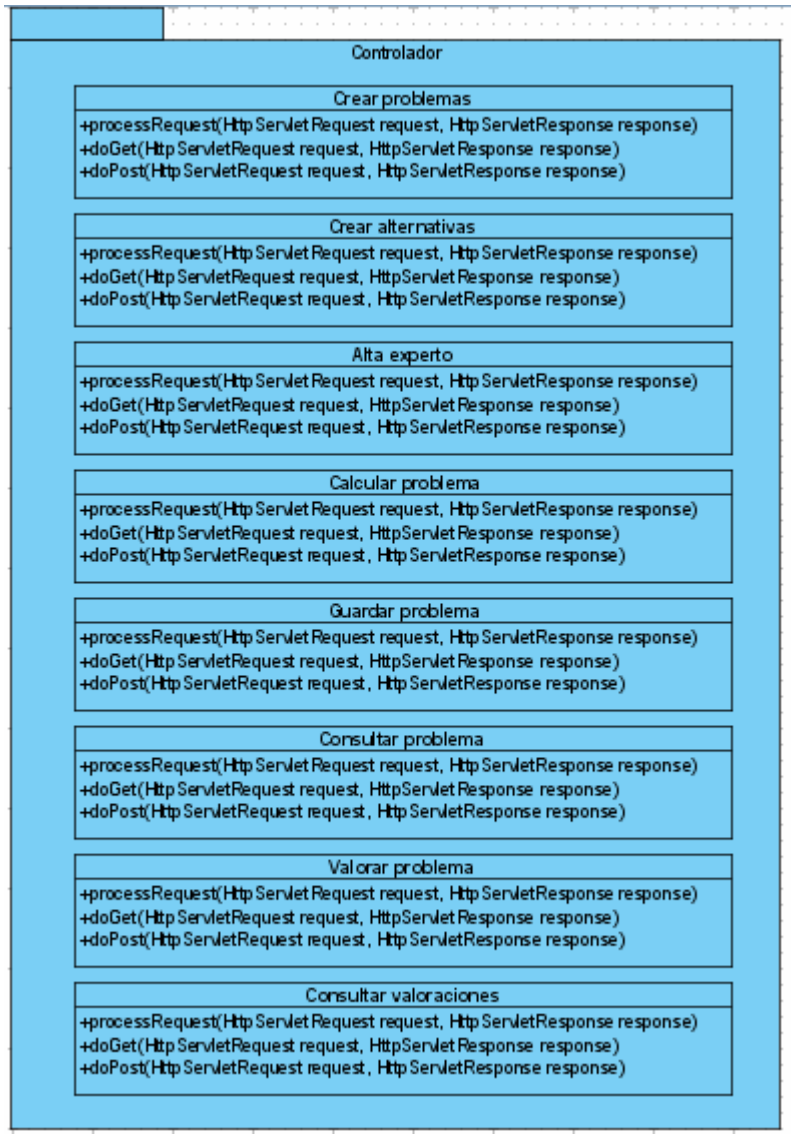


Figura 16. Paquete Controlador.

3.4.2. Diseño de los datos

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 va a trabajar. Estos elementos son:

- *Administrador*, conocemos su nombre, apellidos y contraseña.
- *Expertos*, conocemos su nombre, apellidos y contraseña.
- *Alternativas*, conocemos su descripción y su código.
- *Problemas,* conocemos su descripción y su código.

Una vez determinados cuales son los elementos de información del sistema, se deben obtener sus representaciones en forma de tablas de una base de datos. Para ello, se debe realizar primeramente un diseño conceptual de la base de datos para, posteriormente, obtener las tablas requeridas. Para realizar este diseño conceptual se utilizará el modelo Entidad-Relación.

Modelo Entidad-Relación

El modelo Entidad-Relación (también conocido por sus iniciales: E-R) es una técnica de modelado de datos que utiliza diagramas entidad-relación. No es la única técnica de modelado pero si es la más extendida y utilizada.

Un diagrama entidad-relación esta compuesto por tres tipos de elementos principales:

- **Entidades**: objetos (cosas, conceptos o personas) sobre los que se tiene información. Se representan mediante rectángulos etiquetados en su interior con un nombre. Una instancia es cualquier ejemplar concreto de una entidad.

- **Relaciones**: interdependencias entre uno o más entidades. Se representan mediante rombos etiquetados en su interior con un verbo. Si la relación es entre

una entidad consigo mismo se denomina reflexiva, si es entre dos entidades se denomina binaria, ternaria si es entre tres y múltiple si es entre más.

- **Atributos**: características propias de una entidad o relación. Se representan mediante elipses etiquetados en su interior con un nombre.

En los diagramas entidad-relación también hay que tener en cuenta otros aspectos como pueden ser:

- **Entidades débiles**: son aquellas que no se pueden identificar unívocamente solo con sus atributos, es decir, necesitan de estar relacionadas con otras entidades para existir. Se representan con dos rectángulos concéntricos de distinto tamaño con un nombre en el interior del más pequeño.

- **Cardinalidad** de las relaciones: existen tres tipos de cardinalidades de una relación según el número de instancias de cada entidad que involucren:

    - *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 de la entidad B. (1:*)
    - *Muchos a muchos*: cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B. (*:*)

- **Claves**: cada entidad de un diagrama entidad-relación debe tener una clave, debe estar formada por uno o más de sus atributos.

Una vez conocidos los elementos que forman parte de un diagrama entidad-relación podemos empezar a desarrollar el modelo entidad-relación. Los pasos a seguir son los siguientes:

1. Convertir el enunciado del problema (o, como es nuestro caso, los elementos del sistema software) en un Esquema Conceptual del mismo.

2. Convertir este Esquema Conceptual (o EC) en uno más refinado conocido como Esquema Conceptual Modificado (ECM).

3. Obtener las tablas de la base de datos a partir del Esquema Conceptual Modificado.

Normalización en el modelo Entidad-Relación

La normalización es un proceso consistente en imponer a las tablas ciertas restricciones mediante una serie de transformaciones consecutivas. Con ello se asegura 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.

Para asegurar la normalización Codd estableció tres formas normales, las cuales hacen que una base de datos (si las cumple) esté normalizada.

Estas formas normales son:

- **Primera forma Normal (FN1)**: Una tabla esta en FN1 si todos los atributos no clave, dependen 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 esta 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.

3.4.2.1. Esquema Conceptual

Necesitamos convertir nuestros elementos del sistema en entidades o relaciones. Es obvio que *Administrador*, *Expertos*, *Problemas* y *Alternativas* se convertirán en Entidades en nuestro Esquema Conceptual.

En cuanto a las relaciones:

- El *Administrador* del sistema puede gestionar *Problemas*, por lo tanto esta relación será de uno a muchos.

- Un *Problema* está formado por un conjunto de *Expertos,* y un *Experto* puede estar incluido en más de un problema (relación de muchos a muchos).

- Un *Problema* también está formado por un conjunto de *Alternativas,* y una *Alternativa* puede estar incluida en más de un problema (relación de muchos a muchos).

Estas entidades con sus atributos y sus relaciones se puede observar en la *Figura 17.*

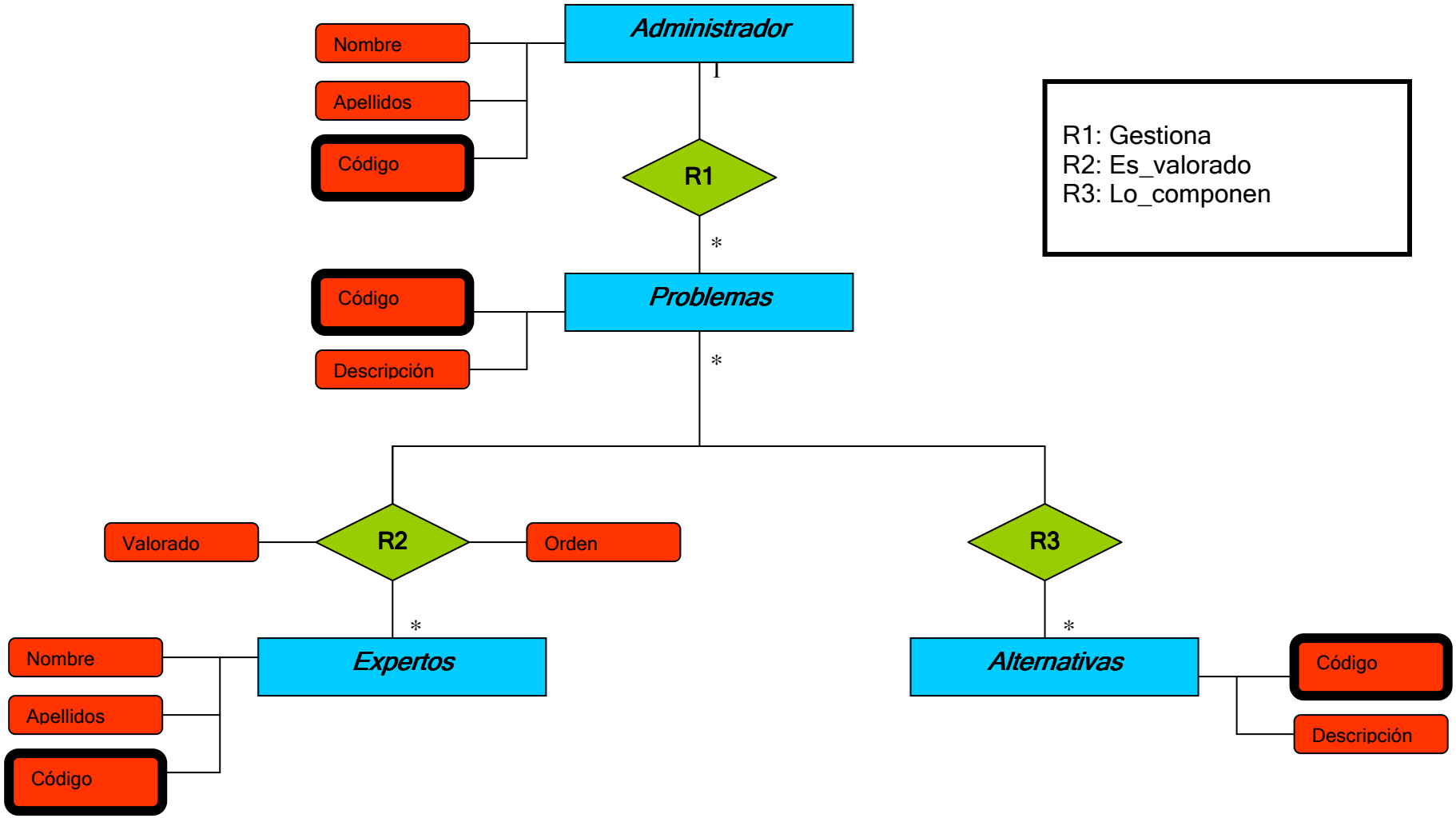


Figura 17. Esquema Conceptual

3.4.2.2. Esquema Conceptual Modificado

Para obtener el Esquema Conceptual Modificado a partir del Esquema Conceptual se deben hacer los cambios que siguen a continuación:

- Eliminar todas las entidades débiles
- Eliminar las relaciones de muchos a muchos
- Eliminar las relaciones con atributos que haya en nuestro Esquema Conceptual.

Por lo tanto, nuestro Esquema Conceptual Modificado (ECM) quedaría como muestra la *Figura 18,* y tendríamos dos entidades más:

- *Exp_prob_val_orden*, que servirá al *Administrador* para ver si un *Experto* ha valorado ya un *Problema* o aún no, y también le servirá para poder almacenar de una forma ordenada las valoraciones de un *Experto*.
- *Prob_alt*, en la que se almacenan las alternativas que forman parte de un determinado problema.

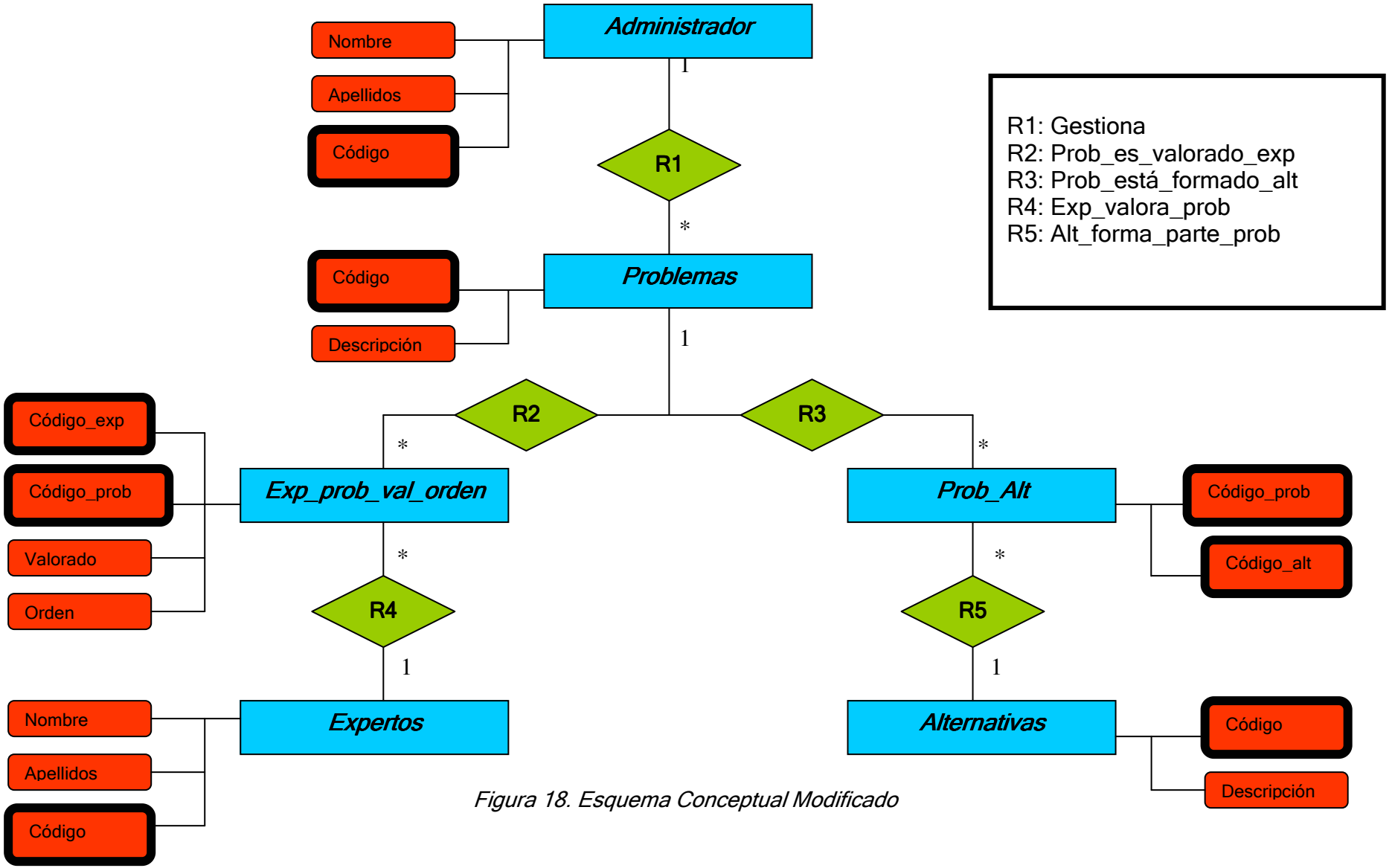


Figura 18. Esquema Conceptual Modificado

3.4.2.3. Tablas de la aplicación

A partir del ECM obtenido previamente podemos determinar las tablas de la base de datos, teniendo en cuenta que:

- Cada entidad del ECM se transforma en una tabla.
- Los atributos de una entidad se convierten en los campos de las tablas respectivas.

Por lo tanto, obtendremos las siguientes tablas: *ADMINISTRADOR*, *EXPERTOS*, *PROBLEMAS*, *ALTERNATIVAS*, *EXP_PROB_VAL_ORDEN* y *PROB_ALT*. A continuación se detallan cada una de estas tablas.

ADMINISTRADOR

CAMPO	TIPO	DESCRIPCIÓN	CLAVE	REQUERIDO
CÓDIGO	INTEGER	Código del administrador	*	SI
NOMBRE	STRING	Nombre del administrador		SI
APELLIDOS	STRING	Apellidos del administrador		SI

Tabla 1: Campos de la tabla ADMINISTRADOR

EXPERTOS

CAMPO	TIPO	DESCRIPCIÓN	CLAVE	REQUERIDO
CÓDIGO	INTEGER	Código del experto	*	SI
NOMBRE	STRING	Nombre del experto		SI
APELLIDOS	STRING	Apellidos del experto		SI

Tabla 2: Campos de la tabla EXPERTOS

PROBLEMAS

CAMPO	TIPO	DESCRIPCIÓN	CLAVE	REQUERIDO
CÓDIGO	INTEGER	Código del problema	*	SI
DESCRIPCIÓN	STRING	Descripción del problema		SI

Tabla 3: Campos de la tabla PROBLEMAS

ALTERNATIVAS

CAMPO	TIPO	DESCRIPCIÓN	CLAVE	REQUERIDO
CÓDIGO	INTEGER	Código de la alternativa	*	SI
DESCRIPCIÓN	STRING	Descripción de la alternativa		SI

Tabla 4: Campos de la tabla ALTERNATIVAS

EXP_PROB_VAL_ORDEN

CAMPO	TIPO	DESCRIPCIÓN	CLAVE	REQUERIDO
CÓDIGO_EXP	INTEGER	Código del experto	*	SI
CÓDIGO_PROB	INTEGER	Código del problema	*	SI
VALORADO	STRING	Indica si el experto ha valorado ya un problema		SI
ORDEN	INTEGER	Orden en el que se guardan las valoraciones del experto en el fichero de valoraciones		SI

Tabla 5: Campos de la tabla EXP_PROB_VAL_ORDEN

PROB_ALT

CAMPO	TIPO	DESCRIPCIÓN	CLAVE	REQUERIDO
CÓDIGO_PROB	INTEGER	Código del problema	*	SI
CÓDIGO_ALT	INTEGER	Código de la alternativa	*	SI

Tabla 6: Campos de la tabla PROB_ALT

3.4.3. Diseño de la interfaz

En esta fase del diseño del sistema software se define cual va a ser la apariencia visual de la aplicación, es decir, se define la interfaz visual entre el usuario y la aplicación. Sin duda, realizar un buen diseño de la interfaz resulta primordial ya que está debe presentarse atractiva al usuario de la aplicación pero a la vez le debe de resultar fácil de entender y trabajar sobre ella.

Esta importancia es mayor en nuestro caso ya que la interfaz de nuestro proyecto es una interfaz Web. Para las aplicaciones con interfaces Web no existe una guía de estilo estándar como existe, por ejemplo, para desarrollar interfaces para aplicaciones de escritorio de Windows XP y que resulten, a la vez, atractivas y familiares. Cada programador, desarrollador o diseñador Web debe definir su propia guía de estilo y procurar que, en base a ella, la interfaz resultante consiga unas cotas dignas de atractivo visual, familiaridad y facilidad de uso.

La ingeniería de usabilidad define en esta fase los siguientes aspectos:

- Definir estilo.
- Metáforas.
- Pantallas.
- Caminos de navegación.
- Secuencias de diálogo.
- Mensajes de error.

Además nosotros hemos añadido otro aspecto que hemos llamado ‘Notificaciones’ para distinguirlas de los mensajes de error.

Cada uno de estos aspectos lo veremos a continuación de forma detallada.

3.4.3.1. Definir estilo

Antes de ponerse a diseñar una interfaz de usuario, se debe definir el estilo de la misma. Esto es de vital importancia cuando el diseño va a ser compartido entre varios diseñadores, ya que ayuda a mantener la coherencia interna de la interfaz.

Sin embargo, en contra de lo que pueda parecer en un principio, también es de mucha utilidad definir una guía de estilo cuando sólo hay un diseñador encargado de la interfaz. Esto se debe a varias razones:

- A veces es posible que mantener la coherencia y consistencia de una interfaz, si esta es muy grande o muy ambiciosa, sea complicado incluso si solo hay un diseñador.

- El diseñador primitivo puede, por las más diversas razones, abandonar el diseño y es de utilidad para sus sustitutos contar con una guía de estilo predefinida para no tener que empezar de cero otra vez. Lo mismo puede aplicarse si no es el diseñador original el que se encarga del mantenimiento o la actualización de la interfaz.

Quedando demostrada la utilidad del uso de guías de estilo pasamos a definir las reglas y normas que contendrá la guía de estilo de nuestra interfaz:

- **Fuentes**: para escribir cualquier texto en nuestra interfaz utilizaremos una tabla cuyas propiedades son las siguientes:

    - Tipo de letra: "Verdana"
    - Tamaño: 18px;
    - Color de fondo de la página: # 999999;
    - Color de la letra: # 141414;
    - Alineación del texto: *center*;

- **Enlaces**: sin subrayado, los enlaces de la aplicación se abren en la misma ventana o pestaña.

- **Cabeceras**: las cabeceras de cada *.java* tendrán las siguientes propiedades:

    - Tipo de letra: "Verdana"
    - Tamaño: 24px;
    - Color de la letra: # 141414;
    - Alineación del texto: *center*.

3.4.3.2. Metáforas

Una metáfora es el empleo de un objeto con un significado o dentro de un contexto diferente al habitual. Al diseñar una interfaz gráfica, la utilización de metáforas resulta muy útil ya que permiten al usuario, por comparación con otro objeto o concepto, comprender de una manera más intuitiva las diversas tareas que la interfaz permite desarrollar.

Al igual que pasa en el ámbito de la literatura, para que una metáfora cumpla con su cometido, el desarrollador de la aplicación y el usuario final de está deben tener una base cultural similar. Es muy posible que el uso de un icono de manera metafórica sea entendido de una manera por el usuario occidental y de otra bien distinta por un usuario oriental. Hay que intentar, por lo tanto, que las metáforas empleadas sean lo más universales posibles para que así sean comprendidas a la perfección por la mayor parte del público potencial.

Las aplicaciones de escritorio de Windows suelen seguir la Guía de Estilo XP y utilizan una serie de metáforas con las que el usuario esta plenamente familiarizado (por ejemplo, una lupa con un signo '+' en su interior establece que la función del icono es, inequivocamente, la de realizar un aumento de zoom). En el mundo de las aplicaciones Web también existen una cantidad de metáforas de amplia difusión como puede ser, por ejemplo, el celebre carrito de la compra que emplean casi todos los comercios online.

Pero las metáforas no sólo dependen del tipo de aplicación (escritorio o Web) sino también del ámbito de la misma. Por ejemplo, el carrito de la compra es

una metáfora conocida por todos pero si nuestra aplicación no va a vender nada al usuario no resulta conveniente utilizarla ya que puede confundir.

En nuestra aplicación, nos encontramos con las metáforas siguientes:

-  Esta metáfora, de una casita, la utilizan gran cantidad de sitios Web como referencia para ir a la página de inicio. En nuestra aplicación, si el usuario pincha sobre esta metáfora, vuelve también a la página de inicio que en nuestro caso es el menú principal del usuario.

-  Esta metáfora también es muy utilizada en los sitios Web para volver a la página anterior. En nuestro caso, esta metáfora aparece en la resolución de un problema, para que el Administrador pueda volver a la página anterior y escoger otro operador para resolver el problema.

3.4.3.3. Pantallas

Como hemos visto en apartados anteriores, para diseñar una buena interfaz, es necesario un concienzudo trabajo de análisis y diseño.

El diseño grafico de la interfaz de usuario juega un papel fundamental, ya que una buena distribución, ordenación, y codificación de colores de los elementos de la interfaz hace que ésta sea más clara y por tanto mas fácil de usar para el usuario. Por el contrario una colocación incorrecta de los elementos puede hacer muy compleja la utilización de una aplicación por parte de un usuario.

Un principio general es que, los elementos que posee la interfaz tienen que estar colocados de tal modo que, a la hora de realizar una tarea, el usuario deba recorrer la interfaz en la misma dirección que lee un texto, es decir, de izquierda a derecha y de arriba abajo. En países orientales no se realizaría de la misma forma.

Una interfaz de usuario visualmente atractiva hace que sea percibida por el usuario como una mejor interfaz, mientras que una interfaz compleja con sus elementos desordenados, provoca en el usuario insatisfacción y desorientación. No hay que olvidar que una buena interfaz es aquella que encuentra el equilibrio entre la funcionalidad y la estética.

En nuestra aplicación se situaran los botones relacionados en el mismo grupo y en tablas diferentes. El botón relacionado con un problema (Seleccionar el problema que se desea) estará en una tabla, mientras que los relacionados con los expertos y alternativas en otra. También hay tablas diferentes con sus correspondientes botones para seleccionar el operador que se desea para resolver el problema, así como para escoger el cuantificador lingüístico utilizado para determinados operadores.

3.4.3.4. Caminos de navegación

Hasta este momento tenemos un diseño visual de la interfaz estática, es decir, cada pantalla diseñada individualmente, pero no tenemos una idea de si en el conjunto de la interacción, la acción va a transcurrir de forma comprensible para el usuario. Para ello vamos a diseñar la interfaz en movimiento y comprobar que es usable.

Para estudiar los caminos de interacción se empleara una herramienta llamada *storyboard*, que consiste en mostrar, a modo secuencia, las distintas pantallas por las que se va pasando al realizar el usuario una determinada acción sobre la aplicación. Mediante flechas se ayuda a entender que es lo que a desencadenado el paso de una pantalla a otra. Los *storyboard*s también están muy ligados a los escenarios vistos anteriormente.

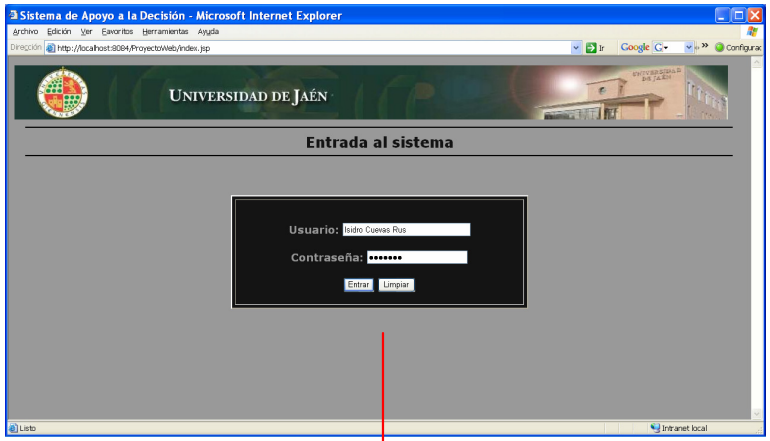
El *storyboard* sirve de prototipo para ser evaluado por el usuario y poder introducir correcciones en fases iniciales, ya que cuanto más tiempo se tarde en validar una interfaz, más coste de tiempo y trabajo nos ocasionará.

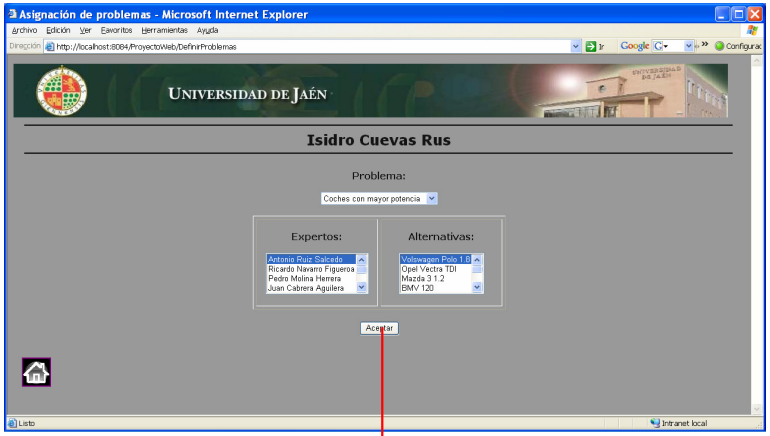
No se han desarrollado *storyboards* para todas las acciones de nuestro sistema por lo que los que se muestran a continuación son los que se han considerado más importantes:

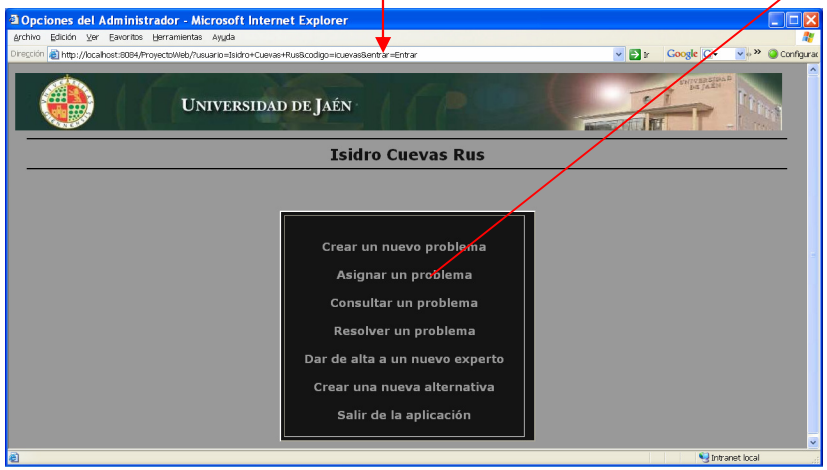
- Storyboard Crear un problema (*Administrador*)
- Storyboard Valorar un problema (*Experto*)
- Storyboard Resolver un problema (*Administrador*)

Una vez diseñados deberían ser validados para comprobar que realmente la aplicación es usable.

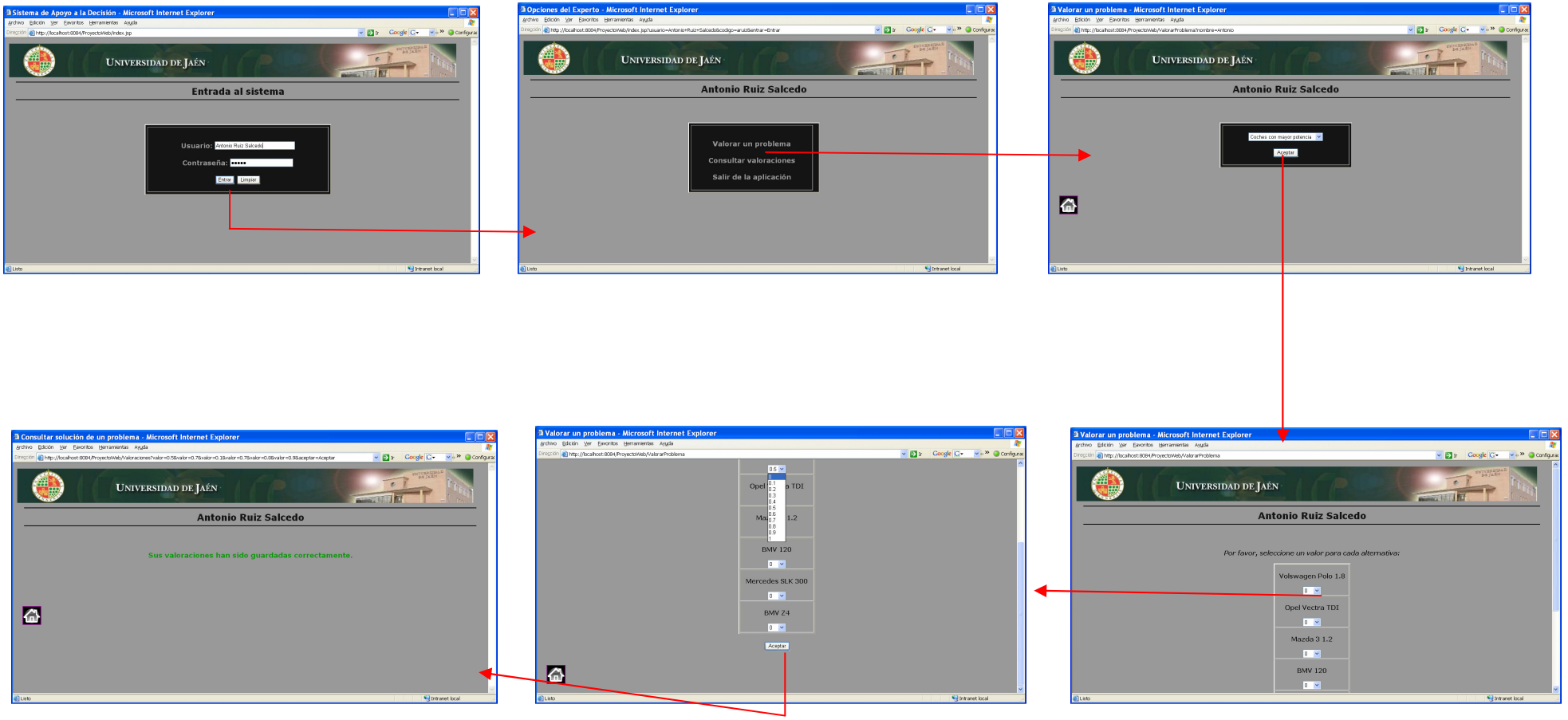
Storyboard Asignar un Problema



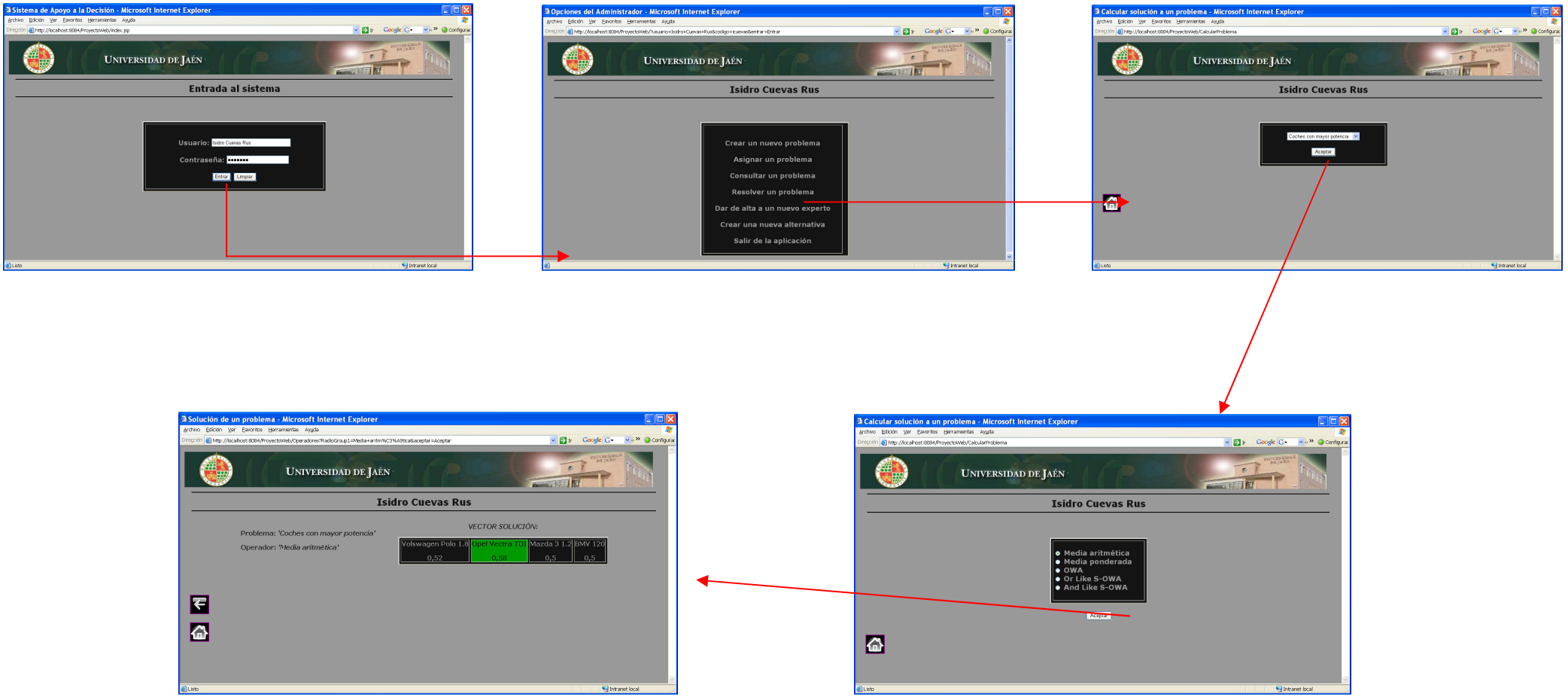




Storyboard Valorar un Problema



Storyboard Resolver un Problema



3.4.3.5. Mensajes de error

Una vez diseñadas las pantallas, los caminos de navegación y las secuencias de diálogo, 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 darse en 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 conocimiento sobre la interfaz, porque se no se ha entendido correctamente el estado del sistema o bien inadvertidamente (por ejemplo se pulsa un botón cuando se intentaba pulsar otro). Ello lleva al usuario a sentirse confuso y aumenta su ansiedad, sobre todo en usuarios noveles en los que su falta de conocimiento y confianza en el uso de la interfaz lleva a amplificar el stress lo que puede llevarles a una experiencia de uso de la interfaz frustrante.

Los mensajes de error son muy importantes 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. Por ello hay que diseñarlos con cuidado. A la hora de diseñar mensajes de error se deberían seguir las siguientes 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 para hacerlo. Por ello hay que diseñar mensajes de error claros, pero breves.

- Ser específico: 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 guía constructiva: nunca recriminar al usuario lo que ha hacho mal, en su lugar siempre que sea posible indicarle que debe hacer para eliminar el error.

- Usar un formato físico apropiado: la mayoría de usuarios encuentran más fácil leer un mensaje donde se mezclan letras mayúsculas y minúsculas de la forma habitual, por tanto este formato es siempre preferible. Los mensajes escritos únicamente en mayúsculas deberían reservarse para avisos breves y graves. Si el mensaje de error debe contener un código numérico, este debería figurar al final del mensaje y entre paréntesis.

Los mensajes de error de nuestra aplicación son los siguientes:

- Identificación: existe la posibilidad de que un usuario se identifique de forma incorrecta. Si esto ocurre el mensaje de error que se muestra es el siguiente: "*Usuario y/o Contraseña incorrecta*", como se puede ver en la *Figura19*.

Figura19. Mensaje de error en la Identificación de usuarios.

- Creación de problemas: cuando en el formulario que aparece para la creación de un nuevo problema no se rellenan todos los campos, aparece el siguiente mensaje de error: "*Por favor, rellene todos los campos*" (*Figura20*).

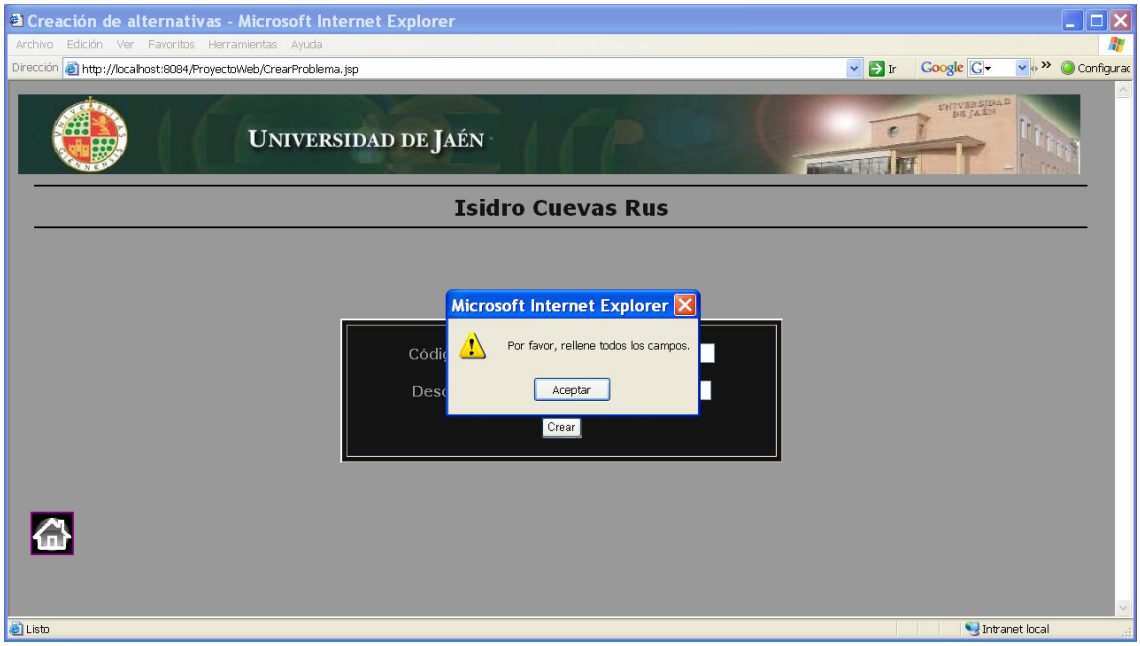


Figura 20. Mensaje de error en 'Creación de un Problema'.

- Creación de alternativas: cuando en el formulario que aparece para la creación de una nueva alternativa no se rellenan todos los campos, aparece el siguiente mensaje de error: "*Por favor, rellene todos los campos*" (*Figura21*).

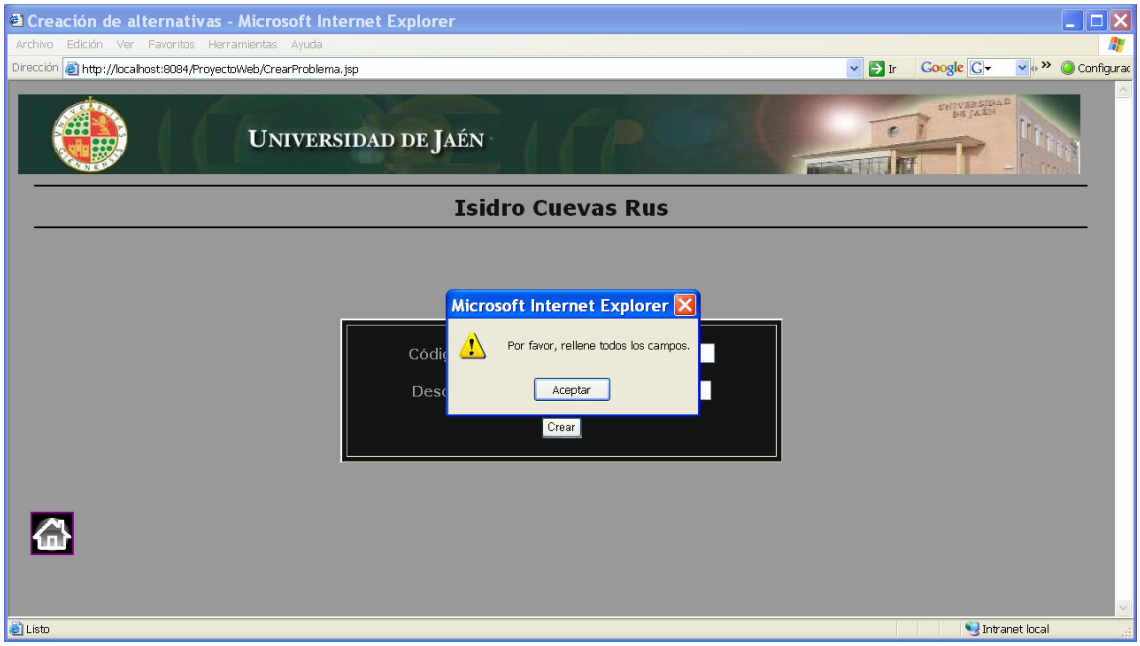


Figura 21 Mensaje de error en 'Creación de una nueva alternativa'.

- Alta de Expertos: cuando en el formulario que aparece para dar de alta a un nuevo experto no se rellenan todos los campos, aparece el siguiente mensaje de error: "*Por favor, rellene todos los campos*" (*Figura22*).

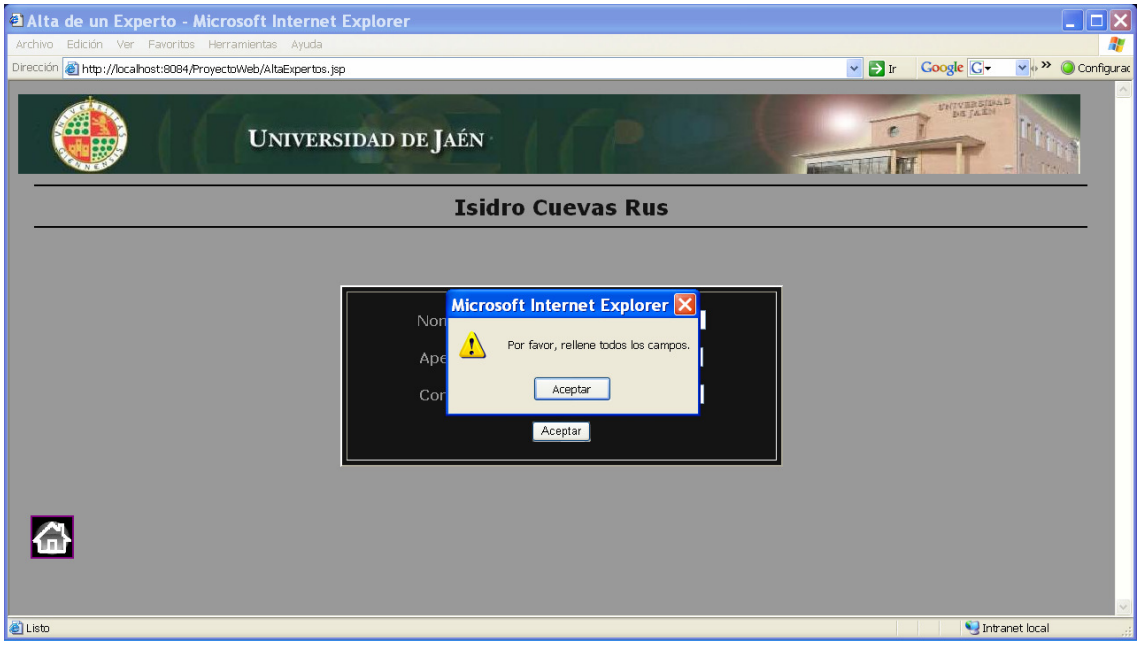


Figura 22 Mensaje de error en 'Alta de un nuevo experto'.

3.4.3.6. Notificaciones

Las notificaciones son también muy importantes de cara a la usabilidad, ya que bien diseñados, permiten también aumentar la confianza del usuario en el uso de la interfaz cuando realiza una operación. Por ello hay que diseñarlos con cuidado. A la hora de diseñar estas notificaciones se deberían seguir las siguientes reglas:

- Ser breves: el usuario es una persona ocupada tratando de llevar a cabo una tarea. Si se le presenta una notificación muy larga, lo normal es que no lo lea ya que no tendrá tiempo para hacerlo. Por ello hay que diseñar mensajes de notificación claros, pero breves.

- Usar un formato físico apropiado: la mayoría de usuarios encuentran más fácil leer un mensaje donde se mezclan letras mayúsculas y minúsculas de la forma habitual, por tanto este formato es siempre preferible.

- Sensibilidad al color: cuando un usuario realiza una operación sobre una aplicación y le aparece un mensaje en color rojo, suele asociar este color con la idea de que ha hecho algo incorrecto. En cambio el color verde suele significar todo lo contrario. Así, el usuario nada más ver el color del mensaje, antes de leer lo que se le dice, ya puede conocer el contenido de dicho mensaje.

Los mensajes de notificación utilizados en nuestro sistema son los siguientes:

- Creación de un problema: si el administrador introduce un código de problema que ya está siendo utilizado por otro problema se muestra en la pantalla una notificación diciendo: "*Lo siento, ese código de problema ya existe.*" (*Figura 23*). Si el código era correcto, la notificación es la siguiente: "*Los datos del problema se han guardado correctamente.*" (*Figura 24*).

Figura 23. Notificación para 'Creación de un Problema' (código erróneo).

Figura 24. Notificación para 'Creación de un Problema' (código correcto).

- Creación de una alternativa: si el administrador introduce un código de alternativa que ya está siendo utilizado por otra alternativa se muestra en la pantalla una notificación diciendo: "*Lo siento, ese código de alternativa ya existe.*" (*Figura 25*). Si el código era correcto, la notificación es la siguiente: "*Los datos de la alternativa se han guardado correctamente.*" (*Figura 26*).

Figura 25. Notificación para 'Creación de una nueva alternativa' (código erróneo).

Figura 26. Notificación para 'Creación de una nueva alternativa' (código correcto).

- Alta de un nuevo experto: si el administrador introduce una contraseña de experto que ya está siendo utilizado por otro experto se muestra en la pantalla una notificación diciendo: "*Lo siento, esa contraseña de experto ya existe.*" (*Figura 27*). Si el código era correcto, la notificación es la siguiente: "*Los datos del experto se han guardado correctamente.*" (*Figura 28*).

Figura 27. Notificación para 'Creación de un nuevo experto (código erróneo).

Figura 28. Notificación para 'Creación de un nuevo experto (código correcto).

- Asignación de problemas: cuando a un problema se le asignan un conjunto de expertos y de alternativas, el sistema muestra la siguiente notificación: "*Los datos han sido guardados correcta*mente." (*Figura 29*).

Figura 29. Notificación en 'Asignación de Problemas'.

- Consultar la solución de un problema: cuando el administrador elige consultar la solución de un problema que aún no ha resuelto el sistema muestra la siguiente notificación: "*No ha resuelto este problema aún*." (*Figura 30*).

Figura 30. Notificación en 'Consultar la solución de un problema'.

- Calcular la solución a un problema: cuando el administrador elige calcular la solución de un problema que aún no ha sido valorado, el sistema muestra la siguiente notificación: "*No puede resolver este problema ya que el/los experto/s siguiente/s no ha/n valorado aún dicho problema: <Lista_de_expertos>*." Un ejemplo de esto lo vemos en la *Figura 31*.

Figura 31. Notificación en 'Calcular solución a un problema'.

- Valoración de un problema: cuando un experto selecciona valorar un problema que ya había valorado, el sistema muestra la siguiente notificación: "*Ese problema ya lo ha valorado*." (*Figura 32*), y si no lo había valorado aún, el sistema muestra esta otra notificación: "*Ese problema ya lo ha valorado*.", y se pregunta si desea valorarlo de nuevo (*Figura 33*).

Figura 32. Notificación de 'Valoración de un Problema' (problema ya valorado).

Figura 33. Notificación de 'Valoración de un Problema' (problema valorado correctamente).

- Consultar la valoración de un problema: cuando un experto selecciona consultar la valoración de un problema que aún no ha valorado, el sistema muestra la siguiente notificación: "*Ese problema no lo ha valorado todavía*." (*Figura 34*).

Figura 34. Notificación de 'Consultar la valoración de un Problema'.

3.5. Implementación

La implementación es la actividad final de la Ingeniería del Software, aquella en la que el modelo obtenido en las actividades anteriores se debe transformar en código fuente. Para ello se debe ser cuidadoso en la elección del lenguaje de programación empleado para la codificación y de la herramienta utilizada para generarla.

En nuestro caso la elección del lenguaje de programación así como el de la herramienta utilizada para su desarrollo viene dado desde la definición del proyecto.

3.5.1. Tipo de arquitectura de la aplicación

En nuestro caso, vamos a desarrollar un sistema con una arquitectura cliente/servidor y una interfaz Web de comunicación con los usuarios. El funcionamiento de las arquitecturas de este tipo es sencilla: la aplicación se encuentra en un servidor central al que los usuarios acceden a través de un software cliente, en nuestro caso un navegador Web. Una vez que ha accedido a la aplicación, el usuario realiza peticiones que el servidor tiene que atender para generar una respuesta comprensible para el cliente.

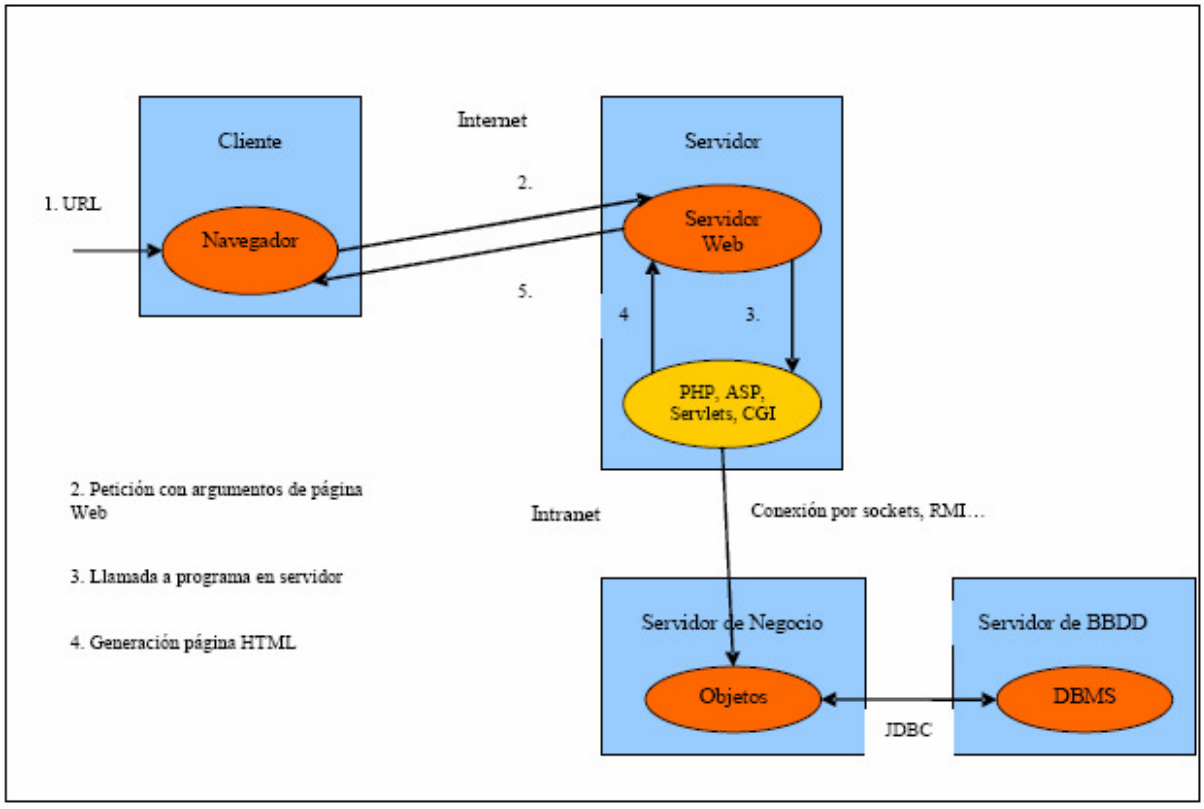


Figura 35. Arquitectura cliente-servidor.

Una arquitectura cliente/servidor Web libera, por lo tanto, al usuario final de la aplicación de tener que instalarla en su máquina y consigue que cada usuario solo pueda acceder a la información que le corresponde. Además, este tipo de arquitectura, gracias a su diseño modular, es fácilmente escalable y ampliable tanto en nuevos clientes como en servidores añadidos.

3.5.2. Lenguajes de programación utilizados

Resulta obvio ante la arquitectura y el funcionamiento previsto de nuestra aplicación que el uso de HTML simple y llano no es adecuado sino que se necesita otro lenguaje capaz de generar contenido dinámico desde el servidor de manera transparente al usuario final. Para este fin hemos utilizado los Servlets de Java.

Los servlets pueden considerarse una versión Java muy potenciada de los tradicionales programas CGI. Un servlet es simplemente una clase Java que extiende la clase *javax.servlet.http.HttpServlet*. Las razones para utilizar la tecnología Java/servlets en el desarrollo de aplicaciones Web son varias:

- Frente a los CGIs tradicionales proporcionan un compromiso entre la portabilidad (de la que carece un binario escrito en C/C++) y la eficiencia (frente a scripts Perl o similares).

- Un servlet tiene una implementación mucho más sencilla que un programa/script CGI y un soporte extenso de características avanzadas.

- Frente a páginas PHP/ASP, Java es un lenguaje más eficiente y mucho más aconsejable para aplicaciones de tamaño medio/grande. Las aplicaciones realizadas con PHP/ASP tienden a incumplir con frecuencia el principio de separación modelo/vistas.

Es necesario un servidor de aplicaciones con soporte de servlets/JSP como puede ser por ejemplo Tomcat.

3.5.3. Herramienta de desarrollo

La herramienta escogida para desarrollar todo el código ha sido *NetBeans* (versión 5.0).

3.5.4. Instalación en el servidor y funcionamiento

La instalación de la aplicación así como la puesta en marcha del servidor viene documentada paso a paso en el Anexo II.

Por su parte, en el Anexo III y Anexo IV se encuentran disponibles los manuales de usuario para el Administrador y para el Experto, respectivamente.

3.6. Implantación y Pruebas

Esta fase de implantación junto con la del mantenimiento del software son partes muy importantes de la Ingeniería del software.

Consisten en desplegar el software realizado y tratar de mejorar u optimizar cualquier problema encontrado.

El mantenimiento de software involucra cualquier tipo de pruebas del software realizadas, las cuales son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación.

3.6.1. Pruebas y Validación

El objetivo de esta fase es realizar un conjunto de pruebas sobre el sistema. Con esto intentaremos conseguir llegar a un sistema sin errores garantizando, como hemos dicho, la calidad del software. Para comprobar esto realizaremos unas pruebas de sistema.

3.6.1.1 Casos de Test

Los test diseñados son los siguientes:

Test 1: Identificación de usuario

Requesitos testeados	RF-01
Acción	Un usuario (ya sea Administrador o Experto) introduce su nombre de usuario y contraseña y pulsa el botón *Aceptar*.
Checkpoint 1	El sistema debe mostrar el menú principal de dicho usuario.

Test 2: Identificación incorrecta de usuario

Requesitos testeados	RF-01
Acción	Un usuario introduce su nombre de usuario y contraseña y pulsa el botón *Aceptar*.
Checkpoint 1	El sistema debe mostrar un mensaje de error informando de que el nombre de usuario o la contraseña o ambos son incorrectos.

Test 3: Crear un nuevo problema

Requesitos testeados	RF-03
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Crear un nuevo problema*'
Checkpoint 1	El sistema debe mostrar el formulario para la creación de un problema.
Acción	Introduce los datos del problema y pulsa *Crear*.
Checkpoint 2	El sistema debe mostrar una notificación de que los cambios se han guardado correctamente.

Test 4: Asignar un problema

Requesitos testeados	RF-04
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Asignar un problema*'
Checkpoint 1	El sistema debe mostrar la pantalla para la asignación de un problema.
Acción	Selecciona el problema, los expertos y las alternativas y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una notificación de que los cambios se han guardado correctamente.

Test 5: Consultar la solución de un problema que no ha sido resuelto.

Requesitos testeados	RF-05
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Consultar un problema*'
Checkpoint 1	El sistema debe mostrar los problemas que el administrador ha asignado.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una notificación diciendo que ese problema no se puede consultar porque aún no ha sido resuelto.

Test 6: Consultar la solución de un problema.

Requesitos testeados	RF-05
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Consultar un problema*'
Checkpoint 1	El sistema debe mostrar los problemas que el administrador ha asignado.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una tabla solamente con las soluciones de los operadores que haya resuelto previamente.

Test 7: Resolver un problema que todavía no ha sido valorado por algún o ningún experto .

Requesitos testeados	RF-06
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Resolver un problema*'
Checkpoint 1	El sistema debe mostrar los problemas que el administrador ha asignado.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una notificación diciendo que ese problema no se puede resolver porque aún no ha sido valorado y da una lista de los expertos que faltan por valorarlo.

Test 8: Resolver un problema utilizando el operador OWA y el cuantificador lingüístico 'Most'.

Requesitos testeados	RF-06
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Resolver un problema*'
Checkpoint 1	El sistema debe mostrar los problemas que el administrador ha asignado.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar el conjunto de operadores.
Acción	Selecciona uno de ellos y pulsa *Aceptar.*
Checkpoint 3	El sistema muestra los cuantificadores lingüísticos.
Acción	Selecciona uno y pulsa *Aceptar.*
Checkpoint 4	El sistema muestra la solución en una tabla.

Test 9: Crear una nueva alternativa con un código que ya existe en la base de datos.

Requesitos testeados	RF-07
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Crear una nueva alternativa'.*
Checkpoint 1	El sistema debe mostrar el formulario para la creación de una nueva alternativa.
Acción	Introduce los datos de la alternativa y pulsa *Crear*.
Checkpoint 2	El sistema debe mostrar una notificación diciendo que ese código de alternativa ya está siendo usado por otra alternativa.

Test 10: Dar de alta a un nuevo experto introduciendo solamente el nombre y la contraseña (campo 'Apellidos' en blanco).

Requesitos testeados	RF-08
Precondiciones	Usuario Administrador identificado, se encuentra en el menú del administrador.
Acción	Accede a: '*Dar de alta a un nuevo experto'.*
Checkpoint 1	El sistema debe mostrar el formulario para la creación de un nuevo experto.
Acción	Introduce algunos datos del experto y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar un mensaje de error pidiendo que se rellenen todos los campos.

Test 11: Valorar un problema.

Requesitos testeados	RF-09
Precondiciones	Usuario Experto identificado, se encuentra en el menú del experto.
Acción	Accede a: '*Valorar un problema*'
Checkpoint 1	El sistema debe mostrar los problemas que el experto puede valorar.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar el conjunto de alternativas.
Acción	Valora las alternativas y pulsa *Aceptar.*
Checkpoint 3	El sistema muestra la solución en una tabla.

Test 12: Valorar un problema que ya ha sido valorado previamente.

Requesitos testeados	RF-09
Precondiciones	Usuario Experto identificado, se encuentra en el menú del experto.
Acción	Accede a: '*Valorar un problema*'
Checkpoint 1	El sistema debe mostrar los problemas que el experto puede valorar.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una notificación diciendo que ese problema ya lo ha valorado.

Test 13: Consultar las valoraciones de un problema.

Requesitos testeados	RF-10
Precondiciones	Usuario Experto identificado, se encuentra en el menú del experto.
Acción	Accede a: '*Consultar las valoraciones de un problema'*
Checkpoint 1	El sistema debe mostrar los problemas de los cuales el experto forma parte.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una tabla con las valoraciones de ese problema.

Test 14: Consultar las valoraciones de un problema que no ha sido valorado todavía.

Requesitos testeados	RF-10
Precondiciones	Usuario Experto identificado, se encuentra en el menú del experto.
Acción	Accede a: '*Consultar las valoraciones de problema*'
Checkpoint 1	El sistema debe mostrar los problemas de los cuales el experto forma parte.
Acción	Selecciona el problema y pulsa *Aceptar*.
Checkpoint 2	El sistema debe mostrar una notificación diciendo que ese problema no ha sido valorado aún.

Test 15: Salir del sistema (usuario Administrador o Experto).

Requesitos testeados	RF-02
Precondiciones	Usuario identificado, se encuentra en su menú correspondiente.
Acción	Hace clic en: '*Salir de la aplicación'.*
Checkpoint 1	El sistema debe mostrar el formulario de entrada para volver a entrar en la aplicación.

3.6.1.2. Resultados obtenidos

A continuación mostramos la tabla de resultados de los test diseñados una vez ya realizados sobre el sistema:

TEST	*RESULTADO*
Test 1	
Checkpoint 1	OK
Test 2	
Checkpoint 1	OK
Test 3	
Checkpoint 1	OK
Checkpoint 2	OK
Test 4	
Checkpoint 1	OK
Checkpoint 2	OK
Test 5	
Checkpoint 1	OK
Checkpoint 2	OK
Test 6	
Checkpoint 1	OK
Checkpoint 2	OK
Test 7	
Checkpoint 1	OK
Checkpoint 2	OK
Test 8	
Checkpoint 1	OK
Checkpoint 2	OK
Checkpoint 3	OK
Checkpoint 4	OK
Test 9	
Checkpoint 1	OK
Checkpoint 2	OK
Test 10	
Checkpoint 1	OK
Checkpoint 2	OK

Test 11	
Checkpoint 1	OK
Checkpoint 2	OK
Checkpoint 3	OK
Test 12	
Checkpoint 1	OK
Checkpoint 2	OK
Test 13	
Checkpoint 1	OK
Checkpoint 2	OK
Test 14	
Checkpoint 1	OK
Checkpoint 2	OK
Test 15	
Checkpoint 1	OK

Figura 36. Resultados de las pruebas de test.

CAPÍTULO 4

CONCLUSIONES

CAPÍTULO 4

CONCLUSIONES

Contenidos

4.1. Conclusiones finales

En este proyecto hemos implementado un prototipo de un Sistema de Ayuda a la Decisión, centrándonos fundamentalmente en el estudio de una de las fases de la *Selección de alternativas*, como es la *Agregación*. Para ello, hemos abordado problemas de decisión con múltiples expertos.

En este modelo de Toma de Decisiones Multiexperto con el que hemos trabajado, hay un conjunto de expertos en el que cada uno de ellos valora una serie de las alternativas de acuerdo a sus preferencias suministrando un vector de utilidad en el que hay un valor de preferencia para cada una de las alternativas con el que establece un orden sobre ellas. A continuación, todas las soluciones de los expertos son agregadas en un único vector de utilidad que será la solución al problema planteado.

Para agregar todas las valoraciones que han hecho los expertos del conjunto de alternativas que forma parte de un problema, hemos implementado diferentes operadores, para así, poder resolver dicho problema.

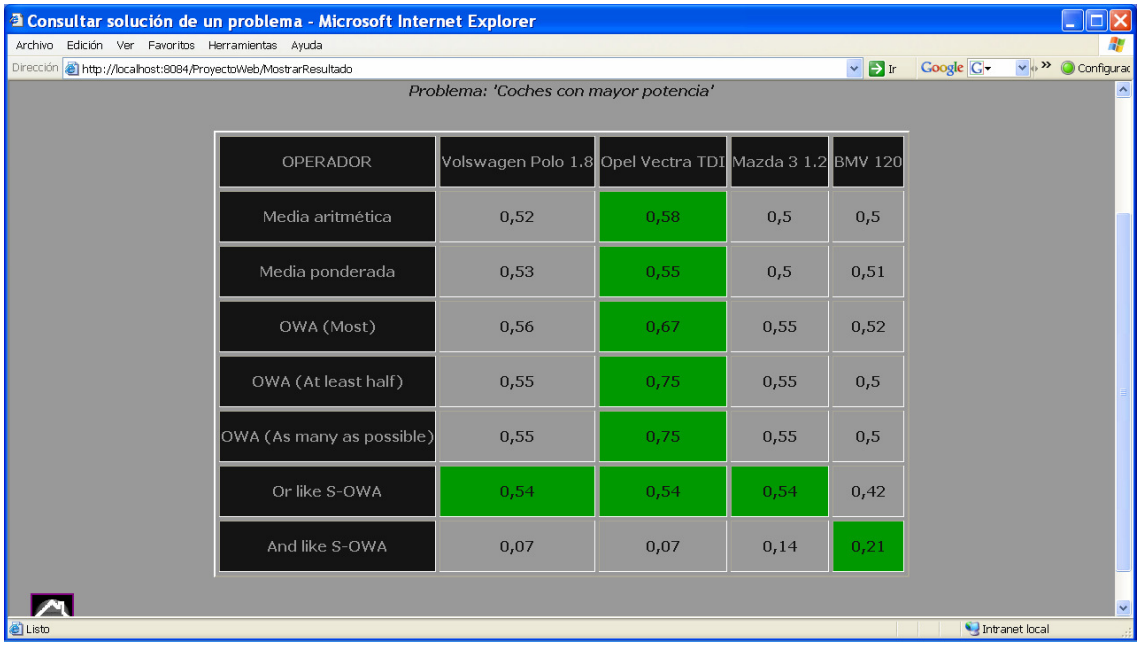
Una vez implementados dichos operadores, hemos resuelto varios problemas variando el número de expertos y de alternativas que lo componían, así como las valoraciones de dichas alternativas, para ver los resultados que nos ofrecían.

Hemos obtenido resultados diferentes dependiendo del operador que se escoja para resolver un problema. Esto es debido a una serie de valores que el Administrador tiene que escoger para resolver un problema, como pueden ser pesos, en el caso de la *Media ponderada*, cuantificador lingüístico, en el caso del operador *OWA*, o un valor adicional que el administrador ha de escoger cuando utiliza los operadores *Or-Like S-OWA* y *And-Like S-OWA*.

Estos resultados, de los cuales podemos ver ejemplos en las figuras *Figura 37, Figura 38, Figura 39, Figura 40* y *Figura 41,* indican que dependiendo del problema que pretendamos solucionar, el operador escogido será distinto, ya que los operadores implementados aportan diferentes soluciones a un mismo problema.

Debido a estas conclusiones, puede ser interesante proponer para trabajos futuros, estudiar la importancia de estos operadores y valores que el Administrador tiene que escoger, para saber cómo influyen en los resultados, y así, sacar conclusiones que nos ayuden a tomar una buena decisión sobre un problema.

Otra posible investigación futura podría ser la ampliación de este proyecto para trabajar no sólo con problemas multiexperto, sino también con problemas multicriterio. Así, también se podrían agregar todos los atributos o criterios que formaran parte de una alternativa, para ser tenidos en cuenta en la resolución del problema planteado.



OPERADOR	Volswagen Polo 1.8	Opel Vectra TDI	Mazda 3 1.2	BMV 120
Media aritmética	0,52	0,58	0,5	0,5
Media ponderada	0,53	0,55	0,5	0,51
OWA (Most)	0,56	0,67	0,55	0,52
OWA (At least half)	0,55	0,75	0,55	0,5
OWA (As many as possible)	0,55	0,75	0,55	0,5
Or like S-OWA	0,54	0,54	0,54	0,42
And like S-OWA	0,07	0,07	0,14	0,21

Figura 37. Ejemplo de solución al problema 'Coches con mayor potencia'.

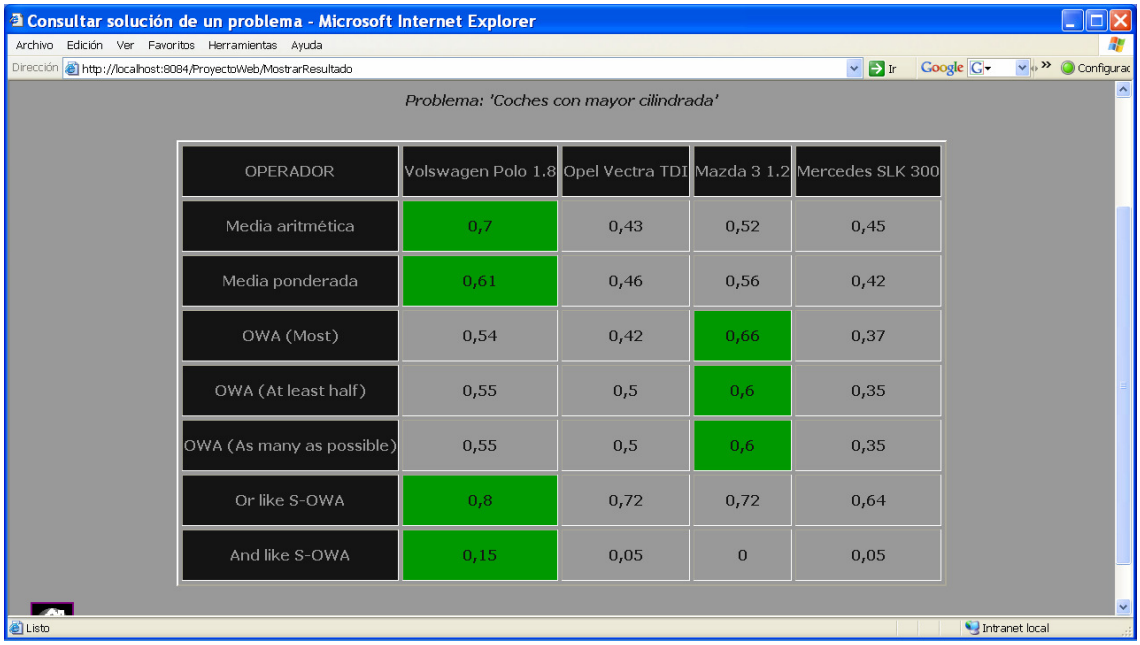


Figura 38. Ejemplo de solución al problema 'Coches con mayor cilindrada'.

Figura 39. Ejemplo de solución al problema 'Coches de flotilla'.

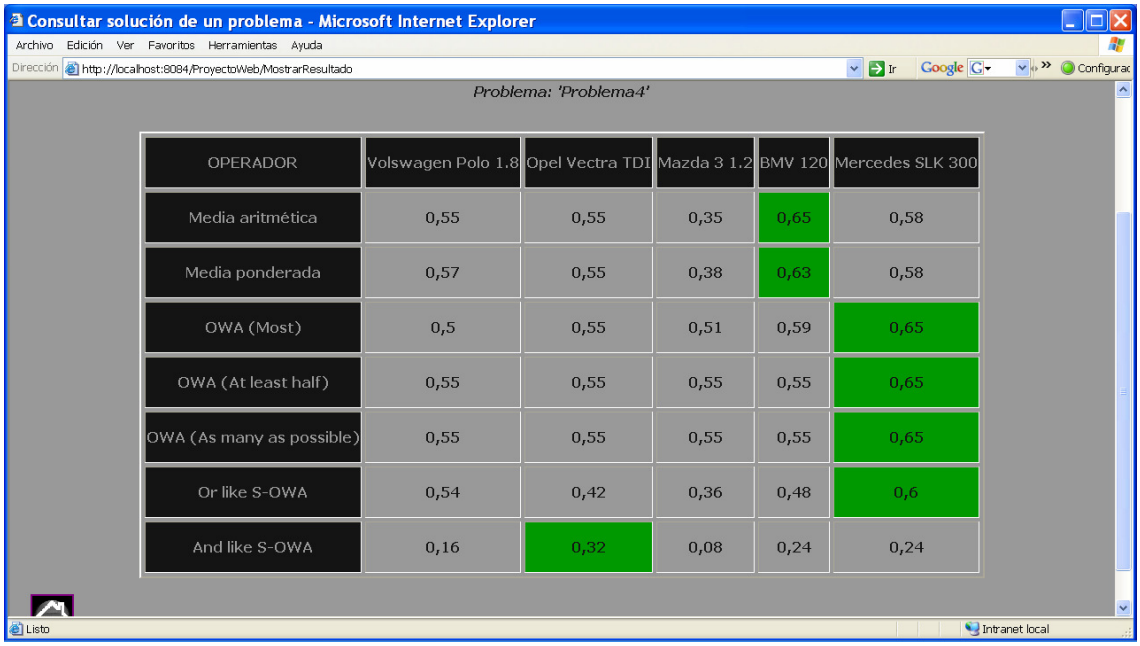


Figura 40. Ejemplo de solución al problema 'Problema4'.

Figura 41. Ejemplo de solución al problema 'Problema5'.

REFERENCIAS Y BIBLIOGRAFÍA

REFERENCIAS Y BIBLIOGRAFÍA

Contenidos

Referencias

[1] L. A. Zadeh, Fuzzy Sets, *Information and Control*, 8 (1965) 338-353.

[2] J. A. Goguen, The Logic of Inexact Concepts, *Synthese*, 19 (1969) 325-373.

[3] J. A. Goguen, L-Fuzzy Sets, JMAA, 18 (1967) 145-174.

[4] P. P. Bonissone, K.S. Decker, Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off and Complexity. In: Kanal, Lemmer, Eds., *Uncertainty in Artificial Intelligence* (North-Holland, 1986) 217-247.

[5] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications* (Kluwer, Dordrecht, 1991).

[6] H.-J. Zimmermann, P. Zysno, Latent Connectives in Human Decision Making, *Fuzzy Sets and Systems*, 4 (1980) 37-51.

[7] R.R. Yager, On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making, IEEE Transaction on Systems, *Man and Cybernetics*, 18 (1988) 183-190.

[8] R.R. Yager, Families of OWA Operators, *Fuzzy Sets and Systems*, 59 (1993) 125-148.

[9] L. A. Zadeh, A Computational Approach to Fuzzy Quantifiers in Natural Languages, *Computers and Mathematics with Applications*, 9 (1983) 149-184.

[10] R.R. Yager, Connectives and Quantifiers in Fuzzy Sets, *Fuzzy Sets and Systems*, 40 (1991) 39-75.

[11] R.R. Yager, Interpreting Linguistically Quantified Propositions, *International Journal of Intelligent Systems*, 9 (1994) 541-569.

[12] R.R. Yager, On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making, *IEEE Transactions on Systems, Man, and Cybernetics* 18 (1988) 183-190.

[13] R.R. Yager, D.P. Filev, Parameterized And-Like and Or-Like OWA Operators, *Int. Journal General Systems* 22 (1993) 297-316.

[14] J. Aczél, OnWeighted Synthesis of Judgements, *Aequationes Math*. 27 (1984) 288-307.

[15] J. Aczél, OnWeighted Synthesis of Judgements, *Aequationes Math*. 27 (1984) 288-307.

[16] D. Dubois, H. Prade, A Review of Fuzzy Sets Aggregation Connectives, Information Sciences, 36 (1985) 85-121.

[17] D. Dubois, J.-L. Koning, Social Choice Axioms for Fuzzy Sets Aggregation, Fuzzy Sets and Systems, 43 (1991) 257-274.

[18] J. Kacprzyk, M. Roubens, Non-Conventional Preference Relations in Decision Making (Springer-Verlag, 1988).

Bibliografía

http://tomcat.apache.org/tomcat-4.1-doc/index.html **Documentación Apache Tomcat4.1**

http://www.appservnetwork.com/ **Página de AppServ**

ANEXO I.

CUANTIFICADORES LINGÜÍSTICOS

ANEXO I.

CUANTIFICADORES LINGÜÍSTICOS

Contenidos

Cuantificadores Lingüísticos Difusos

Los cuantificadores se usan para representar la cantidad de elementos que satisfacen un predicado. La Lógica Clásica se restringe al uso de dos cuantificadores, "existe" y "para todo", los cuales se relacionan con los conectivos "o" e "y", respectivamente.

Sin embargo, en el lenguaje natural se utilizan muchos y diversos cuantificadores, por ejemplo "bastantes", "casi todos", "muchos más que la mitad", etc., los cuales no pueden manejarse por métodos formales convencionales, pues en éstos suelen considerarse tan sólo dos cuantificadores, que son "al menos uno" y "todos". Afortunadamente, en años recientes han sido propuestos cálculos de proposiciones cuantificadas lingüísticamente basados en lógica difusa [1, 2, 3]. Estos cálculos han sido aplicados por Kacprzyk y otros [4, 5, 6, 7] para introducir una mayoría difusa, representada por un cuantificador lingüístico difuso, tanto en los modelos de toma de decisiones en grupo como en la implementación de sistemas soporte para la obtención de consenso.

Los cuantificadores pueden utilizarse para representar la cantidad de elementos que satisfacen una determinada propiedad. Los cuantificadores que habitualmente se utilizan en el quehacer diario son muchos más, como por ejemplo*, aproximadamente 5, casi todos, unos pocos, muchos, la mayor parte de, casi la mitad, al menos la mitad*, etc. En un intento de proporcionar una representación de tales cuantificadores, Zadeh introdujo el concepto de cuantificador lingüístico difuso [1], el cual fue definido como un conjunto difuso. Él distinguió entre dos tipos de cuantificadores lingüísticos, *absolutos* y *proporcionales* o *relativos*. Los primeros se utilizan para representar cantidades que son absolutas en naturaleza tales como por ejemplo *aproximadamente 5, más de 5 o menos de 5*. Estos cuantificadores lingüísticos absolutos están relacionados con el concepto de número de elementos. Zadeh definió estos cuantificadores como subconjuntos difusos, Q, del conjunto de números reales no negativos, R^+, de forma que para $r \in R^+$ el grado de pertenencia de r en Q, $Q(r)$ indica el grado con el que la cantidad r es compatible con el cuantificador representado por Q. Los cuantificadores proporcionales, como por ejemplo al menos la

mitad o la mayor parte, pueden representarse mediante subconjuntos difusos del intervalo [0,1].

Para $r \in [0,1]$, $Q(r)$ indica el grado con el que la proporción r es compatible con el significado del cuantificador que representa. Cualquier cuantificador del lenguaje natural puede ser representado como un cuantificador proporcional o, supuesto conocida la cantidad de elementos en consideración, como un cuantificador absoluto. Funcionalmente, un cuantificador lingüístico puede ser de tres tipos, creciente, decreciente o unimodal.

Un cuantificador creciente viene caracterizado por la relación

$$Q(r_1) \geq Q(r_2) \quad si \quad r_1 \geq r_2$$

Ejemplos de este tipo de cuantificador son la mayor parte de, al menos la mitad, la mayor cantidad posible.

Los cuantificadores unimodales tienen la propiedad

$$Q(a) \leq Q(b) \leq Q(c) = 1 \geq Q(d)$$

para algunos $a \leq b \leq c \leq d$. Estos cuantificadores son útiles para representar términos como *aproximadamente q*.

Un cuantificador absoluto, $Q : R^+ \rightarrow [0,1]$, verifica

$$Q(0) = 0 \ \wedge \ \exists k \ tal \ que \ Q(k) = 1$$

mientras que un cuantificador relativo, Q:[0,1] $\rightarrow$ [0,1], satisface la propiedad

$$Q(0) = 0 \ \wedge \ \exists r \rightarrow [\,0,1] \ \text{tal que} \ Q(r) = 1$$

La función de pertenencia de un cuantificador relativo creciente puede representarse como sigue:

$$Q(r) = \begin{cases} 0 & si\ r < a \\ r-a/b-a & si\ a \le r \le b \\ 1 & si\ r > b \end{cases}$$

con a, b, r $\rightarrow$ [0,1]. Algunos ejemplos de cuantificadores relativos crecientes se muestran en la figura 2 donde los parámetros (a,b) son (0,0.5), (0.5,1) y (0.3,0.8) respectivamente.

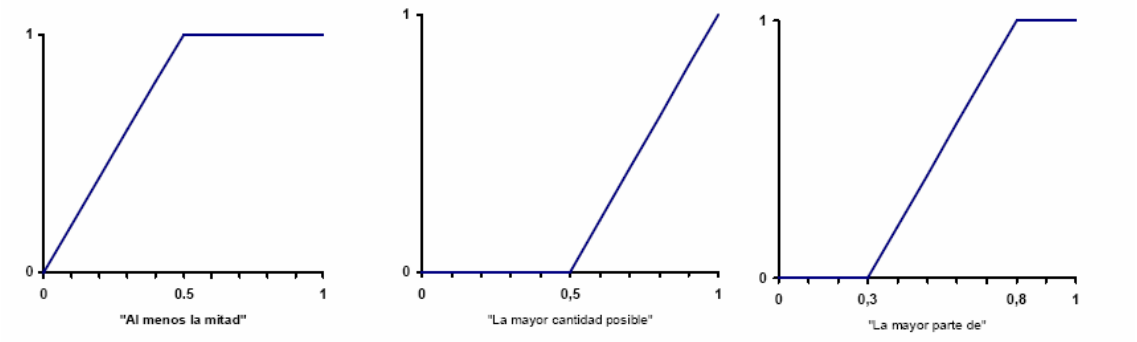


Figura 2. Cuantificadores lingüísticos difusos relativos

Yager [1] sugirió una interesante forma de calcular los pesos del operador de agregación OWA usando un cuantificador lingüístico difuso, que en el caso de un cuantificador proporcional creciente Q, viene dado por la ecuación:

$$w_i = Q(i/n) - Q(i-1/n), \quad i=1,...,n.$$

Cuando para calcular los pesos del operador OWA Φ se utiliza un cuantificador lingüístico difuso Q, entonces dicho operador se simboliza por Φ_Q.

Referencias Anexo I

[1] L. A. Zadeh, A Computational Approach to Fuzzy Quantifiers in Natural Languages, *Computers and Mathematics with Applications*, 9 (1983) 149-184.

[2] R.R. Yager, Connectives and Quantifiers in Fuzzy Sets, *Fuzzy Sets and Systems*, 40 (1991) 39-75.

[3] R.R. Yager, Interpreting Linguistically Quantified Propositions, *International Journal of Intelligent Systems*, 9 (1994) 541-569.

[4] J. Kacprzyk, M. Fedrizzi, Multiperson Decision Making Models Using Fuzzy Sets and Possibility Theory (Kluwer Academic Pub., Dordrecht, 1990).

[5] F. Herrera, E. Herrera-Viedma, J.L. Verdegay, A Sequential Selection Process in Group Decision Making with a Linguistic Assessment Approach, Information Sciences, 85 (1995) 223-239.

[6] F. Herrera, E. Herrera-Viedma, J.L. Verdegay, A Model of Consensus in Group Decision Making Under Linguistic Assessment, Fuzzy Sets and Systems, 78 (1996) 73-87.

[7] E. Herrera-Viedma, Modelos Lingüísticos para la Toma de Decisiones en Grupo, Tesis Doctoral, Granada, Abril, 1996.

ANEXO II.

MANUAL DE INSTALACIÓN DEL SERVIDOR

Este anexo está dedicado ha realizar las configuraciones necesarias para poner en marcha nuestra aplicación.

Las únicas consideraciones previas que han de tenerse en cuenta son que durante todo este manual se ha supuesto que la unidad principal de disco duro es C:, que la unidad principal de disco flexible es D:.

Hemos de decir que nuestro sistema trabajará con un servidor basado en la plataforma Microsoft Windows, por lo que todas estas instalaciones y configuraciones hay que realizarlas teniendo en cuenta dicho Sistema Operativo.

Todo el material necesario para instalar y dejar operativo el servidor se encuentra disponible en el CD que acompaña a esta memoria. Sitúese en el dispositivo de CD-ROM D:\TomaDecision y compruebe que se encuentran los siguientes archivos:

- jdk-6-windows-i586.exe
- apache-tomcat-4.1.36-LE-jdk14
- appserv-win32-2.5.5.exe
- ProyectoWeb.war

Si es así podemos proceder a la instalación de nuestro servidor inmediatamente. Si falta algún archivo o alguno de ellos se encuentra dañado póngase en contacto con el responsable de la aplicación para subsanar el percance.

Si ya dispone de un servidor Apache Tomcat 4.1 o superior sólo tendrá que copiar el archivo ProyectoWeb.war en el directorio webapp, que se encuentra donde tenga instalado el servidor. Una vez hecho esto podrá acceder a la aplicación tecleando en su navegador http://www.ip-del-servidor/ProyectoWeb. En caso contrario siga los siguientes pasos.

Instalar *AppServ*

AppServ es un software que nos permite instalar sobre Windows los siguientes paquetes de forma conjunta:

- Apache Web Server
- Lenguaje PHP
- Base de datos MySQL
- Manejador de base de datos phpMyAdmin

Es una aplicación muy útil para empezar a familiarizarse con Gestores de Contenidos, aprender a configurarlos e instalarlos, ya que permite realizar pruebas sobre nuestro PC.

1. El primer paso es ejecutar el software, para ello hacemos doble clic en el ejecutable *appserv-win32-2.5.5.exe*. Esperamos la precarga:

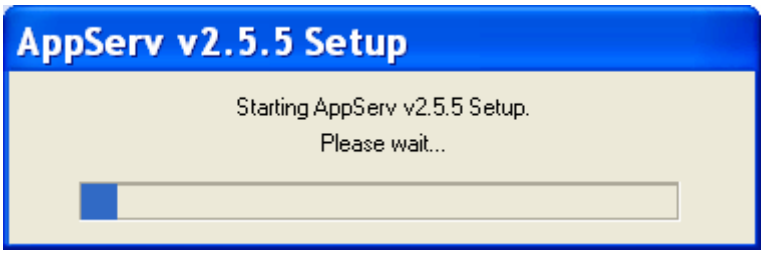


Figura II_1. Precarga.

y aparece la ventana de bienvenida:

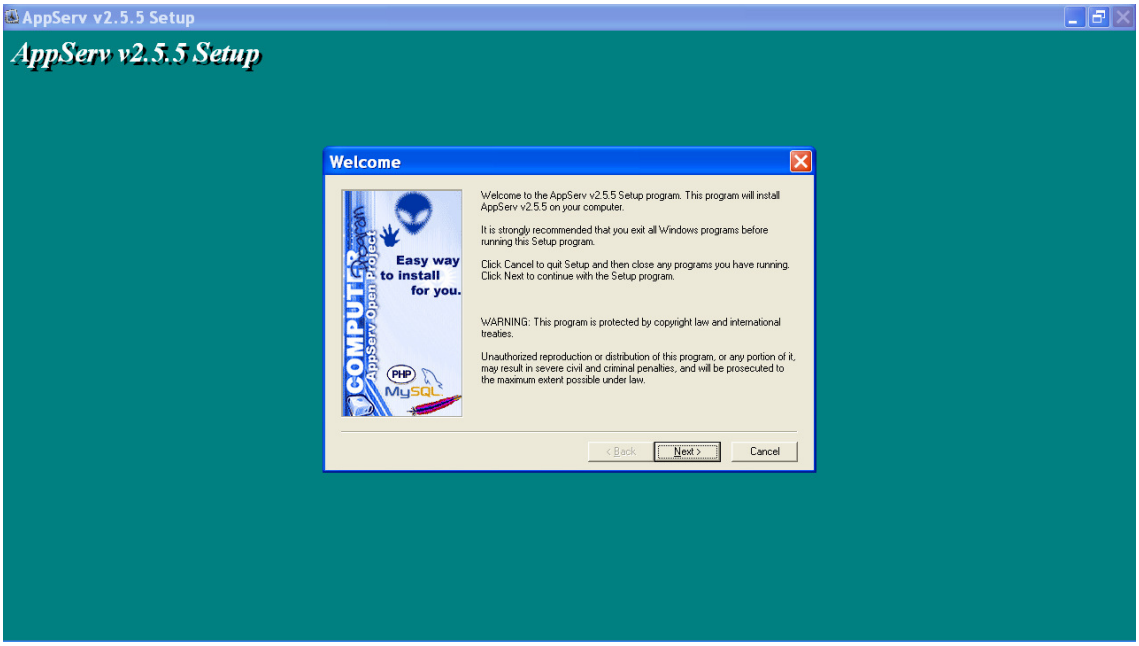


Figura II_2. Ventana de bienvenida.

2. Hacemos clic en *Next* y nos aparece la ventana en donde escogemos la ruta de instalación, la que aparece por defecto es: C:\Appserv, esta ruta se recomienda dejarla así:

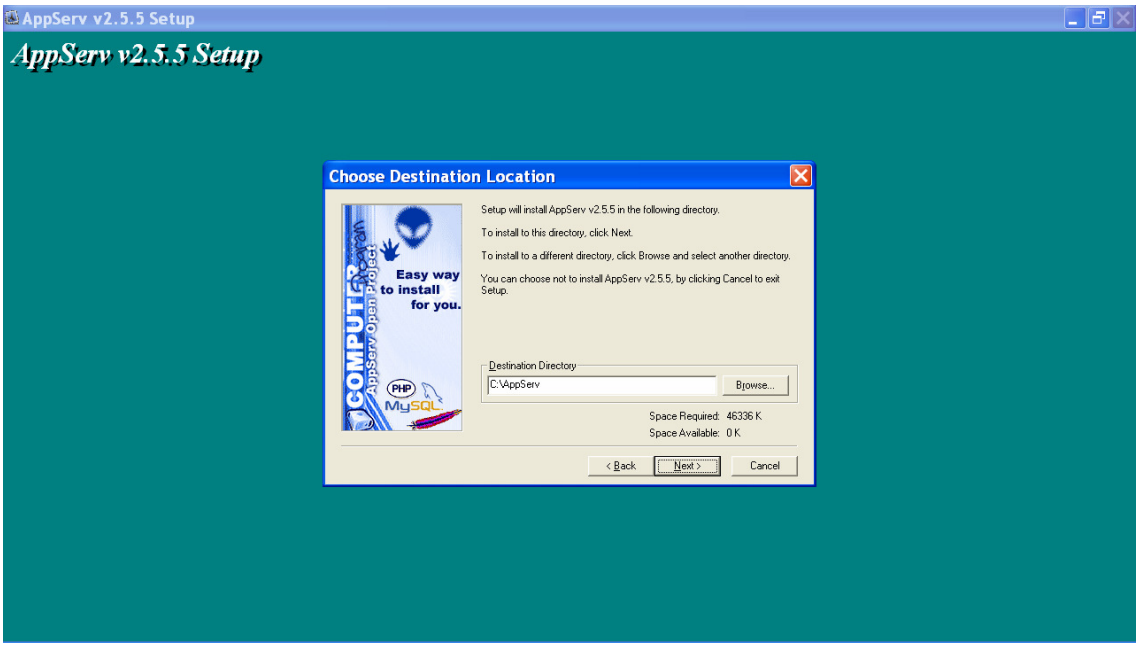


Figura II_3. Selección del directorio de instalación.

3. Hacemos clic en *Next*, y ahora pide escoger el tipo de instalación, por defecto la dejaremos en *Typical*, además de esto muestra la cantidad de espacio en disco que se necesita y la cantidad de espacio que se posee realmente. Si el espacio requerido es mayor que el que se tiene, se puede cambiar la ruta de instalación o liberar espacio en disco:

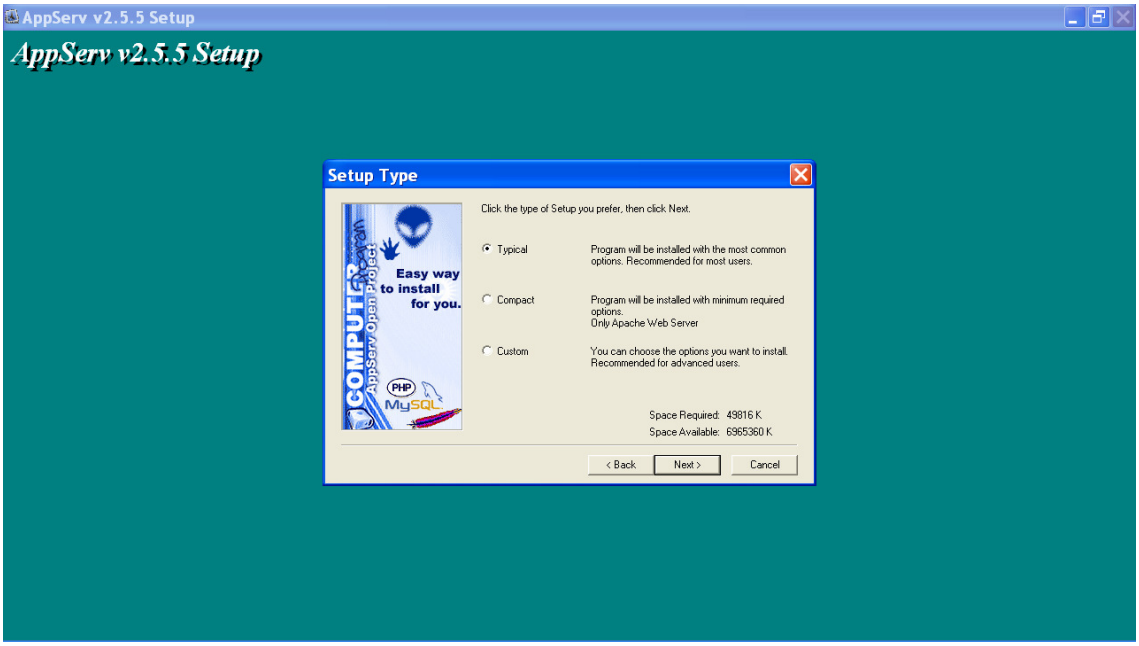


Figura II_4. Selección del tipo de instalación.

4. Después hacemos clic en *Next* y aparece a continuación la pantalla de configuración del servidor: En "*Server Name*" colocamos como vamos a llamar al Appserv desde nuestro explorador de Internet (comúnmente se coloca **localhost**, pero si se puede poner el nombre de una página Web, aunque no es recomendable). El "*Administrator's Email*" se deja así como aparece o se pone una dirección de e-mail. El campo "*HTTP Port*" se deja el que viene por defecto:

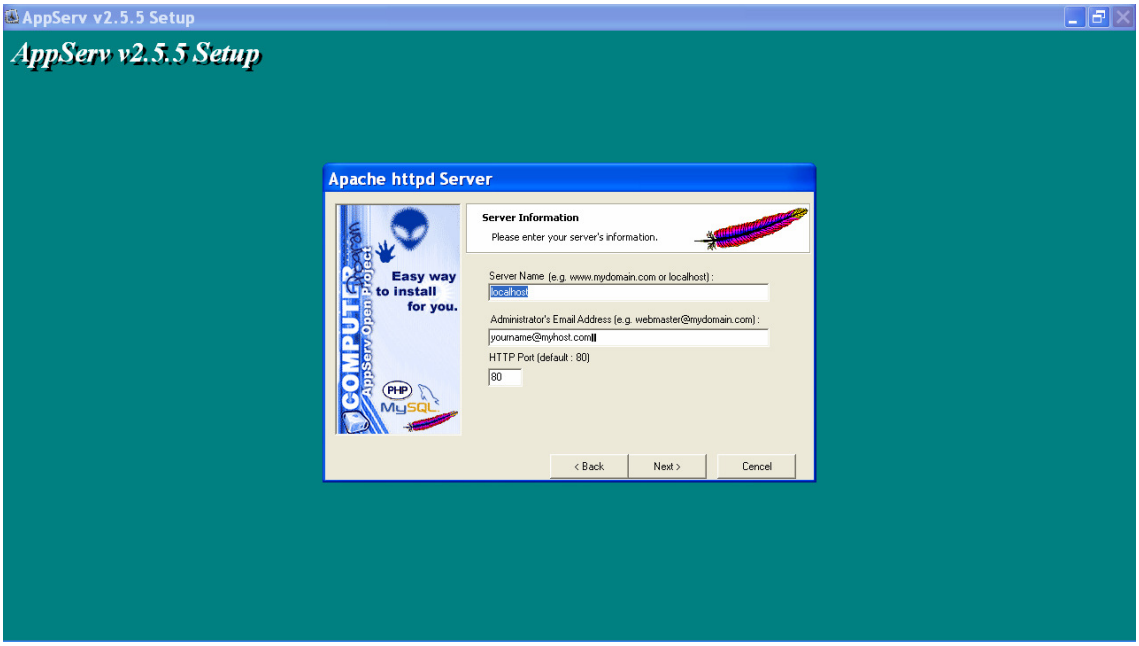


Figura II_5. Selección del nombre del servidor, mail y puerto.

5. Hacemos clic en *Next* y ahora necesitamos configurar la segunda parte del servidor que es el acceso. En el campo "*User Name*" debemos colocar un nombre de usuario cualquiera, por ejemplo: *root*. En el campo "*Password*" se puede poner una contraseña que no es obligatoria y en el campo "*Charset*" se deja por defecto en *latin1*.

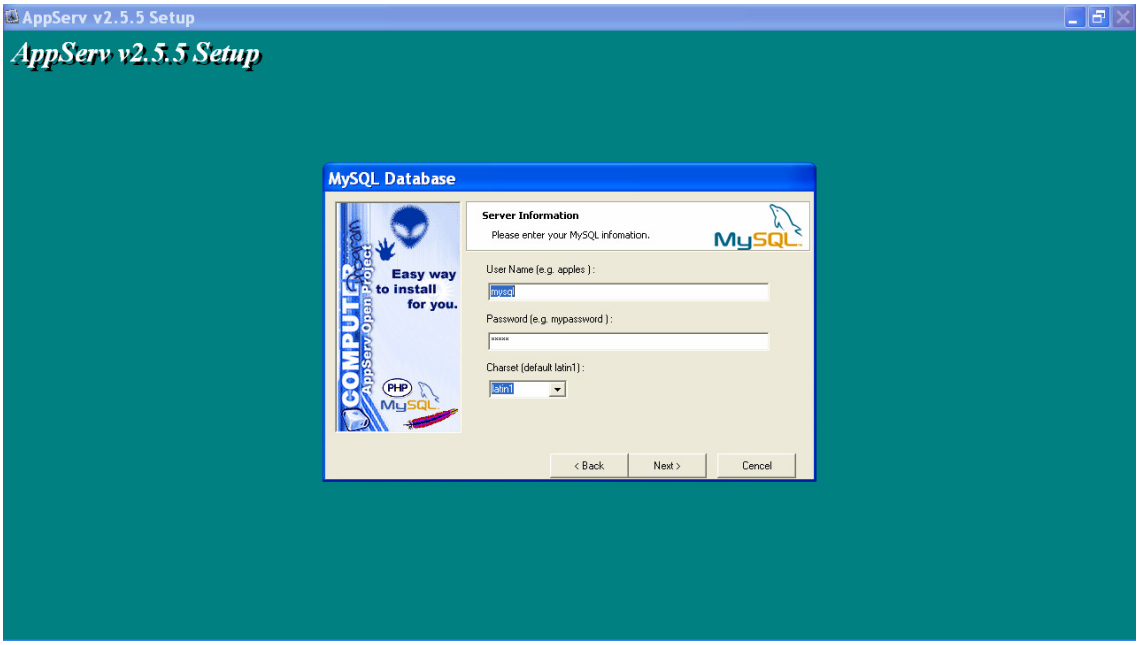


Figura II_6. Selección del acceso.

6. Ahora dejamos que el proceso de instalación termine.

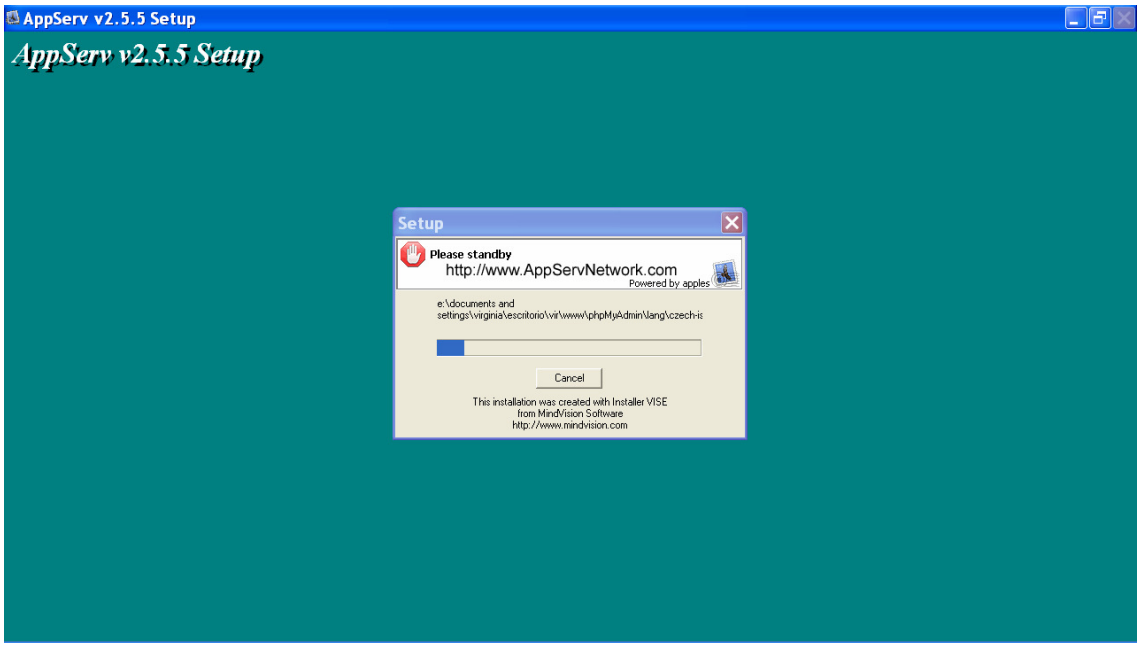


Figura II_7. Proceso de instalación.

7. En la siguiente pantalla dejamos las 2 opciones que se muestran activas, y hacemos clic en *Close*, esperamos, y ya tenemos instalado Appserv en nuestro PC.

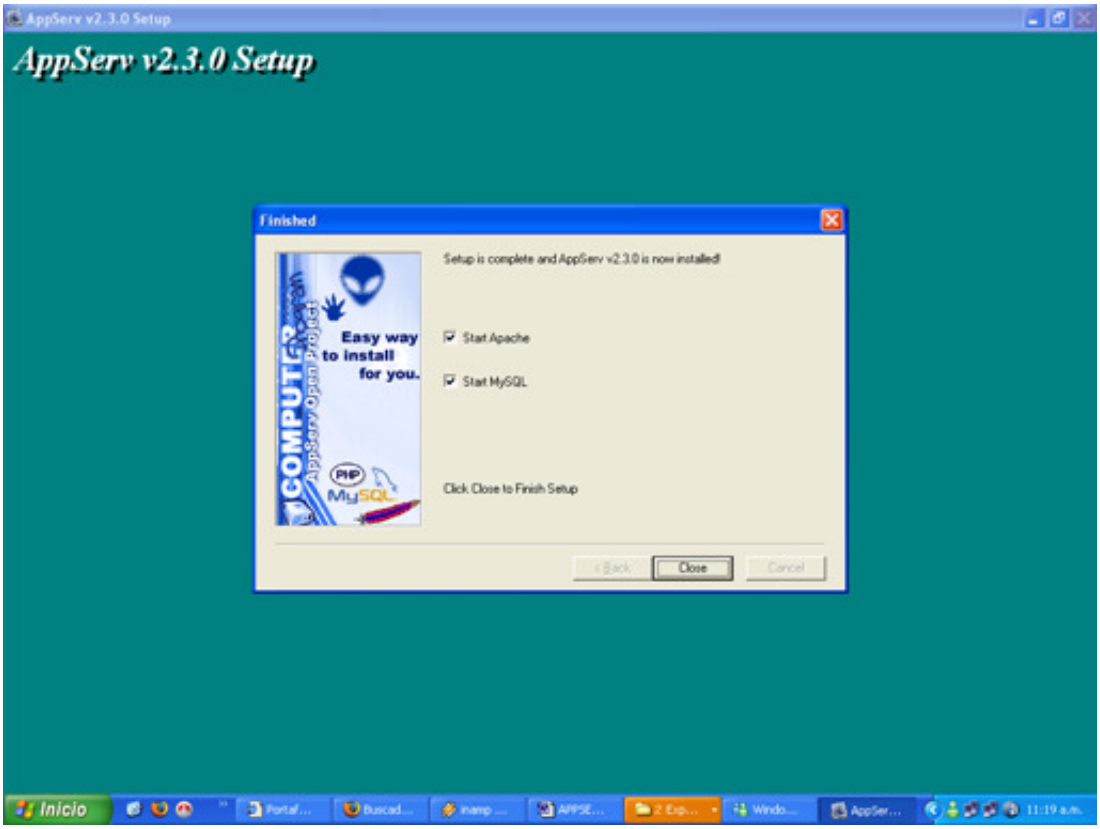


Figura II_8. Instalación finalizada.

Si se han seguido bien estos pasos, al poner en nuestro navegador: *http://localhost*, debe aparecer la pantalla siguiente:

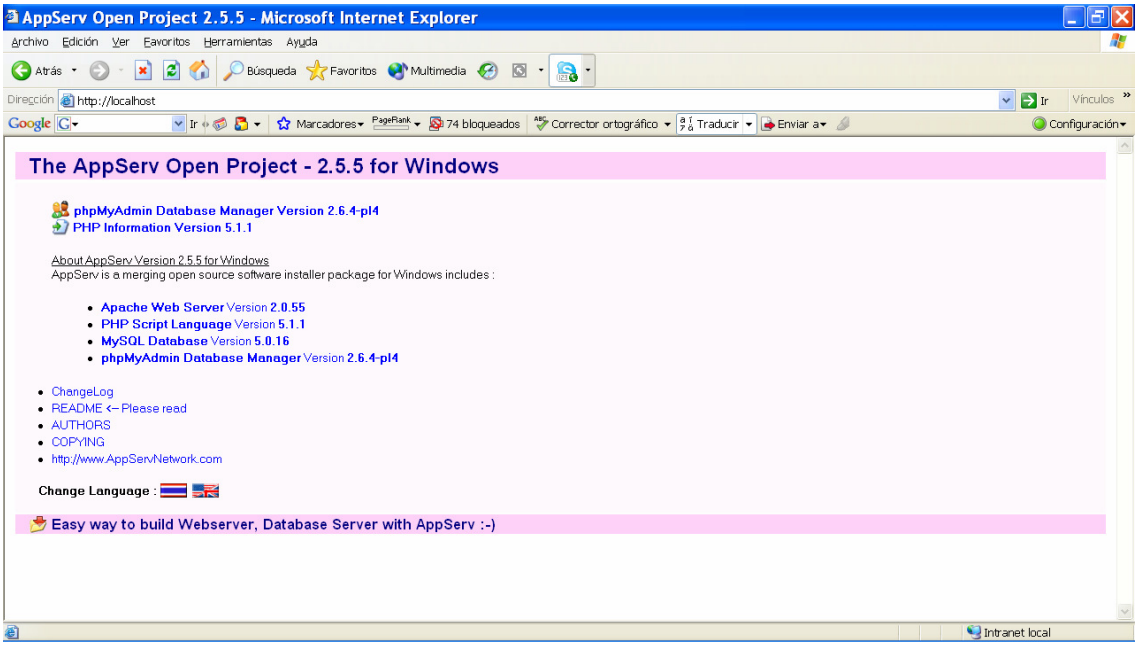


Figura II_9. Página de inicio de AppServ.

Instalar Apache-Tomcat

Al tratarse nuestro proyecto de una aplicación Web, es lógico necesitar un servidor Web para que proporcione los servicios o responda a las peticiones realizadas por cualquier cliente que se quiera conectar al sistema. Para ello utilizaremos el servidor Web Apache Tomcat.

En esta sección explicaremos los pasos a seguir para realizar la instalación y configuración del Servidor Apache Tomcat.

1. En primer lugar necesitaremos disponer del programa de instalación. Este se encuentra en el CD-ROM entregado. El nombre del archivo de instalación es: *apache-tomcat-4.1.36-LE-jdk14.*

2. La primera pantalla, figura II.1, que aparece nos indica que tenemos instalado un JDK superior a la versión 1.4, por lo que podemos seguir con la instalación.

Figura II_10. JDK instalado

3. Una vez que pulsamos aceptar nos aparece una ventana con los términos de licencia. Pulsamos el botón "*I Agree"* para continuar con la instalación.

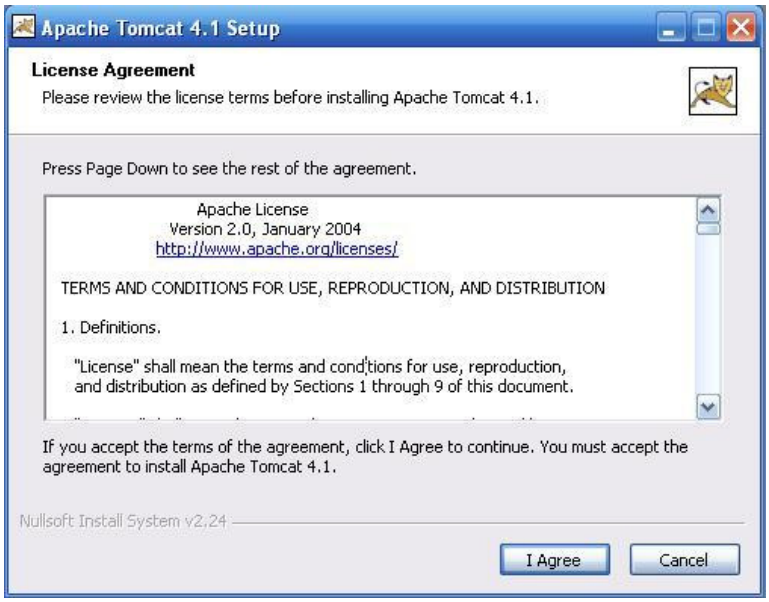


Figura II_11. Términos de la licencia de uso

4. Seleccionamos la instalación *Normal* y pulsamos *Next*.

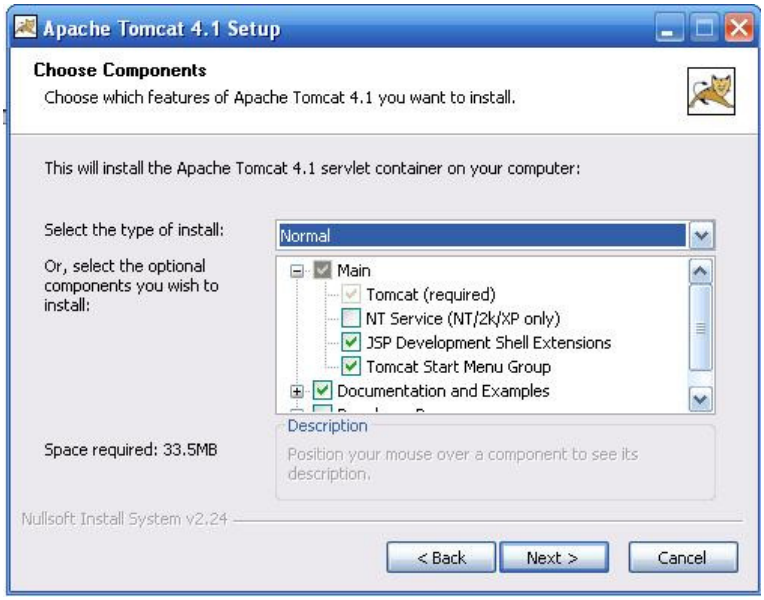


Figura II_12. Selección del tipo de instalación.

5. Elegimos el lugar donde se va a realizar la instalación y pulsamos *Install*.

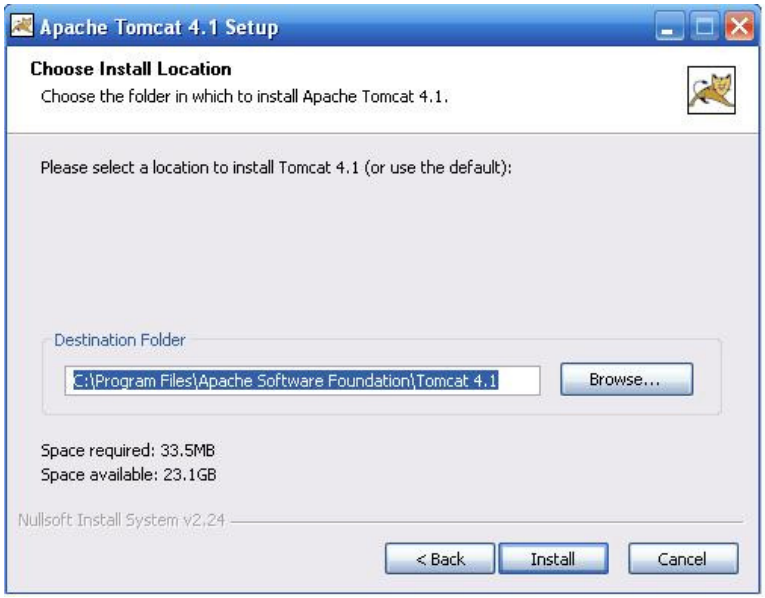


Figura II_13. Selección del directorio de instalación

6. Comienza el proceso de instalación.

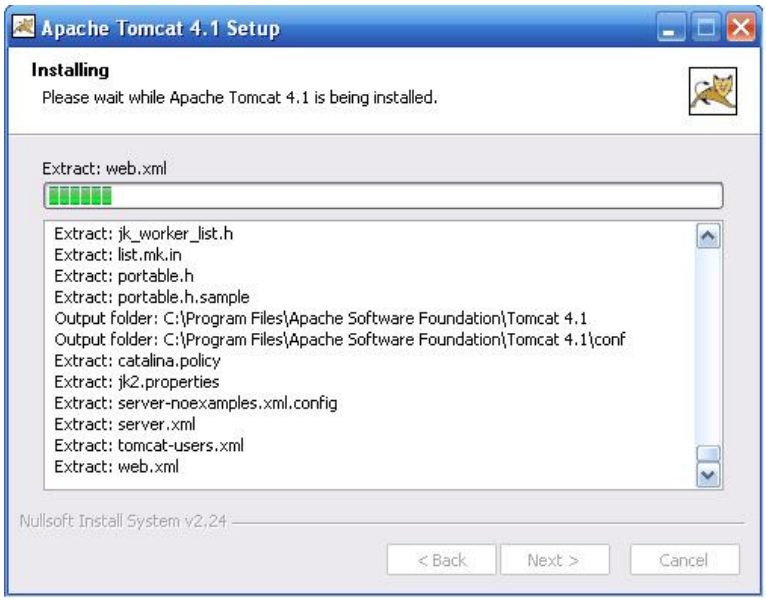


Figura II_14. Durante la instalación.

7. Una vez finalizada la instalación pulsamos *Next* y nos aparece una pantalla

como la de la *Figura II_15*, en la que debemos introducir un usuario y una contraseña para el administrador y el puerto por el cual queremos que se lleve a cabo la comunicación, por defecto el 8080.

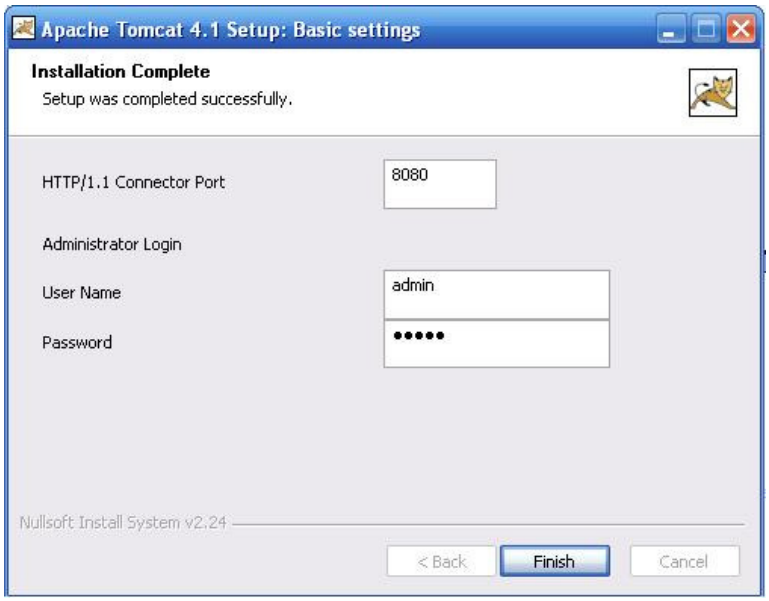


Figura II_15. Selección de usuario y puerto

8. Pulsando *Finish* termina la instalación de Apache Tomcat.

Una vez finalizada la instalación debemos poner en marcha el servidor Apache Tomcat para ello, existen dos opciones:

- Mediante variable de entorno:

  Para esto, tenemos que establecer la variable de entorno CATALINA_HOME al directorio donde tenga instalado Tomcat 4.1. Ejecutando el siguiente comando: %CATALINA_HOME%\bin\startup

- Ejecutando el siguiente comando:

  cd %CATALINA_HOME%\bin
  startup

Después de iniciar, las aplicaciones por defecto incluidas con Tomcat 4.1 deben ser visualizadas si en: http://localhost:8080/

Para detener el servidor dispone de dos opciones:

- Mediante variable de entorno:

  Para esto, tenemos que establecer la variable de entorno CATALINA_HOME al directorio donde tenga instalado Tomcat 4.1. Ejecutando el siguiente comando: %CATALINA_HOME%\bin\shutdown

- Ejecutando el siguiente comando:

  cd %CATALINA_HOME%\bin
  shutdown

Puede encontrar más información sobre como configurar y ejecutar Tomcat 4.1 en la siguiente dirección: http://tomcat.apache.org/

ANEXO III.

MANUAL DE USUARIO: ADMINISTRADOR

Este manual de usuario está organizado como una visita guiada por la aplicación, cuando accedemos a la misma como Administrador.

El primer paso para utilizar la aplicación es abrir el Navegador (Internet Explorer) y teclear la dirección en la que tenemos alojada ésta, que en nuestro caso será: http://localhost:8084/ProyectoWeb/index.jsp.

Al realizar lo anteriormente comentado, nos encontramos con la página de inicio, tal y como se ve en la *Figura III_1*.

Figura III_1. Entrada al sistema.

Para entrar como administrador debemos introducir el nombre de usuario y contraseña del administrador. Si alguno de estos datos (o los dos) no es correcto, el sistema nos muestra un mensaje de error como se observa en la *Figura III_2*.

Figura III_2. Mensaje de error en la Identificación.

En el caso de una identificación correcta nos aparece el menú principal del administrador, como podemos ver en la *Figura III_3*.

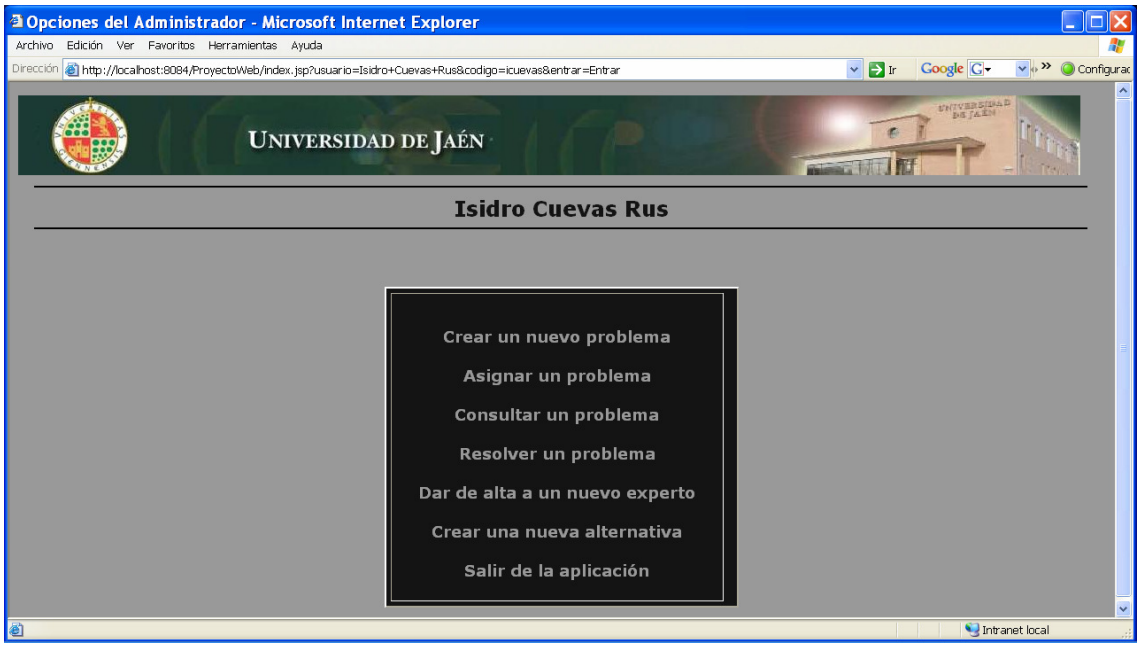


Figura III_3. Menú principal del Administrador.

En este menú principal podemos encontrar todas las operaciones disponibles para el Administrador del sistema.

A continuación veremos de forma detallada cada una de estas partes de la aplicación.

1. **Crear un nuevo problema.**

   Si hacemos clic en la primera de las opciones del Administrador nos aparece un formulario como el que se muestra en la *Figura III_4* para crear un nuevo problema.

Figura III_4. Formulario para la creación de un nuevo problema.

Si al rellenar el formulario para la creación del problema, el administrador se deja algún campo por rellenar nos aparecerá el mensaje de error de la *Figura III_5.*

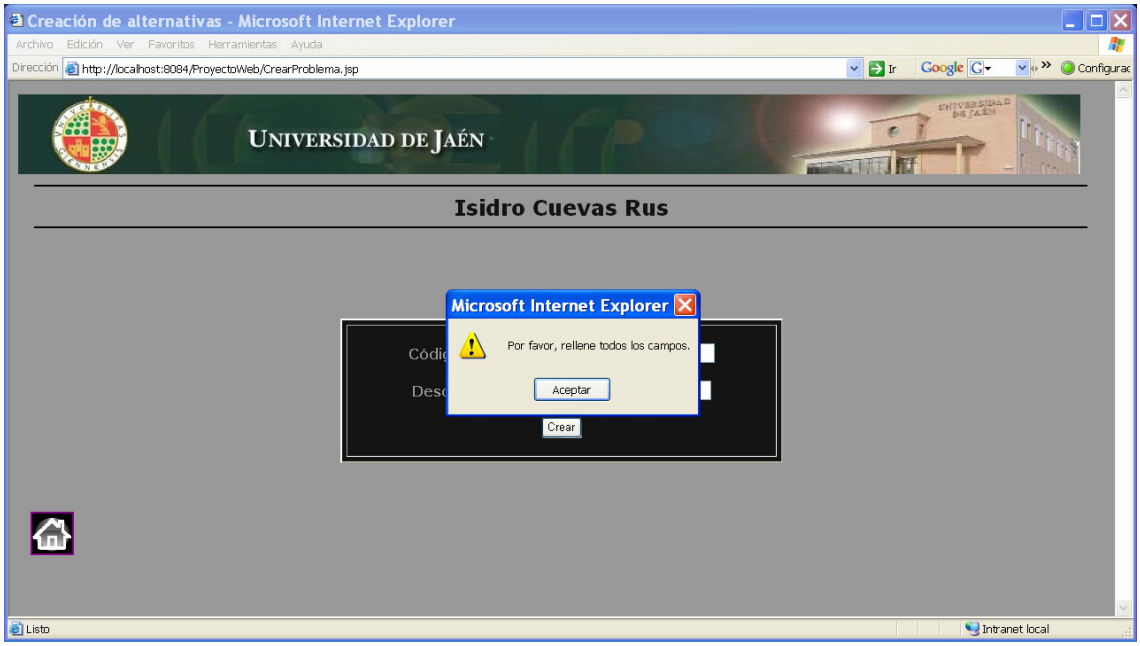


Figura III_5. Mensaje de error de 'Creación de un Problema'.

Si el Administrador introduce un código de problema que ya está siendo usado por otro problema, nos aparece la notificación que se observa en la *Figura III_6.*

Figura III_6. Notificación para 'Creación de un Problema' (código erróneo).

Si no existe aún ningún código de problema como el que introduce el Administrador, nos aparece la notificación de la *Figura III_7.*

Figura III_7. Notificación para la 'Creación de un Problema'.

2. **Asignar un problema.**

   Al hacer clic en esta opción nos aparece la pantalla que muestra la *Figura III_8.*:

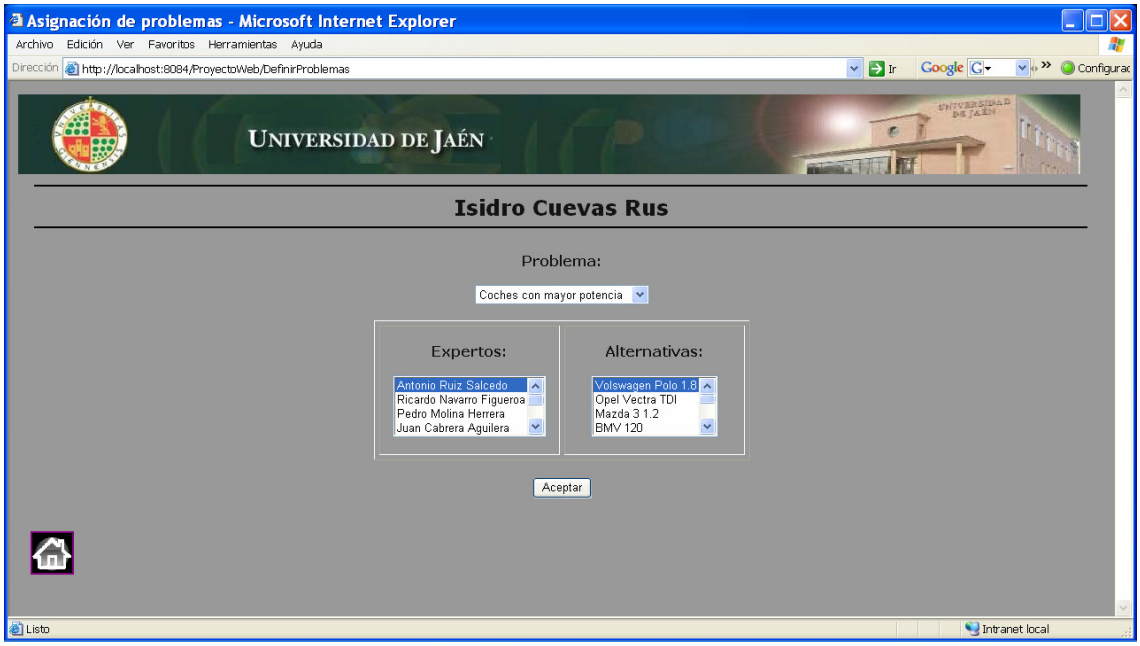


Figura III_8. Pantalla de 'Asignación de Problemas'.

Esta opción ayuda al Administrador a asignarle a un problema un conjunto de expertos y alternativas. Al Administrador se le muestran todos los problemas, expertos y alternativas que ha creado previamente.

Cuando se termine de asignar un problema, se pulsa *Aceptar* y nos aparecería la notificación que se muestra en la *Figura III_9.*

Figura III_9. Notificación en 'Asignación de Problemas'.

3. Consultar un problema.

Si el Administrador quiere consultar los resultados de un problema tendrá que hacer clic en esta opción y aparecerá la siguiente pantalla:

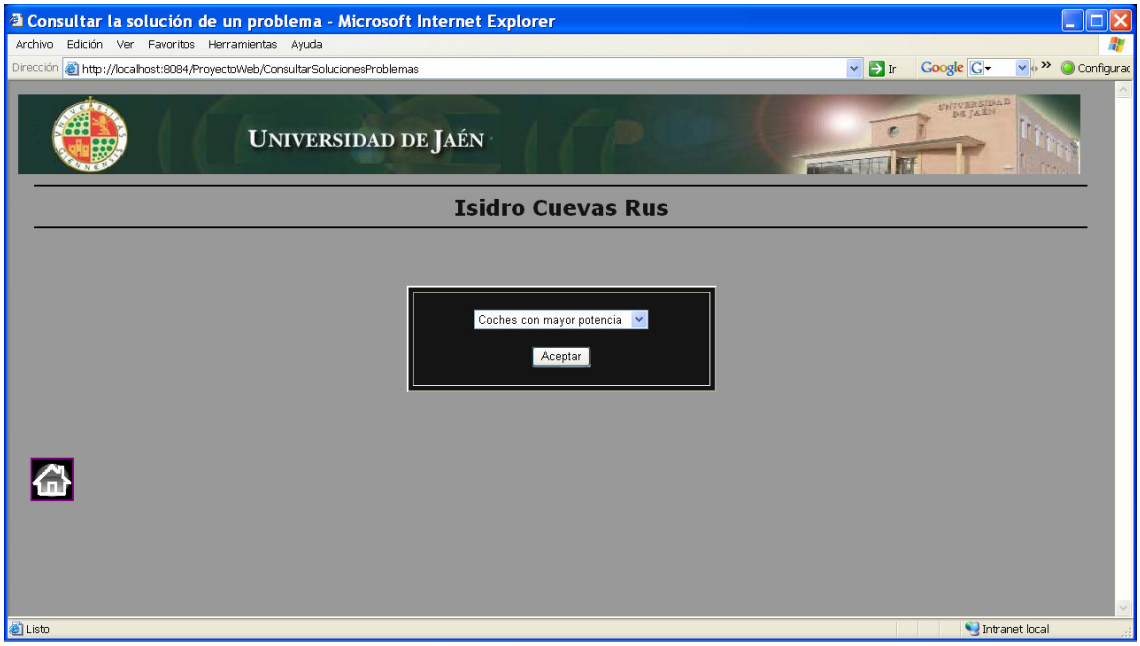


Figura III_10. Pantalla 'Consultar la solución de un problema'.

Si el Administrador elige un problema que aún no ha resuelto, nos aparece la notificación de la *Figura III_11*.

Figura III_11. Notificación en 'Consultar la solución de un problema'.

Si el problema que escoge ya lo había resuelto previamente, nos aparecerá una pantalla que contiene una tabla con los resultados. Un ejemplo de esto se muestra en la *Figura III_12.*

Figura III_12. Ejemplo de resultados en 'Consultar la solución de un problema'.

4. Resolver un problema.

Si el Administrador quiere resolver un problema tendrá que hacer clic en esta opción y aparecerá la siguiente pantalla:

Figura III_13. Pantalla de 'Calcular solución a un problema'.

Si el Administrador elige resolver un problema que aún no ha sido valorado por algún experto, nos aparece la notificación de la *Figura III_14*.

Figura III_14. Notificación de 'Calcular solución a un problema'.

Si el problema que escoge ya había sido valorado por los expertos previamente, entonces podemos pasar a resolverlo mediante alguno de los operadores que aparecen en la pantalla que se muestra en la *Figura III_15.*

Figura III_15. Selección del operador en 'Calcular solución a un problema'.

Si el Administrador escoge el operador '*Media aritmética*' y pulsa *Aceptar* le aparecerá una pantalla con la solución. Un ejemplo de esto lo podemos ver en la *Figura III_16*.

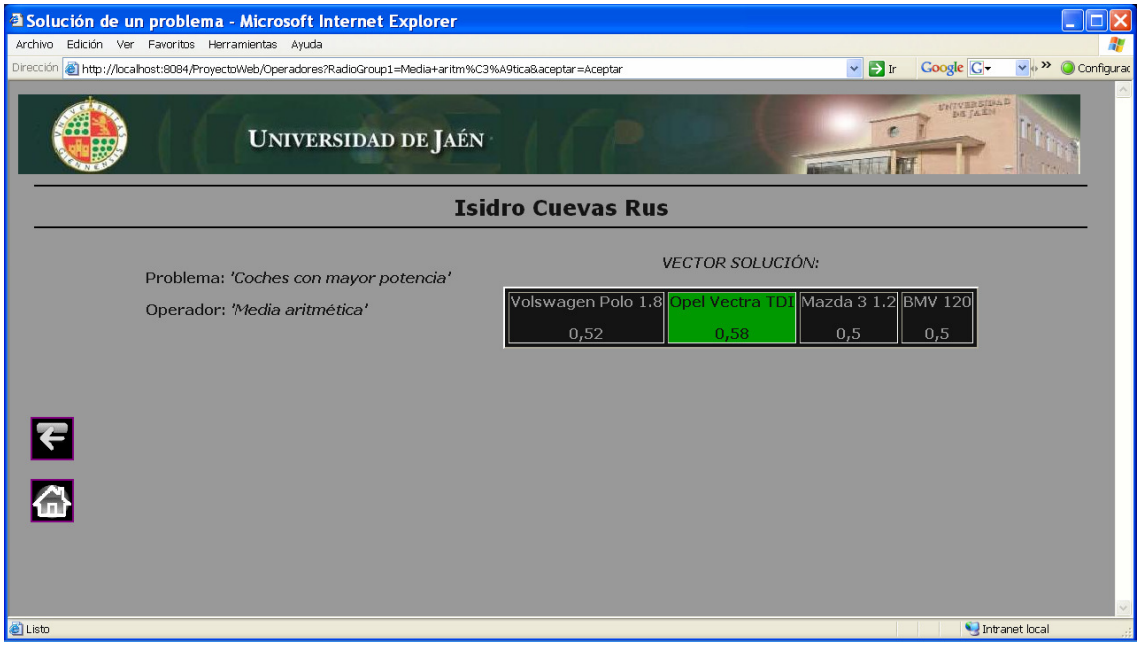


Figura III_16. Ejemplo de solución de un problema usando el operador 'Media aritmética'.

Si el Administrador escoge el operador '*Media ponderada*' y pulsa *Aceptar* le aparecerá una pantalla como la que muestra la *Figura III_17* para que escoja los pesos necesarios para poder utilizar este operador.

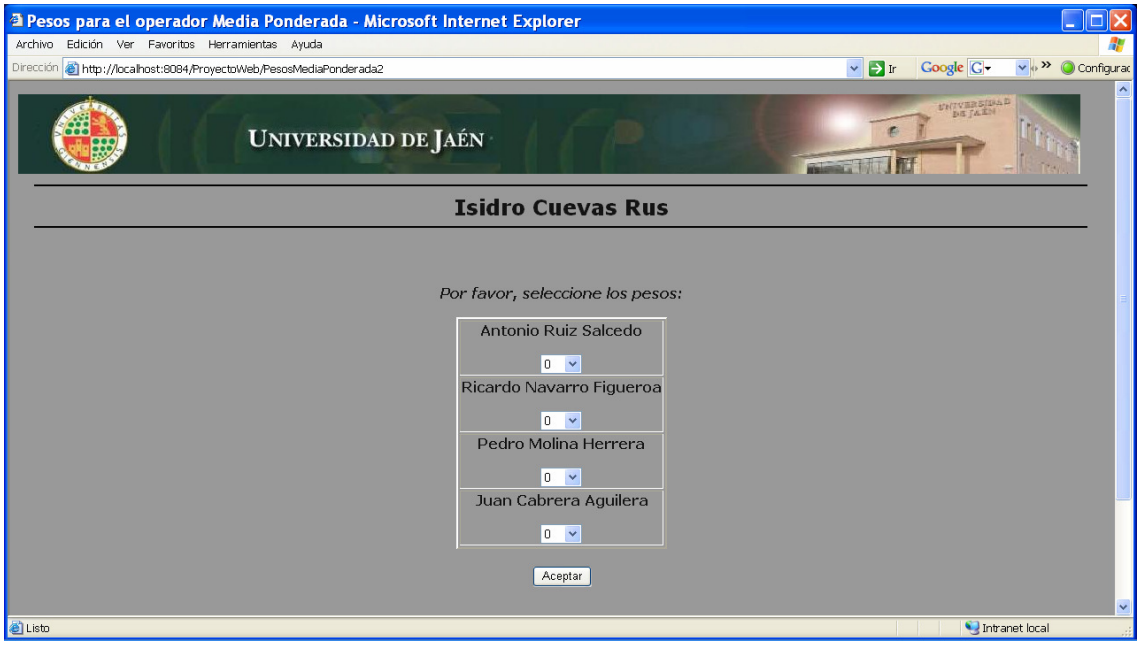


Figura III_17. Selección de pesos para el operador 'Media ponderada'.

Esto sucede cuando es la primera vez que se resuelve este problema, pero si el Administrador quiere volver a resolver dicho problema, le aparecerá la pantalla de la *Figura III_18* por si desea utilizar otros pesos. Si la respuesta es '*Si*', al Administrador le aparecerá la pantalla que muestra la figura *Figura III_17*, y cuando pulse *Aceptar,* o si responde '*No*' aparecerá una pantalla con la solución del problema y los pesos utilizados. Un ejemplo de esto lo podemos ver en la *Figura III_19*.

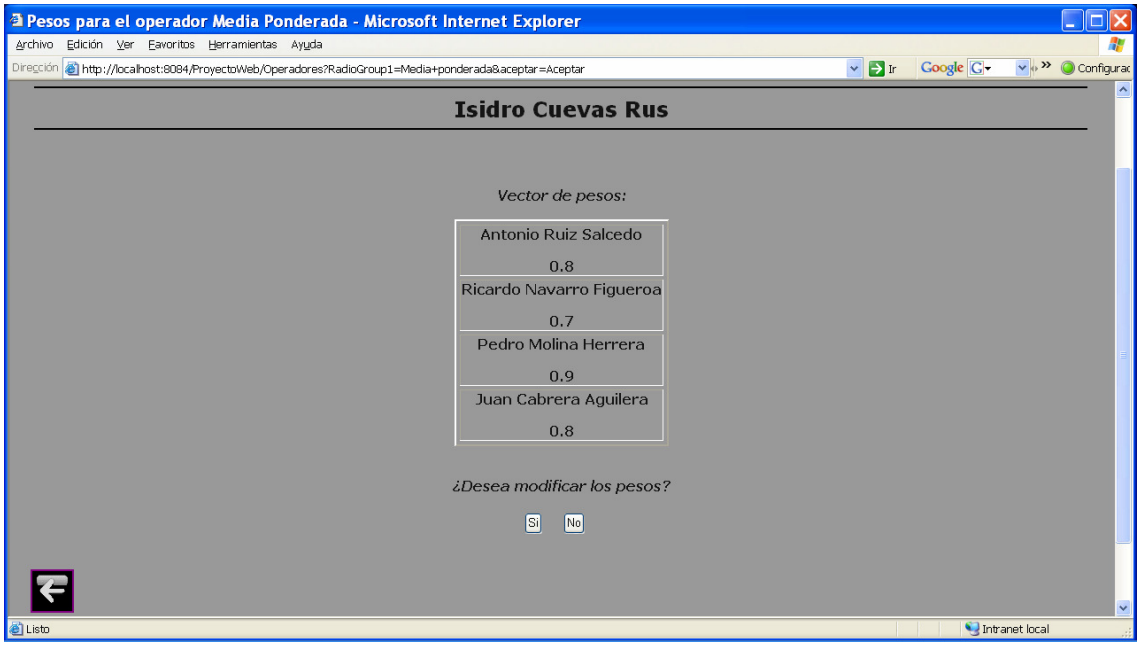


Figura III_18. Modificar pesos.

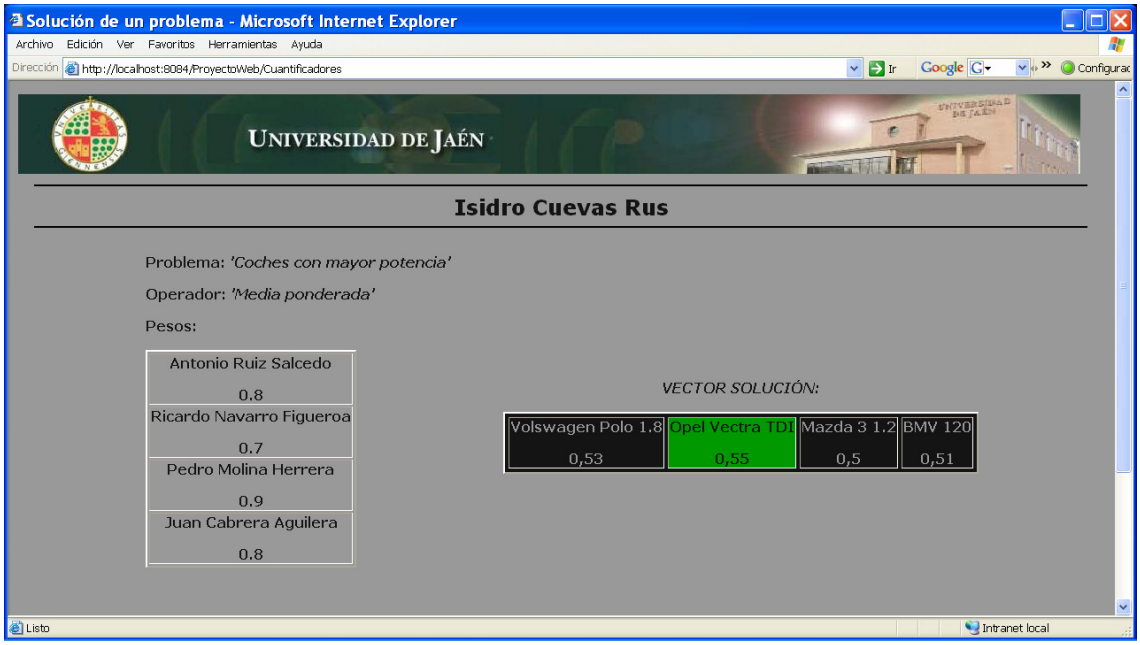


Figura III_19. Ejemplo de solución de un problema usando el operador 'Media ponderada'.

Si el Administrador escoge el operador '*OWA*' y pulsa *Aceptar* le aparecerá una pantalla como la que muestra la *Figura III_20* para que escoja el cuantificador lingüístico que desee.

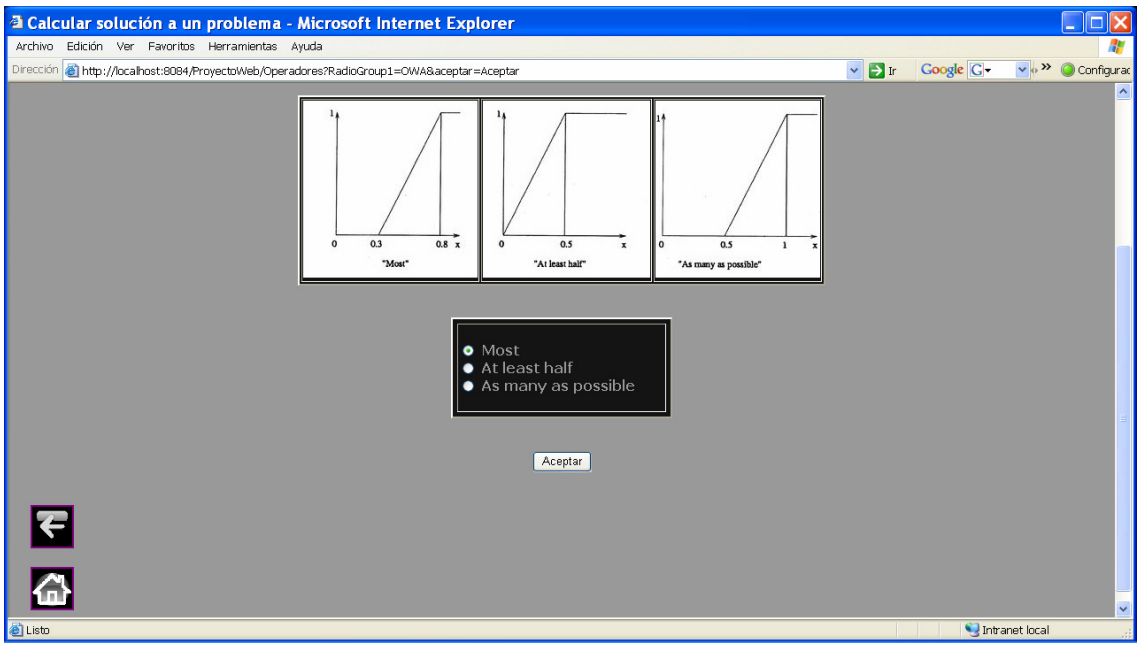


Figura III_20. Selección del cuantificador lingüístico para el operador 'OWA'.

Cuando se escoja el cuantificador lingüístico y se pulse *Aceptar* aparecerá una pantalla con la solución del problema y los pesos utilizados. Un ejemplo de esto lo podemos ver en la *Figura III_21* en el que se ha escogido el cuantificador lingüístico '*Most*'. (Para escoger los otros cuantificadores lingüísticos se procede de la misma manera).

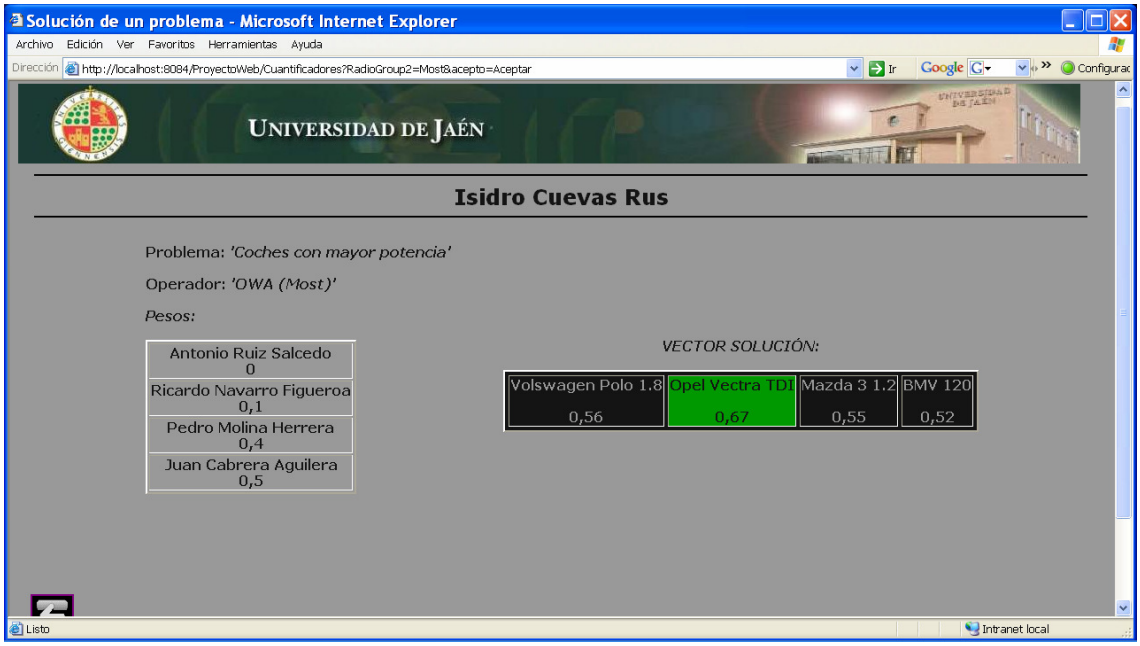


Figura III_21. Ejemplo de solución de un problema usando el operador 'OWA' y el cuantificador lingüístico 'Most'.

Si el Administrador escoge el operador '*Or Like S-OWA* y pulsa *Aceptar* le aparecerá una pantalla como la que muestra la *Figura III_22* para que escoja un valor necesario para resolver el problema utilizando este operador.

Figura III_22. Selección de un valor necesario para el operador 'Or Like S-OWA'.

Cuando se escoja dicho valor y se pulse *Aceptar* aparecerá una pantalla con la solución del problema y el valor que ha escogido para resolverlo. Un ejemplo de esto lo podemos ver en la *Figura III_23*.

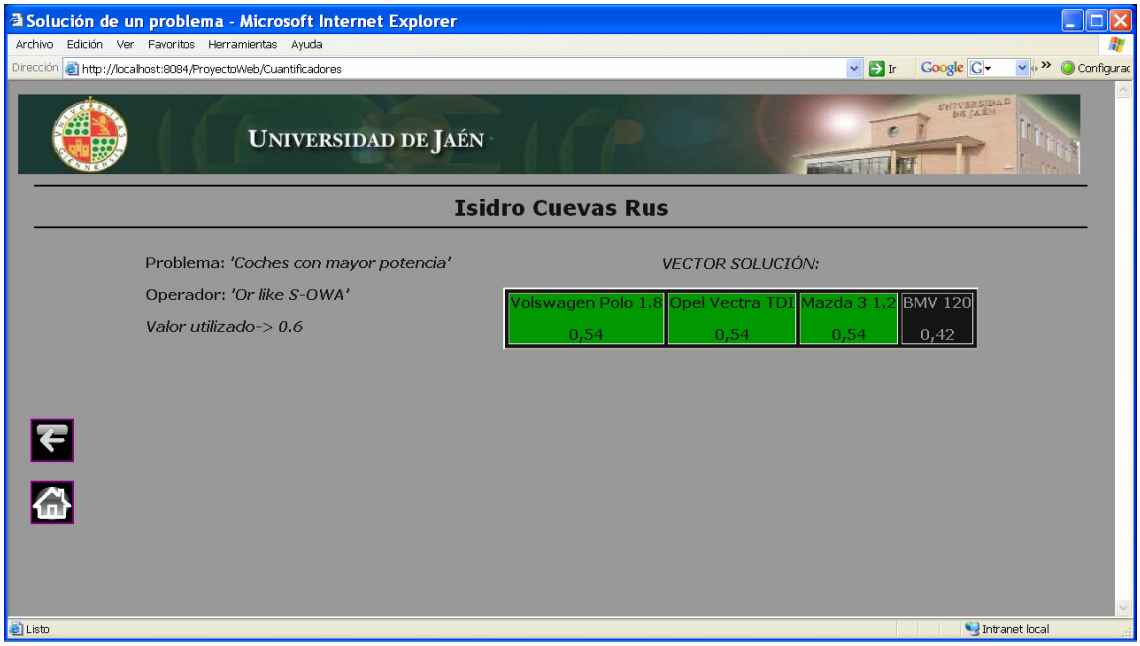


Figura III_23. Ejemplo de solución de un problema usando el operador 'Or Like S-OWA'.

Si el Administrador escoge el operador '*And Like S-OWA*' y pulsa *Aceptar* le aparecerá una pantalla como la que muestra la *Figura III_22* para que escoja un valor necesario para resolver el problema utilizando este operador.

Cuando se escoja dicho valor y se pulse *Aceptar* aparecerá una pantalla con la solución del problema y el valor utilizado. Un ejemplo de esto lo podemos ver en la *Figura III_24*.

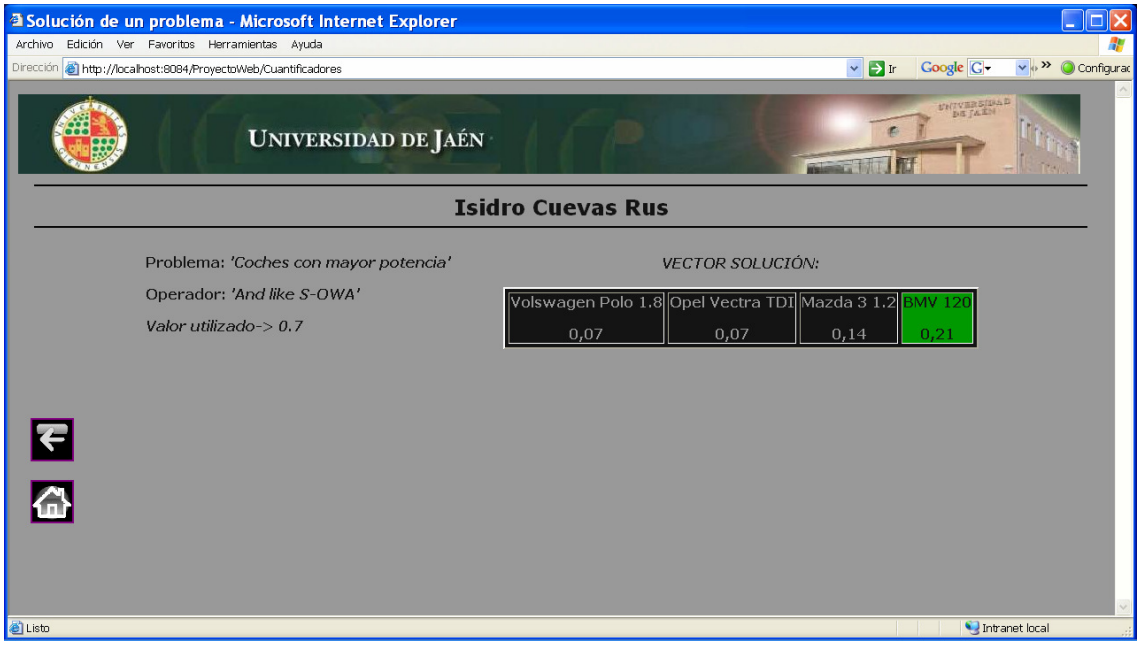


Figura III_24. Ejemplo de solución de un problema usando el operador 'And Like S-OWA'.

En la pantalla donde se muestra la solución del problema, sea cual sea el operador escogido, y en las pantallas donde el Administrador tiene que escoger los pesos o un valor determinado, aparece un icono ( ) llamado '*Seleccionar otro operador*', que nos llevaría a la pantalla que muestra la *Figura III_15*.

5. Dar de alta a un nuevo *Experto*.

Si el Administrador quiere dar de alta a un nuevo experto tendrá que hacer clic en esta opción y aparecerá la siguiente pantalla:

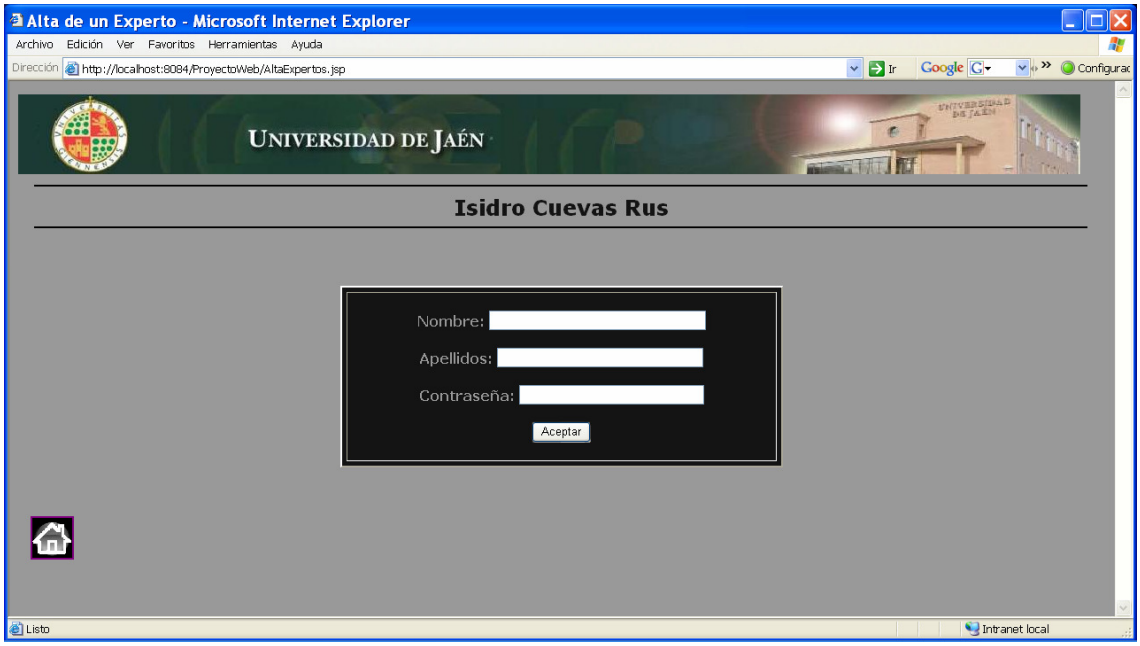


Figura III_25. Formulario para dar de alta a un nuevo experto.

Si al rellenar el formulario para el alta de un nuevo experto, el administrador se deja algún campo por rellenar nos aparecerá el mensaje de error de la *Figura III_26.*

Figura III_26. Mensaje de error en 'Alta de un nuevo Experto'.

Si el Administrador introduce una contraseña para el experto que ya está siendo usada por otro experto, nos aparece la notificación que se observa en la *Figura III_27.*

Figura III_27. Notificación para 'Alta de un nuevo Experto' (contraseña errónea).

Si no existe aún ninguna contraseña de experto como la que introduce el Administrador, nos aparece la notificación de la *Figura III_28.*

Figura III_28. Notificación para la 'Alta de un nuevo Experto'.

6. **Crear una nueva alternativa.**

   Si el Administrador quiere crear una nueva alternativa tendrá que hacer clic en esta opción y aparecerá la siguiente pantalla:

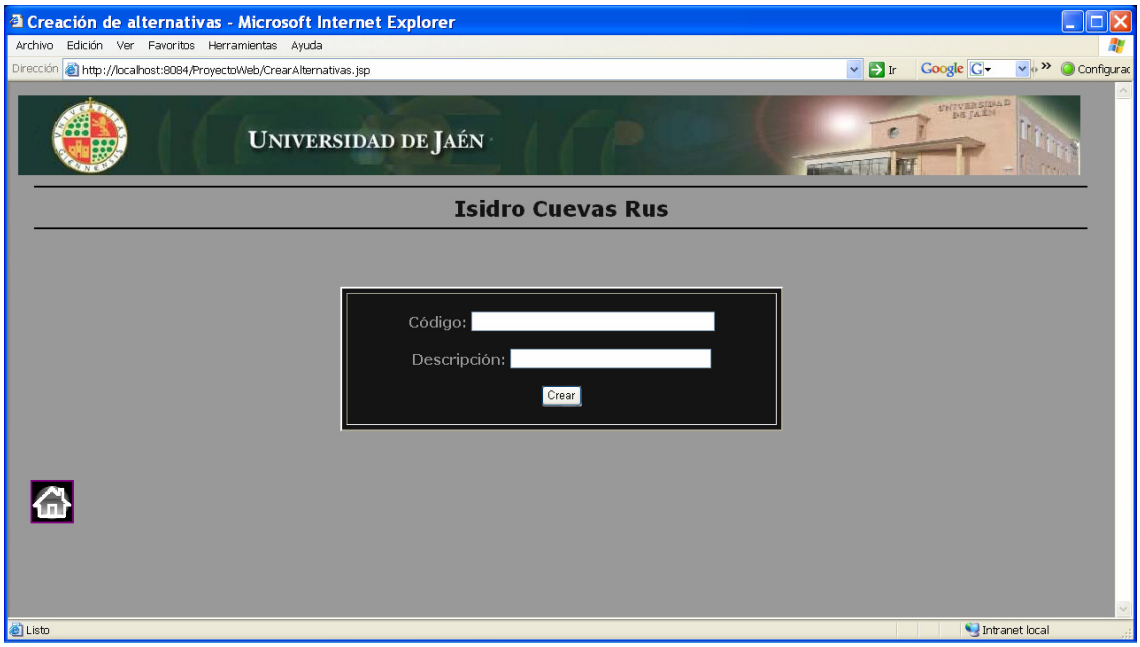


Figura III_29. Formulario para la creación de una nueva alternativa.

Si al rellenar el formulario para el alta de un nuevo experto, el administrador se deja algún campo por rellenar nos aparecerá el mensaje de error de la *Figura III_30.*

Figura III_30. Mensaje de error en Creación de una nueva alternativa'.

Si el Administrador introduce un código de alternativa que ya está siendo usado por otra alternativa, nos aparece la notificación que se observa en la *Figura III_31.*

Figura III_31. Notificación para 'Creación de una nueva alternativa' (código erróneo).

Si no existe aún ningún código de alternativa como el que introduce el Administrador, nos aparece la notificación de la *Figura III_32.*

Figura III_32. Notificación para 'Creación de una nueva alternativa'.

En todas pantallas que se han mostrado en las figuras anteriores aparece un icono () llamado '*Volver al menú principal'* en el que si hacemos clic nos lleva al menú principal del Administrador (*Figura III_3*).

7. Salir de la aplicación.

Si el Administrador desea salir de la aplicación, pulsaría esta opción y aparecería la pantalla de la *Figura III_1*.

ANEXO IV.

MANUAL DE USUARIO:
EXPERTO

Este manual de usuario está organizado como una visita guiada por la aplicación, cuando entramos como experto.

El primer paso para utilizar la aplicación es abrir el Navegador (Internet Explorer) y teclear la dirección en la que tenemos alojada esta, que en nuestro caso será: http://localhost:8084/ProyectoWeb/index.jsp.

Al realizar lo anteriormente comentado, nos encontramos con la página de inicio, tal y como se ve en la *Figura IV_1*.

Figura IV_1. Entrada al sistema.

Para entrar como experto debemos introducir el nombre de usuario y contraseña de un experto. Si alguno de estos datos (o los dos) no es correcto, el sistema nos muestra un mensaje de error como se observa en la *Figura IV_2*.

Figura IV_2. Mensaje de error en la Identificación.

En el caso de una identificación correcta nos aparece el menú principal del experto, como podemos ver en la *Figura IV_3*.

Figura IV_3. Menú principal del Administrador.

En este menú principal podemos encontrar las operaciones disponibles para el Experto:

1. Valorar problema.
2. Consultar las valoraciones de un problema.
3. Salir de la aplicación.

A continuación veremos de forma detallada cada una de estas partes de la aplicación.

1. **Valorar problema.**

   Si hacemos clic en la primera de las opciones del Experto nos aparece una pantalla como la que se muestra en la *Figura IV_4* para valorar problema.

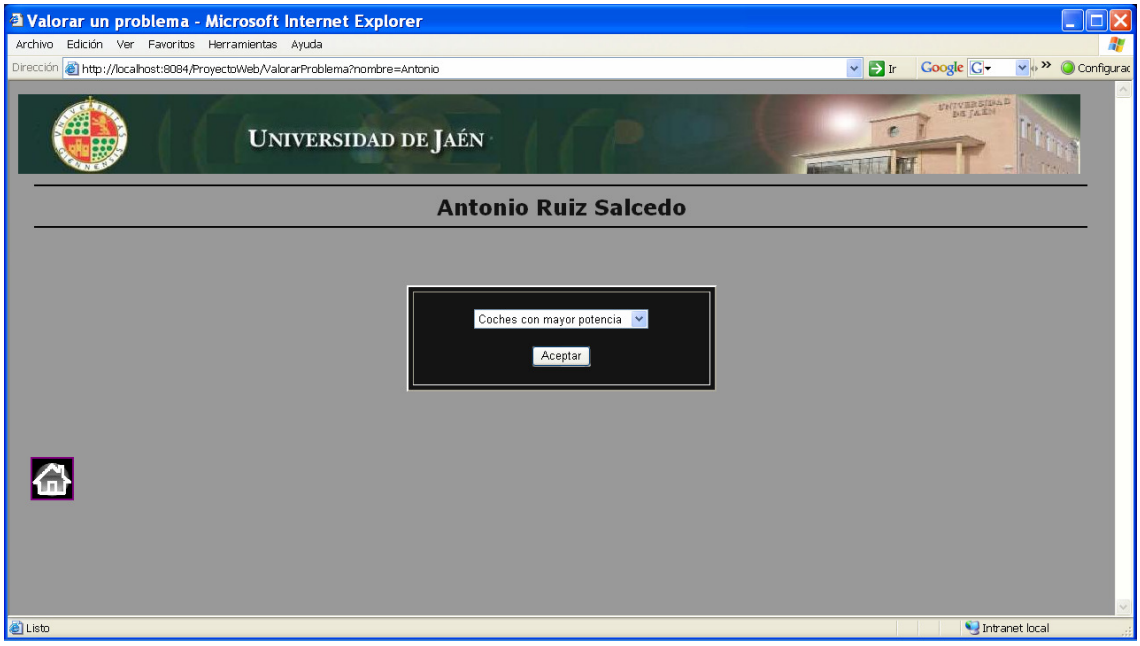


Figura IV_4. Selección del problema en 'Valoración de un Problema'.

Si el experto elige un problema que ya había valorado, aparecerá una pantalla como la de la *Figura IV_5.*

Figura IV_5. Notificación de 'Valoración de un Problema' (problema ya valorado).

Si pulsa el botón *'No'* se volverá a la pantalla de la *Figura IV_4.* Si pulsa el botón *'Si'* o elige, en cambio, un problema que aún no ha valorado, aparece la siguiente pantalla, con la/s alternativa/s que tenga dicho problema para que sean valoradas:

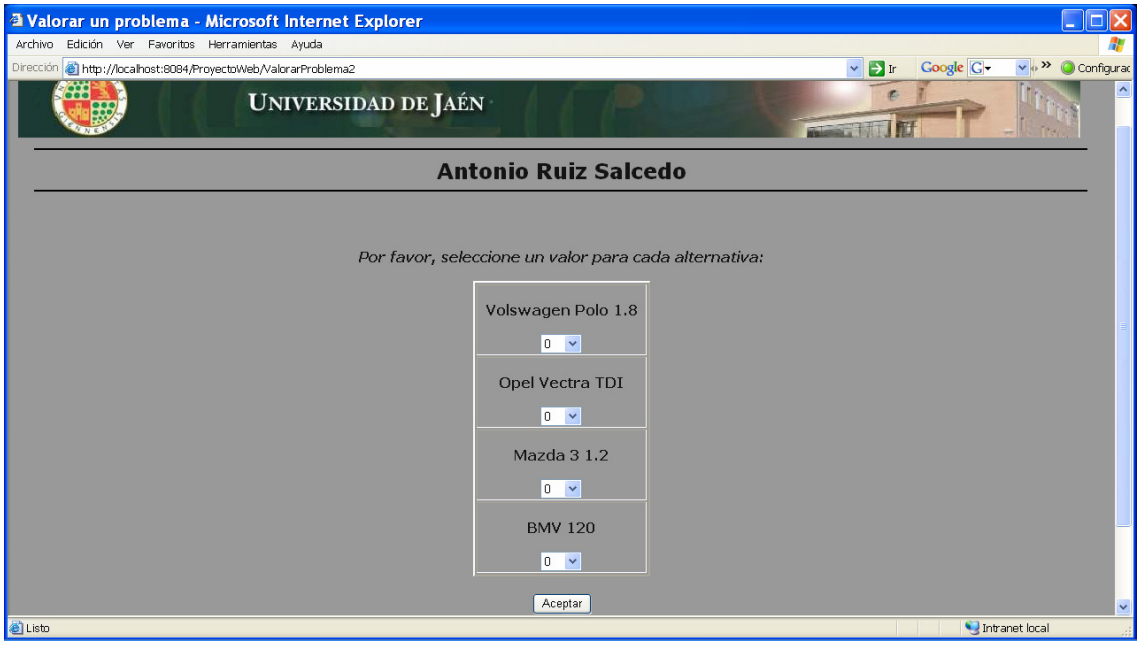


Figura IV_6. Valoración de alternativas en 'Valoración de un Problema'.

El experto tendrá que valorar la/s alternativa/s y pulsar el botón *Aceptar* y aparecerá la notificación que muestra la *Figura IV_7.*

Figura IV_7. Notificación en 'Valoración de un Problema'.

2. **Consultar las valoraciones de un problema.**

   Si el experto desea consultar las valoraciones que hizo previamente de un problema, podrá hacerlo haciendo clic en esta opción, y le aparecerá la siguiente pantalla para que seleccione el problema que desea valorar:

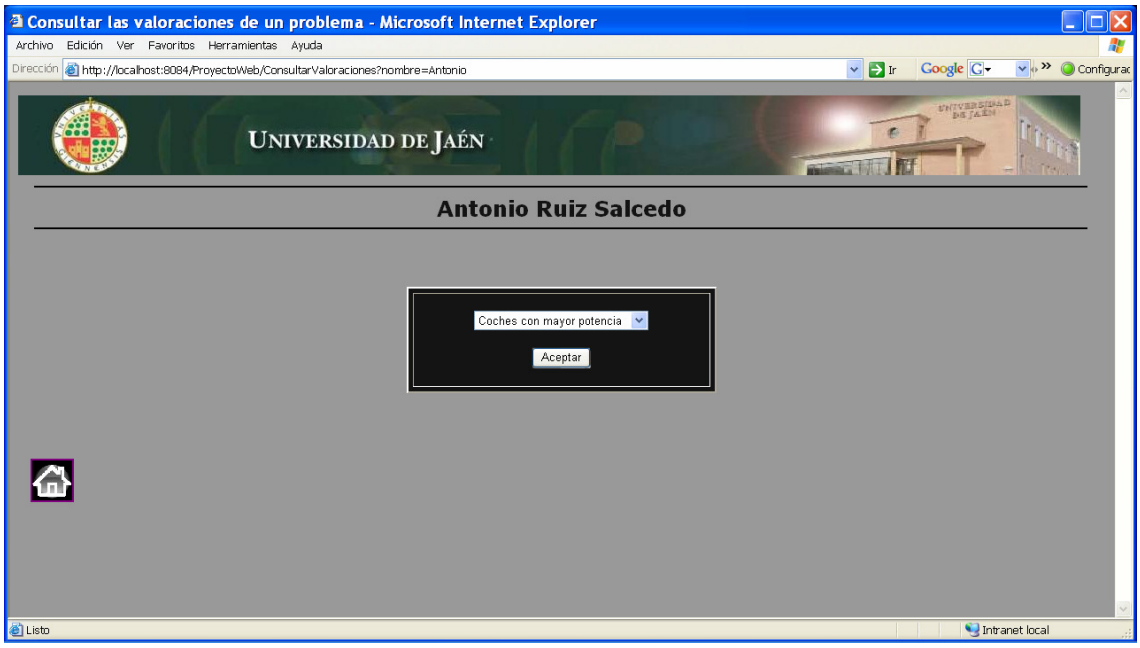


Figura IV_8. Selección del problema en 'Consultar la valoración de un Problema'.

Si el experto elige consultar un problema que no ha valorado aún, le aparecerá una notificación como la de la *Figura IV_9.*

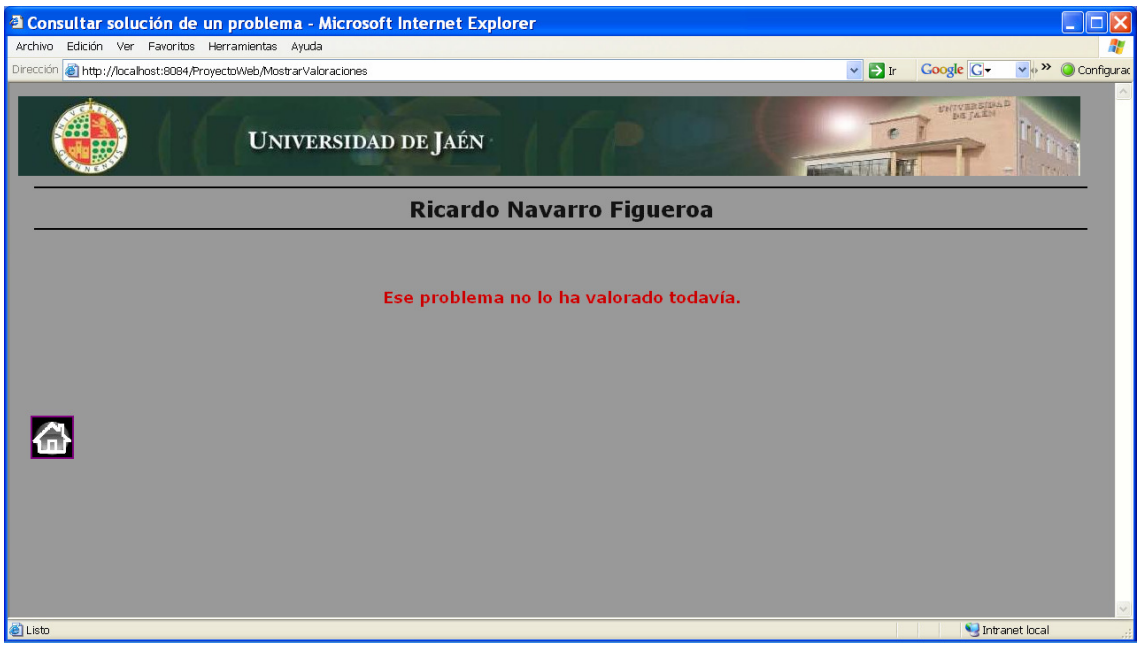


Figura IV_9. Notificación de 'Consultar la valoración de un Problema'.

Si elige, en cambio, un problema que ya había valorado previamente, aparece, como muestra la *Figura IV_10,* una tabla con la/s alternativa/s que tenga dicho problema y su valoración correspondiente, y pregunta si quiere modificar sus valoraciones. Si la respuesta es '*Si'* volveríamos a la pantalla de la *Figura IV_6,* y si la respuesta es '*No'* volveríamos a la pantalla de la *Figura IV_8*, para seleccionar otro problema que deseemos consultar.

Figura IV_10. Alternativas valoradas en 'Consultar la valoración de un Problema'.

En todas pantallas que se han mostrado en las figuras anteriores aparece un icono ( ) llamado '*Volver al menú principal'* en el que si hacemos clic nos lleva al menú principal del Administrador (*FiguraIV_3*).

3. **Salir de la aplicación.**

   Si el Experto desea salir de la aplicación, pulsaría esta opción y aparecería la pantalla de la *Figura IV_1*.