Análisis y Diseño de un motor de reglas IF – THEN para dispositivos móviles

MEMORIA	
Trabajo Final de Carrera	
Titulación	Ingeniería Técnica en Informática de Gestión
Semestre	Marzo – Junio 2006
Área	Ingeniería del Software
Autor	José Luís Salmerón Silvera
Consultor	Juan José Cuadrado Gallego

CONTROL DE MODIFICACIONES

EDICIÓN	FECHA	ENTREGA	COMENTARIOS
00	13/03/2006	PEC1 Plan de Trabajo	Diseño del formato del documento
01			Descripción del TFC
02			Objetivos generales y específicos
03			Planificación con hitos y temporización
04	23/03/2006	PEC2 Captura de requisitos y análisis	Investigación motores de reglas
05	26/03/2006		Identificación de los subsistemas
06	01/04/2006		Especificación de los requisitos
07	03/04/2006		Identificación de actores
08	07/04/2006		Diagramas de casos de uso
09	11/04/2006		Especificación de casos de uso
10	16/04/2006		Documento de visión
11	18/04/2006		Documento de análisis
12	26/04/2006	PEC3 Diseño del motor	Análisis de la arquitectura
13	03/05/2006		Diseño de la arquitectura
14	08/05/2006		Diseño de los casos de uso
15	12/05/2006		Diseño de la persistencia
16	15/05/2006		Documento de arquitectura
17	20/05/2006		Documento de diseño técnico
18	25/05/2006		Diseño de la interfaz gráfica de usuario

A mi esposa y mi hijo

INDICE

Índice de Tablas

Índice de Figuras

1. Introducción

Existen muchas clases diferentes de proyectos informáticos, ya que el campo que engloban es inmenso. Sin embargo, hoy en día, está cada vez más ampliamente reconocido en las instituciones académicas que los proyectos informáticos son algo más que desarrollar un programa de software. Desde este punto de vista, el proyecto debe incluir algún elemento de investigación, esto es, debe estar debidamente razonado y alcanzar de forma sistemática unos resultados. El mero desarrollo de una herramienta o un algoritmo sin hacer ninguna evaluación técnica, ni situarla debidamente en un contexto, puede ser aceptable en la industria donde las soluciones comerciales son necesarias y a veces prioritarias; sin embargo, en el mundo académico, todo proyecto debe contener, en mayor o menor medida, algún elemento de investigación.

La realización de un Proyecto Final de Carrera o Trabajo Fin de carrera (en adelante TFC) nos da la oportunidad de hacer nuestra propia contribución. Coincidiremos que no tiene sentido que el estudiante se limite a imitar simplemente el trabajo de otros. Los pensamientos, ideas y desarrollos propios son importantes para el propio proceso de madurez del estudiante como ingeniero en informática. Durante el proceso citado se desarrollan una serie de habilidades [DAW02] que procedemos a enumerar de forma somera:

- **Independencia**. Uno de los objetivos del TFC estriba en el desarrollo en el estudiante de una capacidad de trabajo de forma independiente. Frecuentemente, las instituciones se refieren a los TFC como proyectos independientes. La capacidad de trabajar por uno mismo sin una supervisión detallada por parte de otra persona constituye una habilidad valiosa. Asimismo, las empresas buscan independencia en los ingenieros.

- **Pensamiento critico**. El TFC debe enseñar al estudiante a pensar de una forma más crítica y más profunda. El desarrollo de ideas y de una forma de pensar independiente representa un grado de madurez que no se adquiere por el mero hecho de asistir a clase. Esta afirmación adquiere más relevancia si consideramos el caso especial de la Universidad Oberta de Cataluña (en adelante UOC), ya que su propia naturaleza como universidad virtual, donde no existe docencia presencial, fomenta esta circunstancia. De nuevo, la habilidad de dar ideas originales, demostrar una comprensión profunda y de utilizar una imaginación viva, son habilidades preciosas para un investigador.

- **Aprendizaje**. El desarrollo del TFC debe enseñar al estudiante a aprender. Como parte del TFC se han aprendido nuevas habilidades y nuevas formas de pensar. Este aprendizaje no se desprende de clases o

tutorías sino del estudio e investigación independientes. Una vez desarrollada dicha habilidad será de gran utilidad en el futuro, ya que dotará al estudiante de la capacidad de adaptación a nuevas tecnologías y retos profesionales.

- **Habilidades técnicas**. También es posible la obtención o profundización de habilidades técnicas en el transcurso del proyecto.

- **Habilidades de comunicación**. Tanto las habilidades de expresión verbal como la escrita constituye una parte de vital importancia para cualquier proyecto final de carrera. Puede ser muy útil mejorar nuestras habilidades en este campo tanto si se va a la empresa como si nos quedamos en la universidad. Las habilidades de expresión oral son muy útiles en la empresa donde se tiene relación con todo tipo de gente: directores, clientes, resto de personal de la empresa, consultores externos, etc. También tendremos que hacer presentaciones y demostraciones de software en el futuro así como elaborar informes claros y concisos. A nivel de tesis doctoral, estas habilidades también son cruciales. Tendremos que asistir a conferencias o dar seminarios. Tendremos que dar clases y tutorías o sesiones de laboratorio. Tendremos que escribir informes, artículos, documentos de todo tipo y, finalmente, una tesis doctoral.

El subepígrafe siguiente describe el TFC elegido.

1.1. Descripción del TFC

El TFC consiste en el análisis y diseño de un sistema que permita administrar las reglas de una organización. Las reglas de negocio son importantes para las empresas, ya que describen como éstas realizan sus actividades. En este TFC se propone un sistema que permitirá gestionar dichas reglas. Para ello, se identificará el modelo de empresa y los diferentes submodelos donde se encuadran las reglas de negocio. Posteriormente, se determinarán los requerimientos y se confeccionarán los modelos necesarios para que el sistema pueda llevarse a cabo.

Por otro lado, el mantenimiento de un sistema como el que propone el presente TFC permitirá el desarrollo de sistemas informáticos acordes con la organización en la que se desarrolla. Ello se debe a que gran parte del conocimiento que deberán considerar los analistas ya se encontraría estructurado. Esta formalización permitirá que dichas reglas resulten más accesibles que si éstas se encuentran en el código de algún lenguaje de programación. De esta forma se favorece la reusabilidad del análisis de un dominio concreto de la empresa. Asimismo, estas reglas podrían servir para

generar avisos en otras aplicaciones de gestión cuando se realicen operaciones que las incumplan.

Otro elemento destacable estriba en la arquitectura para la cual se ha diseñado. Nos referimos a los dispositivos móviles. El crecimiento exponencial de usuarios de estos dispositivos demanda, cada vez en mayor número, soluciones incorporadas que permitan optimizar su uso.

1.2. Objetivos generales y específicos

El presente TFC se realiza con el fin de alcanzar el grado de Ingeniero Técnico en Informática de Gestión en la Universidad Oberta de Cataluña. Siendo su motivación a nivel personal.

Respecto al ámbito del proyecto partimos de dos objetivos generales. El primero de ellos consiste en afianzar los conocimientos adquiridos durante la carrera. Tal y como hemos argumentado anteriormente, el TFC precisa que el estudiante acuda a los conocimientos que se han adquirido en otras asignaturas y los aplique. De esta forma, se encuentran nuevos retos y se profundiza en dichas habilidades. El segundo objetivo general estriba en el análisis y diseño de un sistema que esa capaz de gestionar las reglas de negocio de cualquier entorno. Las organizaciones operan con una serie de reglas. En muchas ocasiones, éstas existen únicamente de forma tácita. Consideramos que un sistema que formalice ese conocimiento resulta valioso, además permitirá mantenerlas adaptándose a los cambios en la organización y su entorno.

Como objetivos específicos podemos destacar los reseñados a continuación.

- El análisis del enfoque denominado reglas de negocio (business rules) para el soporte a las decisiones en la empresa.
- La identificación de los requisitos funcionales y no funcionales de un sistema de este tipo.
- La identificación de los actores, así como de los diferentes casos de uso de estos sistemas.
- Un análisis de la arquitectura del sistema para dispositivos móviles.
- El diseño y especificación de los casos de uso.
- El diseño de la persistencia.
- El diseño de las interfaces gráficas de usuario.

2. Reglas de negocio

2.1. Importancia del análisis de reglas

Cuando los seres humanos aplicamos reglas a hechos y utilizamos el conocimiento para la toma de decisiones estamos empleando una combinación de emociones y consideraciones lógicas. Por ello, todos no reaccionamos de la misma forma cuando afrontamos situaciones idénticas.

En el mundo empresarial también se adoptan decisiones basadas en hechos [BRC06]. No obstante, las consideraciones emocionales no deben dirigir la forma de actuación de las empresas, por ello se refuerza el otro componente citado con anterioridad, la lógica de negocio. En este escenario las reglas de negocio adquieren una mayor importancia al erigirse como principal soporte a las decisiones.

Para que las decisiones empresariales sean de calidad, resulta preciso que también las reglas que se utilicen lo sean. Para que un decisor adopte decisiones consistentes ha de confiar en reglas consistentes y de calidad. Dichas reglas son conocidas como reglas de negocio, las cuales suministradas en el momento oportuno permiten que se adopten decisiones fundamentadas y con menor nivel de incertidumbre que en su ausencia. En este sentido, podemos afirmar que las reglas de negocio implementadas en Sistemas de Información sirven de guía al comportamiento organizacional. De hecho, constituyen una formalización del comportamiento deseado de la organización. Asimismo, podríamos interpretar las reglas de negocio como una formalización de las mejores prácticas de una organización.

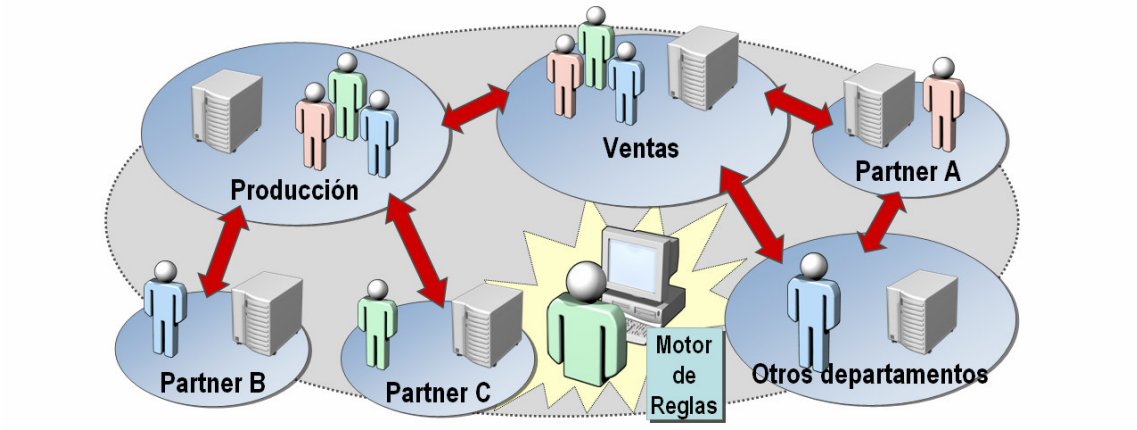

Figura 1. Complejidad empresarial

De cualquier forma, las reglas de negocio siempre han estado presentes [BRF06], aunque no formalizadas como propugnan los motores de reglas. No obstante, considerando la complejidad empresarial actual (Figura 1) resulta

necesario un soporte para afrontar las relaciones tanto intra-empresa como con el entorno de ésta.

De igual forma, la formalización de reglas también ostenta gran relevancia en los entornos técnicos. Hemos de considerar que las aplicaciones no se encuentran aisladas. Existen un gran número de ellas, así como de componentes, entrelazadas para dar soporte a las relaciones dentro de la propia empresa.

El código fuente de los Sistemas de Información recoge un gran número de reglas de negocio, aunque de forma que no permite acceder a ellas directamente. Por ello, el enfoque de reglas de negocio apuesta por su extracción facilitando su gestión y uso por parte de usuarios no técnicos.

Seguidamente profundizaremos en los beneficios que genera el uso de reglas, así como los inconvenientes surgen.

2.1.1. Beneficios del uso de reglas

Un sistema basado en el enfoque de reglas tendrá los siguientes beneficios [VHA02] derivados de su empleo.

- Reuso del conocimiento. Todo el conocimiento subyacente en las reglas se encuentra disponible para su posterior utilización en otras decisiones o proyectos.

- Su desarrollo será más rápido que los sistemas no basados en reglas. Ello se debe a que se reusa el conocimiento formalizado en las reglas. De esta forma, no se ha de identificar todos los requerimientos, sino que una parte de ellos ya se hayan formalizados en dichas reglas.

- Se basa en modelos más estables. El uso de conocimiento ya formalizado, probado y depurado permite desarrollar sistemas con mayor seguridad que si no se dispone de ellas.

- Soporte integral de la organización. Las reglas modelan conocimiento de toda la organización, por ello los sistemas parten de una organización integrada.

- Los cambios en las reglas son simples. Partiendo de unas reglas formalizadas en un motor de reglas, su alteración resulta muy sencilla.

- Simplicidad. Las reglas son comprendidas tanto por técnicos como por no técnicos debido a que resulta un concepto muy intuitivo. Se necesitan muy pocos conceptos técnicos para comprenderlas en su integridad.

- Independencia tecnológica. Las reglas se expresan en un lenguaje y van dirigidas a un fin que las convierten en independientes de la tecnología subyacente. Por ello, permite a los que tienen que interpretarlas centrarse más en el negocio que es su cometido.

- Permite la trazabilidad de objetivos de negocio a requerimientos y a reglas formalizadas. De esta forma, se puede controlar que reglas soportan cada objetivo de negocio. En caso de que existan reglas no relacionadas con objetivos de negocio serían superfluas y podrían ser suprimidas.

No obstante, en todos los casos no está recomendado el uso de motores de reglas. A continuación se detallan ambos escenarios.

2.1.2. Evaluación del uso de reglas

Objetivamente el uso de reglas resulta positivo. No obstante, existen escenarios donde su aportación no compensa el esfuerzo que requiere su empleo, ya que su contribución no resulta significativa. Se debe utilizar en los casos que se detallan seguidamente [VHA02].

- La aplicación en desarrollo tiene un componente decisional destacado. Sobre todo si estas aplicaciones consisten en Sistemas de Soporte a la Decisión (DSS) o Sistemas de Información para Directivos (EIS). Ambos se encuentran dirigidos a los integrantes del nivel táctico y estratégico de una organización cuya misión estriba en la toma de decisiones de diferente nivel. Las reglas facilitan la adopción de decisiones, reduciendo la incertidumbre inherente a ellas.

- Las reglas son complejas. En estos casos, los motores de reglas se erigen en la única forma de gestionarlas y acceder a ellas. Es más se trata de reglas no obvias que se extraerán de un sistema, de forma que no podrían derivarse de otra forma.

- Las reglas cambia frecuentemente. En estos casos, su mantenimiento precisa de un motor para adaptarse a dichas reglas cambiantes.

- Las reglas se refieren a diferentes departamentos, organizaciones y aplicaciones. Un motor de reglas resulta imprescindible para mantener la coherencia debido a la dispersión de fuentes.

- Las reglas están referidas a un sector o ámbito de actuación muy cambiante o con una regulación muy extensa. Dichos escenarios producen unas reglas poco estables o muy profusas. En ambos casos se precisa del soporte de un motor.

Por el contrario, existen escenarios donde los motores de reglas no aportan un valor significativo. Podemos sintetizarlo en los siguientes casos.

- Las reglas son estáticas. Se mantienen en el tiempo, por ello resulta más simple acceder a ellas de formas no formalizadas.

- Las reglas son simples, siendo conocidas por todos. No compensa la existencia de un motor para tratar reglas simples en exceso, ya que será un sistema superfluo.

- No existe un compromiso organizativo con el uso de las reglas. Este escenario es común a cualquier sistema. Si la propia organización no apuesta por su uso, quedará relegado y caerá en desuso inexorablemente.

En el siguiente apartado abordaremos la conceptualización de regla de negocio, su evolución en la arquitectura de los sistemas y sus características.

2.2. Conceptualización de reglas de negocio

No existe una definición estándar para el concepto de regla de negocio. Vamos a recoger la definición propuesta por el Business Rule Group [BRG00] que se expresa en los siguientes términos.

> *Una regla de negocio es una sentencia que define o restringe algunos aspectos del negocio. Se fundamenta en la estructura del negocio o en el control o influencia sobre el comportamiento del negocio.*

Con el ánimo de establecer el alcance de las reglas de negocio vamos a recoger [VHA02] el dominio de las reglas de negocio. En este sentido, identificamos tres componentes, términos, hechos y reglas.

- Términos. Es un sustantivo o frase sustantivada. Podemos citar como ejemplos un concepto como cliente, una propiedad de un concepto como codigoValoracionCliente, un valor como femenino o un conjunto de valores como DiasDeLaSemana. Estos términos se transformarán en entidades, atributos, constantes o dominios. Siguiendo el ejemplo anterior, cliente sería una entidad, codigoValoracionCliente probablemente un atributo de la entidad cliente, femenino podría ser una constante empleada en una regla, mientras que el cuarto ejemplo podría llegar a ser un dominio.

- Hechos. Se trata de una sentencia sobre el negocio que conecta términos mediante preposiciones y verbos. Un ejemplo podría ser <el

cliente puede efectuar un pedido>. Estos hechos serán las relaciones entre entidades o la asociación de un atributo a una entidad. En el ejemplo planteado se recoge la relación entre las entidades cliente y pedido. Tanto los hechos como los términos constituyen la semántica que recogen las reglas.

- Reglas. Constituyen sentencias declarativas que aplica la lógica de negocio a valores recogidos en datos. Una regla sugiere una línea de acción respecto a los términos y hechos que recoge. Un ejemplo sería <los clientes nuevos no deben hacer pedidos a cuenta por valor mayor a 1.000 €>. De esta forma, en una regla una combinación de términos y hechos establecen una línea de acción para la situación que se plantea.

Las reglas de negocio se encuentran integradas en la arquitectura informática de los Sistemas de Información, aunque ésta ha variado a lo largo del tiempo. En el siguiente apartado revisaremos su evolución.

2.2.1. Evolución de la reglas de negocio en la arquitectura informática

El enfoque de reglas de negocio ha seguido una evolución a lo largo del tiempo produciendo un impacto incremental sobre los sistemas informáticos y, por ende, en la organización.

La arquitectura de tres capas clásica se dispone como se presenta en la Figura 2. En primer lugar tenemos los datos, donde se recogen todas las transacciones procedentes de las aplicaciones de procesamiento transaccional. Sobre esa capa se sitúa la capa de lógica de aplicación compuesta por las reglas de negocio y los procesos.

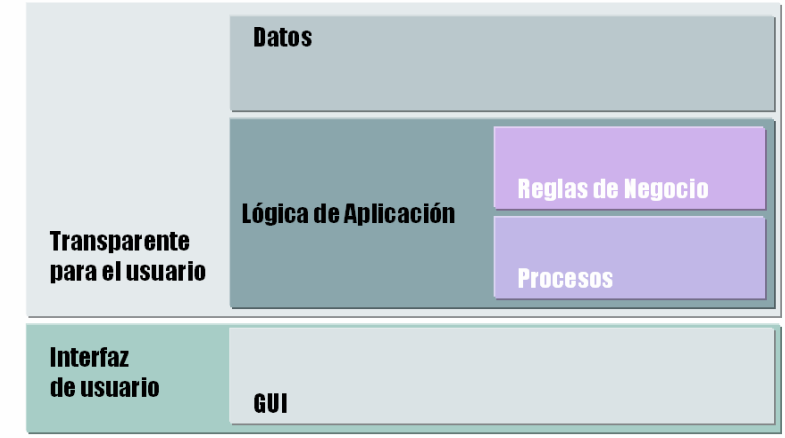

Figura 2. Reglas de negocio en la arquitectura de tres capas

En esta capa se modela el funcionamiento de la propia organización en la que está implantado el Sistema de Información. Por último, tenemos la capa del interfaz de usuario, a través del cual éste accede a las demás capas aunque si interactuar con ellas de forma directa.

Para analizar la evolución que han seguido las reglas de negocio en la arquitectura interna de los Sistemas de Información revisaremos sus componentes a alto nivel.

Originariamente, los datos, las reglas de negocio y los procesos se encontraban en un solo bloque (Figura 3).

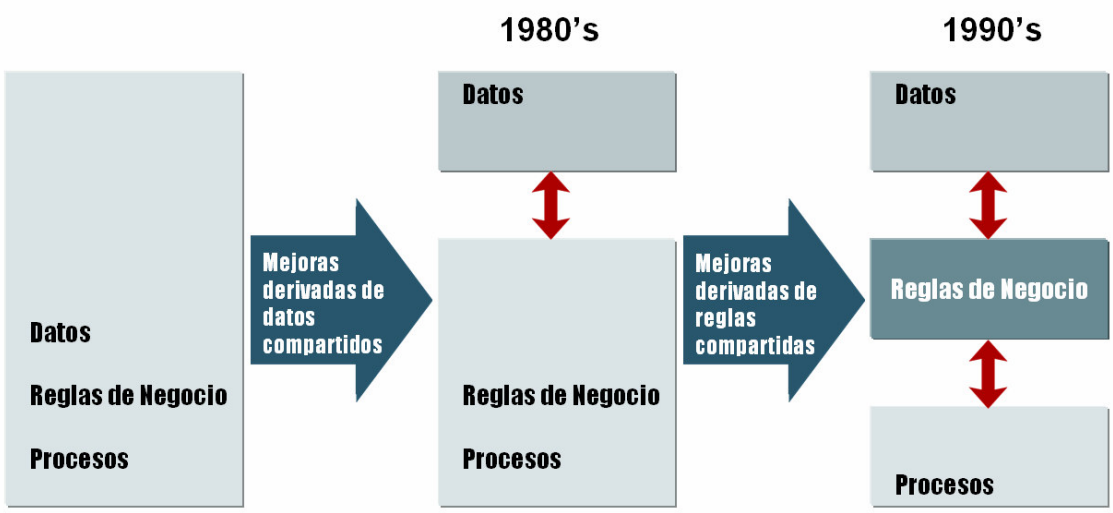

Figura 3. Hacia las reglas de negocio compartidas

Esta arquitectura no permitía una gestión adecuada de los tres componentes citados. En particular, los datos suponían el problema más acuciante debido al incremento desmesurado que estaban experimentando. Para afrontar esta situación se separaron los datos de los otros dos bloques. Ello permitió una mejor administración de los datos. No obstante, aun era preciso separar las reglas de negocio de los procesos para facilitar su acceso y mantenimiento. Finalmente en la década de los noventa se inició este camino.

La arquitectura final permite que las reglas de negocio puedan compartirse con otras aplicaciones y sistemas. De esta forma se favorece el reuso de éstas y se asegura la coherencia de la lógica de negocio en toda la organización.

No obstante, el escenario planteado hace referencia los Sistemas de Información empresariales. En contraposición a éstos, existían motores de inferencia e inducción de reglas de forma diferenciada desde cierto tiempo anterior (Figura 4). Ello se inició con los sistemas expertos en los albores de la década de los setenta, seguidamente se incorporaron los primeros Sistemas de Gestión de Bases de Datos (SGBD, DBMS).

En esta línea han surgido metodologías de desarrollo de sistemas basadas en reglas [ROS03]. Asimismo, han surgido sistemas basados en reglas orientados a objetos. Entre ello podemos citar a ILOG, JESS, OPS/J y Blaze Advisor.

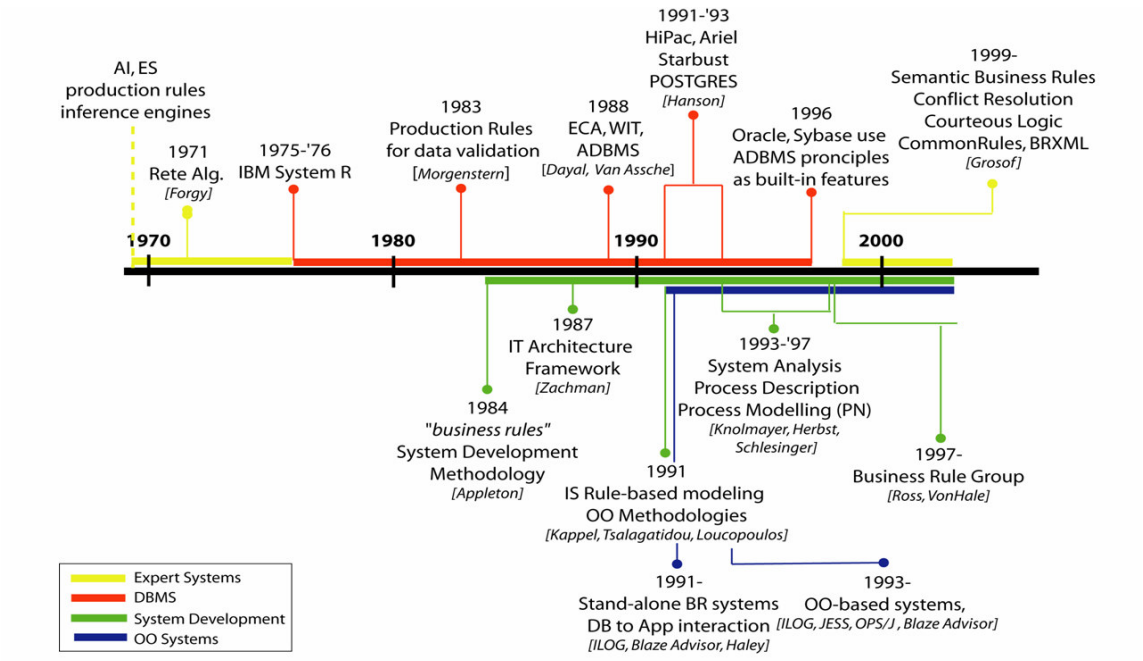

Figura 4. Sistemas relacionados

De lo expuesto podemos concluir que las reglas de negocio se constituyen como un dominio que ha despertado, y continúa haciéndolo, gran interés en la industria. Lo cual queda patente por la aparición de herramientas, cada vez sofisticadas tecnológicamente, a lo largo del tiempo.

Asimismo, tampoco ha pasado inadvertida para la comunidad científica que desarrolla trabajos en esta línea [BAJ05, BUB93, LAY88, PET94, VAS88].

2.2.2. Principios de las reglas de negocio

El enfoque de reglas de negocio ostenta una serie de características o principios, los cuales se detallan en la Tabla 1.

Todas estas características giran en torno al conocimiento que representan y a la facilidad que proporciona el uso de un motor para su administración y uso.

Tabla 1 Principios de las reglas	
Principio	**Propósito del principio**
Separación	• Reuso de reglas • Aplicación de técnicas para optimizar la calidad de las reglas • Cambio de las reglas independientemente de otros aspecto o reglas
Trazabilidad	• Estimación de su corrección y alineamiento con la estrategia empresarial. • Valoración del impacto del cambio de reglas
Externalización	• Permite a usuarios autorizados donde están las reglas • Permite a usuarios autorizados cuales son las reglas
Posicionamiento	• Permite a usuarios autorizados cambios en las reglas de forma simple. • Permite a usuarios autorizados cambios en las reglas de forma rápida.

Seguidamente se aborda la estructura estática del entorno de las reglas de negocio.

2.2.3. Estructura estática del entorno de la reglas de negocio.

Con el objetivo de establecer el marco donde se sitúan, tanto las reglas IF-THEN como el presente TFC, hemos seguido las especificaciones establecidas por el OMG (Object Management Group) en sus especificaciones de Semántica de Negocio y Reglas de Negocio [OMG06]. Dicha estructura se refleja en la Figura 5.

Las directivas definen o restringen algún aspecto de la empresa. Su objetivo es controlar la estructura del negocio o influir sobre el comportamiento de la empresa. Su formulación depende del una parte autorizada de la empresa. Se encuentra íntimamente relacionada con la estrategia empresarial. Algunas directivas son ejecutables (entendido como la capacidad de producir directamente una acción), teniendo en cuenta que en la empresa existen puesto en los que sus ocupantes deciden si la empresa está cumpliendo o no la directiva en cuestión.

La directiva tiene dos componentes, la política de negocio y los elementos de guía. La política de negocio define cursos de acción elegidos entre diferentes alternativas respecto a unas condiciones dadas. Se diseña para guiar decisiones presentes y futuras. La política de negocio no es ejecutable, siendo su propósito servir de guía a la organización. Comparada con la regla de negocio, la política de negocio es menos estructurada, no es atómica, no se expresa de forma tan cuidada como ésta y no es directamente ejecutable. Los elementos de guía, que en su conjunto conforman el cuerpo compartido de guía insertándose en un marco de significados compartidos, tienen como propósito

resolver problemas o dificultades y son ejecutables. Los elementos de guía se basan en los hechos.

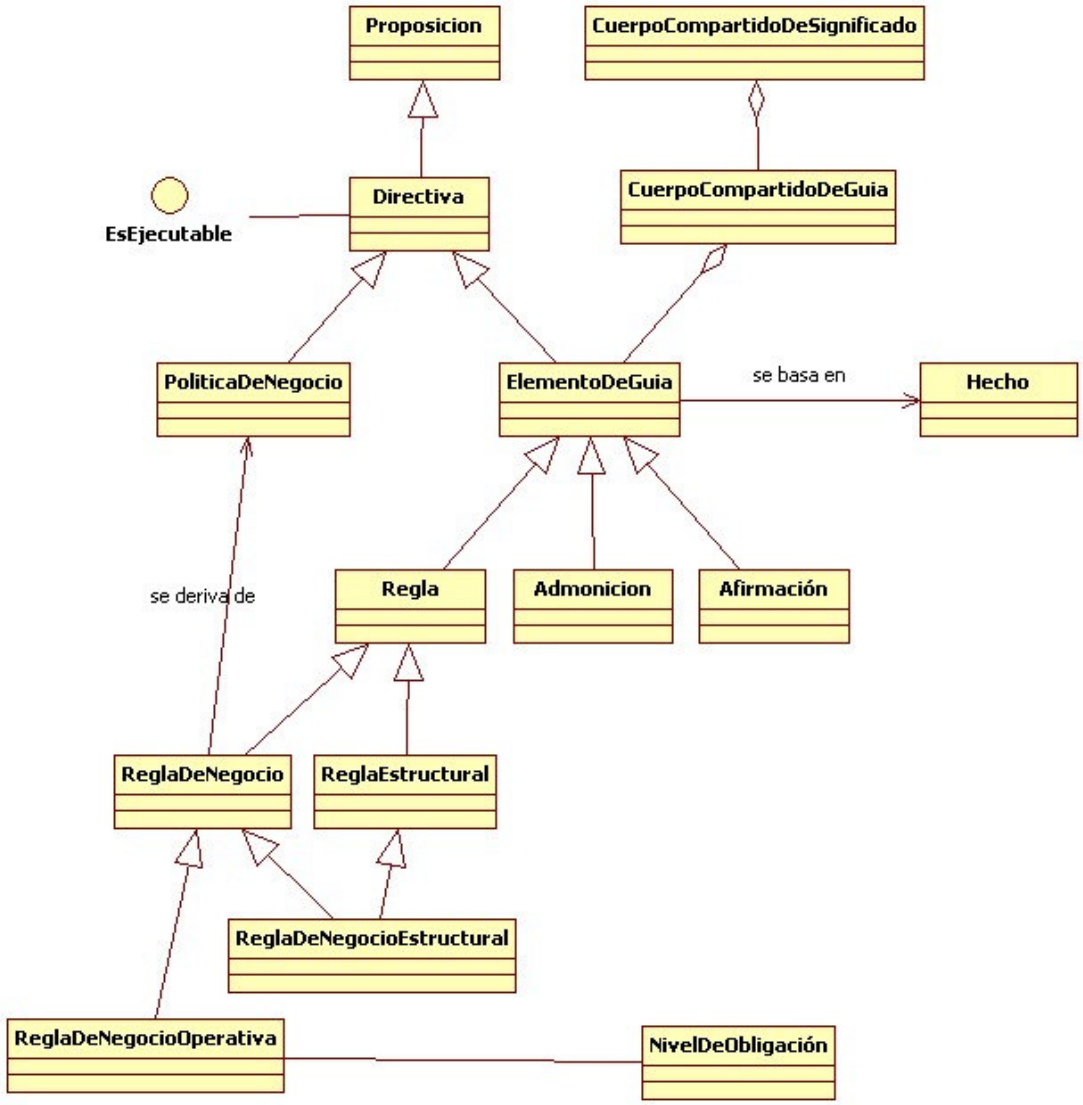

Figura 5. Entorno estático de reglas

Como elementos de guía encontramos, reglas, admoniciones o avisos sobre el comportamiento de la organización o el entorno de ésta y afirmaciones. La regla es un elemento de guía que genera una obligación o necesidad. A su vez las reglas pueden ser reglas de negocio o reglas estructurales. Las reglas de negocio se basan en la política de negocio. Mientras que las reglas estructurales definen una situación.

Las reglas de negocio, que a su vez son reglas estructurales, conforman las reglas de negocio estructurales. El resto de las reglas de negocio son reglas de

negocio operativas. Dichas reglas de negocio operativas establecen la acción a emprender ante la presencia de unas determinadas precondiciones. Estas reglas se pueden traducir a reglas IF-THEN que es el objeto del motor de reglas que estamos analizando y diseñando en este documento. Dichas reglas de negocio operativas tiene asociado un nivel de obligación el cual sitúa en una escala la relevancia y las características de su ejecución. En la Tabla 2 se recoge una escala de niveles de obligación a modo de ejemplo.

Tabla 2 Escala de niveles de obligación para reglas de negocio operativas

Nivel de Obligación	Descripción
Estricto	Debe ejecutarse estrictamente
Obligación retrasada	Debe ejecutarse estrictamente, aunque ésta puede ser retrasada
Pre-autorización	De obligada ejecución, aunque pueden existir excepciones y su ejecución requiere una autorización previa.
Post-justificada	De obligada ejecución, siendo precisa su justificación posterior
Opcional	Puede no ser ejecutada, requiriéndose una justificación posterior
Guía	No es obligada ni se requiere justificación, sólo es una sugerencia

Las admoniciones y las afirmaciones constituyen elementos de guía no obligatorios pudiendo ser asumidas por la práctica o clientes específicos. La diferencia entre ambas estriba en que las admoniciones no son obligaciones y las afirmaciones presentan posibilidades disponibles.

En este marco se han situado las reglas IF-THEN, las cuales constituyen el elemento más formalizado de las reglas de negocio. En el siguiente apartado se recogen las principales corrientes en la literatura especializada sobre la implementación de reglas.

2.3. Implementación de reglas

Existen una serie de alternativas para implementar reglas de negocio. Son dos corrientes basadas en tecnologías diferentes. La primera de ellas se fundamenta en la tecnología de objetos, para ello se emplea la notación OCL (Lenguaje de Restricciones de Objetos, Object Constraint Language) que es un sublenguaje de UML [BOO99;RUM00] para la especificación de restricciones. La segunda se basa en el metalenguaje XML. Seguidamente se detallan cada una de estas aproximaciones.

2.3.1. Notación UML-OCL

La construcción genérica para definir mediante UML reglas formales es mediante restricciones. El OMG ha propuesto un lenguaje formal para expresar

restricciones libres denominado OCL. No es un lenguaje de programación, sino un sublenguaje de UML para modelar restricciones. Podría haberse utilizado el lenguaje natural, pero su obvia ambigüedad provocó la necesidad de la incorporación del OCL. Se puede decir que se trata de un lenguaje de expresión, porque se utiliza únicamente para expresar un valor, sin provocar cambios en el modelo. También se considera un lenguaje de modelado, porque no es ejecutable. Asimismo, también se trata de un lenguaje formal, porque todos sus elementos han sido definidos formalmente.

2.3.2. RuleML

RuleML o Rule Markup Language [RML06] es un lenguaje en construcción basado en el metalenguaje XML o eXtensible Markup Language. Su objetivo estriba en adoptar el rol de lenguaje Web canónico para reglas usando XML, semántica formal e implementaciones eficientes.

RuleML cubre el espectro completo de la reglas, desde las reglas operativas a las estructurales. También puede especificar consultas e inferencias en ontologías y soportar servicios Web y agentes. En la actualidad se encuentra en la versión 0.9. Seguidamente se expone un ejemplo del código RuleML en el que se recoge implicaciones y consultas.

```
<RuleML>
  <Implies>
    <And>
      <Atom>
        <op><Rel>promocion</Rel></op>
        <Var>cliente</Var>
      </Atom>
      <Atom>
        <op><Rel>regular</Rel></op>
        <Var>producto</Var>
      </Atom>
    </And>
    <Atom>
      <op><Rel>descuento</Rel></op>
      <Var>cliente</Var>
      <Var>producto</Var>
      <Ind>5.0 por ciento</Ind>
    </Atom>
  </Implies>
  ...
  <Query>
    <Atom>
      <op><Rel>descuento</Rel></op>
      <Var>cliente</Var>
      <Var>producto</Var>
      <Var>cantidad</Var>
    </Atom>
  </Query>
</RuleML>
```

2.3.3. Lenguaje natural

El lenguaje natural es el que utilizamos los seres humanos. Su empleo en las reglas las acerca a la comprensión por parte del usuario no técnico, aunque dificulta su procesamiento.

Como solución para simplificar su uso se perfila la limitación del lenguaje natural a las condiciones y los consecuentes. De esta forma, se mantendría el formato de regla IF-THEN y su manejo sería más simple. En este sentido, una regla podría formularse en lenguaje natural de las formas que sugiere la Tabla 3.

Tabla 3. Reglas en lenguaje natural

Regla en lenguaje natural libre	Regla en lenguaje natural restringido
Hay que tener en cuenta que los clientes que compran todas las semanas, pero pagan al contado, deberíamos considerarlos como clientes preferentes.	IF <cliente que compra todas las semanas> AND <cliente que paga al contado> THEN <cliente preferente>

Podemos observar como en la columna derecha quedan perfectamente definidas las condiciones y el consecuente o acción a emprender. El uso del lenguaje natural permite la comprensión por parte del usuario no técnico, mientras que su estructuración facilita su procesamiento.

3. Identificación de los subsistemas

Como punto de partida este apartado persigue identificar a alto nivel los grandes elementos de que constará el motor de reglas. Tomamos como punto de partida la clásica arquitectura de tres capas (Figura 6).

La primera capa será la que actúe de interfaz con el usuario (front-end). La capa intermedia recoge la lógica de aplicación, mientras que la última capa es la encargada de la persistencia (back-end).

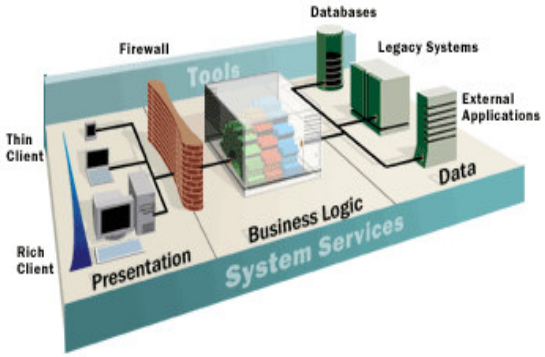

Figura 6. Arquitectura de tres capas

Tanto el front-end como el back-end realizarán las funciones que tienen asignadas como cualquier aplicación clásica. En este epígrafe nos interesa analizar con mayor detalle la capa intermedia referida lógica de aplicación (Figura 7). Necesitaremos una gestión de las reglas de negocio, así como una clasificación de éstas en categorías. De igual forma, necesitamos modelar el proceso que ejecutan las reglas. Además, será preciso un módulo para la gestión de perfiles de acceso. En un sistema de este tipo resulta esencial el control de permisos de acceso, ya que contienen información crítica para el funcionamiento de la organización.

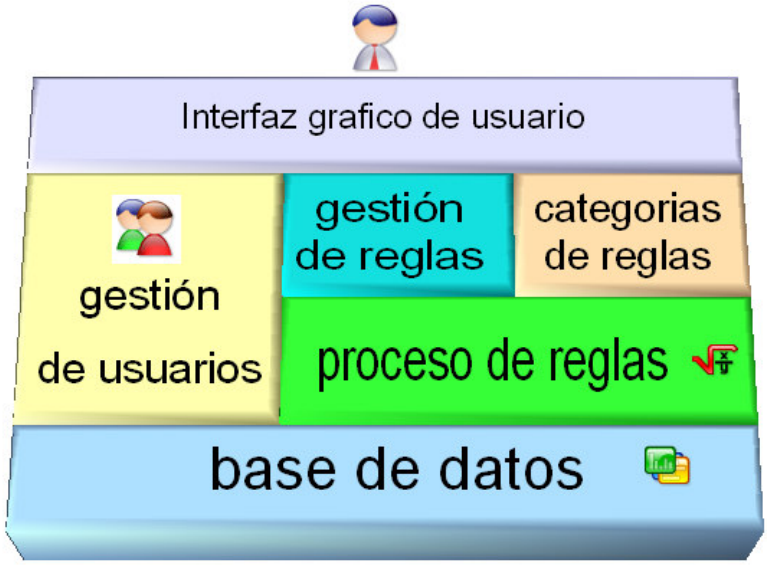

Figura 7. Arquitectura del sistema

4. Especificación de requisitos

Los requisitos constituyen la solución que ofrece el software al problema que ha generado su aparición. Son descripciones del comportamiento, propiedades y restricciones del software a desarrollar.

Para la especificación de requisitos vamos a distinguir entre requisitos funcionales y no funcionales. Los requisitos funcionales describen aquellas funciones que debe realizar el software para sus usuarios. Estos requisitos quedan recogidos en los casos de uso.

Los requisitos no funcionales no van asociados a casos de uso concretos y consisten en restricciones impuestas por el entorno o la tecnología, especificaciones sobre tiempo de respuesta o volumen de información tratado por unidad de tiempo, requisitos en cuanto a interfaces, extensibilidad, facilidad de mantenimiento, entre otros.

4.1. Recogida de requisitos

En esta etapa se lleva a cabo la definición, análisis y validación de los requisitos a partir de la información facilitada por el usuario. El objetivo de esta actividad es obtener un catálogo detallado de los requisitos, a partir del cual se pueda comprobar que los productos generados en las actividades de modelado se ajustan a los requisitos de usuario.

Consiste en un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido [FER01]. La importancia de esta fase es esencial puesto que los errores más comunes y más costosos de reparar, así como los que más tiempo consumen se deben a una inadecuada ingeniería de requisitos.

Esta recogida se descompone en un conjunto de tareas que, si bien tienen un orden, exige continuas realimentaciones y solapamientos, entre sí y con otras tareas realizadas en paralelo. No es necesaria la finalización de una tarea para el comienzo de la siguiente. Lo que se tiene en un momento determinado es un catálogo de requisitos especificado en función de la progresión del proceso de análisis.

Las técnicas más frecuentemente usadas en la recogida de requisitos son las entrevistas, el Desarrollo Conjunto de Aplicaciones (Joint Application Development o JAD), la tormenta de ideas (brainstorming) y el uso de escenarios [WEI98, ROL98], más conocidos como casos de uso [BOO99, JAC00].

Dichas técnicas suelen complementarse con otras técnicas [IEE98] como la observación directa, el estudio de documentación, los cuestionarios, la inmersión en el negocio del cliente [GOG93] o haciendo que los ingenieros de requisitos sean aprendices del cliente [BEY95]. La revisión la documentación del software en uso en la actualidad constituye otra alternativa. Ello será posible cuando exista un sistema que esté cumpliendo en alguna medida las necesidades que cubrirá el sistema a desarrollar. Deberán estudiarse los manuales de la aplicación y también los procedimientos manuales que se utilizan, sobre todo si no existe un sistema.

Asimismo, suelen resultar útil como fuente común de recogida de requisitos los colegas de los usuarios. Es posible que los usuarios estén acostumbrados a realizar su trabajo de la misma manera durante mucho tiempo. Ello puede provocar cierto grado de automatismo en sus tareas, lo cual conlleva la ausencia de reflexión consciente sobre éstas. El contacto con colegas de los usuarios puede aportar diferentes visiones sobre las tareas que se pretende cubrir. De igual forma, puede resultar de utilidad la revisión de sistemas análogos al objeto del proyecto que existan en el mercado. Los usuarios pueden considerar evidente que determinadas funciones forman parte del dominio o del sistema que pueden considerar innecesario mencionarlas. Dichas funciones se encontrarán en software comercial ya existente con gran probabilidad.

Procedemos a revisar someramente las principales técnicas de recogida de requisitos citadas.

4.1.1. Entrevistas

Las entrevistas son la técnica de recogida más comúnmente utilizada. En realidad resultan prácticamente inevitables en cualquier desarrollo, ya que son una de las formas de comunicación más naturales entre personas.

En las entrevistas se pueden identificar tres fases: preparación, realización y análisis [PIA96]. Una vez elaborada la información, se puede enviar al entrevistado para confirmar los contenidos. También resulta importante la evaluación de la propia entrevista para determinar los aspectos mejorables.

4.1.2. Desarrollo Conjunto de Aplicaciones (JAD)

La técnica denominada JAD fue desarrollada por IBM en 1977. Constituye una alternativa a las entrevistas individuales. Se desarrolla mediante un conjunto de reuniones en grupo durante un periodo de 2 a 4 días. En dichas reuniones se ayuda a los clientes y usuarios a formular problemas y explorar posibles soluciones, involucrándolos y haciéndolos partícipes del desarrollo.

Esta técnica se base en cuatro principios [RAG94]: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación (diagramas, transparencias, multimedia, herramientas CASE, etc.), mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (What You See Is What You Get, lo que se ve es lo que se obtiene), por la que durante las reuniones se trabaja directamente sobre los documentos a generar. El JAD tiene dos grandes etapas, el JAD/Plan cuyo objetivo es la elicitación y especificación de requisitos, y el JAD/Design, en el que se aborda el diseño del software.

Debido a las necesidades de organización que requiere y a que no suele adaptarse bien a los horarios de trabajo de los clientes y usuarios, esta técnica no suele emplearse con frecuencia, aunque cuando se aplica suele tener buenos resultados, especialmente para recoger requisitos en el campo de los Sistemas de Información [RAG94].

En comparación con las entrevistas individuales, JAD presenta las siguientes ventajas:

- Ahorra tiempo al evitar que las opiniones de los clientes se contrasten por separado.
- Todo el grupo, incluyendo los clientes y los futuros usuarios, revisa la documentación generada, no sólo los ingenieros de requisitos.
- Implica más a los clientes y usuarios en el desarrollo.

4.1.3. Tormenta de ideas

La tormenta de ideas o brainstorming consiste en una serie de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios [GAU98, RAG94]. Las sesiones de brainstorming suelen estar formadas por un número de cuatro a diez participantes, uno de los cuales es el jefe de la sesión, encargado más de comenzar la sesión que de controlarla.

Como técnica de recogida de requisitos, el brainstorming puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo cuando los requisitos son todavía muy difusos.

Frente al JAD, el brainstorming tiene la ventaja de que su aprendizaje resulta muy simple y requiere poca organización. Incluso existen propuestas de realización de brainstorming por vídeo–conferencia [RAG94]. Como desventaja podemos encontrar que al resultar un proceso poco estructurado, puede no producir resultados con la misma calidad o nivel de detalle que otras técnicas.

4.1.4. Casos de uso

Los casos de uso constituyen una técnica para la especificación de requisitos funcionales propuesta inicialmente en [JAC93] y que actualmente forma parte de la propuesta de UML [BOO99].

Un caso de uso consiste en la descripción de una secuencia de interacciones entre el sistema y uno o más actores en la que se considera al sistema como una caja negra y en la que la que los actores obtienen resultados observables. Los actores son personas u otros sistemas que interactúan con el sistema cuyos requisitos se están describiendo [SCH98]. Los casos de uso presentan ventajas sobre la descripción meramente textual de los requisitos funcionales, ya que facilitan la recogida de requisitos y resultan fácilmente comprensibles por los clientes y usuarios. Además, pueden servir de base a las pruebas del sistema y a la documentación para los usuarios [WEI98]. En cualquier caso consideramos que la descripción textual y los casos de uso no son excluyentes, sino complementarios.

A pesar de ser una técnica ampliamente aceptada, existen múltiples propuestas para su utilización concreta [COC97]. Para la descripción textual de los casos de uso se propone una plantilla, en las que las interacciones se numeran siguiendo las propuestas de [COC97] y [SCH98]. La plantilla detallada en la Tabla 4 se basa en la propuesta realizada por [DUR00] para la especificación de requisitos funcionales.

Tabla 4. Plantilla de especificación textual de casos de uso

UC - <id>	<denominación descriptiva>	
Versión	<nº de la versión actual> [<fecha de la versión actual>]	
Descripción	Descripción del comportamiento del sistema {durante la realización de los casos de uso <lista de casos de uso> | cuando <evento de activación>}	
Precondición	<precondición del caso de uso>	
Secuencia normal	*Pasos*	*Acción*
	p_1	{El actor <actor>| El sistema} <acción/es realizada/s por actor/sistema>
	p_2	Se realiza el caso de uso <caso de uso (UC–x)>
	p_3	Si <condición>, {el actor <actor>| el sistema} <acción/es realizada/s por {actor | sistema}>
	p_4	Si <condición>, se realiza el caso de uso <caso de uso (UC–x)>
	…	…
Postcondición	<postcondición del caso de uso>	
Excepciones	*Pasos*	*Acción*
	p_i	Si <condición de excepción>, {el actor <actor>| el sistema} <acción/es realizada/s por {actor | sistema}>, a continuación este caso de uso {continúa | finaliza}
	p_j	Si <condición de excepción>, se realiza el caso de uso <caso de uso (UC–x)>}, a continuación este caso de uso {continúa| finaliza}
	…	…
Frecuencia esperada	<no de veces relativas>	
Importancia	<importancia del caso de uso respecto a los objetivos del sistema>	
Urgencia	<urgencia del requisito>	
Comentarios	<comentarios adicionales sobre el caso de uso>	

El significado de los campos de la plantilla se detalla a continuación.

- **Identificador y denominación descriptiva**. Los identificadores de los casos de uso comenzarán con UC. Mientras que la denominación descriptiva suele coincidir con el objetivo que los actores esperan alcanzar al realizar el caso de uso. No se debe confundirse este objetivo con los objetivos de la aplicación. El objetivo que los actores esperan alcanzar al realizar un caso de uso es de más bajo nivel, por ejemplo dar de alta una regla o realizar una búsqueda de éstas.

- **Descripción**. Si el caso de uso es abstracto (es decir no lo realiza directamente el actor), deben indicarse los casos de uso en los que se debe realizar (<lista de casos de uso>), es decir, aquellos desde los que es incluido o a los que extiende. Si, por el contrario, se trata de un caso de uso concreto (lo realiza directamente el actor), se debe indicar el evento de activación que provoca su realización. Un caso de uso es abstracto si no puede ser realizado por sí mismo, por lo que sólo tiene significado cuando se utiliza para describir alguna funcionalidad que es común a otros casos de uso. Por otra parte, un caso de uso será concreto si puede ser iniciado por un actor y realizado por sí mismo.

- **Precondición**. Se expresan en lenguaje natural las condiciones necesarias para que se pueda realizar el caso de uso.

- **Secuencia normal**. Contiene la secuencia normal de interacciones del caso de uso. En cada paso, un actor o el sistema realiza una o más acciones, o se realiza (se incluye) otro caso de uso. Un paso puede tener una condición de realización, en cuyo caso si se realizara otro caso de uso se tendría una relación de extensión. Se asume que, después de realizar el último paso, el caso de uso termina.

- **Postcondición**. Recoge en lenguaje natural las condiciones que se deben cumplir después de la terminación normal del caso de uso.

- **Excepciones**. Especifica el comportamiento del sistema en el caso de que se produzca alguna situación excepcional durante la realización de un paso determinado. Después de realizar las acciones o el caso de uso asociados a la excepción (una extensión), el caso de uso puede continuar la secuencia normal o terminar, lo que suele ir acompañado por una cancelación de todas las acciones realizadas en el caso de uso dejando al sistema en el mismo estado que antes de comenzar el caso de uso.

- **Frecuencia esperada**. Indica la frecuencia esperada relativa de realización del caso de uso, que aunque no es realmente un requisito, es una información interesante para los desarrolladores.

Tanto la secuencia normal como las excepciones pueden ser representadas diagramáticamente por medio de diagramas de actividad. También se podría incluir en la plantilla información sobre los siguientes apartados.

- Datos relativos al autor del caso de uso.
- Fuentes del caso de uso.
- Rendimiento.
- Estado.
- Estabilidad.

Los datos expuestos se han obviado, ya que en el marco del presente TFC todos los casos de uso presentarían los mismos valores para ellos.

Los casos de uso tienen una representación gráfica en los denominados diagramas de casos de uso [Booch et al. 1999]. En estos diagramas, los actores se representan en forma de pequeños monigotes y los casos de uso se representan por elipses contenidas dentro de un rectángulo que representa al sistema. La participación de los actores en los casos de uso se indica por una flecha entre el actor y el caso de uso que apunta en la dirección en la que fluye la información. Un ejemplo de este tipo de diagramas puede verse en la Figura 8.

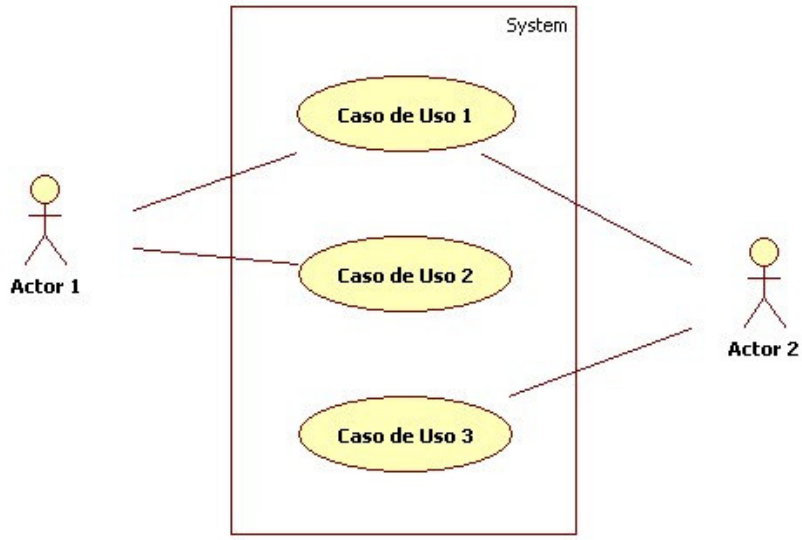

Figura 8. Ejemplo de Diagrama de Casos de Uso

Los diagramas de casos de uso sirven para proporcionar una visión global del conjunto de casos de uso de un sistema así como de los actores y los casos de

uso en los que éstos intervienen. Las interacciones concretas entre los actores y el sistema no se muestran en este tipo de diagramas.

En ocasiones se establecen relaciones entre distintos casos de uso para simplificar su descripción. Las dos relaciones posibles y sus semánticas según [BOO99] son las detalladas a continuación (Figura 9).

- **Include**. El caso de uso 4 (UC-4) incluye el caso de uso 1 (UC-1), cuando UC-1 es una parte del caso de uso UC-4, es decir, la secuencia de interacciones de UC-1 forma parte de la secuencia de interacciones de UC-4. El caso de uso UC-1 se realiza siempre dentro del caso de uso UC-4. Es más, siempre que ocurre UC-4 ocurre también UC-1, por lo que se dice que UC-1 es un caso de uso abstracto [JAC97]. A partir de las especificaciones 2.0 de UML [UML04] los casos de uso incluidos en otros pueden ser realizados directamente por actores, por lo que dejarían de ser abstractos.

  Se suele utilizar esta relación cuando se detectan subsecuencias de interacciones comunes a varios casos de uso. Dichas subsecuencias comunes se extraen de los casos de uso que las contienen y se les da forma de casos de uso que son incluidos por los casos de uso de los que se han extraído. De esta forma se evita repetir las mismas subsecuencias de interacciones una y otra vez en varios casos de uso.

- **Extends**. El caso de uso 3 (UC-3) extiende a otro caso de uso 2 (UC-2) cuando UC-3 es una subsecuencia de interacciones de UC-2 que ocurre en una determinada circunstancia. En cierta forma, UC-3 completa la funcionalidad de UC-2. El caso de uso UC-3 puede realizarse o no cuando se realiza el caso de uso UC-2, según se den las circunstancias. Por otro lado, el caso de uso UC-3 puede ser un caso de uso abstracto o concreto, en cuyo caso puede ocurrir sin necesidad de que ocurra el caso de uso UC-2.

- **Herencia**. El caso de uso 5 (UC-5) hereda del caso de uso 1 (UC-1). Al igual que en la herencia entre clases, el caso de uso hijo hereda todo lo perteneciente al caso de uso padre. El caso de uso padre se trata de un caso de uso abstracto, que no está definido completamente. Este tipo de relación se utiliza mucho menos que las dos anteriores, únicamente cuando un caso de uso definido de forma abstracta se particulariza por medio de otro caso de uso más específico.

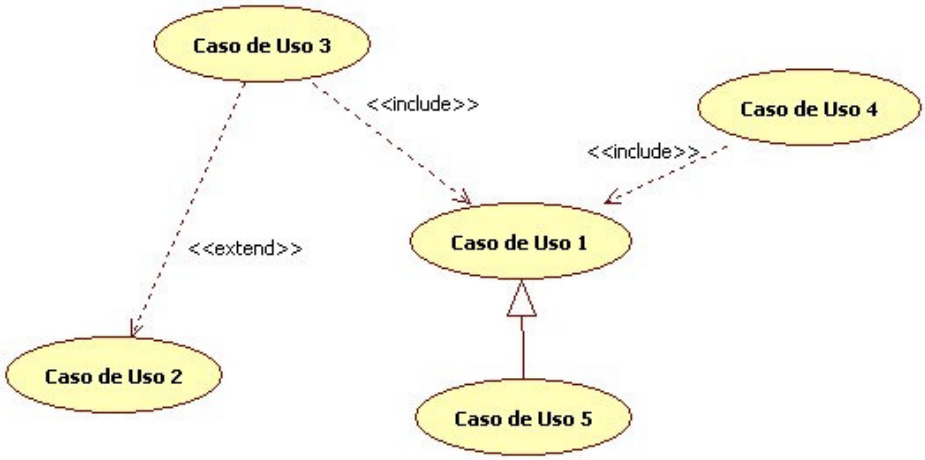

Figura 9. Relaciones entre casos de uso

Seguidamente procedemos a exponer a alto nivel los requisitos funcionales y no funcionales del motor de reglas.

4.2. Requisitos funcionales

El motor de reglas gestionará las reglas que los usuarios autorizados deseen almacenar para su adecuado procesamiento, almacenamiento y mantenimiento. Esta gestión estará basada en la catalogación de las reglas que realizarán los propios usuarios. Las reglas estarán relacionadas con productos o servicios que ofrezca la organización. Concretamente el motor debe permitir las funciones que procedemos a recoger en la Tabla 5.

Tabla 5. Requisitos funcionales

	Requisito	Subsistema
1	Alta de una regla	Gestión de reglas
2	Baja de una regla	
3	Modificar una regla	
4	Relacionar reglas	
5	Ejecución de reglas	Proceso de reglas
6	Búsqueda de regla	
7	Notificación de reglas nuevas	
8	Alta de usuario	Gestión de usuarios
9	Baja de usuario	
10	Identificación (log in)	
11	Cierre Sesión (log out)	
12	Crear administrador	
13	Alta de categoría de reglas	Gestión de categorías de reglas
14	Baja de categoría de reglas	
15	Modificación de categoría de reglas	

A continuación procedemos a detallar los requisitos funcionales anteriormente expuestos.

4.2.1. Alta, baja y modificación de una regla

El usuario autorizado podrá dar de alta una regla en el motor. Las instancias de reglas caracterizaran su estado mediante los valores de los siguientes atributos.

- Denominación.
- Autor.
- Antecedente.
- Consecuente.
- Fecha de creación.
- Producto o servicio al que hace referencia.
- Departamento responsable.
- Grado de importancia.
- Destinatarios de la notificación.
- Observaciones.

El usuario autorizado podrá eliminar o modificar una regla en el motor.

4.2.2. Búsqueda de una regla

Mediante la selección de uno o más criterios de búsqueda en torno a los descriptores de los documentos (atributos y/o categorías). Una vez localizada la regla, según los permisos del usuario, este podrá revisarla o realizar tareas de administración.

La regla localizada por cualquier criterio se muestra en su integridad. El usuario podrá utilizarla en su proceso decisional desde que es localizada.

4.2.3. Relacionar reglas

Se permitirá el anidamiento de reglas, de forma que el consecuente de una regla puede constituir el antecedente de otra. Ello permitirá propagar un efecto sobre varias reglas.

Para ilustrarlo proponemos un ejemplo. Sea el conjunto de hechos $H=\{h_1,h_2,h_3\}$. Si tenemos dos reglas de la siguiente forma.

IF h_1=TRUE THEN h_2=TRUE (1)

IF h_2=TRUE THEN h_3=FALSE (2)

Tenemos que el consecuente de la regla (1) actúa como antecedente de la regla (2). Si relacionamos dichas reglas podemos inferir que si h_1=TRUE entonces h_3=FALSE. Dicha relación debe establecerse por parte del diseñador de reglas cuando ostente suficiente confianza en ésta.

4.2.4. Notificación de reglas nuevas

El alta de nuevas reglas en el motor será notificado a través del correo electrónico a una serie de interesados en dicha regla, los cuales se encuentran recogidos en el atributo Destinatarios de la notificación.

4.2.5. Alta y baja de usuario

El administrador de usuarios podrá incorporar nuevos usuarios al motor. Así como eliminar alguno de los existentes. Las instancias de usuario caracterizan su estado asignando valores a los siguientes atributos.

- Nombre.
- Departamento.
- Correo electrónico. El contenido de este atributo se usará para las notificaciones.

4.2.6. Identificación y cierre de sesión

Un usuario debe identificarse ante el motor para usarlo. Para ello debe estar de alta como usuario. Una vez realizada la autenticación, se abre una sesión que se mantiene hasta el momento en que el usuario decide abandonar el sistema y cierra la sesión.

4.2.7. Crear administrador

Un administrador del sistema podrá crear otro administrador a partir de un usuario ya registrado, concediéndole así los mismos permisos que él posee.

4.2.8. Alta, baja y modificación de categorías de reglas

Las reglas podrán clasificarse atendiendo al criterio que se desee. No es obligatorio incluir una regla en todas las categorías se tengan definidas, sino sólo aquellas que sean relevantes según el criterio del usuario autorizado. El objetivo de las categorías es, al igual que los atributos, clasificar y organizar las reglas, aunque en este caso de una forma mucho más cerrada y concreta. Algunas categorías pueden ser las siguientes.

- Nombre del proyecto relacionado.
- Departamento del autor del documento.
- Reglas relacionadas.

4.3. Requisitos no funcionales

Se incluyen en este apartado los requisitos no funcionales que no se especificaron en los requisitos funcionales detallados en el epígrafe anterior. En la Tabla 6 recogemos los requisitos no funcionales relacionados con la calidad.

Tabla 6. Requisitos no funcionales

	Categoría	Requisito no funcional
1	**Fiabilidad**	Recuperación frente a fallos de conexión. Se ha de asegurar que no se pierdan los datos del perfil definido por el usuario. Incluyendo no perderlos en el envío al servidor o en el envío a otras máquinas, como si hay un fallo de conexión entre el receptor del usuario y el servidor.
		Recuperación frente a fallos del sistema, posibilitando el reinicio del sistema
		Fiabilidad en la autentificación de los usuarios
2	**Usabilidad**	Interfaz (GUI) sencilla y amigable.
		Existencia de un Manual de Usuario escrito para describir el funcionamiento y el uso del sistema al usuario final.
3	**Soporte**	Facilidad de instalación.
		Facilidad de mantenimiento, lo que requiere código y diseño documentado.
		Facilidad de actualización hacia versiones más modernas.
4	**Información**	En todo momento debe informarse al usuario de todas las acciones que puede realizar en el sistema.

5	**Seguridad**	Debido a la importancia de la información almacenada, deberá realizarse una copia de seguridad diaria de todos los archivos de la aplicación.
6	**Portabilidad**	Debe garantizarse la instalación en diferentes plataformas, por lo que habrá que utilizarse de estándares de mercado posible (SQL, XML para la comunicación), evitando siempre elementos de arquitectura propietaria.

4.4. Identificación de actores

En este apartado se analizan los actores y sus relaciones con los diferentes paquetes de casos de uso.

4.4.1. Paquetes de casos de uso y jerarquía de actores

La jerarquía de actores y sus atribuciones se presentan de forma gráfica en la Figura 10.

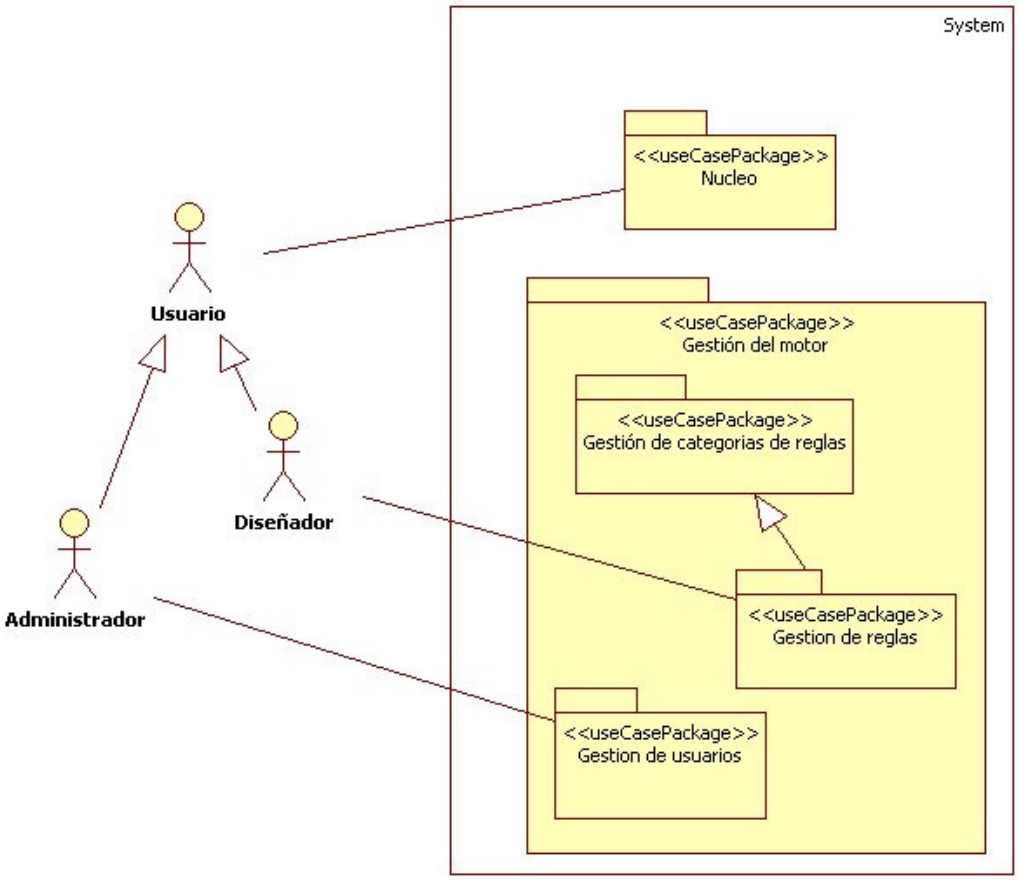

Figura 10. Jerarquía de actores

4.4.2. Descripción textual de los actores

La descripción textual de los actores queda detallada a continuación en la Tabla 7.

Tabla 7. Descripción textual de las atribuciones de los actores		
Usuario	**Paquetes**	**Descripción**
Usuario	Núcleo	El usuario podrá ejecutar las reglas y será notificado por el motor por las reglas para las cuales haya sido identificado como destinatario. También podrá realizar búsquedas de reglas.
Diseñador	Gestión de categorías de reglas Gestión de reglas	El diseñador podrá dar de alta, de baja o modificar reglas, realizar búsquedas y relacionar reglas. También podrá dar de alta, baja y modificar categorías de reglas
Administrador	Gestión de usuarios	Podrá dar de alta y baja usuarios y podrá otorgar a usuarios o diseñadores los privilegios de administrador.

4.5. Identificación de Casos de Uso

4.5.1. Paquetes de análisis

Dentro del paquete de casos de uso que forma el modelo de análisis (Figura 11), se han separado los casos de uso en dos paquetes distintos.

- Núcleo. En este paquete situamos los casos de uso relacionados directamente con el objetivo de la aplicación, es decir, el uso de las reglas de negocio. En este paquete incluimos los casos de uso relativos a la autentificación y al cierre de la sesión, ya que son necesarios para el uso del motor.

- Gestión del motor. Este paquete contiene aquellos casos de uso que cubren funcionalidades no relacionadas directamente con el objetivo de la aplicación. Sobre todo los casos de uso relacionados con la gestión del propio motor.

El objetivo de la separación de este último paquete del núcleo estriba en la localización de las funcionalidades que pueden variar debido a factores distintos de los de la lógica evolución de la aplicación: cambios en la seguridad, en la gestión de usuarios, entre otros.

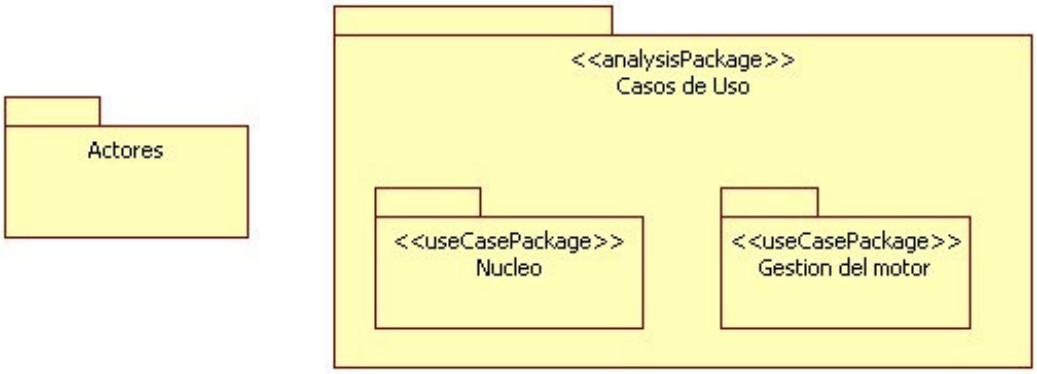

Figura 11. Paquetes del modelo de Análisis

A continuación analizamos los paquetes que componen el paquete denominado Gestión del motor.

4.5.2. Paquetes de la gestión del motor

El paquete de Gestión del motor contiene tres paquetes más (Figura 12) para separar los casos de uso relacionados con la gestión de la reglas de aquellos relativos a la gestión de usuarios.

Los paquetes que componen la gestión del motor son los detallados a continuación.

- Gestión de categorías de reglas. En este paquete se engloban los casos de uso relativos al alta, baja y modificación de las categorías de reglas de negocio.

- Gestión de reglas. Este paquete, además de heredar del paquete de Gestión de categorías de reglas, recoge los casos de uso relativos al alta, baja o modificación de reglas, así como la realización de búsquedas y la relación de reglas.

- Gestión de usuarios. En este paquete se incluyen los casos de uso relacionados con la gestión de usuarios. Concretamente el alta y baja usuarios, así como la asignación de los privilegios de administrador a usuarios que no los ostentaban.

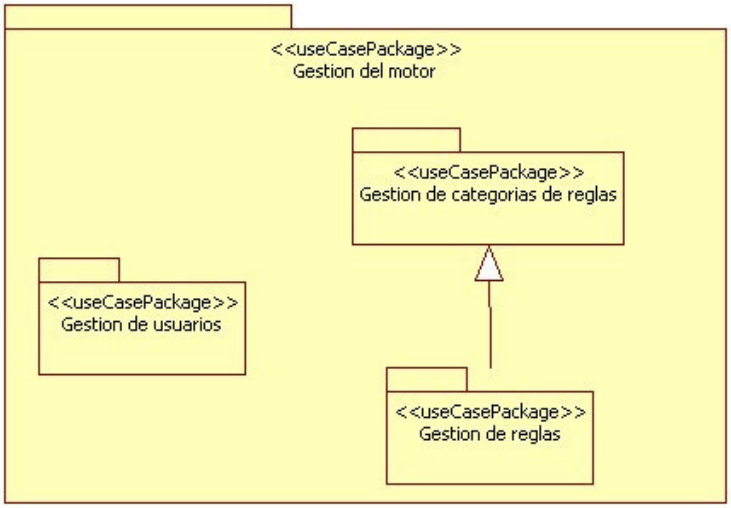

Figura 12. Paquetes de la gestión del motor

4.5.3. Casos de uso del núcleo

En este paquete situamos los casos de uso (Figura 13) relacionados con el uso de las reglas de negocio, la autentificación y el cierre de la sesión. En el diagrama de casos de uso únicamente aparece el actor Usuario, ya que es el que se relaciona con este paquete directamente. El resto de los actores heredan del citado actor teniendo acceso a todos casos de uso como se indica en la jerarquía de actores (Figura 10).

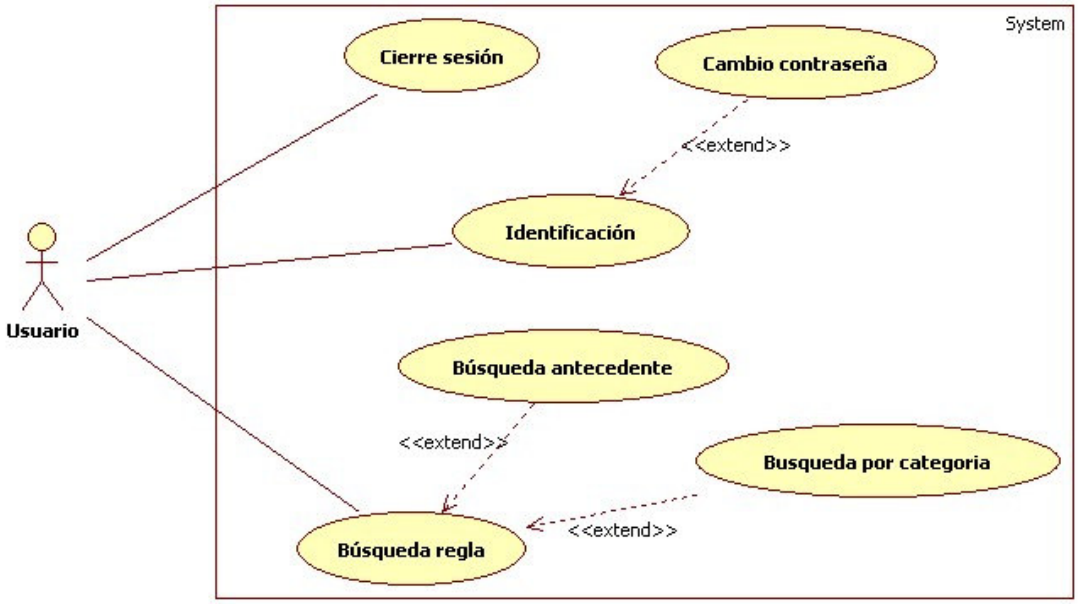

Figura 13. Diagrama de casos de uso del núcleo

4.5.4. Casos de uso de la gestión de categorías de reglas

En este paquete se engloban los casos de uso (Figura 14) relativos al alta, baja y modificación de categorías de reglas de negocio.

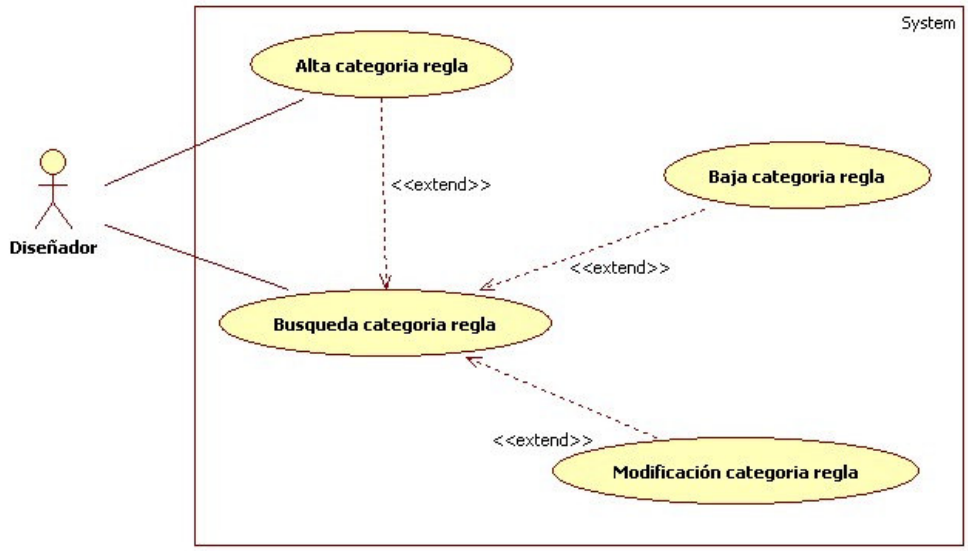

Figura 14. Diagrama de casos de uso de la gestión de categorías de reglas

4.5.5. Casos de uso de gestión de reglas

Este paquete, además de heredar del paquete de Gestión de categorías de reglas, recoge los casos de uso (Figura 15) relativos al alta, baja o modificación de reglas, así como la realización de búsquedas y la relación entre reglas.

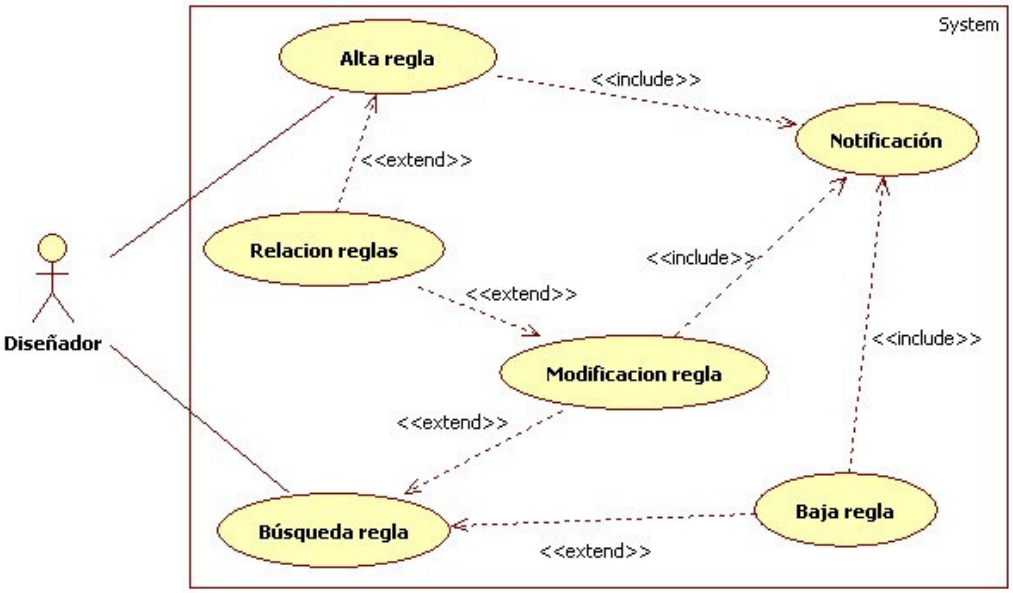

Figura 15. Diagrama de casos de uso de gestión de reglas

4.5.6. Casos de uso de gestión de usuarios

En este paquete se incluyen los casos de uso (Figura 16) relacionados con la gestión de usuarios. Concretamente el alta y baja usuarios, así como la asignación de los privilegios de administrador a usuarios que no los ostentaban.

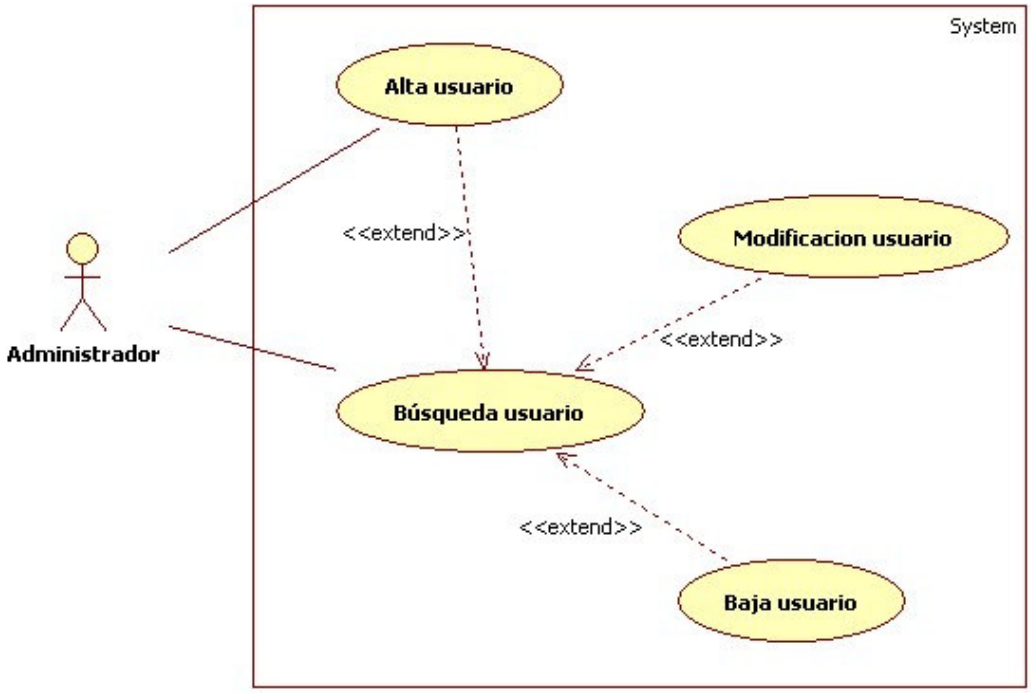

Figura 16. Diagrama de casos de uso de gestión de usuarios

4.6. Especificación de los casos de uso

En este apartado se especifican los casos de uso identificados en el apartado anterior. La especificación se llevará a cabo de las dos maneras siguientes.

- Textual. Se describirán textualmente los detalles asociados a cada caso de uso. Para ello se usará la plantilla especificada en la Tabla 4.
- Diagrama. Se usarán los diagramas de actividad para representar el comportamiento dinámico de los casos de uso.

4.6.1. Casos de uso del núcleo

Los casos de uso englobados en el núcleo se detallan a continuación.

Tabla 8. Especificación textual del casos de uso <Identificación>		
UC-<1>	Identificación	
Versión	1.0	
Descripción	El usuario del sistema se identifica en el sistema para su utilización	
Precondición	El usuario ha sido dado de alta en el sistema correctamente y dispone de un username y un password correctos	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor usuario accede a la ventana de identificación del sistema
	P_2	Si el actor selecciona el comando correspondiente, el actor realiza el caso de uso UC-2
	P_3	El actor introduce su usuario y su contraseña
	P_4	El actor confirma la operación
	P_5	El caso de uso concluye
Postcondición	El actor ha iniciado una sesión en el sistema y puede acceder a las funciones para las que tenga permiso.	
Excepciones	*Pasos*	*Acción*
	P_3	El actor introduce username o password erróneos, por lo que se reinicia UC-1
	P_4	El actor decide no identificarse y cancela la realización de UC-1
Frecuencia esperada	Una vez por sesión	
Importancia	Alta	
Urgencia	Alta	
Comentarios	El sistema contiene información confidencial de la organización, por ello el control de acceso adquiere gran importancia	

El diagrama de actividad recogido en la Figura 17 detalla el UC-1. Hay que destacar que el UC-2 aparece en el diagrama de actividad como un estado de subactividad, ya que el UC-2 extiende al UC-1.

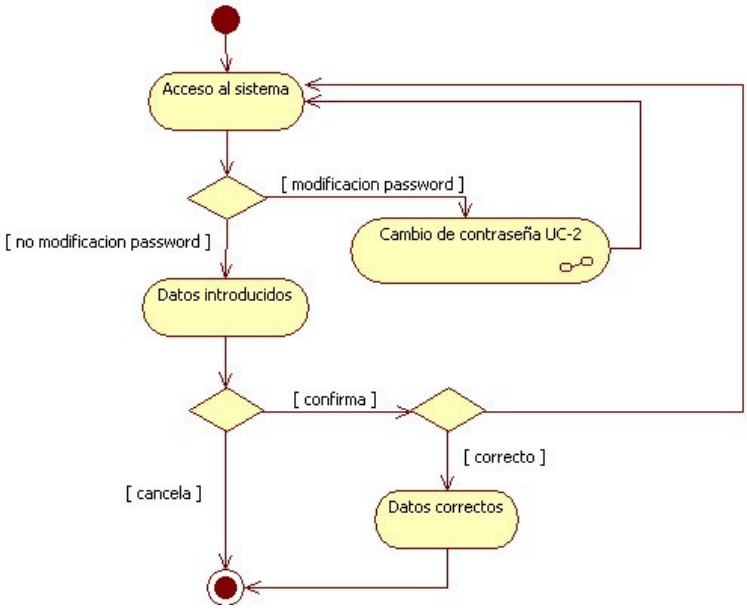

Figura 17. Diagrama de actividad de UC-1

Tabla 9. Especificación textual del caso de uso <Cambio de contraseña>		
UC-<2>	Cambio de contraseña	
Versión	1.0	
Descripción	El usuario del sistema cambia la contraseña que tiene asignada	
Precondición	El usuario se encuentra realizando el UC-1	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor usuario introduce su usuario y su contraseña actuales
	P_2	El actor introduce la nueva contraseña
	P_3	El actor vuelve a introducir la nueva contraseña que es idéntica a la introducida en el paso p_2.
	P_4	El actor confirma
	P_5	El caso de uso concluye
Postcondición	El actor usuario ha cambiado su contraseña	
Excepciones	*Pasos*	*Acción*
	P_3	El actor introduce una contraseña nueva errónea, por lo que se reinicia el caso de uso
	P_4	El actor decide no cambiar la contraseña y cancela UC-2
Frecuencia esperada	Escasa	
Importancia	Media	
Urgencia	Media	
Comentarios	Se recomendará al usuario el cambio de la contraseña asignada inicialmente	

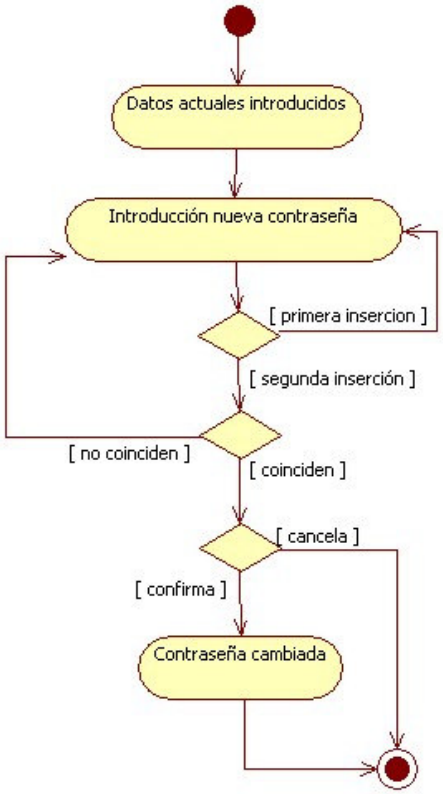

Figura 18. Diagrama de actividad de UC-2

Tabla 10. Especificación textual del caso de uso <Cierre sesión >		
UC-<3>	Cierre sesión	
Versión	1.0	
Descripción	El actor usuario cierra la sesión que mantenía abierta en el sistema	
Precondición	El actor usuario realizó completamente el caso de uso UC-1 y lo concluyó con éxito. El usuario no tiene ningún proceso en ejecución.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor usuario elige la opción cierre sesión.
	p_2	La sesión se cierra
	P_3	El caso de uso concluye
Postcondición	El actor usuario ha cerrado la sesión.	
Excepciones	*Pasos*	*Acción*
	P_2	El actor usuario tiene un proceso en ejecución. El sistema espera a que concluya el proceso y envía a memoria externa los datos persistentes para cerrar la sesión
Frecuencia esperada	Una vez por sesión	
Importancia	Media	
Urgencia	Alta	
Comentarios	Su importancia es media porque el usuario ya se identificó en UC-1 y este caso de uso no quiebra la seguridad.	

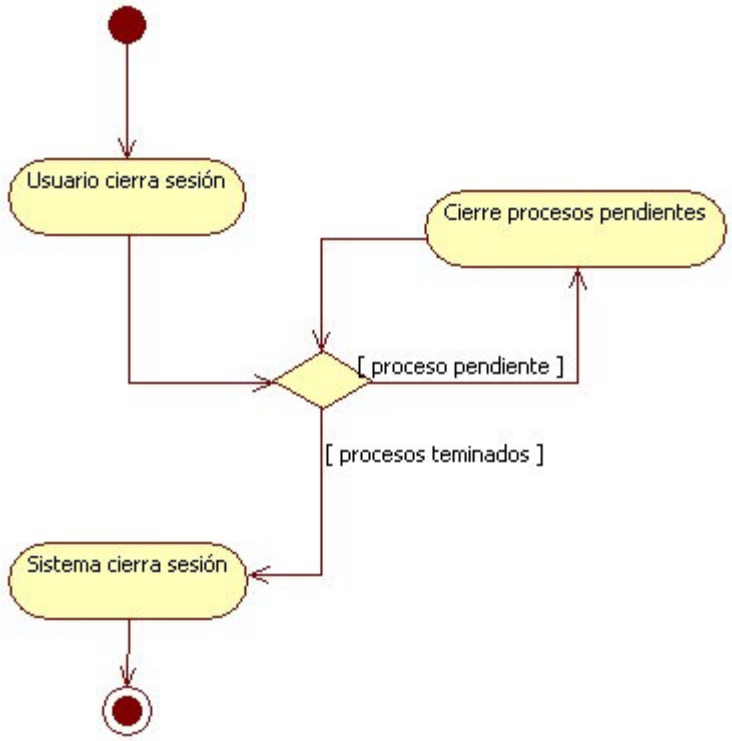

Figura 19. Diagrama de actividad de UC-3

Tabla 11. Especificación textual del caso de uso <Búsqueda regla>		
UC-<4>	Búsqueda regla	
Versión	1.0	
Descripción	El actor usuario busca una regla en el repositorio	
Precondición	El actor usuario realizó completamente el caso de uso UC-1 y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor usuario elige la opción de búsqueda.
	P_2	Si el actor usuario elige búsqueda por antecedente se realiza el caso de uso UC-5
	P_3	Si el actor usuario elige búsqueda por categoría se realiza el caso de uso UC-6
	P_4	El actor usuario elige un atributo como criterio predefinido de búsqueda
	P_5	La regla buscada es localizada.
	P_6	El caso de uso concluye
Postcondición	La regla localizada se muestra al actor usuario	
Excepciones	*Pasos*	*Acción*
	P_5	La regla buscada no es localizada. El sistema ofrece otra búsqueda.
Frecuencia esperada	Frecuentemente	
Importancia	Alta	
Urgencia	Alta	
Comentarios	En todas las sesiones será usual la búsqueda de reglas	

En el diagrama de actividad recogido en la Figura 20 aparecen dos estados de subactividad, ya que el UC-5 y el UC-6 extienden al UC-4.

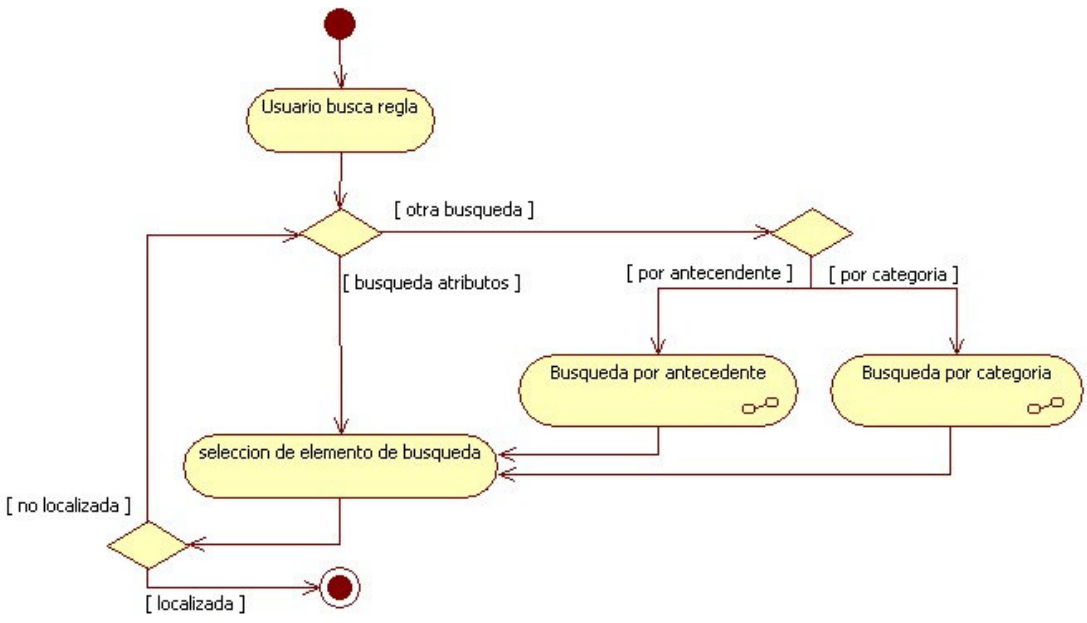

Figura 20. Diagrama de actividad de UC-4

Tabla 12. Especificación textual del caso de uso <Búsqueda antecedente>		
UC-<5>	Búsqueda antecedente	
Versión	1.0	
Descripción	El actor usuario busca un antecedente en el repositorio	
Precondición	El actor usuario realizó completamente el caso de uso UC-1 y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor usuario elige la opción de búsqueda por antecedente.
	P_2	El actor usuario elige el antecedente deseado
	P_3	La regla buscada es localizada
	P_4	El caso de uso concluye
Postcondición	La regla localizada se muestra al actor usuario	
Excepciones	*Pasos*	*Acción*
	P_3	La regla buscada no es localizada. El sistema ofrece otra búsqueda.
Frecuencia esperada	Frecuentemente	
Importancia	Alta	
Urgencia	Alta	
Comentarios	En todas las sesiones será usual la búsqueda de reglas por antecedentes	

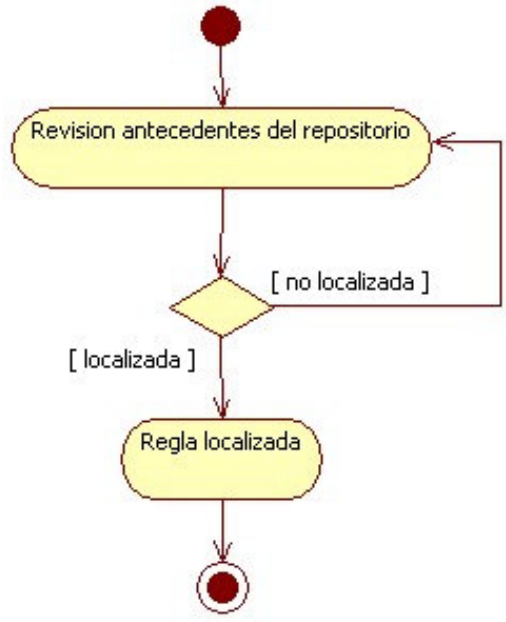

Figura 21. Diagrama de actividad de UC-5

Tabla 13. Especificación textual del caso de uso <Búsqueda por categoría>		
UC-<6>	Búsqueda por categoría	
Versión	1.0	
Descripción	El actor usuario busca en el repositorio de reglas por un criterio predefinido (atributo)	
Precondición	El actor usuario realizó completamente el caso de uso UC-1 y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor usuario elige búsqueda de regla por criterio
	P_2	El actor usuario elige la categoría deseada
	P_3	El actor usuario introduce el dato
	P_4	La regla buscada es localizada
	P_5	El caso de uso concluye
Postcondición	La regla localizada se muestra al actor usuario	
Excepciones	*Pasos*	*Acción*
	P_3	La regla buscada no es localizada. El sistema ofrece la elección de otro criterio.
	P_3	La regla buscada no es localizada. El actor usuario elige cancelar el caso de uso.
Frecuencia esperada	Frecuentemente	
Importancia	Alta	
Urgencia	Alta	
Comentarios	En todas las sesiones será usual la búsqueda de reglas por criterio	

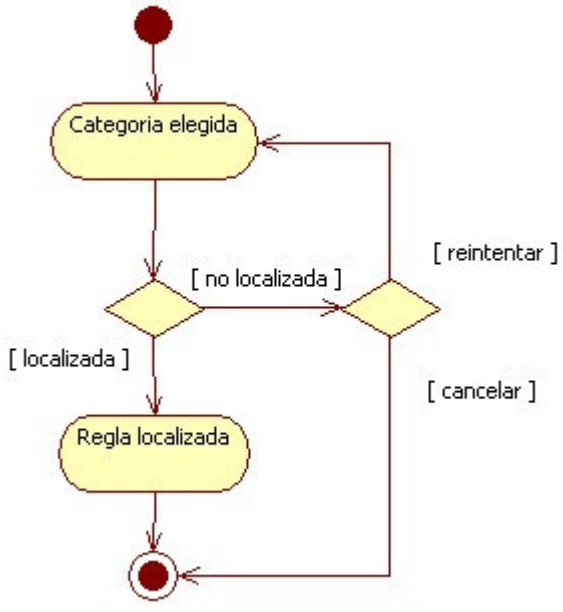

Figura 22. Diagrama de actividad de UC-6

4.6.2. Casos de uso de gestión de categorías de reglas

Tabla 14. Especificación textual del caso de uso <Búsqueda categoría regla>		
UC-<7>	Búsqueda categoría regla	
Versión	1.0	
Descripción	El actor diseñador realiza un búsqueda en el repositorio de categorías de reglas	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción de búsqueda de categoría de regla.
	P_2	Si el actor diseñador desea dar de alta una categoría y no la ha encontrado en el repositorio realiza el caso de uso UC-8
	P_3	El actor diseñador localiza la categoría en el repositorio
	P_4	Si el actor diseñador desea modificar la categoría se realiza el caso de uso UC-9
	P_5	Si el actor diseñador desea eliminar la categoría se realiza el caso de uso UC-10
	P_6	El caso de uso concluye
Postcondición	La categoría localizada se muestra al actor usuario	
Excepciones	*Pasos*	*Acción*
	P_{4-5}	El actor diseñador no localiza la categoría a modificar/eliminar. El sistema le ofrece otra búsqueda
Frecuencia esperada	Escasa	
Importancia	Baja	
Urgencia	Baja	
Comentarios	La modificación de categorías de reglas se realizará esporádicamente y usualmente justo antes de la puesta en producción del sistema	

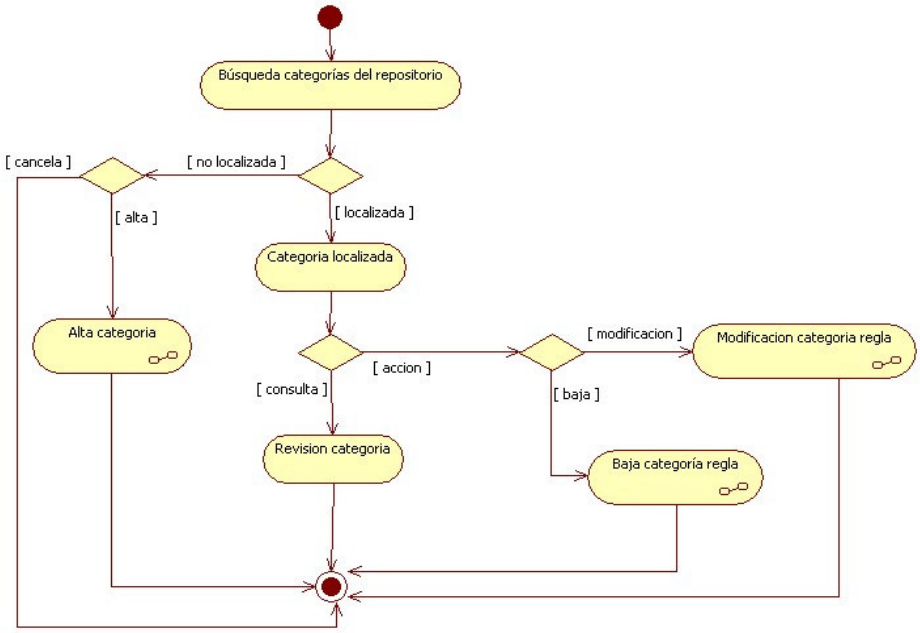

Figura 23. Diagrama de actividad de UC-7

Tabla 15. Especificación textual del caso de uso <Alta categoría regla>		
UC-<8>	Alta categoría regla	
Versión	1.0	
Descripción	El actor diseñador realiza un alta en el repositorio de categorías de reglas	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito. La categoría no existe en el repositorio	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción alta de categoría de regla.
	P_2	El actor diseñador parametriza la categoría deseada
	P_3	El actor diseñador confirma la categoría y ésta se introduce en el repositorio
	P_4	El caso de uso concluye
Postcondición	La nueva categoría se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_3	El actor diseñador cancela el alta
Frecuencia esperada	Escasa	
Importancia	Media	
Urgencia	Media	
Comentarios	El alta de categorías de reglas se realizará esporádicamente y usualmente justo antes de la puesta en producción del sistema	

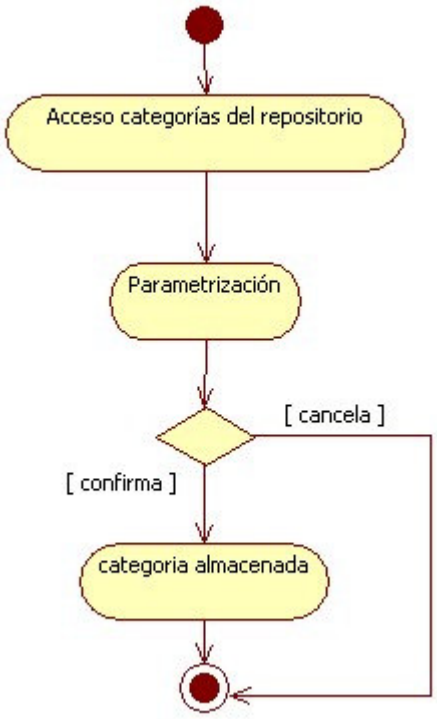

Figura 24. Diagrama de actividad de UC-8

Tabla 16. Especificación textual del caso de uso <Modificación categoría regla>		
UC-<9>	Modificación categoría regla	
Versión	1.0	
Descripción	El actor diseñador modifica una categoría en el repositorio	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito. La categoría existe en el repositorio	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción modificación de categoría de regla.
	P_2	El actor diseñador modifica los parámetros deseados de la categoría
	P_3	El actor diseñador confirma los cambios en la categoría y ésta se actualiza en el repositorio
	P_4	El caso de uso concluye
Postcondición	La categoría actualizada se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_3	El actor diseñador cancela la modificación
Frecuencia esperada	Escasa	
Importancia	Media	
Urgencia	Baja	
Comentarios	El alta de categorías de reglas se realizará esporádicamente y usualmente justo antes de la puesta en producción del sistema	

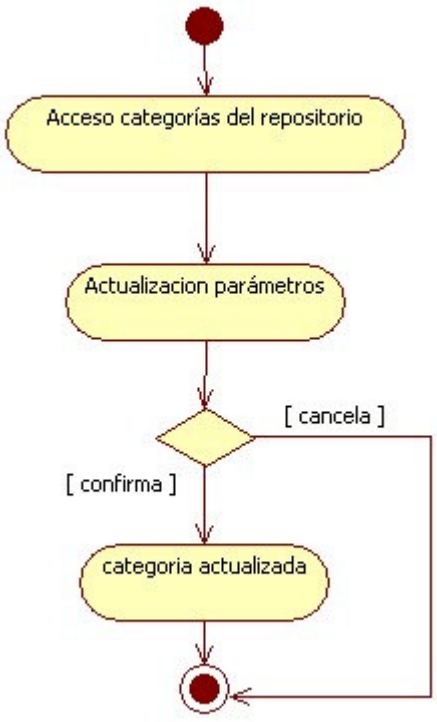

Figura 25. Diagrama de actividad de UC-9

Tabla 17. Especificación textual del caso de uso <Baja categoría regla>		
UC-<10>	Baja categoría regla	
Versión	1.0	
Descripción	El actor diseñador elimina una categoría del repositorio	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito. La categoría existe en el repositorio	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción Baja de categoría de regla.
	P_2	El actor diseñador confirma la baja de la categoría y ésta se elimina del repositorio
	P_3	El caso de uso concluye
Postcondición	La categoría eliminada no se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_2	El actor diseñador cancela la eliminación
Frecuencia esperada	Escasa	
Importancia	Media	
Urgencia	Media	
Comentarios	El alta de categorías de reglas se realizará esporádicamente y usualmente justo antes de la puesta en producción del sistema	

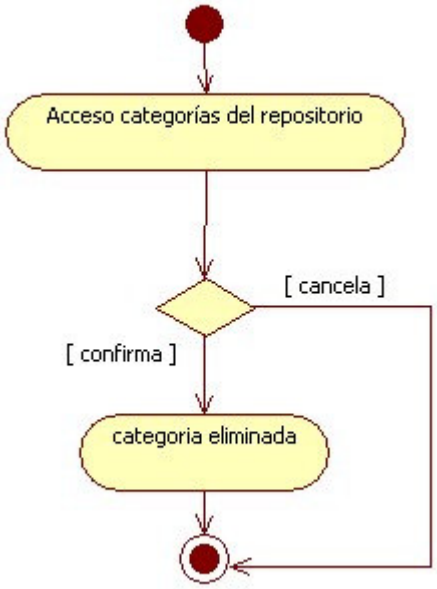

Figura 26. Diagrama de actividad de UC-10

4.6.3. Casos de uso de gestión de reglas

Tabla 18. Especificación textual del caso de uso <Búsqueda regla>		
UC-<11>	Búsqueda regla	
Versión	1.0	
Descripción	El actor diseñador realiza un búsqueda en el repositorio de reglas	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción de búsqueda de reglas.
	P_2	Si el actor diseñador desea dar de alta una regla y no la ha encontrado en el repositorio realiza el caso de uso UC-12
	P_3	El actor diseñador localiza la regla en el repositorio
	P_4	Si el actor diseñador desea modificar la regla se realiza el caso de uso UC-13
	P_5	Si el actor diseñador desea eliminar la regla se realiza el caso de uso UC-14
	P_6	El caso de uso concluye
Postcondición	La regla localizada se muestra al actor usuario	
Excepciones	*Pasos*	*Acción*
	P_{4-5}	El actor diseñador no localiza la regla a modificar/eliminar. El sistema le ofrece otra búsqueda
Frecuencia esperada	Frecuentemente	
Importancia	Alta	
Urgencia	Alta	
Comentarios	La búsqueda de reglas se realizará frecuentemente durante toda la vida útil del motor	

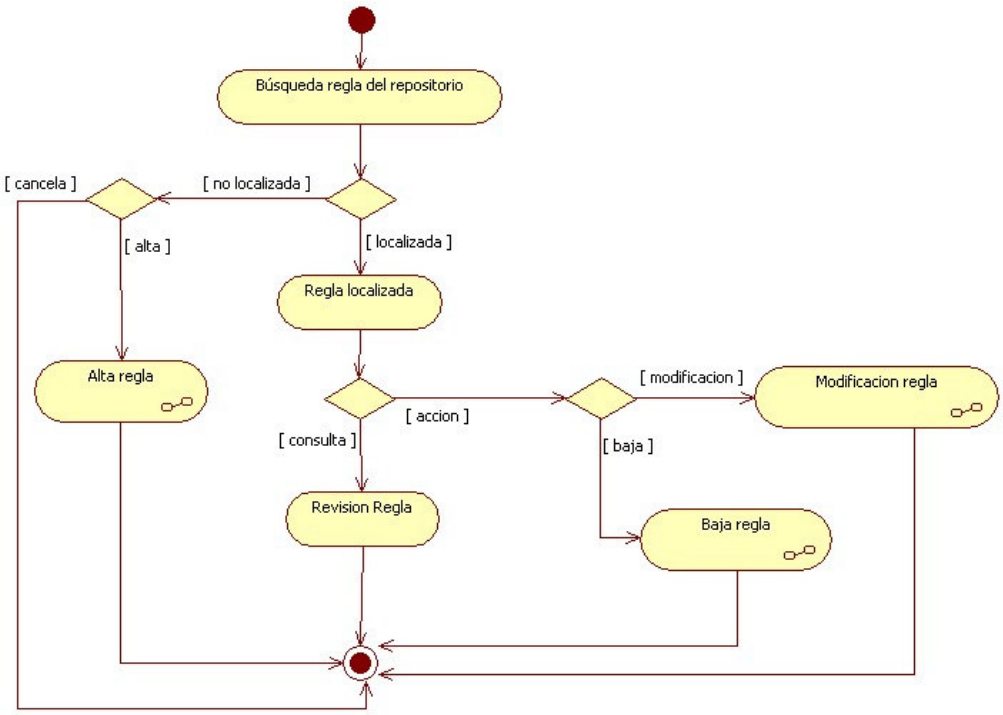

Figura 27. Diagrama de actividad de UC-11

Tabla 19. Especificación textual del caso de uso <Alta regla>		
UC-<12>	Alta regla	
Versión	1.0	
Descripción	El actor diseñador realiza un alta en el repositorio de reglas	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción alta de regla.
	P_2	El actor diseñador parametriza la regla deseada
	P_3	Si el actor diseñador desea relacionar la regla se realiza UC-15
	P_4	El actor diseñador confirma la regla y ésta se introduce en el repositorio
	P_5	El sistema realiza el UC-16
	P_6	El caso de uso concluye
Postcondición	La nueva regla se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_2	La regla existe en el repositorio. El sistema muestra un mensaje de error y ofrece la entrada de una nueva regla
	P_4	El actor diseñador cancela el alta
Frecuencia esperada	Media	
Importancia	Alta	
Urgencia	Alta	
Comentarios	El alta de reglas se realizará a menudo	

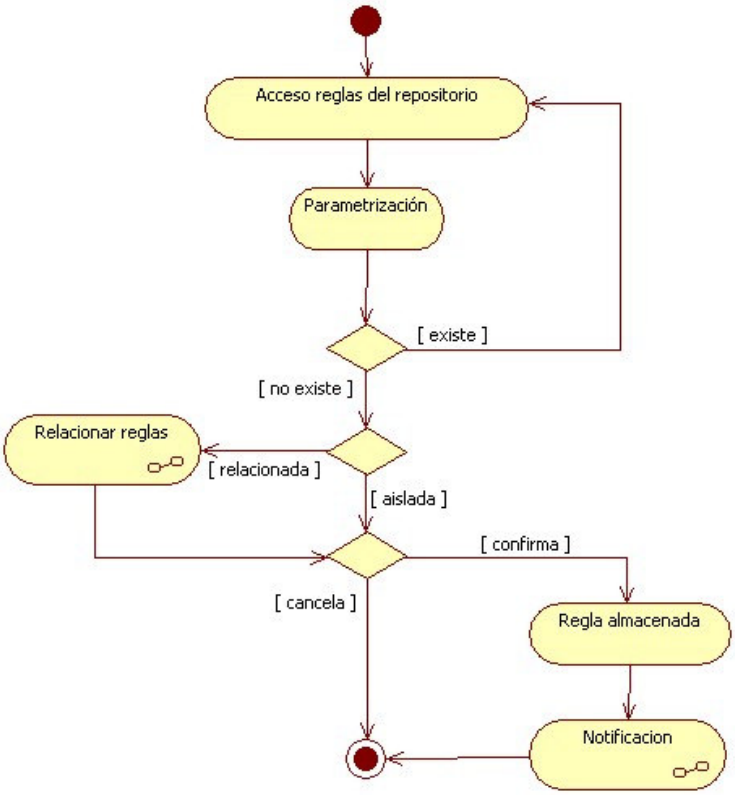

Figura 28. Diagrama de actividad de UC-12

Tabla 20. Especificación textual del caso de uso <Modificación regla>		
UC-<13>	Modificación regla	
Versión	1.0	
Descripción	El actor diseñador realiza una modificación de una regla	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito. La regla existe en el repositorio	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción modificación de regla.
	P_2	El actor diseñador actualiza la regla deseada
	P_3	Si el actor diseñador desea relacionar la regla se realiza UC-15
	P_4	El actor diseñador confirma la actualización de regla y ésta se introduce en el repositorio
	P_5	El sistema realiza el UC-16
	P_6	El caso de uso concluye
Postcondición	La regla actualizada se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_4	El actor diseñador cancela la actualización
Frecuencia esperada	Baja	
Importancia	Media	
Urgencia	Media	
Comentarios	La modificación de reglas se realizará de forma esporádica. Su frecuencia se reducirá cuando se estabilice el contenido del repositorio de reglas	

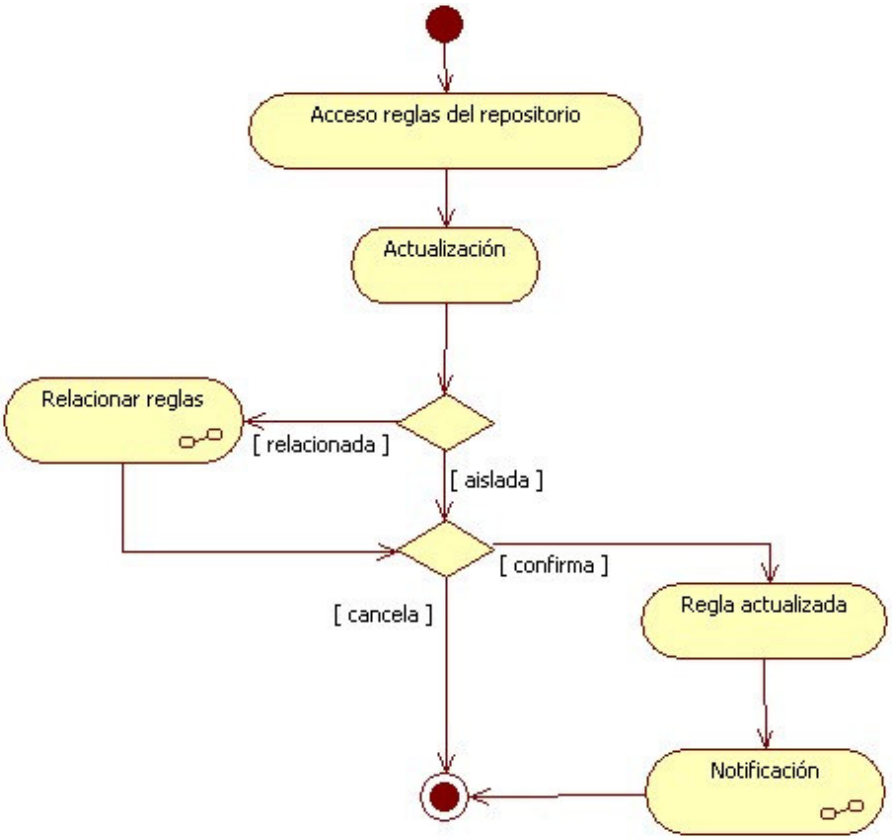

Figura 29. Diagrama de actividad de UC-13

Tabla 21. Especificación textual del caso de uso <Baja regla>		
UC-<14>	Baja regla	
Versión	1.0	
Descripción	El actor diseñador realiza una baja de una regla	
Precondición	El actor diseñador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito. La regla existe en el repositorio	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción baja de regla.
	P_2	El actor diseñador confirma la baja de la regla y ésta se elimina del repositorio
	P_3	El sistema realiza el UC-16
	P_4	El caso de uso concluye
Postcondición	La regla ha sido eliminada del repositorio	
Excepciones	*Pasos*	*Acción*
	P_4	El actor diseñador cancela la baja
Frecuencia esperada	Baja	
Importancia	Alta	
Urgencia	Alta	
Comentarios	La baja de reglas se realizará de forma esporádica, aunque resulta de gran importancia, ya que sería grave mantener en el repositorio reglas que no estén en vigor. Su frecuencia se reducirá aún más cuando se estabilice el contenido del repositorio de reglas	

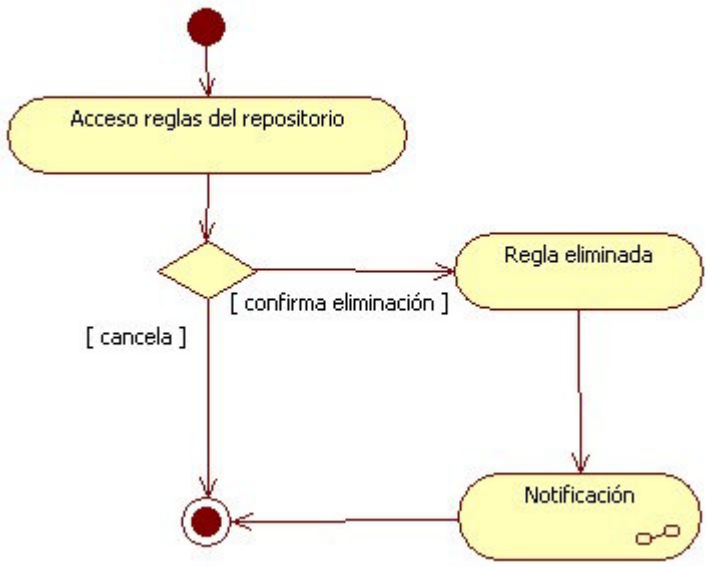

Figura 30. Diagrama de actividad del UC-14

Tabla 22. Especificación textual del caso de uso <Relacionar regla>		
UC-<15>	Relacionar regla	
Versión	1.0	
Descripción	El actor diseñador relaciona una regla con otra/s	
Precondición	El actor diseñador realiza el caso de uso UC-12 o el UC-13.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor diseñador elige la opción relacionar regla.
	P_2	El actor diseñador elige la regla con la que relacionar la actual regla.
	P_3	El actor diseñador confirma la relación y ésta se incluye en el repositorio
	P_4	El caso de uso concluye
Postcondición	La nueva relación se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_2	El actor diseñador elige incorporar más relaciones
	P_3	El actor diseñador cancela la relación
Frecuencia esperada	Media	
Importancia	Alta	
Urgencia	Media	
Comentarios	La relación de reglas se realizará usualmente en el alta de reglas. Su frecuencia se reducirá aún más cuando se estabilice el contenido del repositorio de reglas	

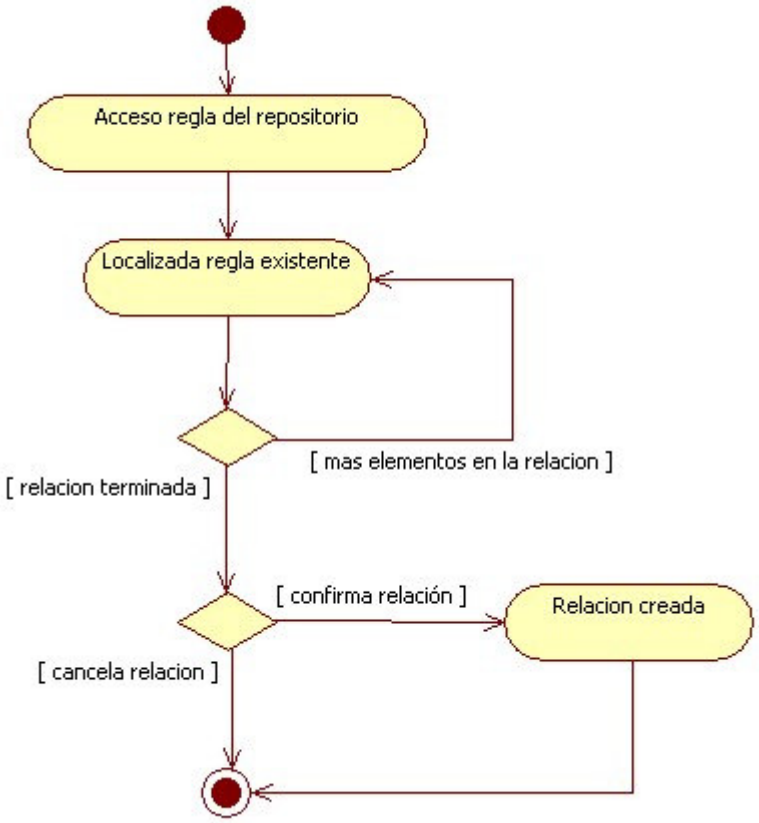

Figura 31. Diagrama de actividad de UC-15

Tabla 23. Especificación textual del caso de uso <Notificación>		
UC-<16>	Notificación	
Versión	1.0	
Descripción	El sistema notifica sobre un evento acaecido	
Precondición	El actor diseñador realiza el caso de uso UC-12, UC-13 o UC-14.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El sistema detecta un evento confirmado que requiere notificación
	P_2	El sistema envía la notificación
	P_3	El caso de uso concluye
Postcondición	Se han enviado todas las notificaciones	
Excepciones	*Pasos*	*Acción*
	P_2	El sistema no encuentra destinatarios y concluye el caso de uso
Frecuencia esperada	Media	
Importancia	Baja	
Urgencia	Baja	
Comentarios	La notificación de modificaciones se realizará usualmente en el alta de reglas. Su frecuencia se reducirá aún más cuando se estabilice el contenido de los repositorios	

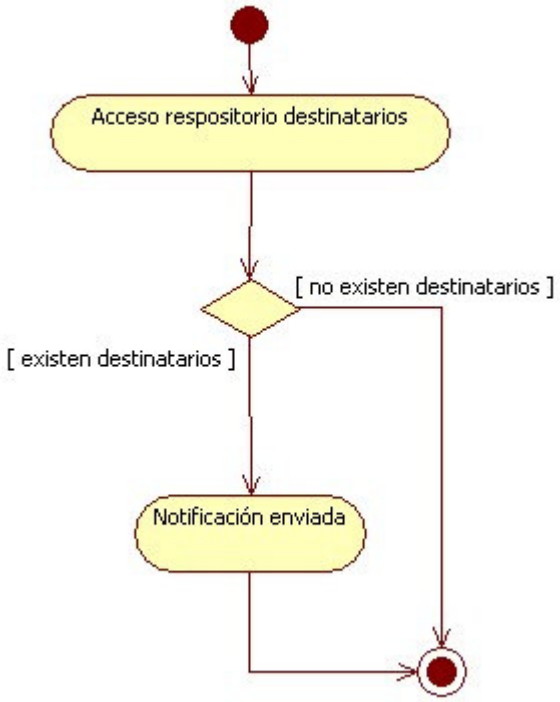

Figura 32. Diagrama de actividad de UC-16

4.6.4. Casos de uso de gestión de usuarios

Tabla 24. Especificación textual del caso de uso <Búsqueda usuario>		
UC-<17>	Búsqueda usuario	
Versión	1.0	
Descripción	El actor administrador realiza un búsqueda en el repositorio de usuarios	
Precondición	El actor administrador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor administrador elige la opción de búsqueda de usuario.
	P_2	El actor administrador elige el criterio de búsqueda
	P_3	Si el actor administrador no encuentra el usuario buscado y desea darlo de alta se realiza el caso de uso UC-18
	P_4	El actor administrador localiza el usuario en el repositorio
	P_5	Si el actor administrador desea modificar el rol del usuario se realiza el caso de uso UC-19
	P_6	Si el actor administrador desea eliminar el usuario se realiza el caso de uso UC-20
	P_7	El caso de uso concluye
Postcondición	El usuario localizado se muestra al actor administrador	
Excepciones	*Pasos*	*Acción*
	P_2	El actor administrador cancela la búsqueda
Frecuencia esperada	Esporádicamente	
Importancia	Media	
Urgencia	Media	
Comentarios	La búsqueda de usuarios se realizará a menudo durante toda la vida útil del motor	

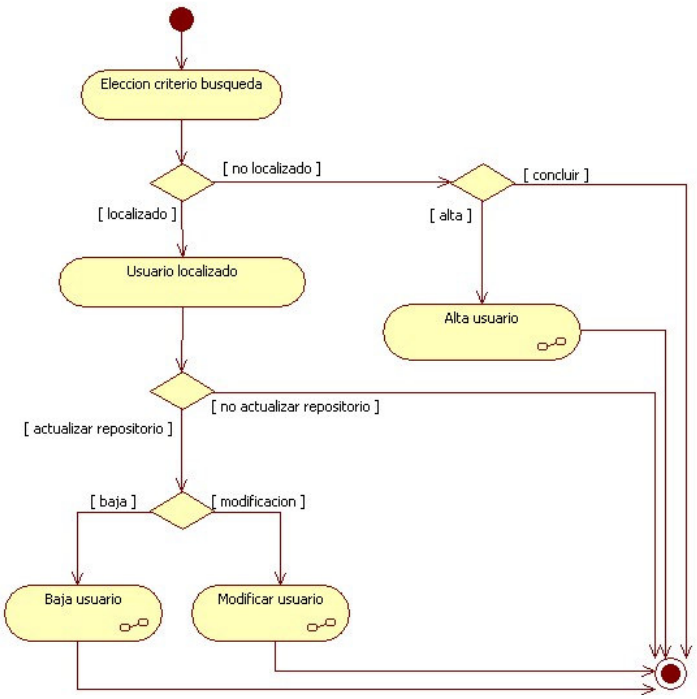

Figura 33. Diagrama de actividad de UC-17

Tabla 25. Especificación textual del caso de uso <Alta usuario>		
UC-<18>	Alta usuario	
Versión	1.0	
Descripción	El actor administrador realiza un alta en el repositorio de usuarios	
Precondición	El actor administrador realizó completamente el caso de uso UC-1 (heredado del actor usuario) y lo concluyó con éxito.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor administrador elige la opción alta de regla.
	P_2	El actor diseñador parametriza el usuario
	P_3	El actor diseñador confirma el usuario y éste se introduce en el repositorio
	P_4	El caso de uso concluye
Postcondición	El nuevo usuario se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_3	El usuario existe en el repositorio. El sistema muestra un mensaje de error y ofrece la entrada de una nueva regla
	P_3	El actor diseñador cancela el alta
Frecuencia esperada	Media	
Importancia	Alta	
Urgencia	Alta	
Comentarios	El alta de usuario se realizará sobre todo justo antes de la puesta en producción del motor	

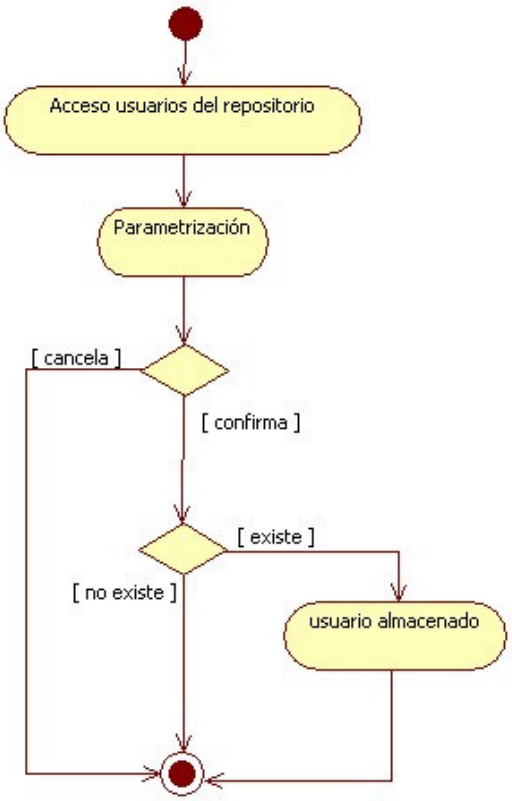

Figura 34. Diagrama de actividad de UC-18

Tabla 26. Especificación textual del caso de uso <Modificación usuario>		
UC-<19>	Modificación usuario	
Versión	1.0	
Descripción	El actor administrador realiza una modificación del rol de un usuario	
Precondición	El actor administrador se encuentra realizando el UC-17. El usuario existe en el repositorio.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor administrador elige la opción modificación de usuario.
	P_2	El actor administrador actualiza el rol del usuario
	P_3	El actor administrador confirma la actualización del rol regla y éste se introduce en el repositorio
	P_4	El caso de uso concluye
Postcondición	El usuario actualizado se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_3	El actor administrador cancela la actualización
Frecuencia esperada	Baja	
Importancia	Media	
Urgencia	Baja	
Comentarios	La modificación de roles se realizará de forma esporádica.	

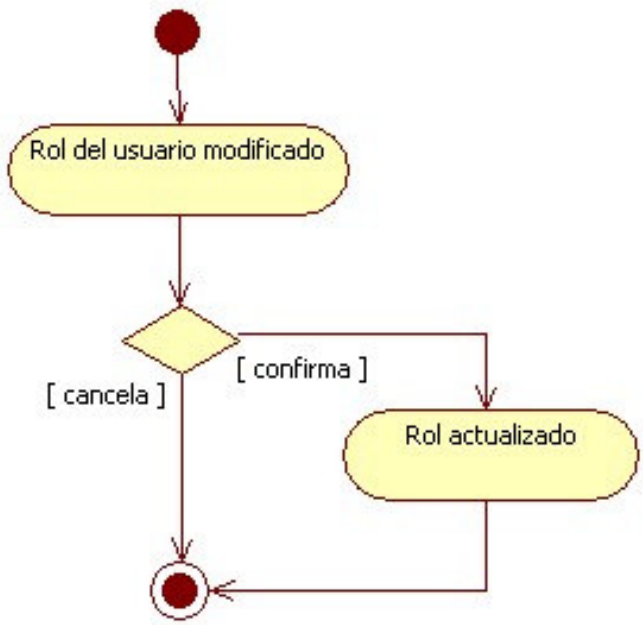

Figura 35. Diagrama de actividad de UC-19

Tabla 27. Especificación textual del caso de uso <Baja usuario>		
UC-<20>	Baja usuario	
Versión	1.0	
Descripción	El actor administrador realiza una baja de un usuario	
Precondición	El actor administrador se encuentra realizando el UC-17. El usuario existe en el repositorio.	
Secuencia normal	*Pasos*	*Acción*
	P_1	El actor administrador elige la opción baja de usuario.
	P_2	El actor administrador confirma la baja del usuario y éste se elimina del repositorio
	P_4	El caso de uso concluye
Postcondición	La regla eliminada no se encuentra en el repositorio	
Excepciones	*Pasos*	*Acción*
	P_4	El actor administrador cancela la baja
Frecuencia esperada	Baja	
Importancia	Alta	
Urgencia	Media	
Comentarios	La baja de usuario se realizará con menor frecuencia que su alta, aunque resulta importante, ya que usuarios no autorizados no deben acceder al motor.	

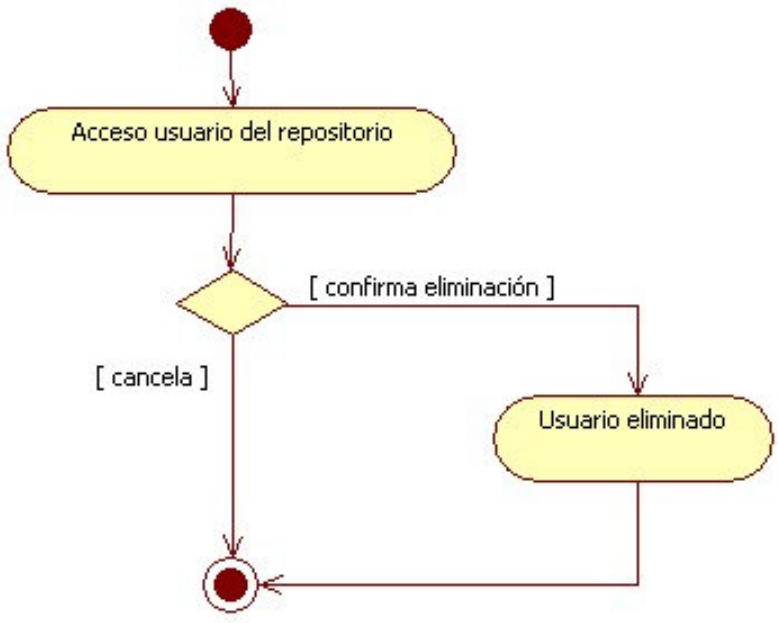

Figura 36. Diagrama de actividad de UC-20

4.6.5. Resumen de los casos de uso

Tabla 28. Resumen de los casos de uso

Paquete	Subpaquete	Actor	<ID>	Denominación	Funcionalidad	Urgencia	Relación
Núcleo		Usuario	UC-1	Identificación	Acceso	Alta	
			UC-2	Cambio de contraseña		Media	Extends UC-1
			UC-3	Cierre sesión		Alta	
			UC-4	Búsqueda regla	Proceso reglas	Alta	
			UC-5	Búsqueda antecedente		Alta	Extends UC-4
			UC-6	Búsqueda por criterio		Alta	Extends UC-4
Gestión del motor	Gestión de categorías de reglas	Diseñador	UC-7	Búsqueda categoría regla	Categorización de reglas	Baja	
			UC-8	Alta categoría regla		Media	Extends UC-7
			UC-9	Modificación categoría regla		Baja	Extends UC-7
			UC-10	Baja categoría regla		Media	Extends UC-7
	Gestión de reglas		UC-11	Búsqueda regla	Administración de reglas	Alta	
			UC-12	Alta regla		Alta	Include UC-16
			UC-13	Modificación regla		Media	Include UC-16
			UC-14	Baja regla		Alta	Include UC-16
			UC-15	Relación reglas	Utilidades de reglas	Media	Extends UC-12 Extends UC-13
			UC-16	Notificación		Baja	
	Gestión de usuarios	Administrador	UC-17	Búsqueda usuario	Búsqueda	Media	
			UC-18	Alta usuario	Administración de usuario	Alta	Extends UC-17
			UC-19	Modificación usuario		Baja	Extends UC-17
			UC-20	Baja usuario		Media	Extends UC-17

4.7. Entidades

Las clases de entidades consisten en unas primeras clases mediante las que se pueda especificar los casos de uso en forma de interacciones. Muchas de estas clases pasarían al diseño y a la implementación, etapas en las que se definirán muchas más clases, cuyo papel será de implementación, y no conceptual.

Se pretenden localizar clases en el sentido habitual de conjuntos de objetos homogéneos que ostentan unos mismos datos (atributos) y comportamiento (operaciones). Por otro lado, a diferencia del modelo del negocio, interesan las clases del sistema y no del entorno.

En este apartado nos basaremos en [CAM04] para la confección del diagrama de entidades.

4.7.1. Identificación de las clases de entidades

La identificación de las clases de entidades se realizará a partir de los casos de uso. Para cada caso de uso se indican las clases que se encuentran. En el caso de que una clase ya se había identificado en un caso de uso anterior, su denominación incorporará un asterisco, mientras que las clases que resulten dudosas irán seguidas de un interrogante. De igual forma, se ha incorporado una columna adicional a la propuesta por [CAM04] para recoger los repositorios.

Tabla 29. Identificación de clases de entidades

UC-<id>	Denominación Caso de Uso	Clases	Repositorios
UC-1	Identificación	• Usuario • Sesión	• Usuarios
UC-2	Cambio de contraseña	• Usuario* • Sesión*	
UC-3	Cierre sesión	• Usuario* • Sesión*	
UC-4	Búsqueda regla	• Usuario* • Regla • Antecedente • Categoría • Consecuente	• Regla • Antecedentes • Consecuentes • Categorías
UC-5	Búsqueda antecedente	• Usuario* • Antecedente* • Regla*	• Usuarios* • Antecedentes* • Consecuentes*
UC-6	Búsqueda por criterio	• Usuario*	• Regla* • Usuarios*
UC-7	Búsqueda categoría regla	• Categoría* • Diseñador	• Categorías*
UC-8	Alta categoría regla	• Categoría* • Diseñador*	
UC-9	Modificación categoría regla	• Categoría* • Diseñador*	
UC-10	Baja categoría regla	• Categoría* • Diseñador*	
UC-11	Búsqueda regla	• Regla*	• Regla*

		• Diseñador*	
UC-12	Alta regla	• Regla* • Diseñador*	
UC-13	Modificación regla	• Regla* • Diseñador*	
UC-14	Baja regla	• Regla* • Diseñador*	
UC-15	Relacionar regla	• Regla* • Diseñador*	
UC-16	Notificación	• Regla* • Diseñador* • EventoDisparador	
UC-17	Búsqueda usuario	• Administrador • Usuario*	• Usuarios*
UC-18	Alta usuario	• Administrador* • Usuario*	
UC-19	Modificación usuario	• Administrador* • Usuario*	
UC-20	Baja usuario	• Administrador* • Usuario*	

4.7.2. Especificación de los atributos de las clases de entidades

Se ha elegido una especificación de atributos simplificada de los atributos de las clases de entidad, ya que podrían tener muchos atributos que no aportarían nada al presente TFC y tan sólo contribuirían a dificultar la comprensión de los diagramas. Por ejemplo la entidad usuario podría tener como atributos la dirección, teléfono, proyectos en los que ha participado y consideraciones laborales, entre otras.

Tabla 30. Especificación de los atributos de las clases de entidades

Clase de entidad	Atributos
Usuario	• DNI (string) • Nombre (string) • Departamento (string) • email (string)
Diseñador Administrador	(no dispondrá de atributos diferentes de su superclase, la especialización se materializará en las operaciones)
Sesión	• Id (string) • Inicio (date) • Finalización (date) • idUsuario (string)
Regla	• Id (string) • Denominación (string) • idAntecedente (string) • idConsecuente (string) • idCategoría (string) • FechaCreación (date) • idProductoDeReferencia (string) • idDepartamentoResponsable (string) • Importancia (string) • idDestinatarios (string)

	• Observaciones (string)
Antecedente	• Hecho (string) • idRegla (string)
Categoría	• Denominación (string)
Consecuente	• Hecho (string) • idRegla (string)
EventoDisparador	• Hecho (string)
Departamento	• idDepartamento
Producto	• idProducto

4.7.3. Relaciones entre clases de entidad

En la Figura 37 se recogen, tanto las relaciones de herencia como las asociaciones entre las diferentes clases de entidad que hemos contemplado.

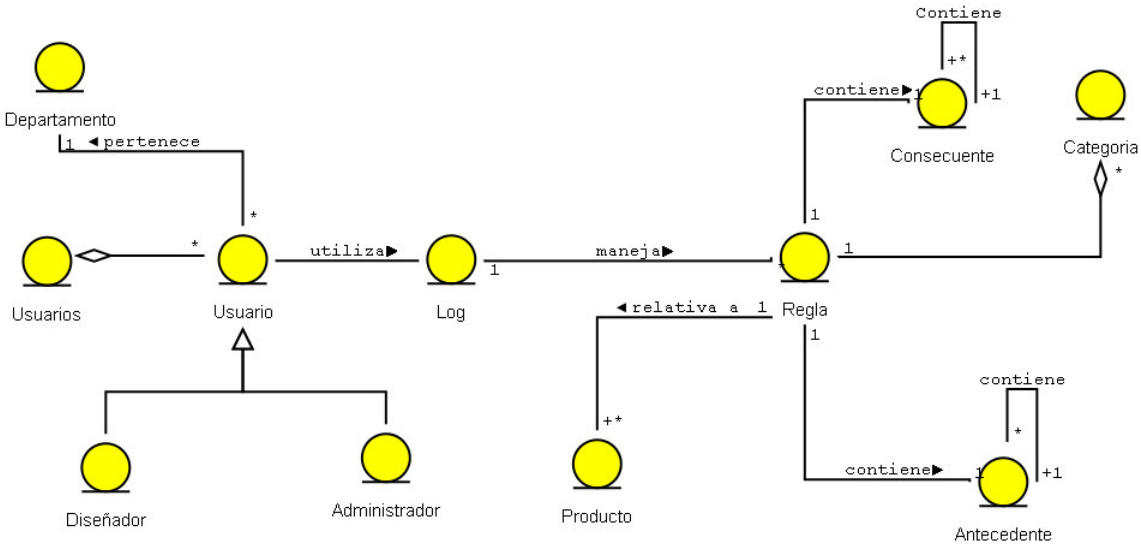

Figura 37. Diagrama de clases de entidad

4.7.4. Asociación recursiva de la clase de entidad consecuente

Esta relación modela la composición de consecuentes. Cualquier consecuente puede estar compuesto por un número indeterminado de elementos hoja, lo que justifica la multiplicidad.

4.7.5. Asociación recursiva de la clase de entidad antecedente

Esta relación modela la composición de antecedentes. Cualquier antecedente puede estar compuesto por un número indeterminado de elementos hoja, lo que justifica la multiplicidad.

4.8. Realizaciones de los casos de uso

Seguidamente se especifican los diagramas de secuencia y colaboración para los diferentes casos de uso del motor de reglas. En cualquiera de los diagramas podemos encontrarnos con que el acceso a datos persistentes no se encuentre disponible devolviendo un error. En tal caso cualquiera de los diagramas terminaría con mostrando el error correspondiente.

En muchas ocasiones se han utilizado algunos gestores, interfaces y entidades genéricos que serán detallados en fase de diseño.

4.8.1. Paquete Núcleo

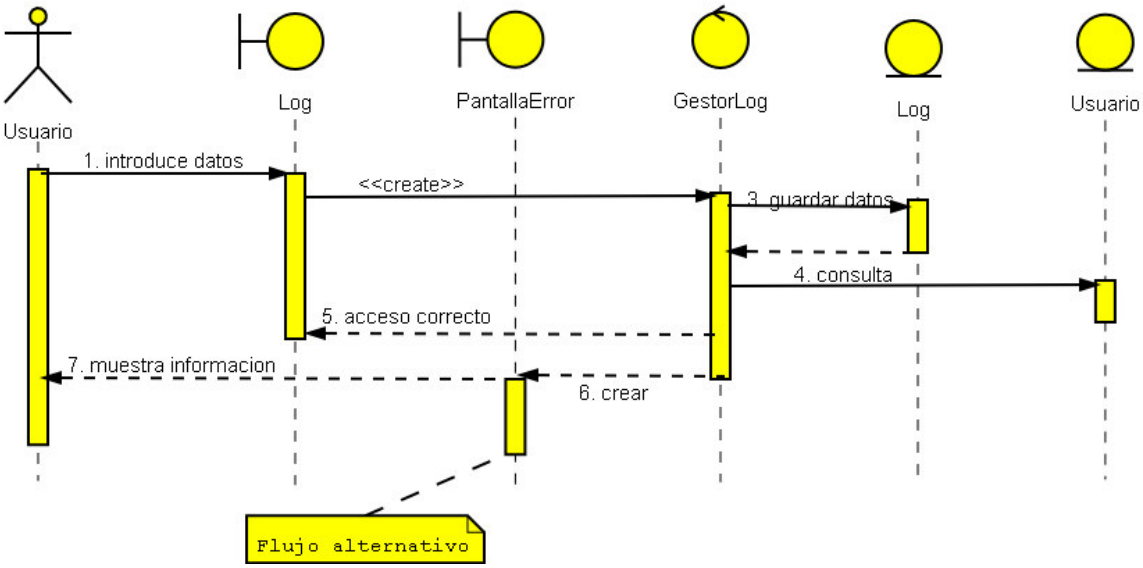

Figura 38. Diagrama de secuencia del modelo de análisis de UC-1

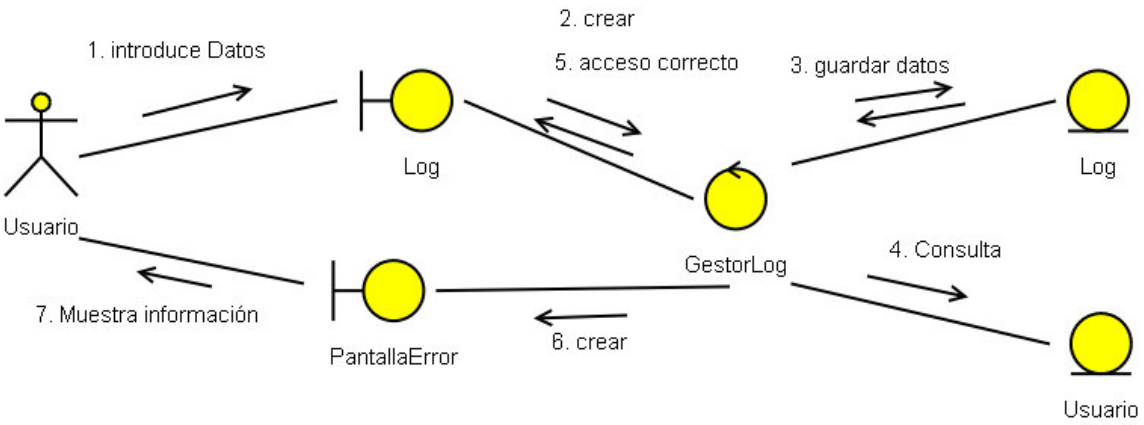

Figura 39. Diagrama de colaboración del modelo de análisis de UC-1

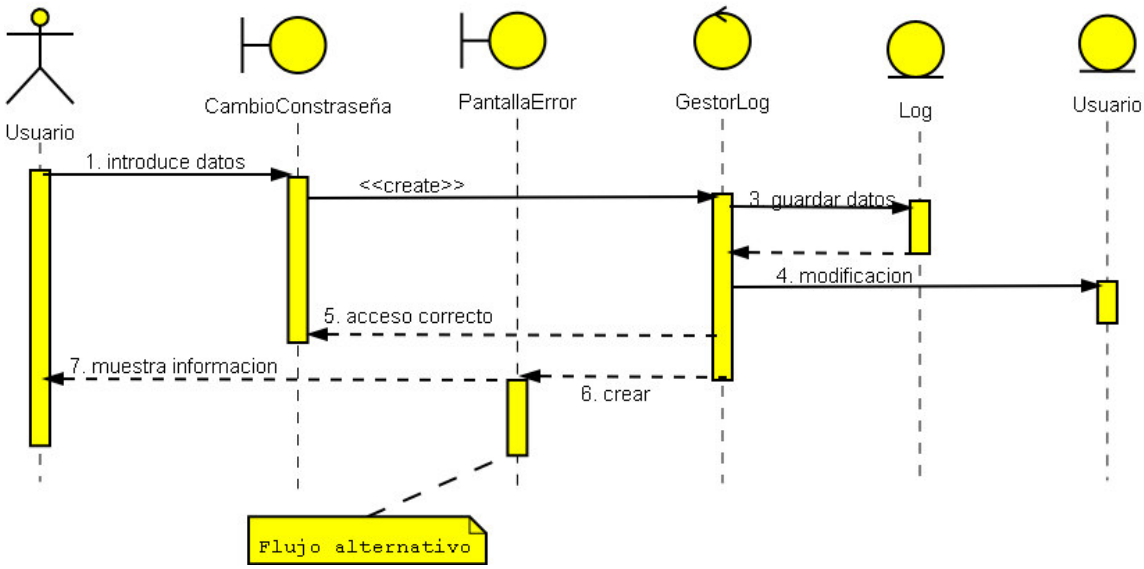

Figura 40. Diagrama de secuencia del modelo de análisis de UC-2

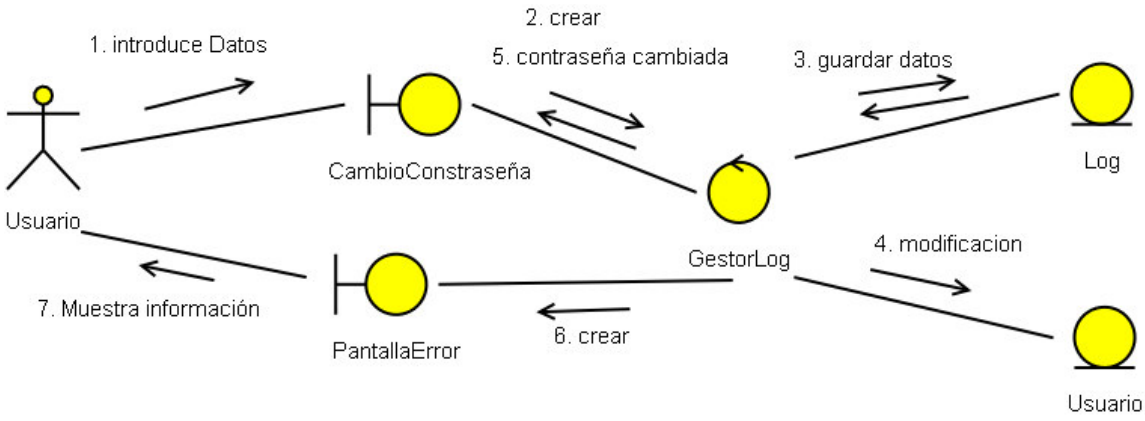

Figura 41. Diagrama de colaboración del modelo de análisis de UC-2

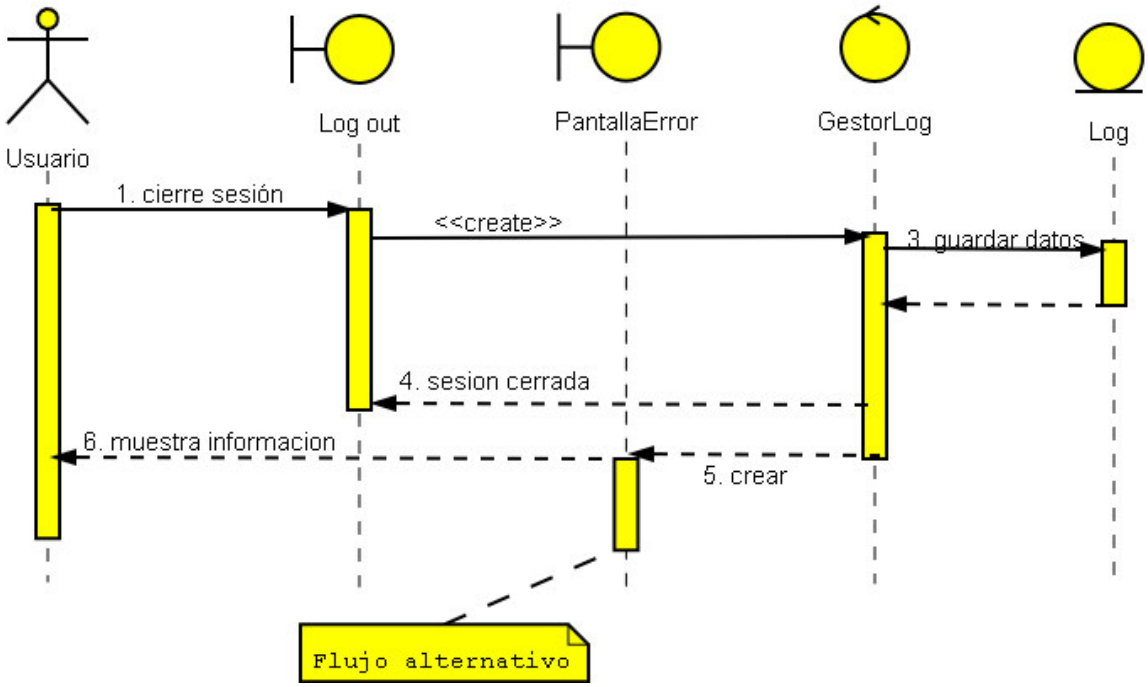

Figura 42. Diagrama de secuencia del modelo de análisis de UC-3

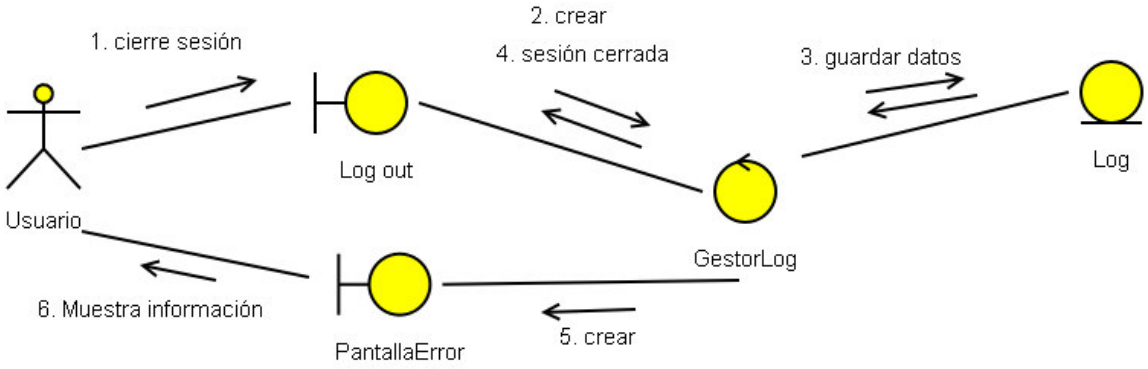

Figura 43. Diagrama de colaboración del modelo de análisis de UC-3

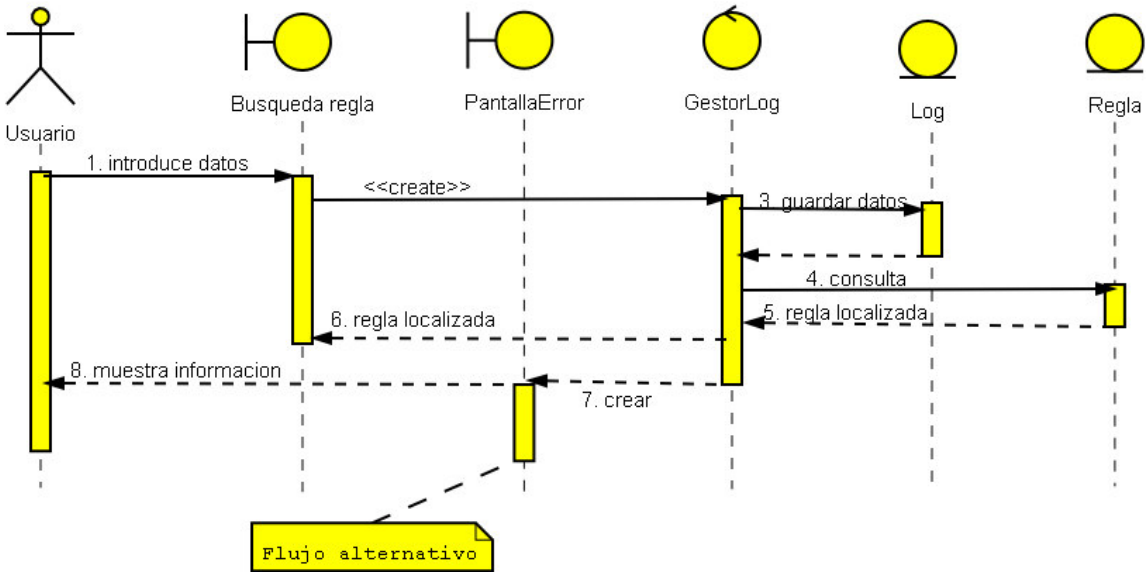

Figura 44. Diagrama de secuencia del modelo de análisis de UC-4

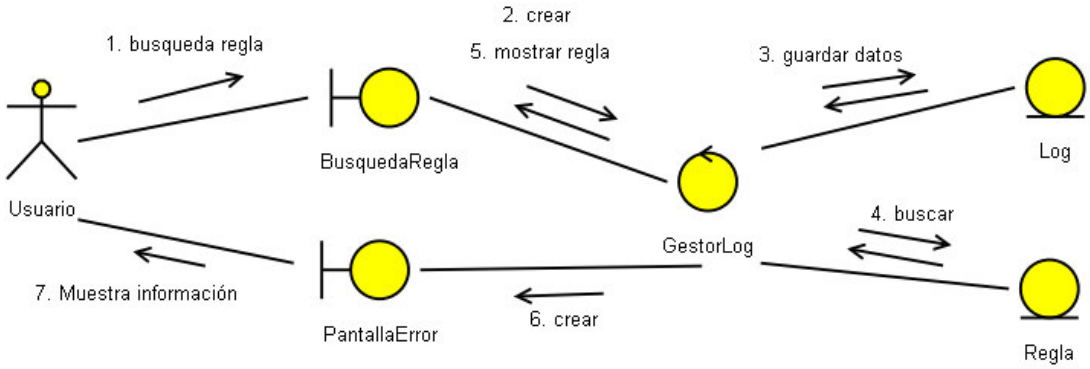

Figura 45. Diagrama de colaboración del modelo de análisis de UC-4

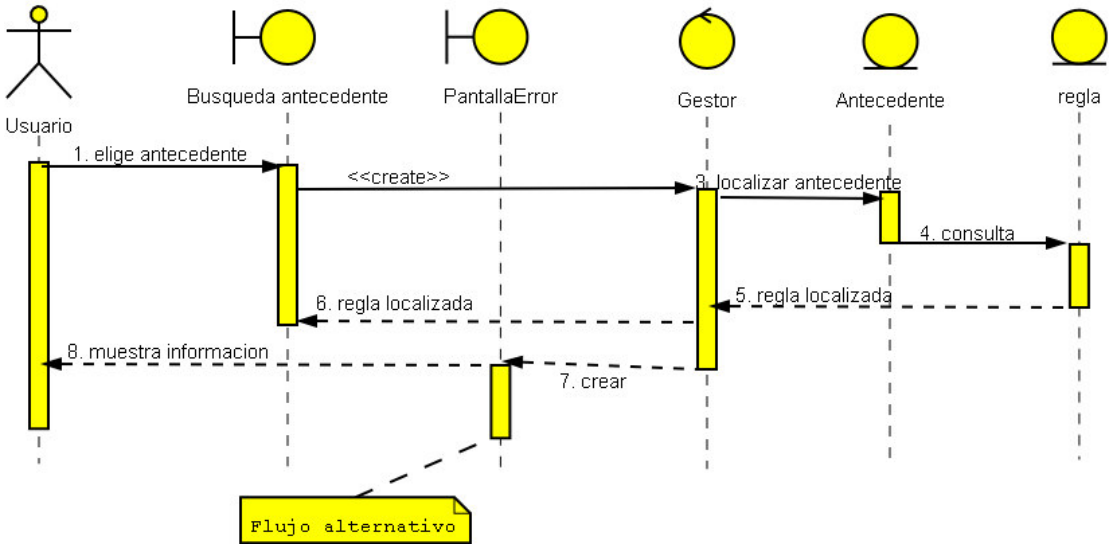

Figura 46. Diagrama de secuencia del modelo de análisis de UC-5

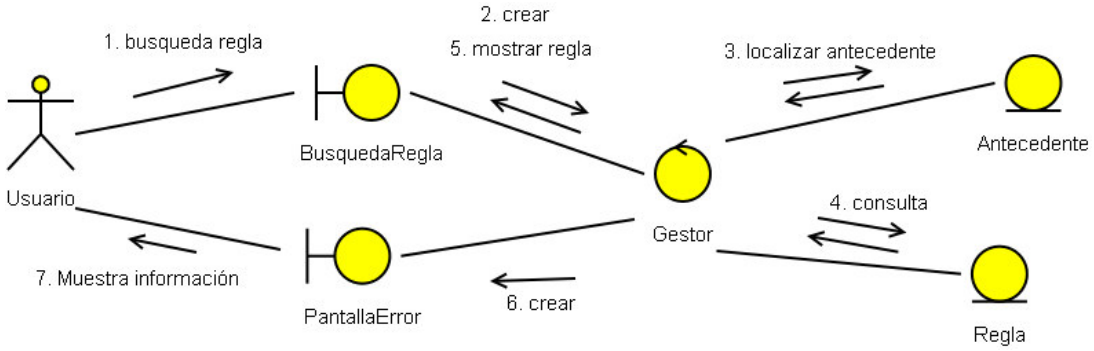

Figura 47. Diagrama de colaboración del modelo de análisis de UC-5

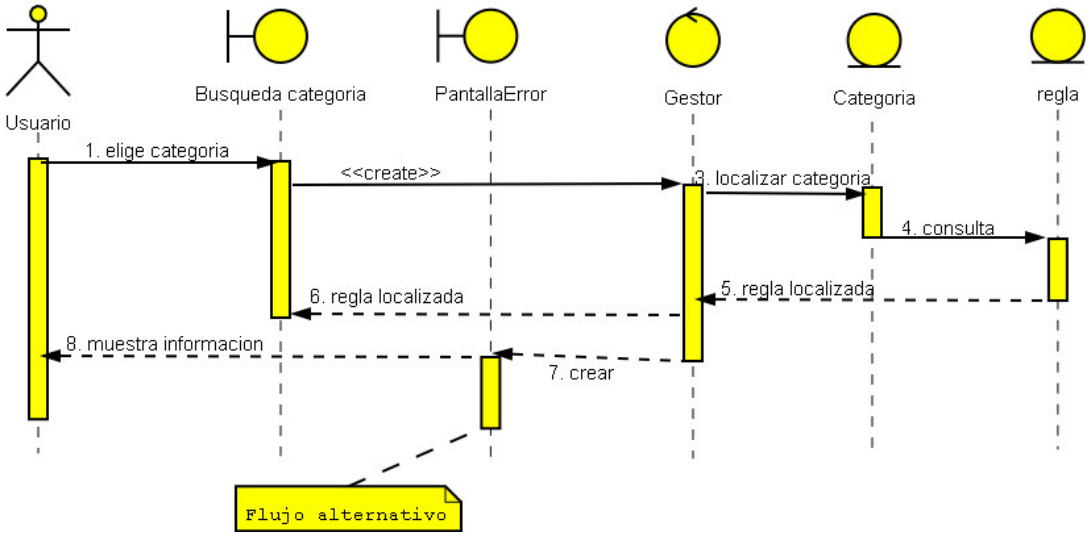

Figura 48. Diagrama de secuencia del modelo de análisis de UC-6

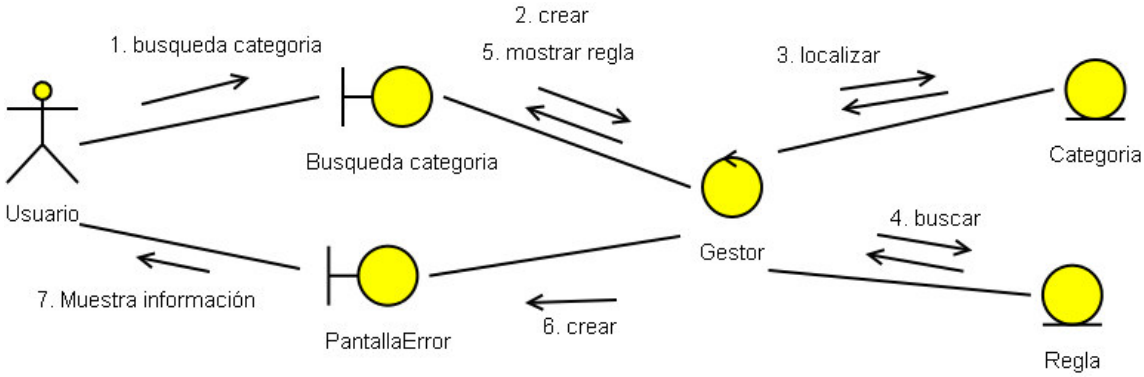

Figura 49. Diagrama de colaboración del modelo de análisis de UC-6

4.8.2. Paquete de gestión de categorías de reglas

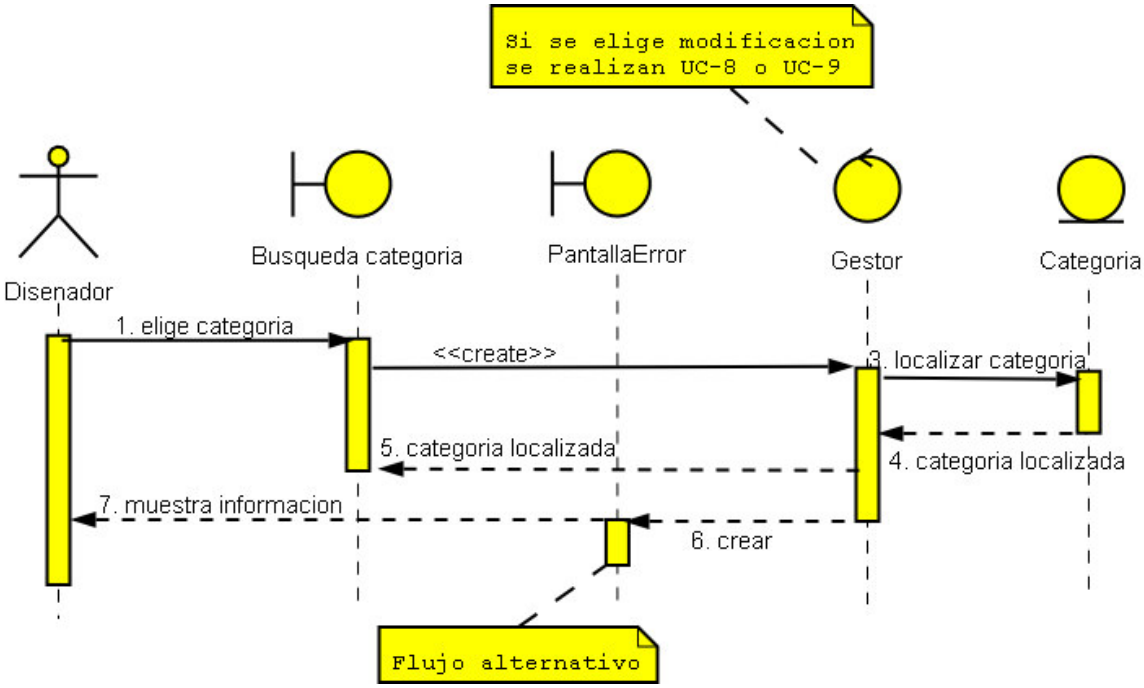

Figura 50. Diagrama de secuencia del modelo de análisis de UC-7

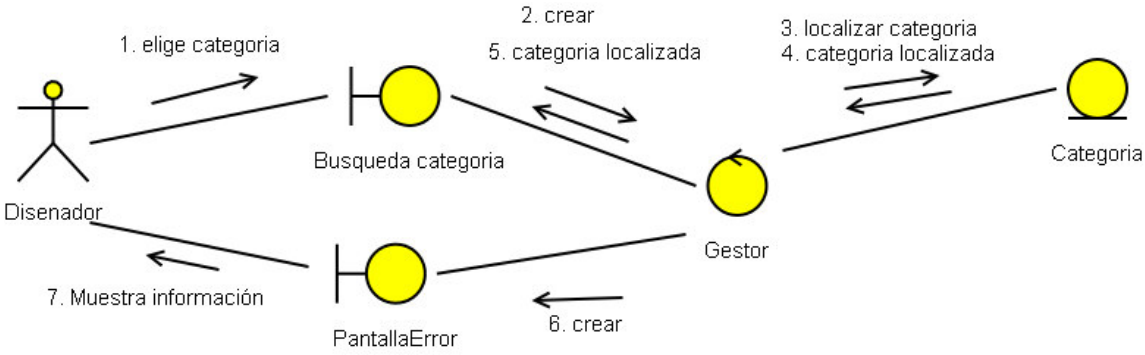

Figura 51. Diagrama de colaboración del modelo de análisis de UC-7

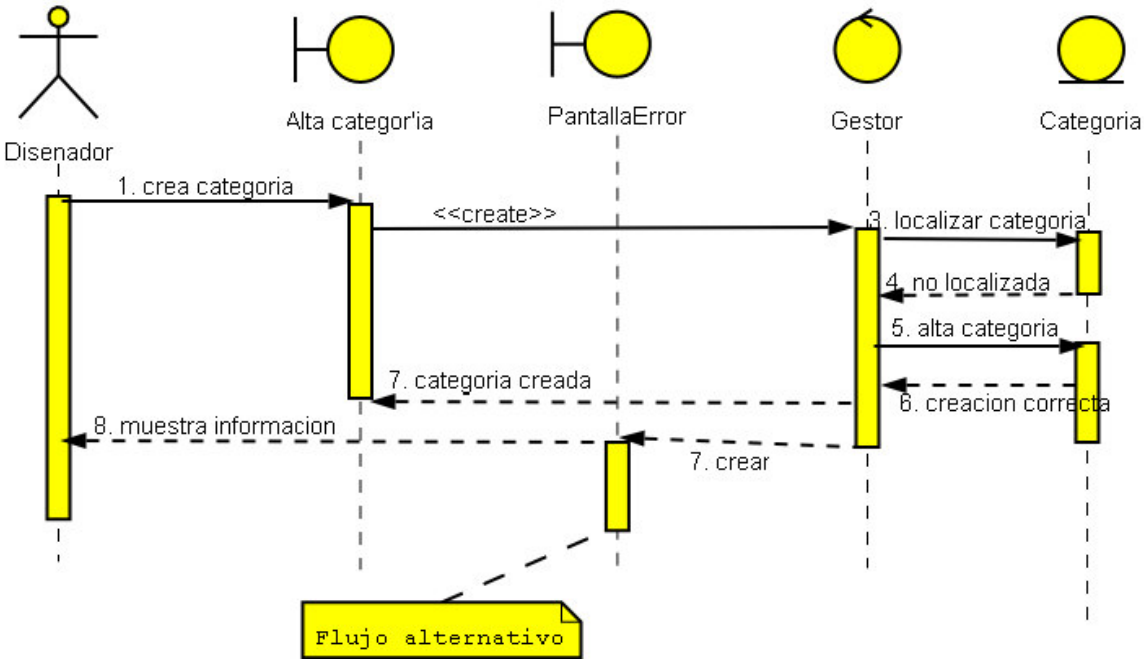

Figura 52. Diagrama de secuencia de UC-8

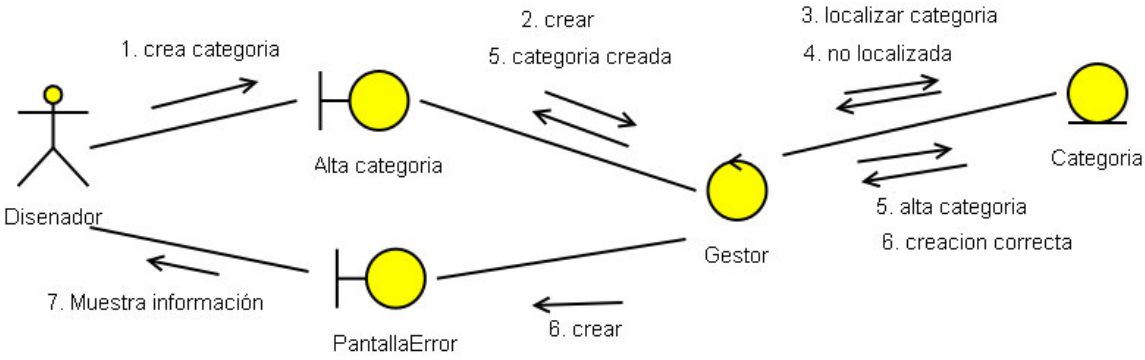

Figura 53. Diagrama de colaboración del modelo de análisis de UC-8

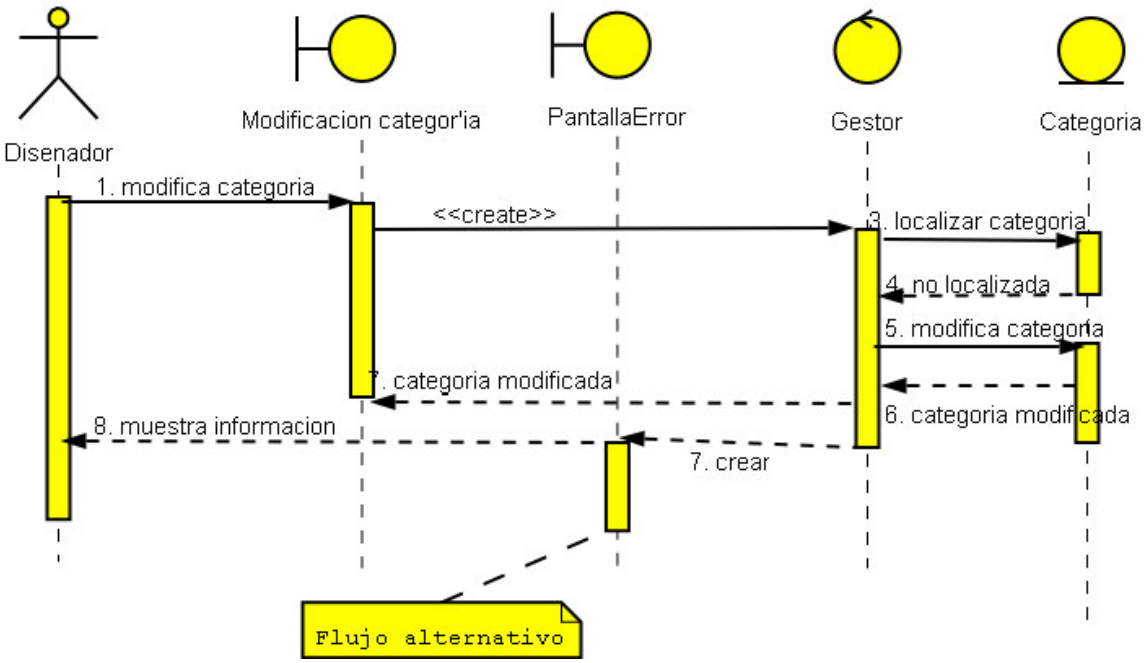

Figura 54. Diagrama de secuencia del modelo de análisis de UC-9

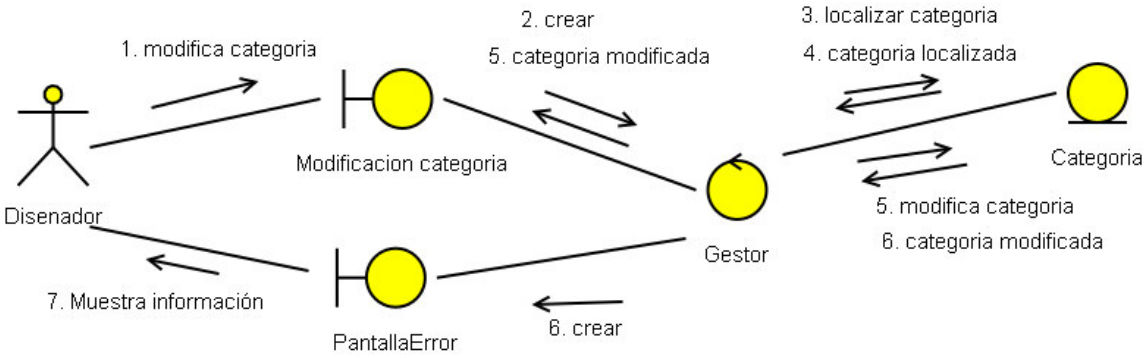

Figura 55. Diagrama de colaboración del modelo de análisis de UC-9

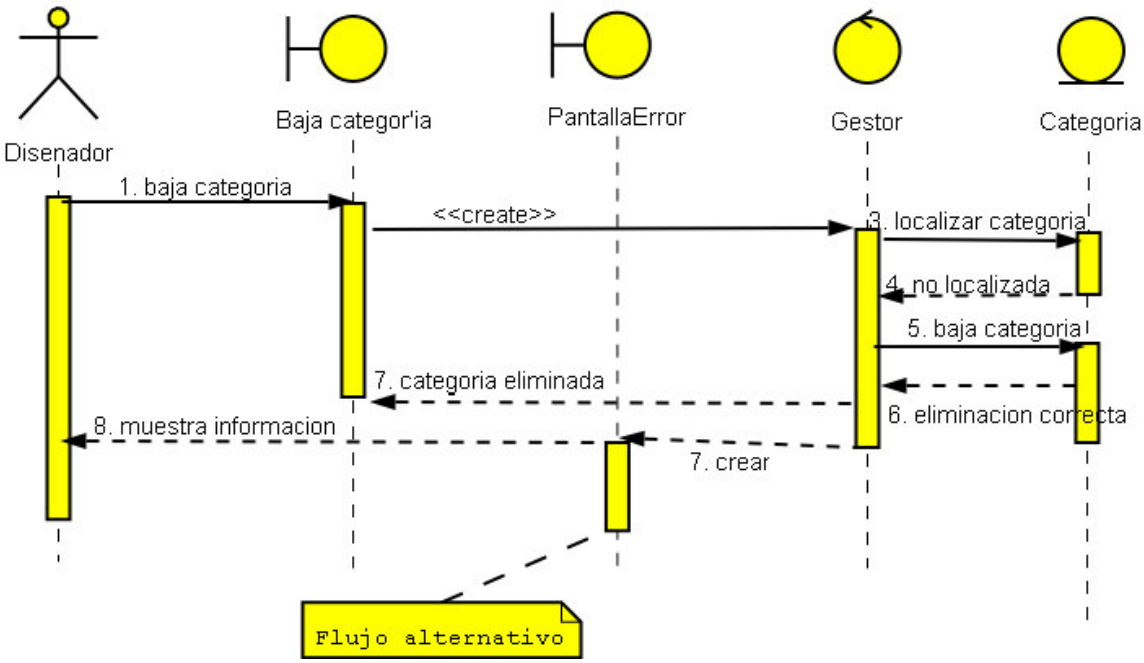

Figura 56. Diagrama de secuencia del modelo de análisis de UC-10

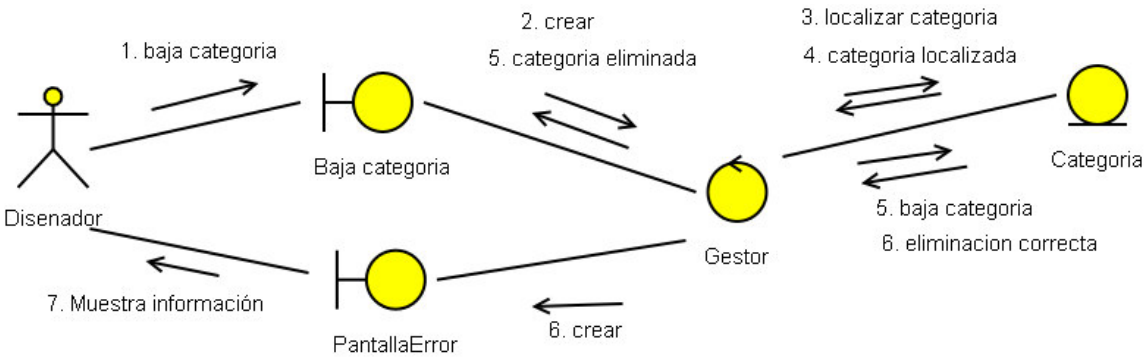

Figura 57. Diagrama de colaboración del modelo de análisis de UC-10

4.8.3. Paquete de gestión de reglas

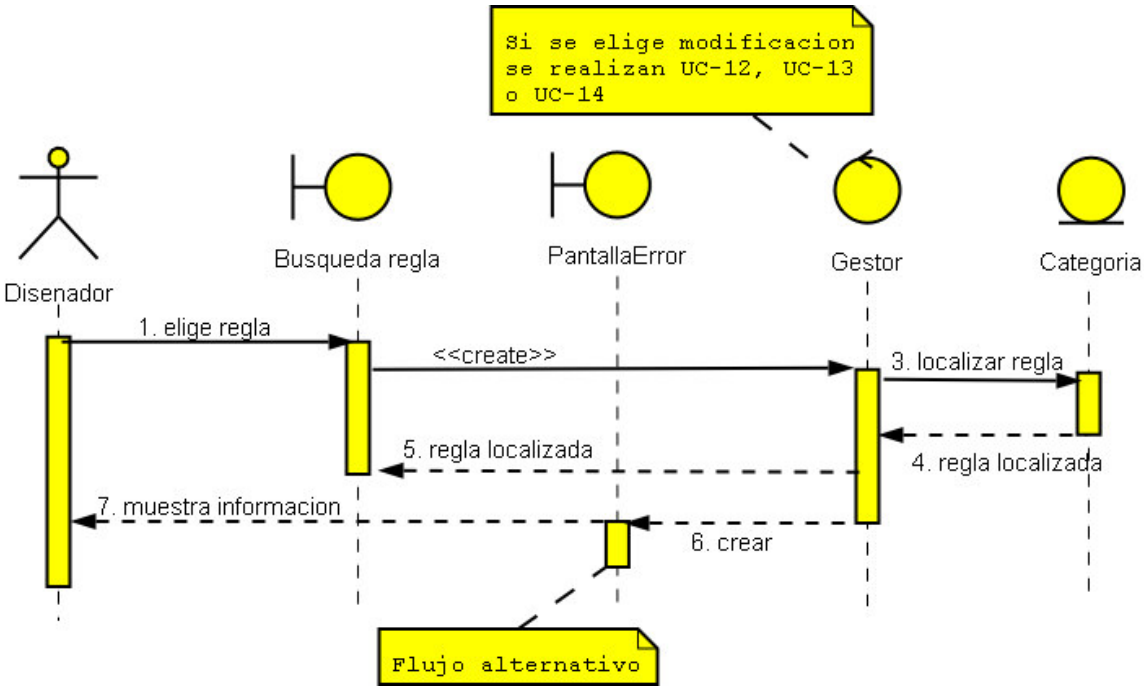

Figura 58. Diagrama de secuencia del modelo de análisis de UC-11

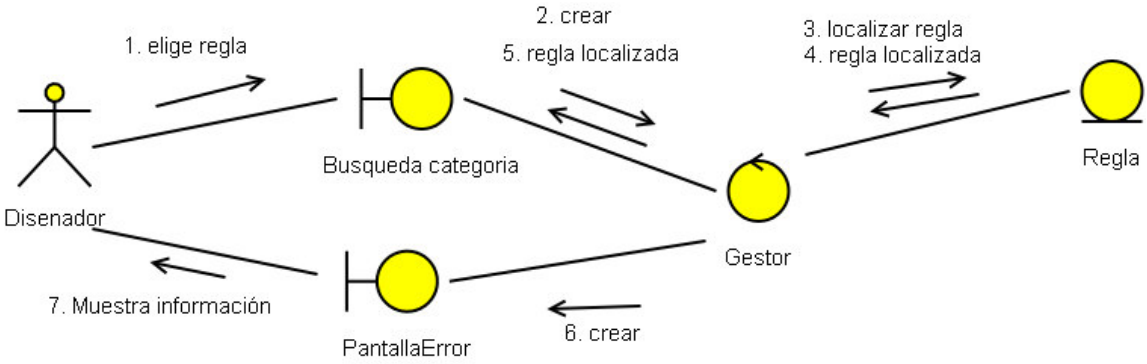

Figura 59. Diagrama de colaboración del modelo de análisis de UC-11

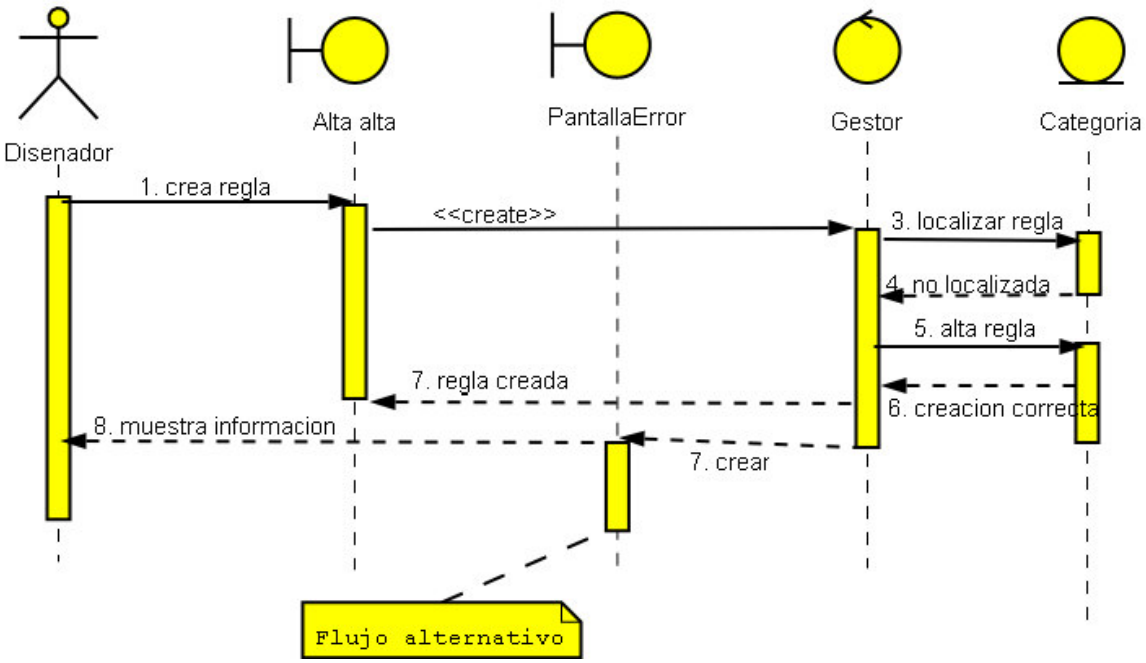

Figura 60. Diagrama de secuencia del modelo de análisis de UC-12

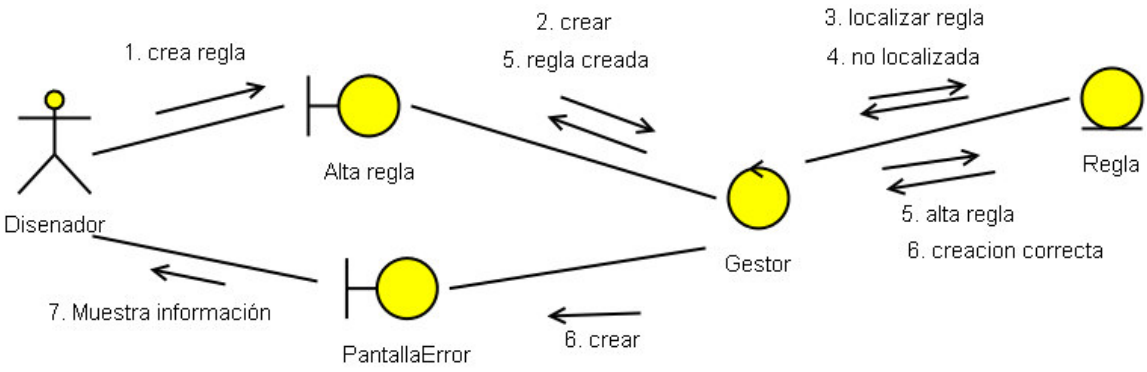

Figura 61. Diagrama de colaboración del modelo de análisis de UC-12

La modificación de regla recogida en el caso de uso UC-13 incluye el acceso a antecedente, consecuente, categorías y cambios de atributos. En esta etapa de análisis se ha optado por englobar todo ello bajo el acceso con el mensaje <5. modifica regla>. En la etapa de diseño se detalla todo lo expuesto.

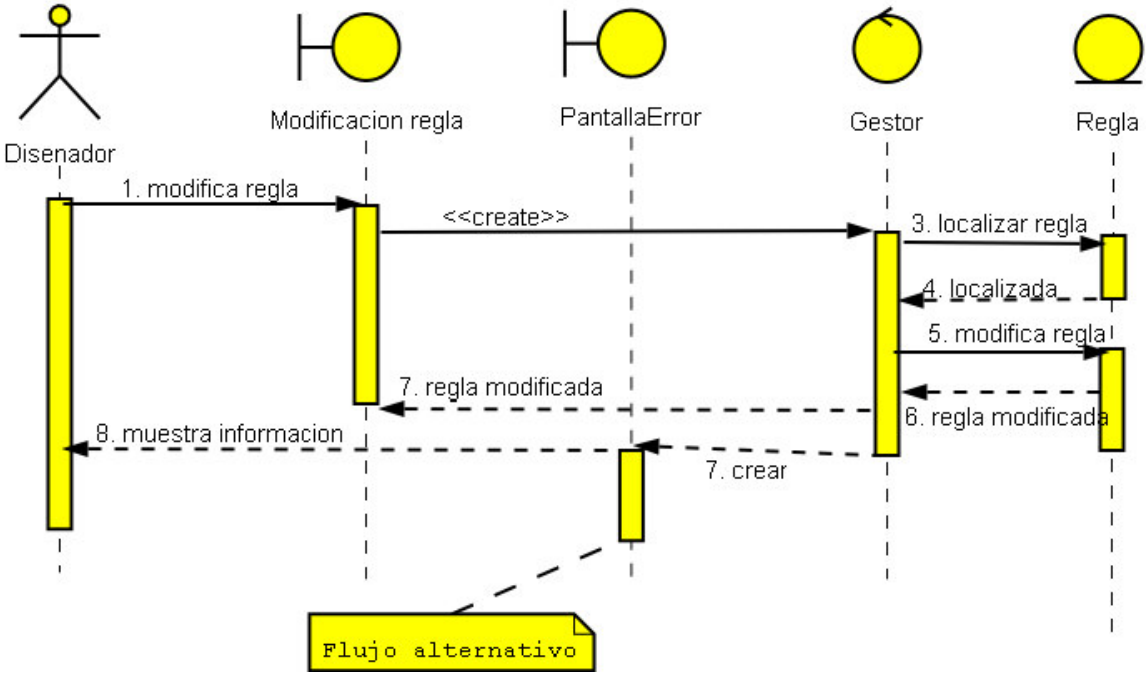

Figura 62. Diagrama de secuencia del modelo de análisis de UC-13

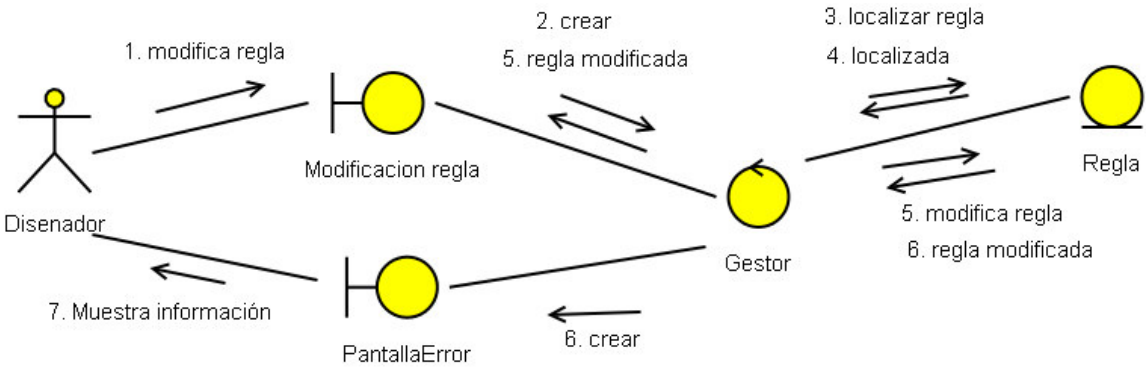

Figura 63. Diagrama de colaboración del modelo de análisis de UC-13

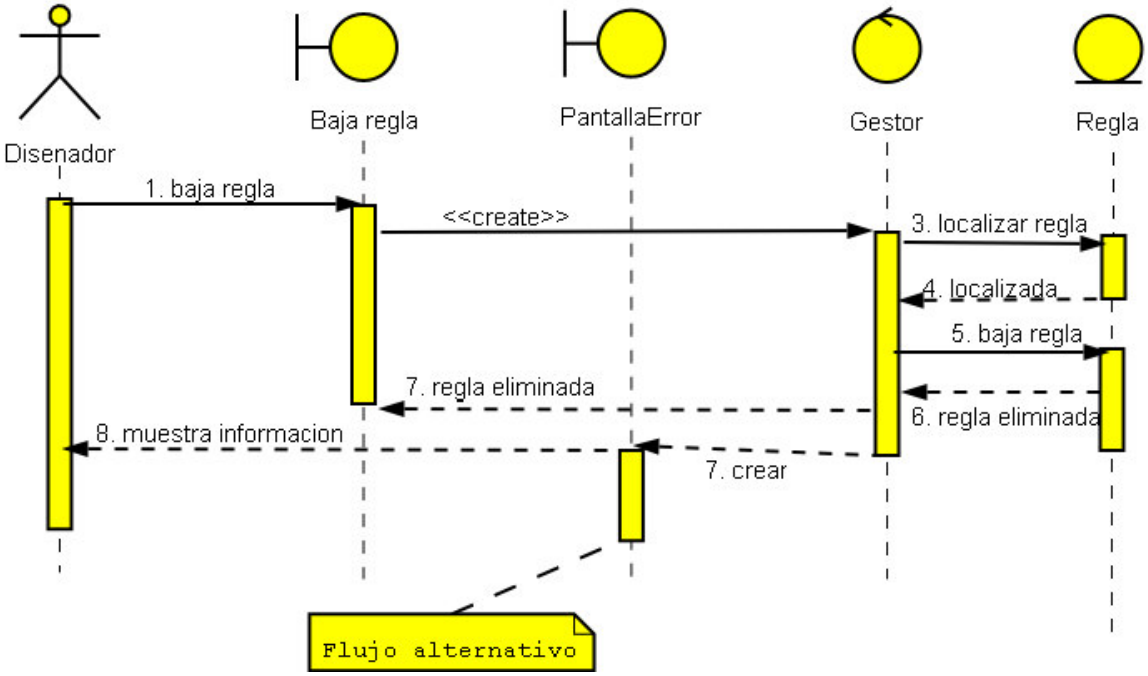

Figura 64. Diagrama de secuencia del modelo de análisis de UC-14

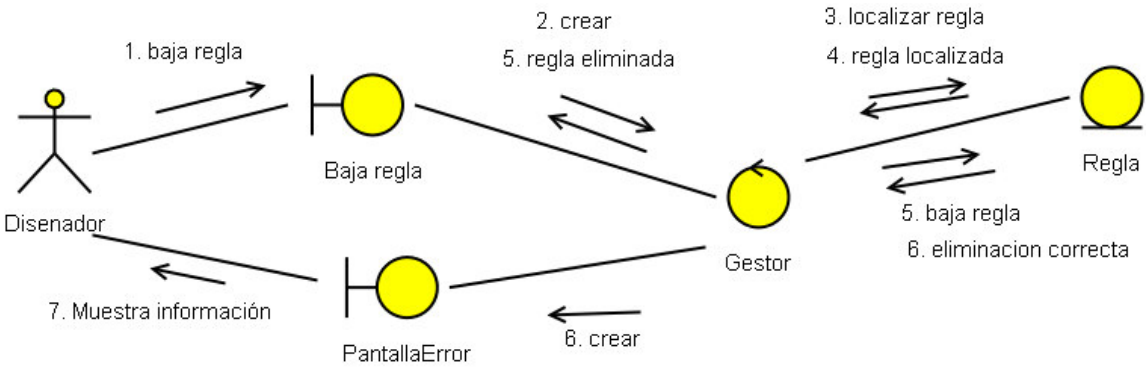

Figura 65. Diagrama de colaboración del modelo de análisis de UC-14

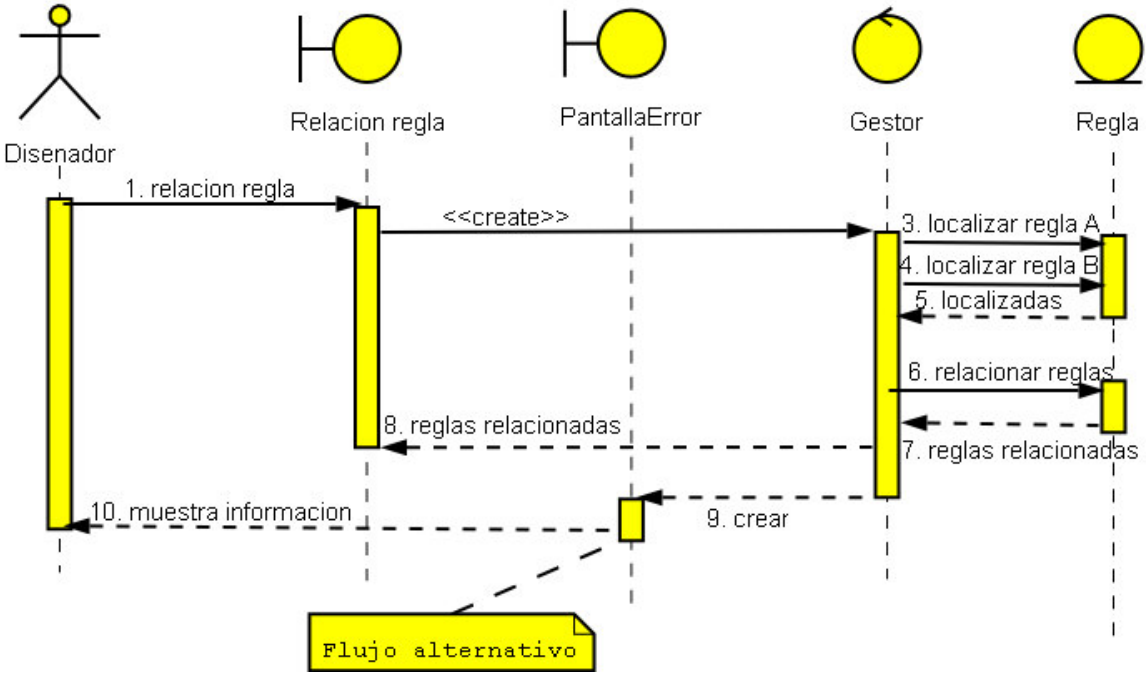

Figura 66. Diagrama de secuencia del modelo de análisis de UC-15

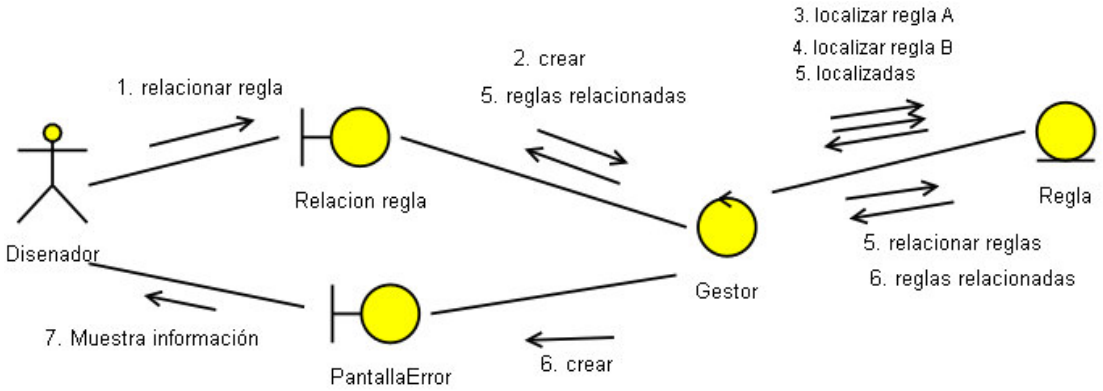

Figura 67. Diagrama de secuencia del modelo de análisis de UC-15

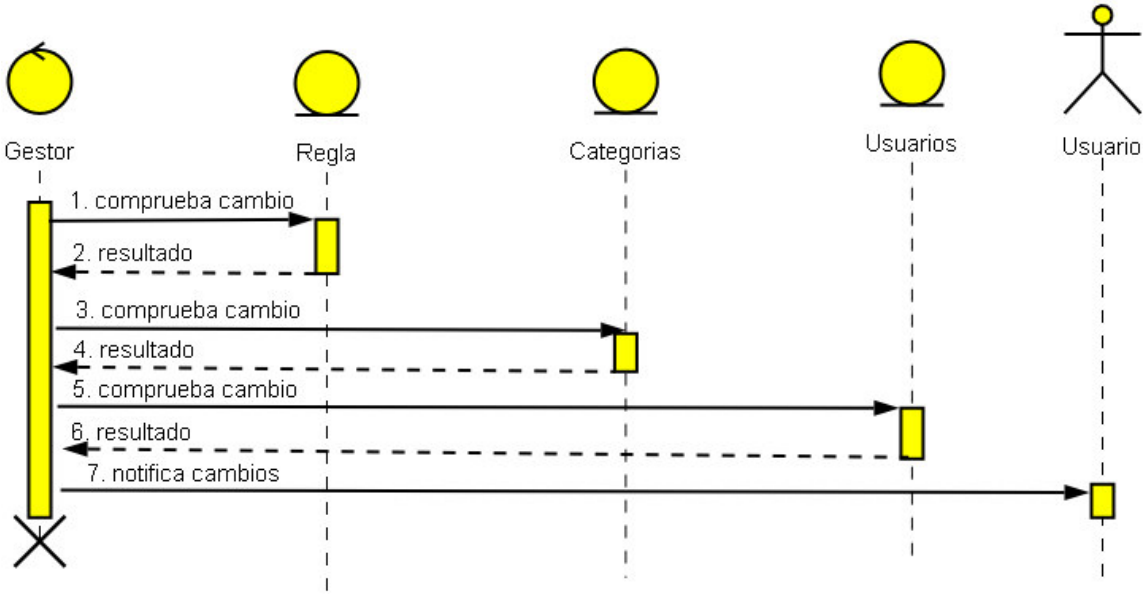

Figura 68. Diagrama de secuencia del modelo de análisis de UC-16

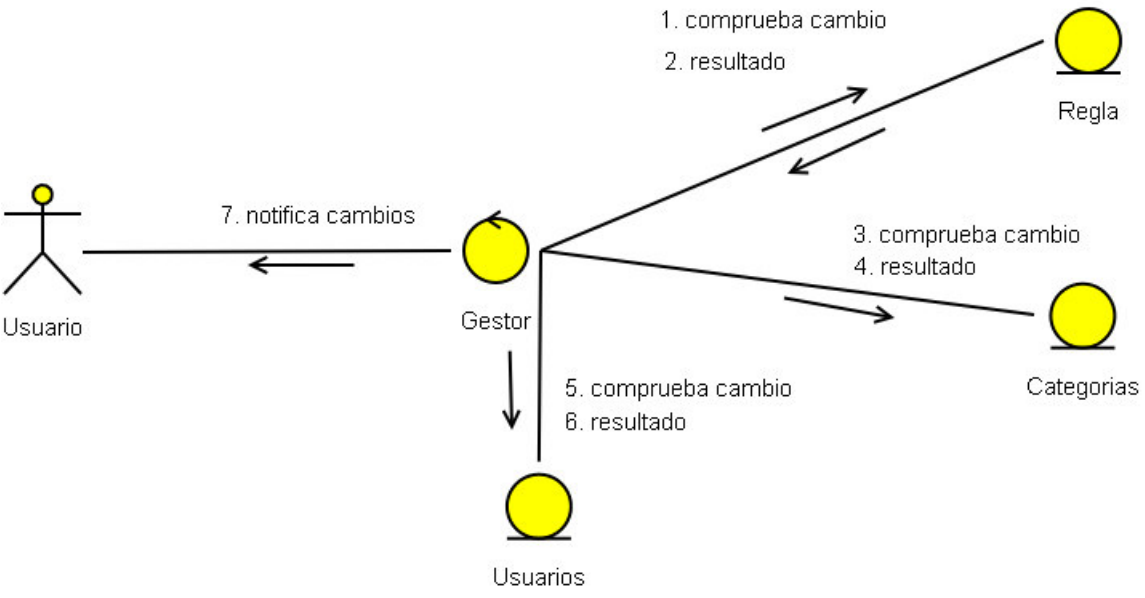

Figura 69. Diagrama de colaboración del modelo de análisis de UC-16

4.8.4. Paquete gestión de usuarios

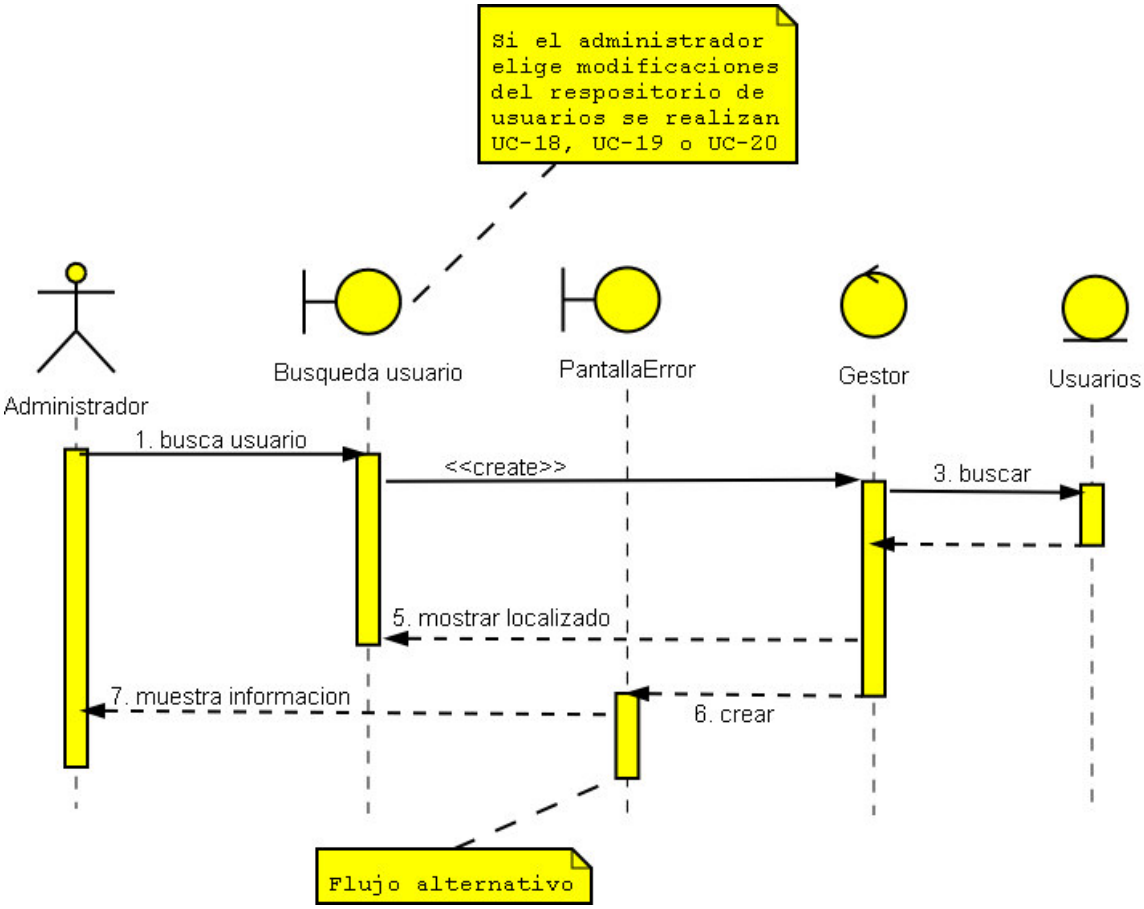

Figura 70. Diagrama de secuencia del modelo de análisis de UC-17

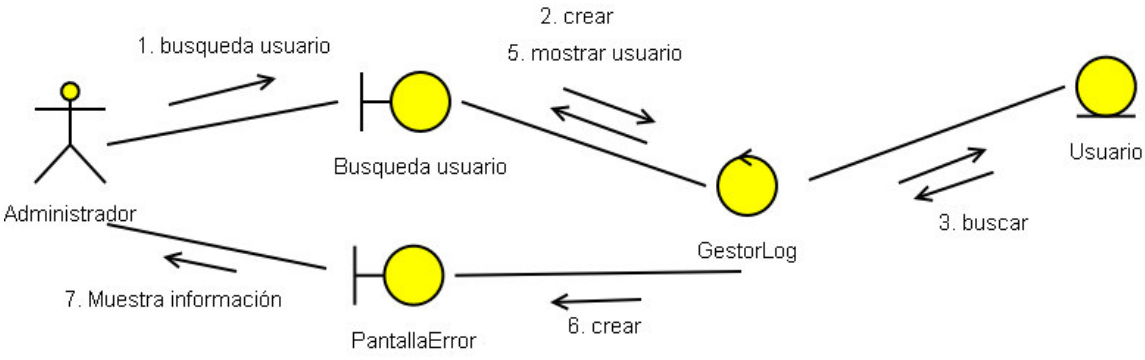

Figura 71. Diagrama de colaboración del modelo de análisis de UC-17

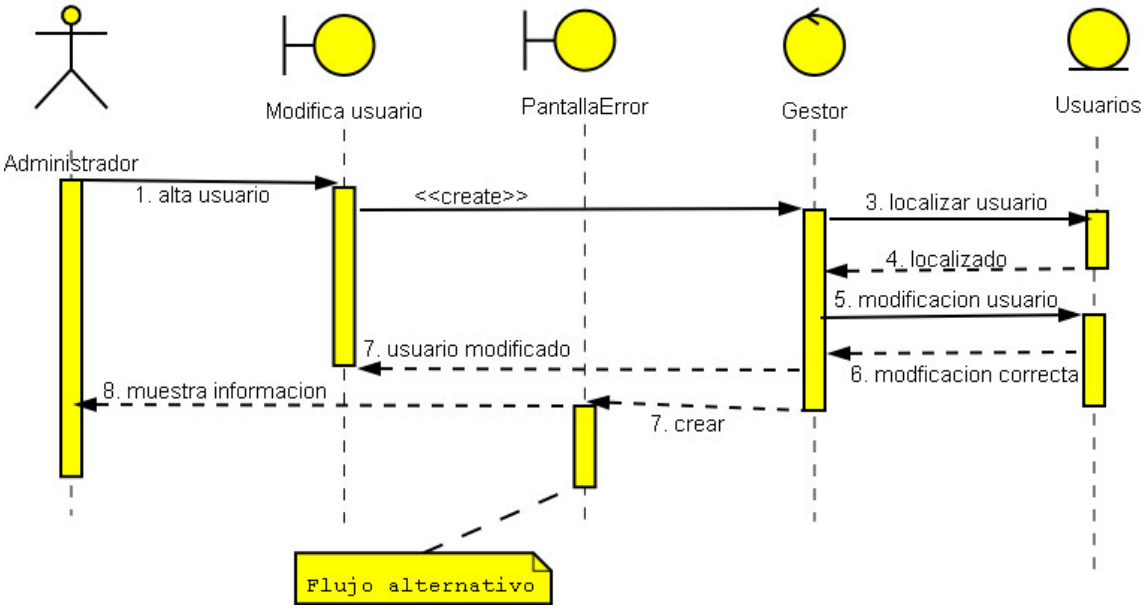

Figura 72. Diagrama de secuencia del modelo de análisis de UC-18

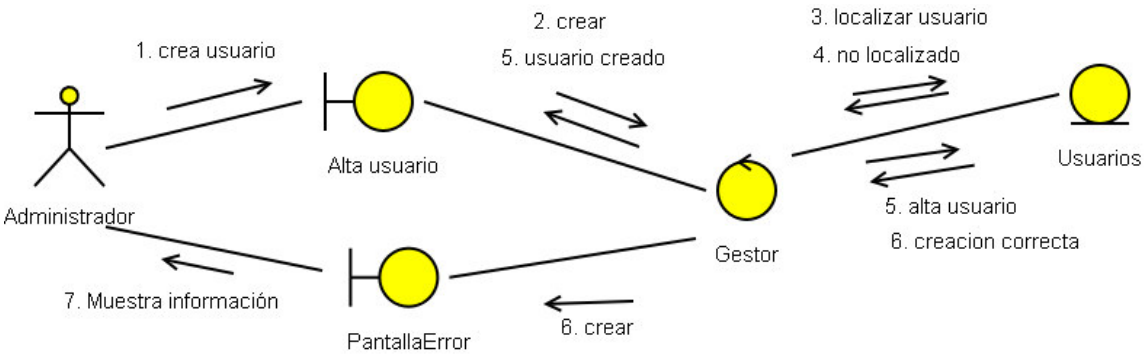

Figura 73. Diagrama de colaboración del modelo de análisis de UC-18

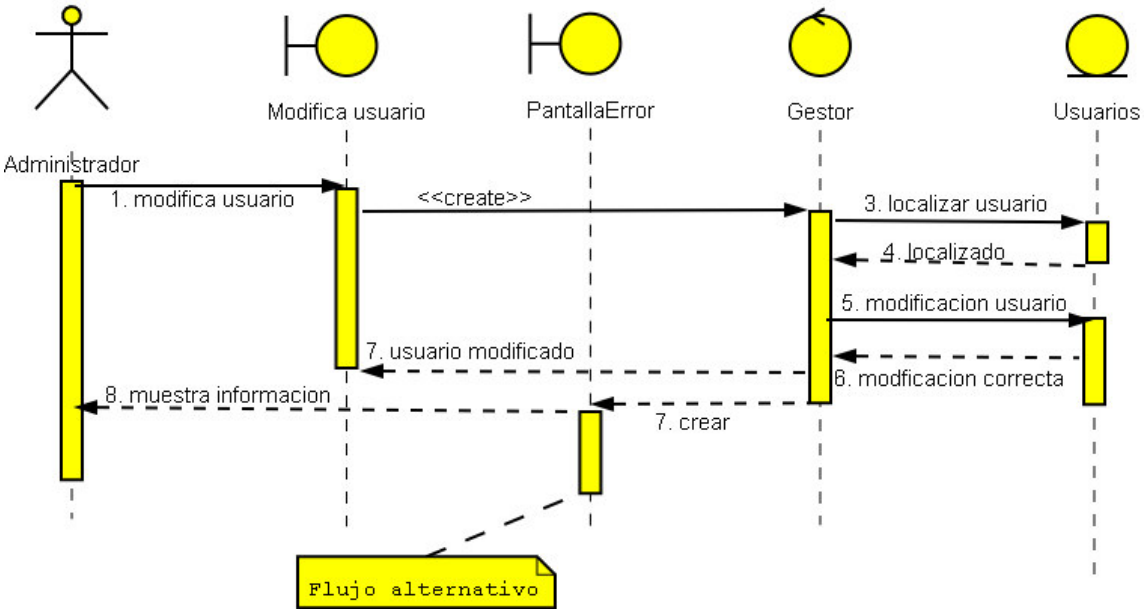

Figura 74. Diagrama de secuencia del modelo de análisis de UC-19

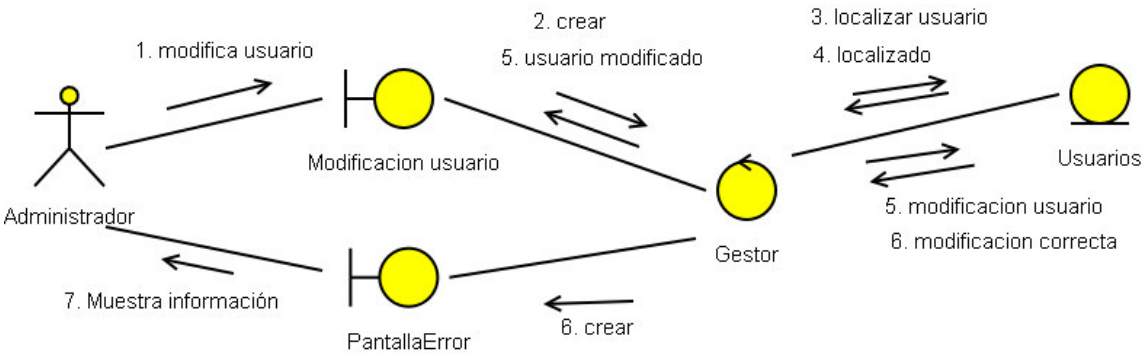

Figura 75. Diagrama de colaboración del modelo de análisis de UC-19

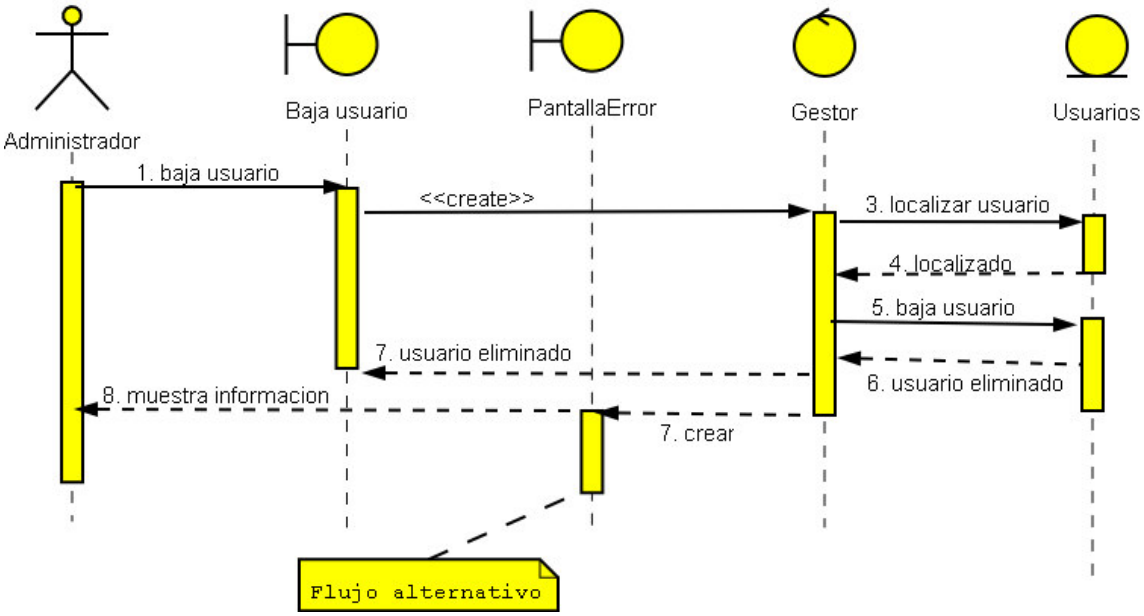

Figura 76 . Diagrama de secuencia del modelo de análisis de UC-20

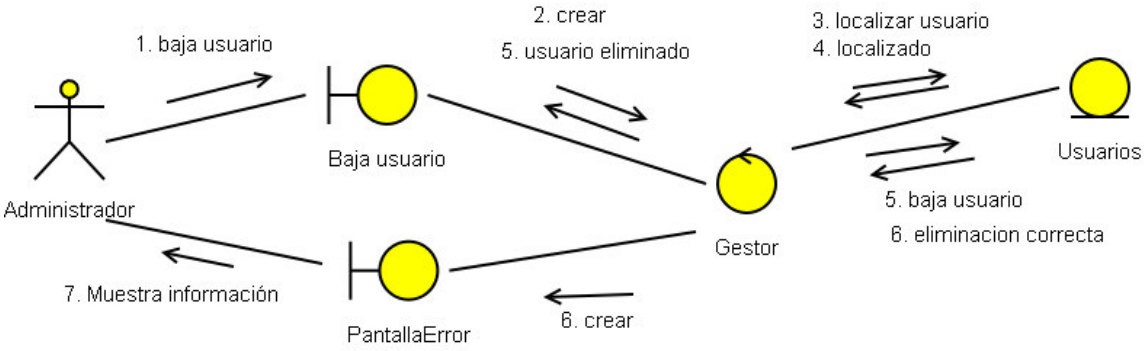

Figura 77. Diagrama de colaboración del modelo de análisis de UC-20

4.9. Conclusiones del análisis

Tomando como punto de partida el acervo de conocimientos existente sobre la representación de reglas de negocio mediante motores de reglas IF-THEN hemos obtenido el análisis del motor de reglas.

Dicho motor cubre un conjunto de funcionalidades muy concreto y sencillo, pero precisamente la sencillez (que no simplicidad) es uno de sus mayores valores de cara a su utilización. Ya que esperamos que, de forma rápida y sencilla, las reglas se encuentren disponibles para los usuarios potenciales.

Los requisitos funcionales nos han permitido identificar un total de una veintena de casos de uso para tres actores diferentes. La realización de los casos de uso se ha llevado a cabo con la prioridad de que el usuario tenga que realizar el menor número posible de acciones, aunque sin perder coherencia y funcionalidad. Consideramos que se ha realizado un análisis bastante completo de la aplicación, sus requisitos y las necesidades a cubrir muy bien identificadas.

En el capítulo siguiente abordaremos el diseño del motor de reglas.

5. Diseño del motor

Respecto al diseño del motor hemos optado por usar la plataforma J2ME de Sun para dispositivos móviles, frente a otras alternativas como Waba o Superwaba. El motivo es que J2ME, en la actualidad, está mejor documentada y depurada que las alternativas citadas.

5.1. ANÁLISIS DE LA ARQUITECTURA

En este apartado seleccionamos y analizamos una arquitectura tanto física como lógica del motor de reglas

5.1.1. JAVA2 MICRO EDITION

Java 2 Micro Edition fue lanzada por Sun Microsystems [J2M06] en 1999 con el fin de habilitar aplicaciones Java para pequeños dispositivos. Java Micro Edition es la versión del lenguaje Java que está orientada al desarrollo de aplicaciones para dispositivos pequeños con capacidades limitadas tanto en pantalla gráfica, como de procesamiento y memoria (teléfonos móviles, PDAs, sistemas inteligentes para la industria del automóvil y electrodomésticos inteligentes entre otros).

La aparición de esta tecnología se debió a las crecientes necesidades de los usuarios de telefonía móvil, las cuales han eclosionado en los últimos años demandando cada vez más servicios y prestaciones de terminales y compañías. En este sentido, el uso de J2ME [J2M06] depende del asentamiento en el mercado de otras, como GPRS, íntimamente asociada a J2ME, y que no ha estado a nuestro alcance hasta hace poco.

J2ME se configura como una tecnología de futuro para la industria de los dispositivos móviles. Esta afirmación se encuentra soportada por el creciente número de implantación de protocolos y dispositivos necesarios para soportarla por parte de los fabricantes.

5.1.2. Arquitectura de J2ME

Esta plataforma usa una máquina virtual denominada KVM (Kilo Virtual Machine, debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) o la máquina CVM (Compact Virtual Machine) en lugar de la JVM clásica, inclusión de un pequeño y rápido recolector de basura y otras diferencias que ya iremos viendo más adelante. En la Figura 78 se encuadra J2ME en el contexto de la plataforma Java de Sun.

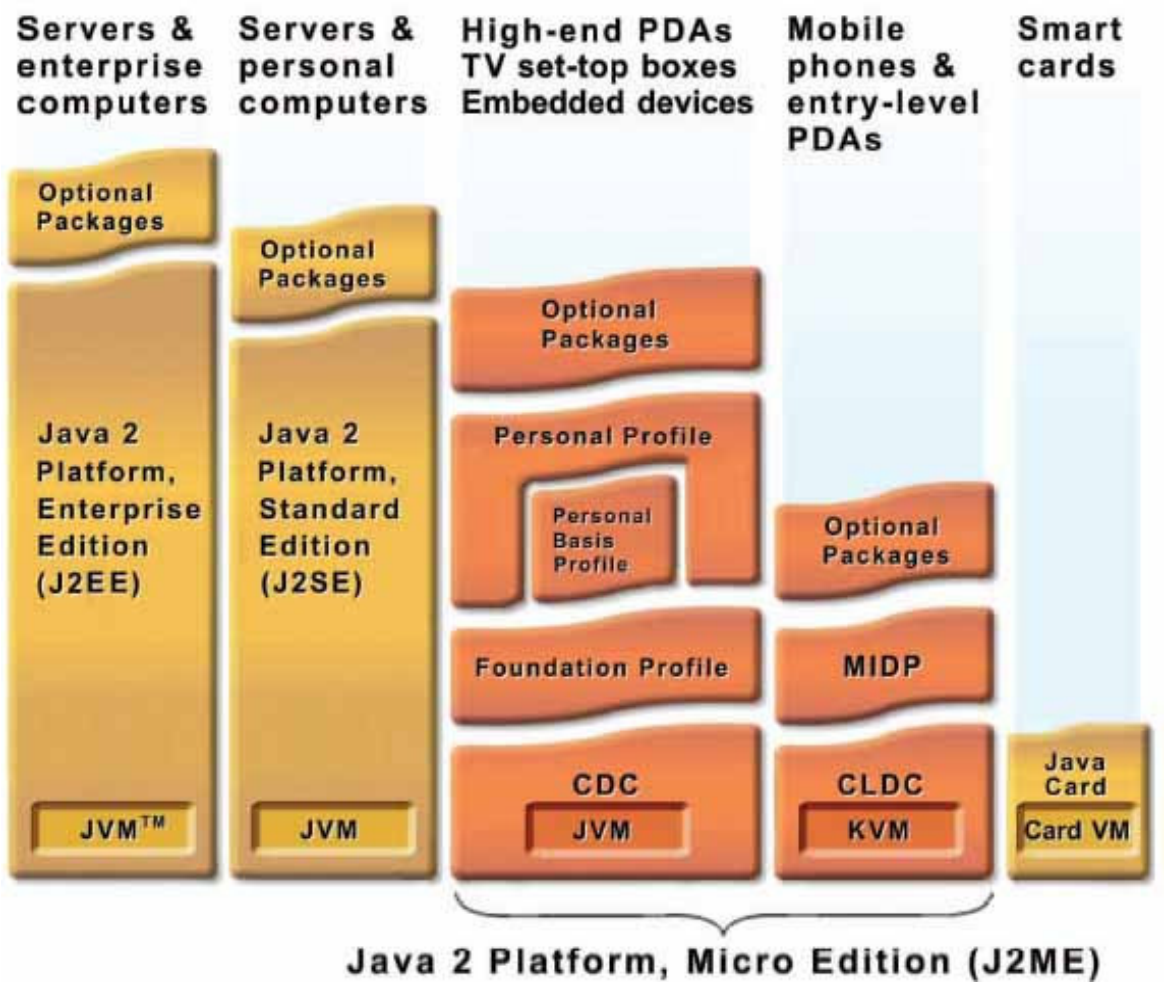

Figura 78. Arquitectura de la plataforma Java

No resulta rigurosa la concepción de Java como un simple lenguaje de programación [GAL03], sería más exacto entenderlo como un conjunto de tecnologías que abarca a todos los ámbitos de la computación con dos elementos en común.

- El código fuente en lenguaje Java es compilado e interpretado por una Java Virtual Machine (JVM), por lo que el código ya compilado es independiente de la plataforma.

- Todas las tecnologías comparten un conjunto más o menos amplio de APIs básicas del lenguaje, agrupadas principalmente en los paquetes java.lang y java.io.

J2ME contiene una mínima parte de las APIs de Java, debido a que la edición estándar de APIs de Java ocupa 20 Megabytes y los dispositivos para los que va destinado J2ME no disponen de una cantidad de memoria semejante. Concretamente, J2ME utiliza treinta y siete clases de la plataforma J2SE, cuyo origen son los paquetes java.lang, java.io, java.util. Esta parte de la API que se mantiene forma parte de lo que se denomina configuración. Otras diferencias con la plataforma J2SE vienen dadas por el uso de una máquina virtual distinta de la clásica JVM denominada KVM. Esta nueva máquina virtual tiene unas restricciones que limitan las capacidades incluidas en la JVM.

Podría decirse que J2EE es un superconjunto de J2SE y que éste lo es a su vez de J2ME. Aunque no es realmente exacta esta afirmación, ya que J2ME dispone de un paquete adicional javax.microedition, tal y como se recoge en la Figura 79. Para no caer en un exceso de simplicidad, hemos de indicar que, en realidad, cada una de las ediciones se encuentra dirigida a unos ámbitos de aplicación diferentes. Las necesidades computacionales y APIs de programación requeridas para un juego ejecutándose en un móvil difieren bastante con las de un servidor distribuido de aplicaciones basado en Enterprise Java Beans.

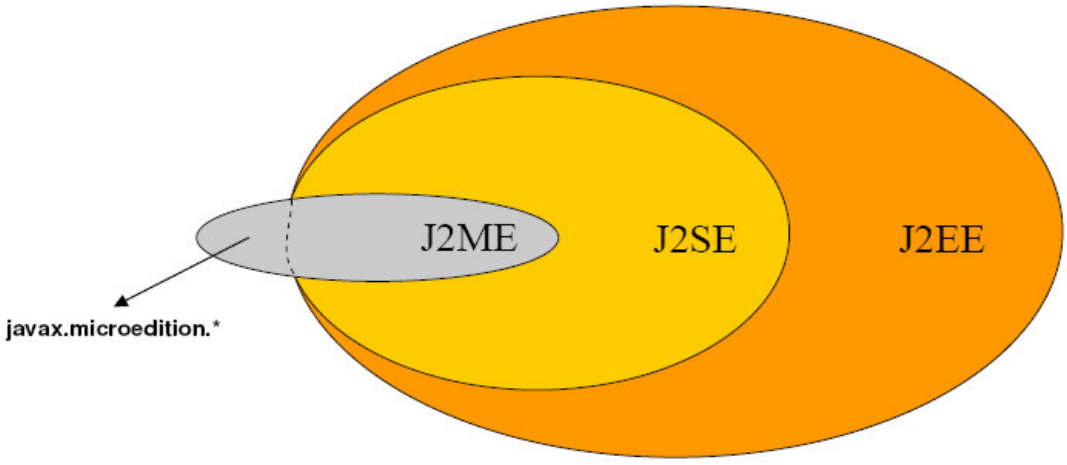

Figura 79. Conjuntos de la plataforma Java

5.1.3. Entorno de ejecución de J2ME

La edición J2ME se compone [GAL03] de una serie de elementos que se detallan a continuación.

- Una serie de máquinas virtuales Java con diferentes requisitos, cada una para diferentes tipos de pequeños dispositivos. Al igual que existen en la edición J2SE para diferentes sistemas operativos, en J2ME se han de tener en cuenta, no sólo los diferentes sistemas operativos existentes (MS Windows CE y Palm OS fundamentalmente), sino también el tipo de disponsitivo.

- Configuraciones. Son un conjunto de clases básicas orientadas a conformar el kernel de las implementaciones para dispositivos de características específicas. Las dos configuraciones definidas en J2ME son, en primer lugar, Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria y que utiliza KVM, y, en segundo, Connected Device Configuration (CDC) enfocada a dispositivos con más recursos y que usa CVM (máquina virtual para la configuración CDC).

- Perfiles. Son bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos.

Podemos situar las diferentes configuraciones que componen la edición J2ME (CDC y CLDC) respecto a la edición estándar de Java (J2SE) tal y como se refleja en la Figura 80. Se advierte que la configuración CLDC se configura como un subconjunto de CDC.

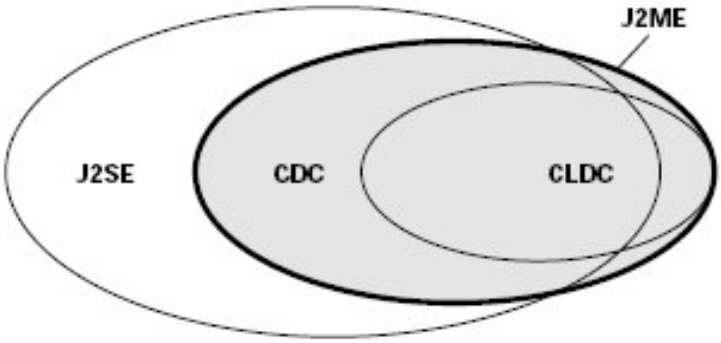

Figura 80. Configuraciones CDC y CDLC

Por otro lado, en un entorno de ejecución determinado de J2ME se compone entonces de una selección de la máquina virtual, la configuración, el perfil y los paquetes opcionales.

Un ejemplo de una instanciación de una arquitectura J2ME se muestra en la Figura 81, donde cada uno de los componentes ha sido sustituido por una elección concreta.

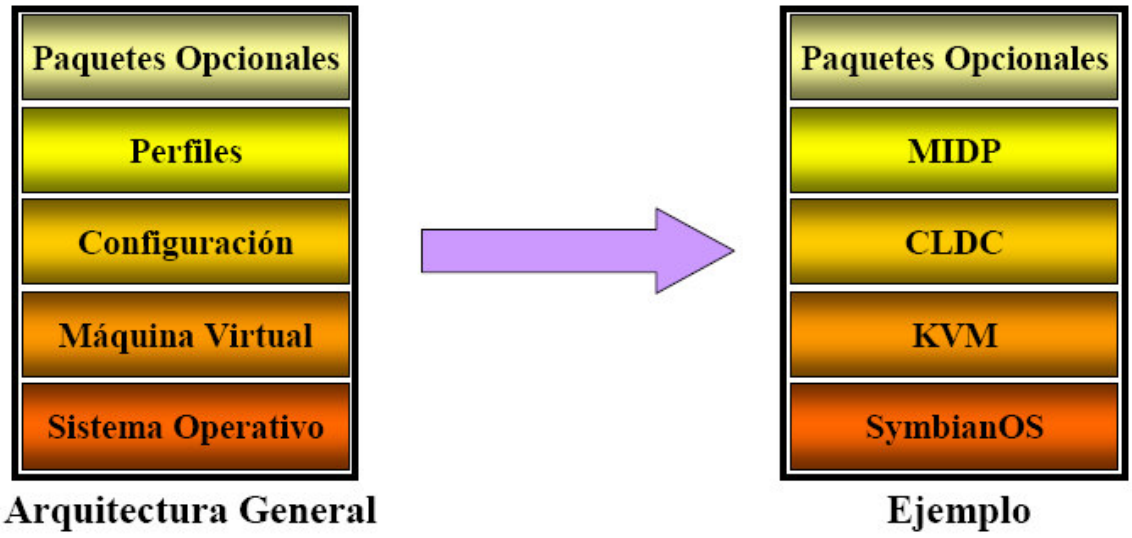

Figura 81. Instanciación de una arquitectura en capas de un entorno de ejecución J2ME

A continuación, una vez instanciada la arquitectura general, se precisan una serie de herramientas para el desarrollo de aplicaciones. En la Tabla 31 se recogen una serie de aplicaciones, toolkits y herramientas ligadas al desarrollo de aplicaciones para J2ME y sus tecnologías asociadas (CDC, CLDC, MIDP, etc.)

Tabla 31. Herramientas para el desarrollo en J2ME

Herramienta	Descripción	Compatibilidad	Proveedor
J2ME Wireless Toolkit	EL J2ME Wireless Toolkit proporciona herramientas para el desarrollo de aplicaciones para dispositivos móviles compatibles con la especificación MIDP. Existen varias versiones del J2ME Wireless Toolkit : • 1.0.4 • 2.0_01 • 2.1 • 2.2 (Beta)	JTWI Roadmap 1, MIDP 2.0 (y 1.x), CLDC 1.1 (and 1.0), MMA 1.1, WMA 1.1, JSR-172 (Web services), JSR-184(3D), JSR-82(Bluetooth) y JSR-75	Sun Microsystems
Nokia Developer's Suite for J2ME	Orientado al desarrollo de aplicaciones para dispositivos Nokia compatibles con J2ME. Existen varias versiones del Nokia Developer's Suite for J2ME: • 1.0	MIDP 2.0 y 1.x, CLDC 1.0	Nokia

	• 2.0		
SDKs y herramientas Motorola iDEN	Desarrollo de aplicaciones J2ME orientadas a móviles iDEN compatibles con J2ME.Existen varias versiones del Motorola iDEN: • SDK 3.0.0 + update • i860 SDK • Java Application Loader • J2ME Open Windowing Toolkit	MIDP 2.0 y 1.x, CLDC 1.0	Motorola
Siemens Mobility Toolkit	Desarrollo de aplicaciones J2ME orientadas a teléfonos móviles Siemens compatibles con J2ME. Está compuesta por el *SMTK Core Pack* y paquetes de emuladores para los modelos *45 y 55* o *65*.	MIDP 1.0 y CLDC 1.0	Siemens
Samsung Java SDK	SDK Java para desarrollo de aplicaciones compatibles J2ME para teléfonos Samsung. Existen dos versiones de la JDK: • 1.0 • 2.0	MIDP 2.0 y 1.0 y CLDC 1.0	Samsung
SonyEricsson	SonyEricsson J2ME SDK y herramientas y documentación asociada. Existen distintas versiones de las SDKs: • J2ME SDK 1.0 • J2ME SDK 2.0 • J2ME SDK 2.1.2 (beta) • J2ME SDK 2.1.3 (beta) • P900 J2ME SDK	MIDP 2.0, MIDP 1.0 y CLDC 1.x, JSR-184(3D)	SonyEricsson

5.2. Diseño de la arquitectura

5.2.1. Paquetes de diseño

La solución de diseño se compondrá de cuatro paquetes UML, los cuales se muestran en la Figura 82. Los diferentes paquetes que se muestran se detallan seguidamente.

- Modelo de análisis y diseño. Contiene todas las especificaciones de de diagramas de secuencia y colaboración referidos a los casos de uso detectados en la fase de análisis, aunque en esta ocasión a nivel de diseño.

- Arquitectura. Contiene la especificación de la arquitectura, que hemos elegido J2ME. La definición de las diferentes capas conforme a las normas de esta edición de la plataforma Java está contenida en el interior de este paquete.

- Esquemas. contiene los esquemas físicos de la base de datos, tablas e índices para ellas que darán soporte a la aplicación.

- Especificación persistencia. Contiene los mecanismos definidos para la propuesta de arquitectura que hacemos; básicamente se trata de la definición del mecanismo de persistencia mediante JDBC y el intercambio de ficheros desde el dispositivo móvil al servidor.

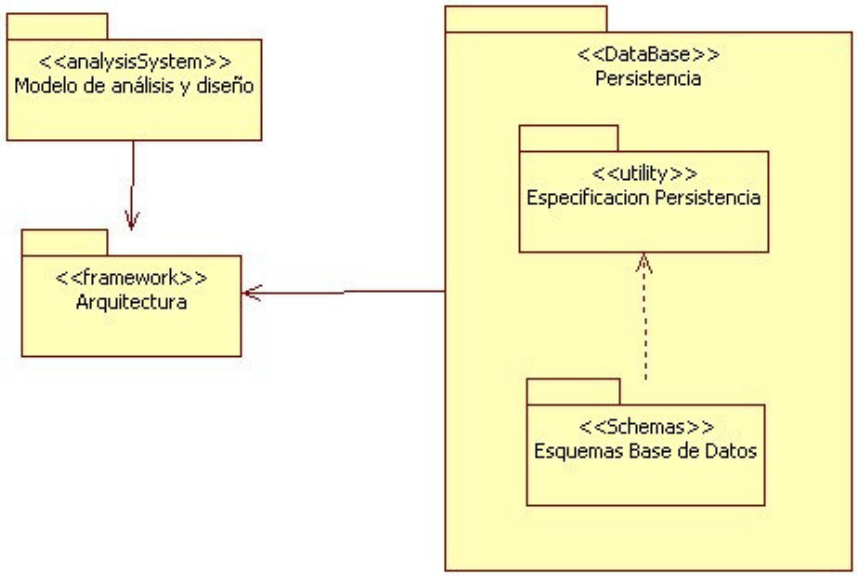

Figura 82. Diagrama de paquetes del motor

El contenido de los diferentes paquetes se detalla seguidamente en los siguientes epígrafes. Se especifican a nivel de diseño los diferentes casos de uso relacionados en el epígrafe 4.5, con un nivel de detalle más elevado.

Los diferentes modelos de diseño se recogerán en los paquetes de casos de uso para su mejor compresión.

Se ha buscado siempre que ha sido posible la ortogonalidad diagramática o, dicho de otra manera, la minimización de excepciones en el funcionamiento del motor, con el fin de facilitar su mantenimiento.

5.2.2. Modelo de diseño del núcleo

Los flujos alternativos se hayan contemplados en los diagramas mediante instancias de excepciones.

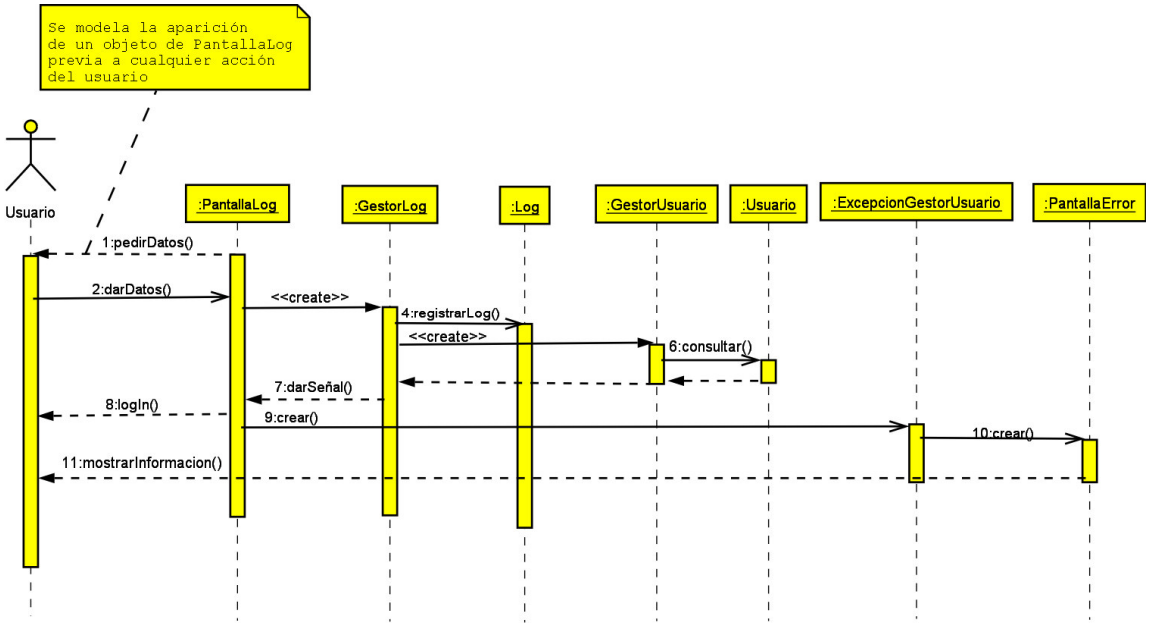

Figura 83. Diagrama de secuencia del modelo de diseño de UC-1

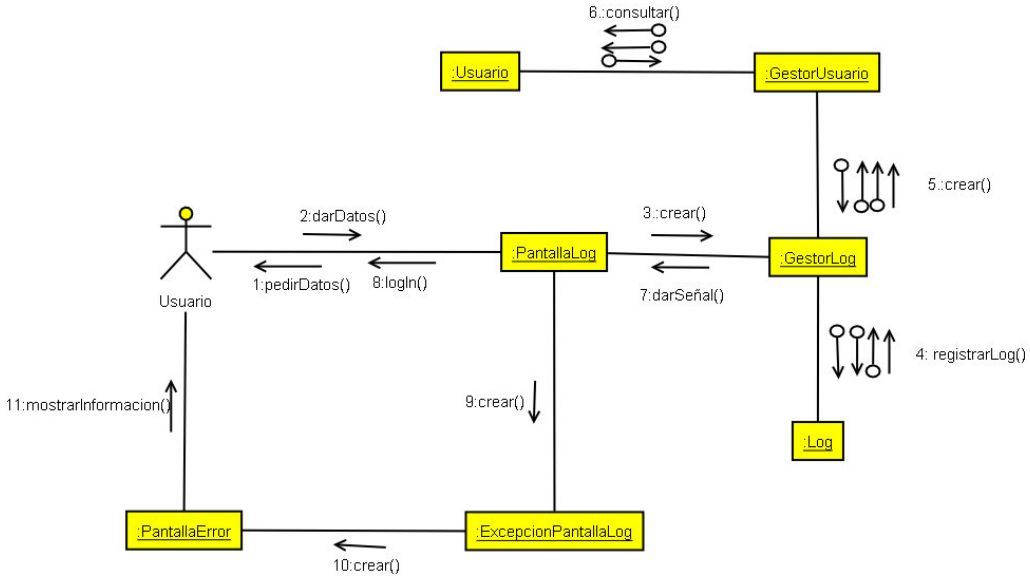

Figura 84. Diagrama de colaboración del modelo de diseño de UC-1

En la Figura 85 y en la Figura 86 podemos advertir como se envían los mensajes 6 y 7 siendo ambos idénticos. Con ello se modela la introducción doble de la nueva contraseña. Ello constituye una medida de seguridad ante un error tipográfico.

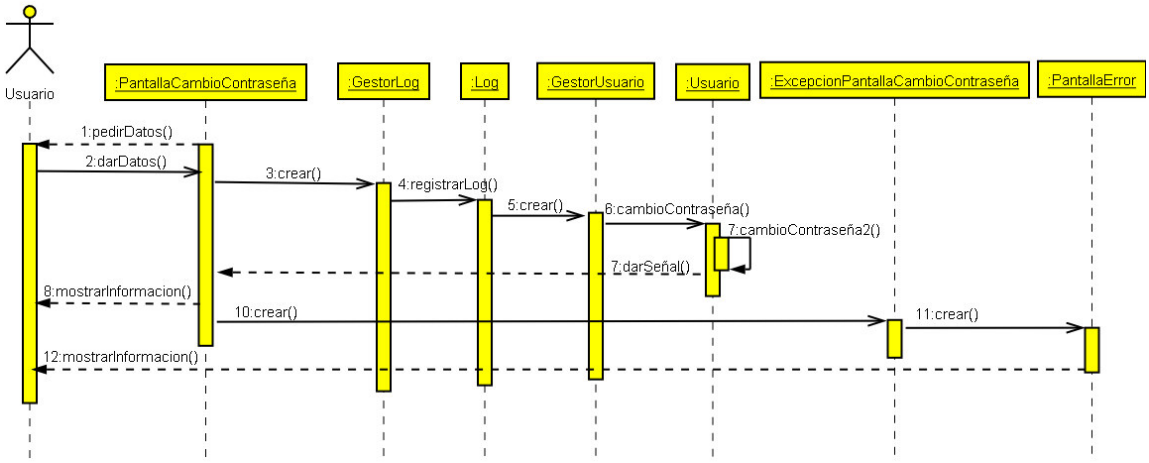

Figura 85. Diagrama de secuencia del modelo de diseño de UC-2

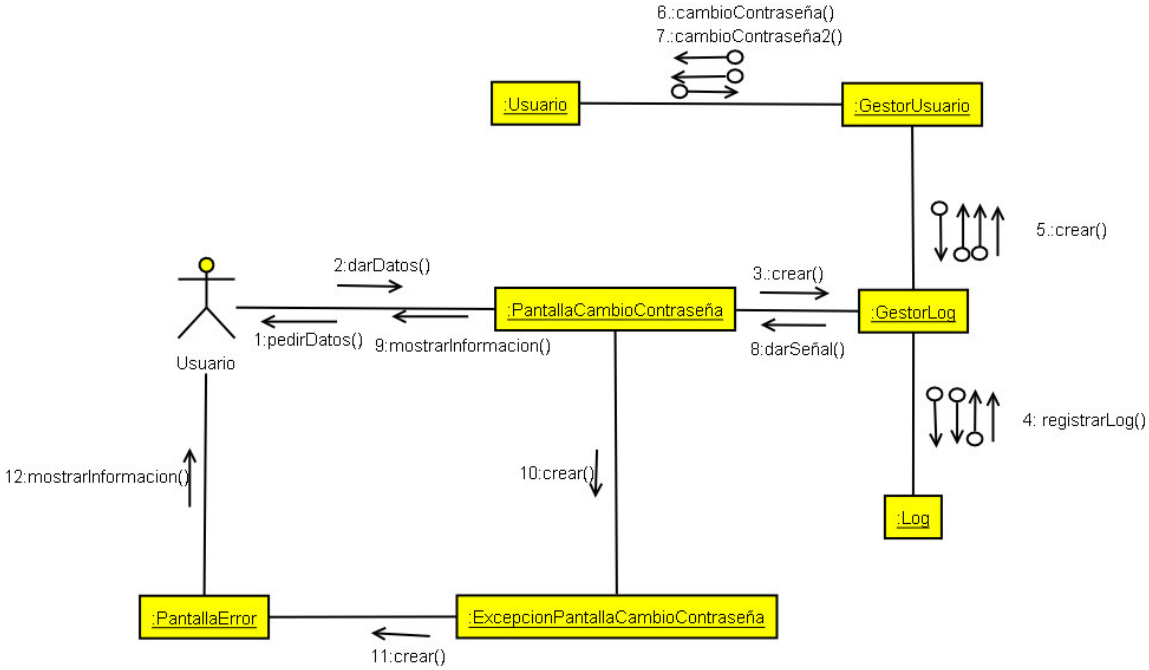

Figura 86. Diagrama de colaboración del modelo de diseño de UC-2

En la Figura 87 se destruyen las líneas de vida de los objetos GestorLog y Log para modelar el cierre de sesión. No se destruye la línea de vida del resto de los objetos para contemplar la secuencia alternativa correspondiente a la clase ExcepciónGestorLog.

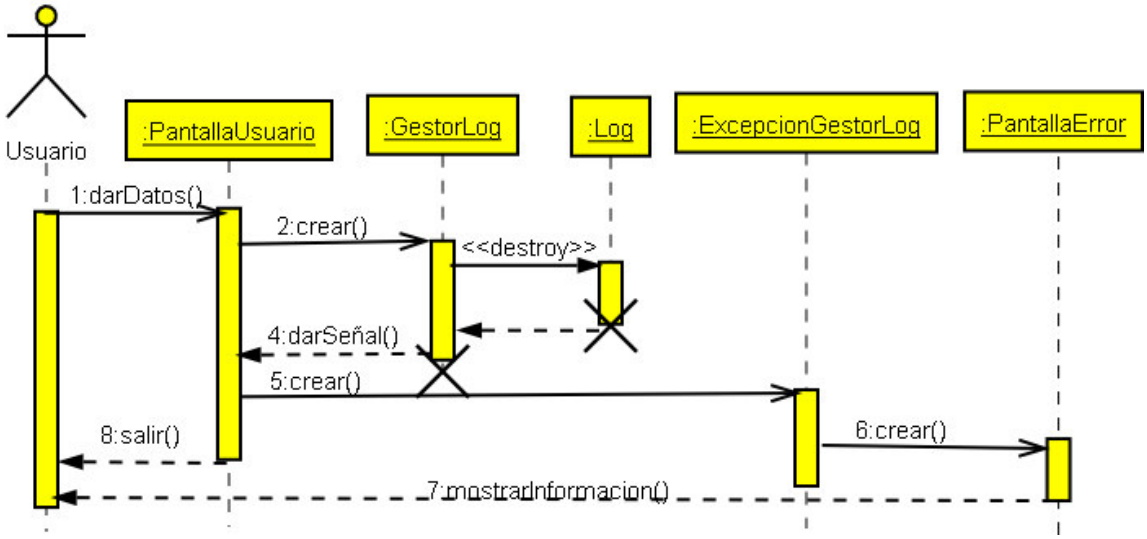

Figura 87. Diagrama de secuencia del modelo de diseño de UC-3

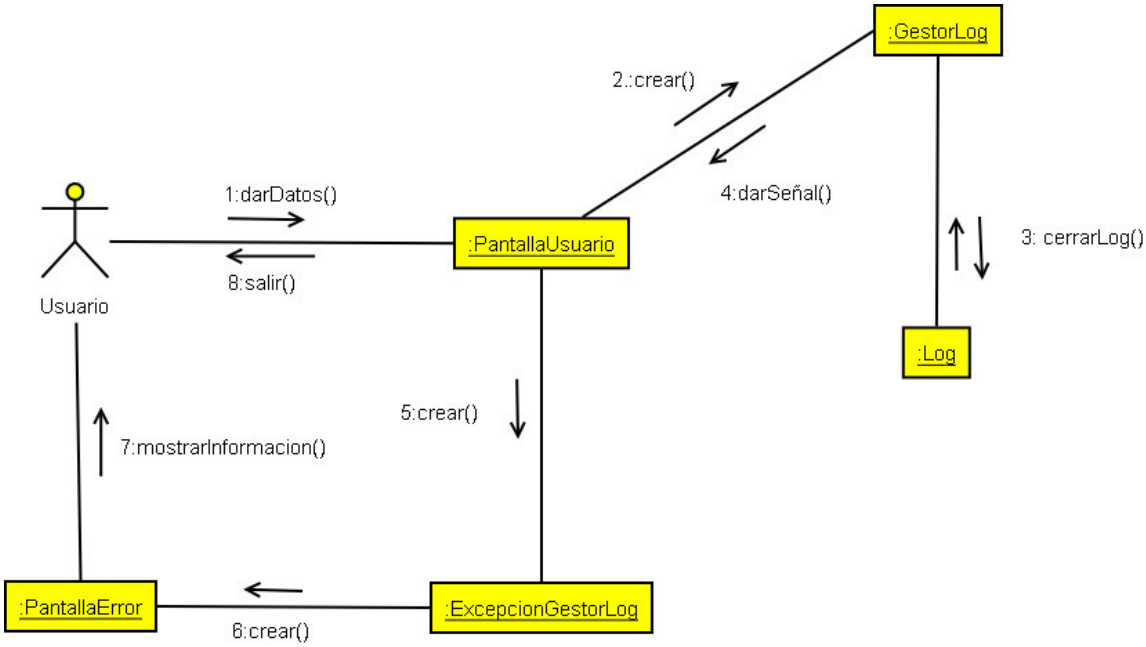

Figura 88. Diagrama de colaboración del modelo de diseño de UC-3

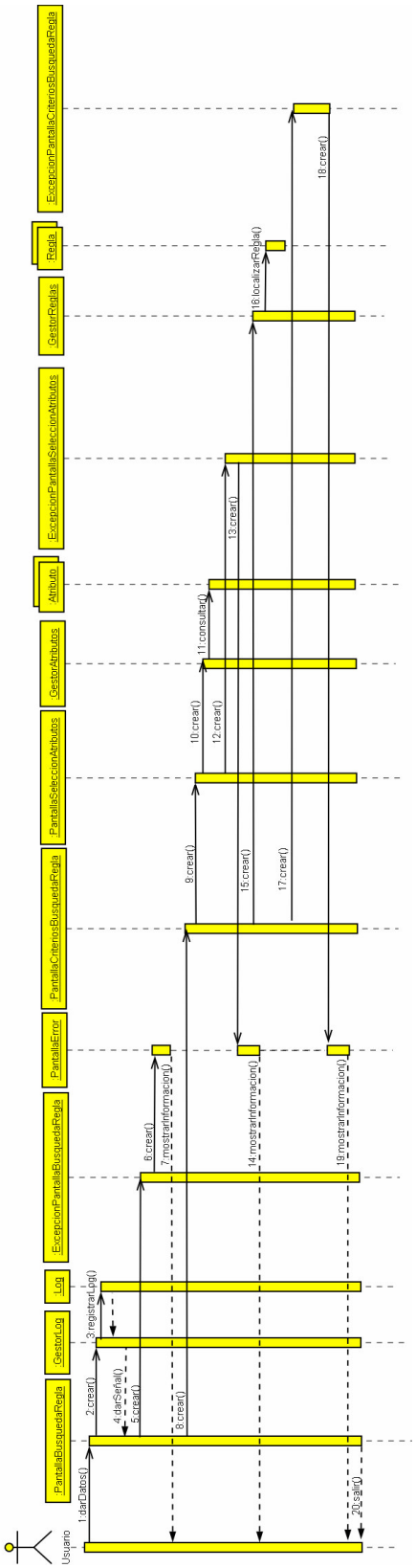

Figura 89. Diagrama de secuencia del modelo de diseño de UC-4

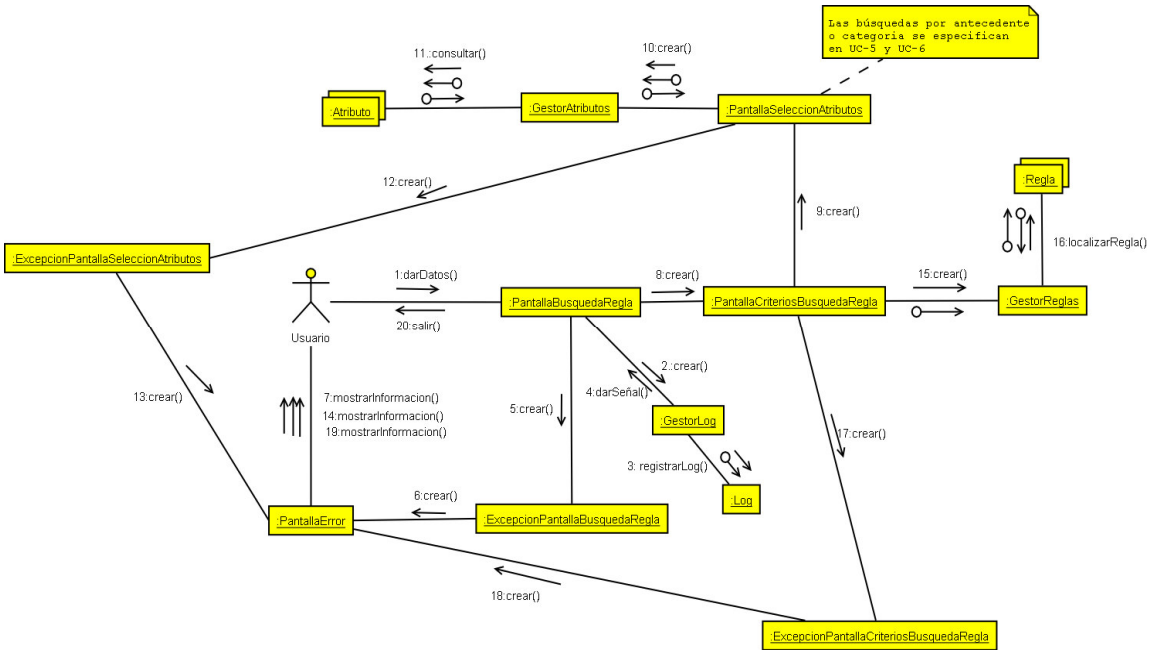

Figura 90. Diagrama de colaboración del modelo de diseño de UC-4

En la Figura 91 y en la Figura 92 se crean tres objetos diferentes correspondientes a la clase PantallaError. Con ello se modela tres posibles excepciones que confluyen en dicha clase.

Tras cada una de estas excepciones se interrumpe la secuencia. No obstante, en el modelo se continúa para modelar la secuencia completa.

De igual forma la búsqueda por criterios o por categoría se modelan en UC- 5 y UC-6, tal y como se indica en la nota que incorpora la Figura 92.

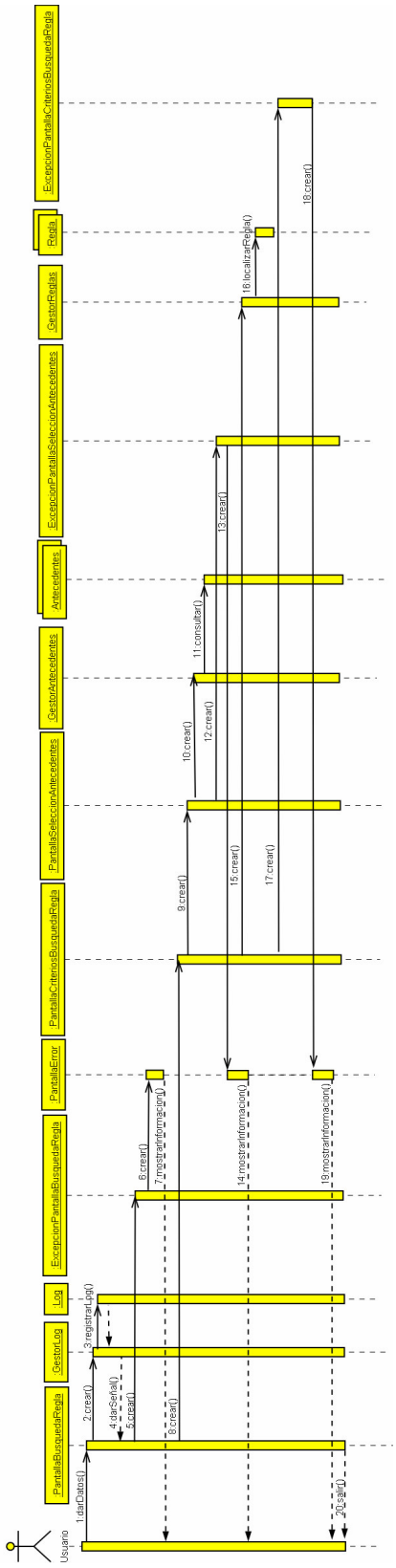

Figura 91. Diagrama de secuencia del modelo de diseño de UC-5

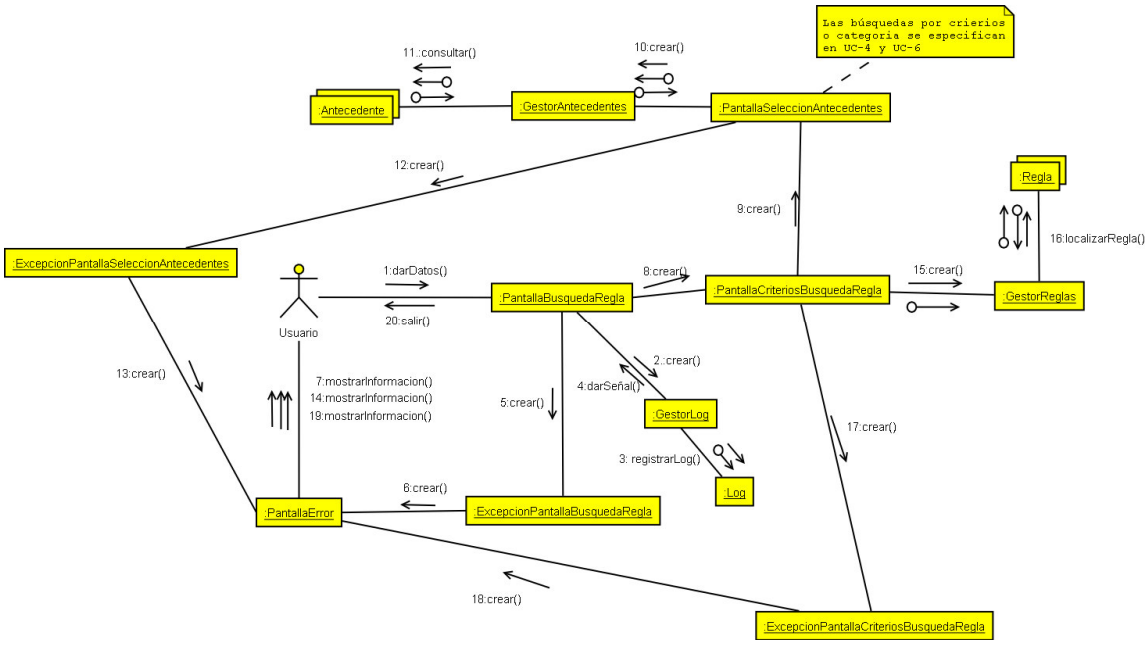

Figura 92. Diagrama de colaboración del modelo de diseño de UC-5

En la Figura 93 y en la Figura 94 se crean tres objetos diferentes correspondientes a la clase PantallaError. Con ello se modelan tres posibles excepciones que confluyen en dicha clase.

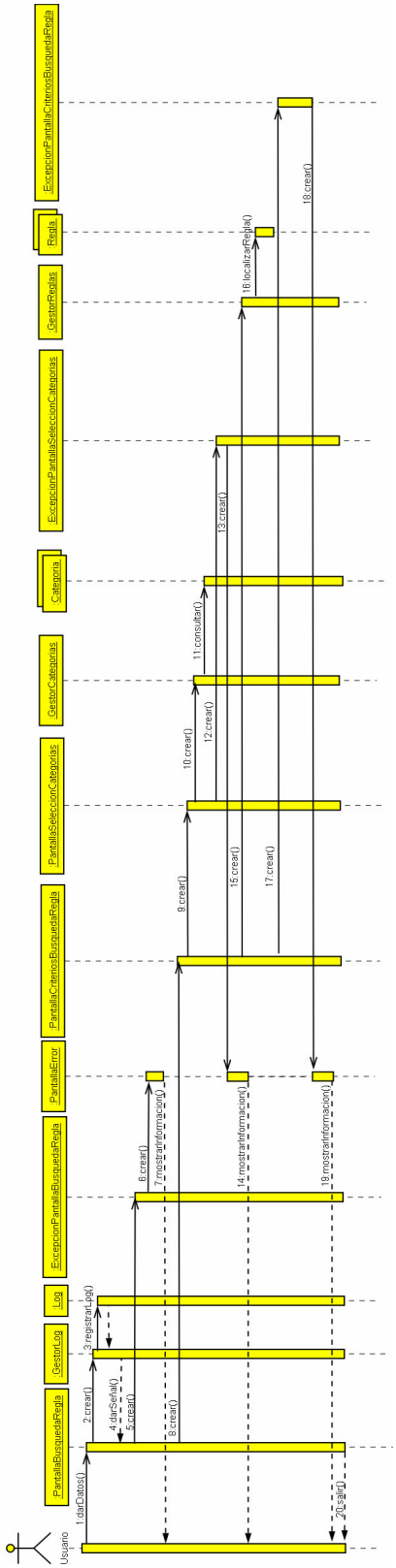

Figura 93. Diagrama de secuencia del modelo de diseño de UC-6

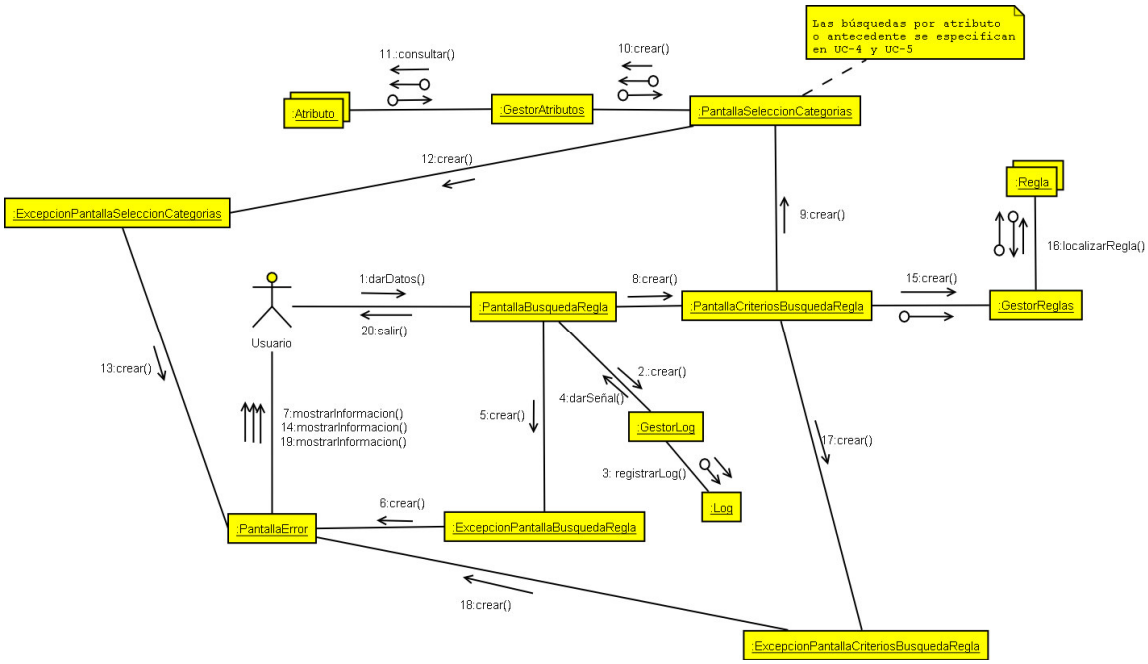

Figura 94. Diagrama de colaboración del modelo de diseño de UC-6

5.2.3. Modelo de diseño de gestión de categorías de reglas

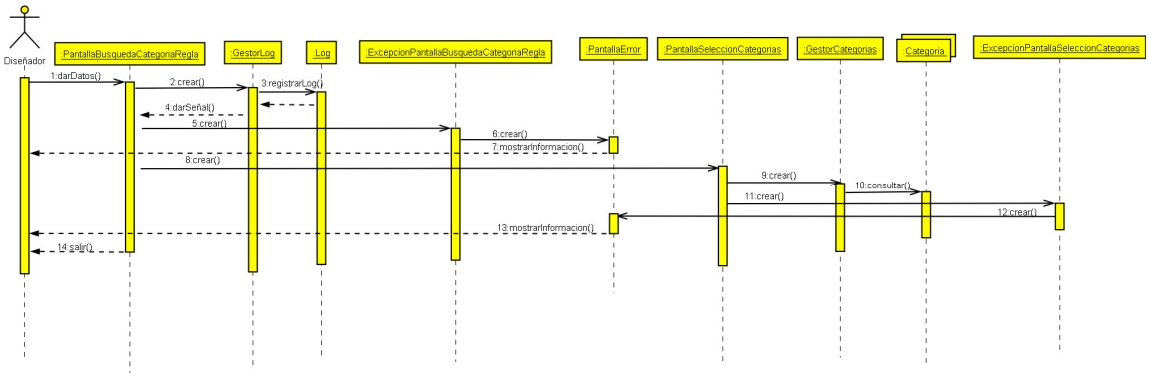

Figura 95. Diagrama de secuencia del modelo de diseño de UC-7

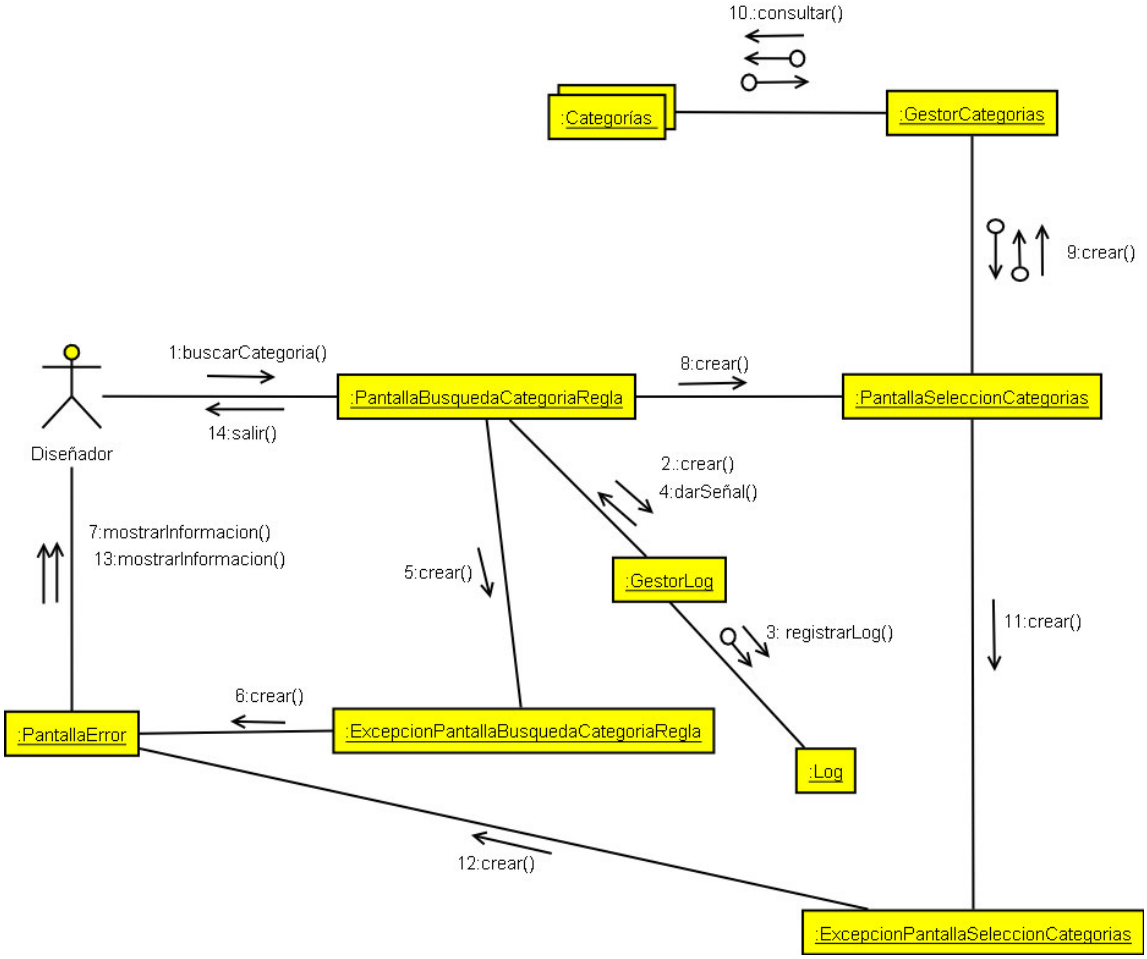

Figura 96. Diagrama de colaboración del modelo de diseño de UC-7

El alta de las categorías se efectuará desde la ventana de búsqueda de categorías, tal y como se detalla en los diagramas siguientes. De esta forma, para el acceso a la pantalla de alta de categorías se hace preciso el acceso previo a la pantalla de búsqueda de categorías.

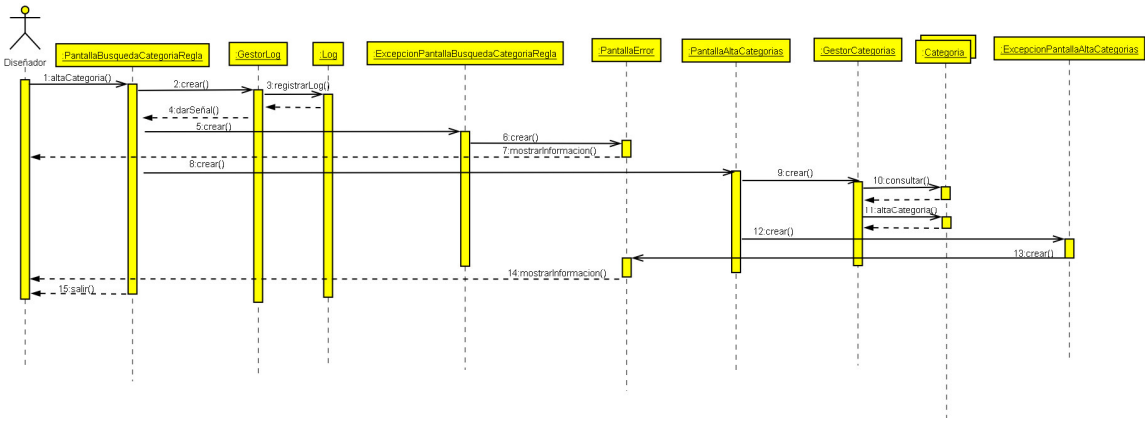

Figura 97. Diagrama de secuencia del modelo de diseño de UC-8

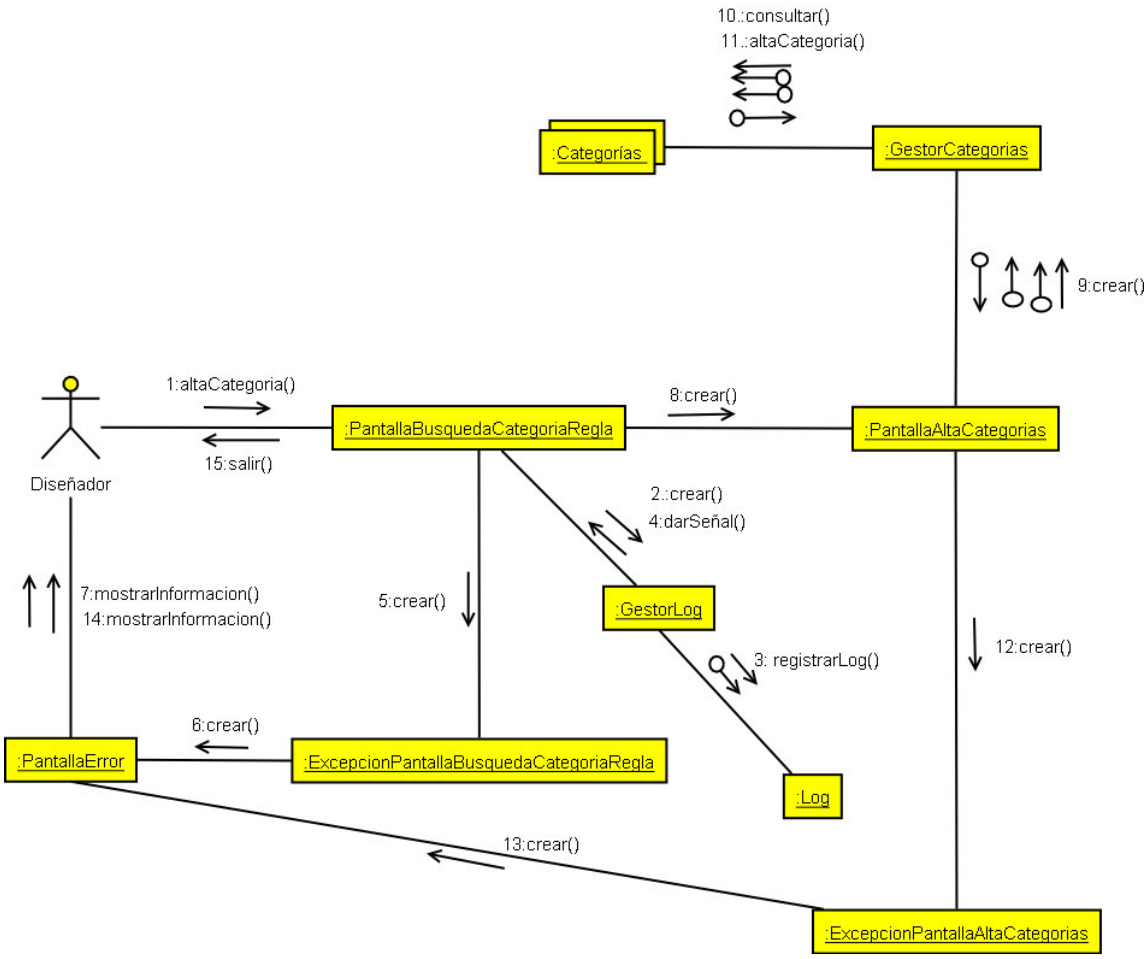

Figura 98. Diagrama de colaboración del modelo de diseño de UC-8

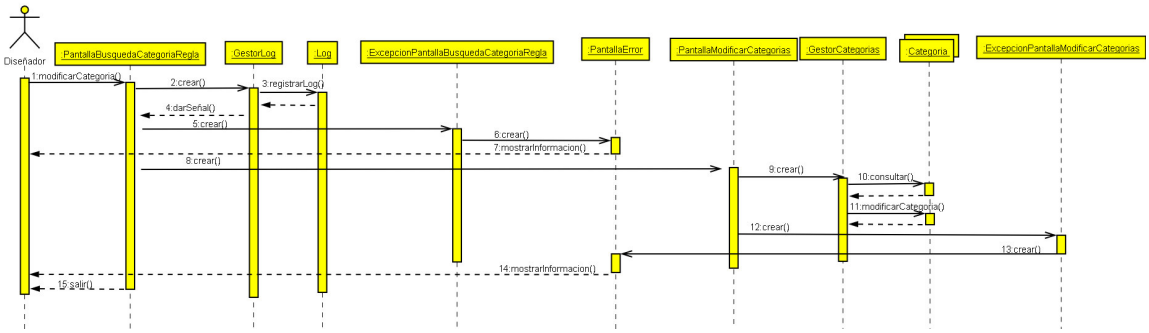

Figura 99. Diagrama de secuencia del modelo de diseño de UC-9

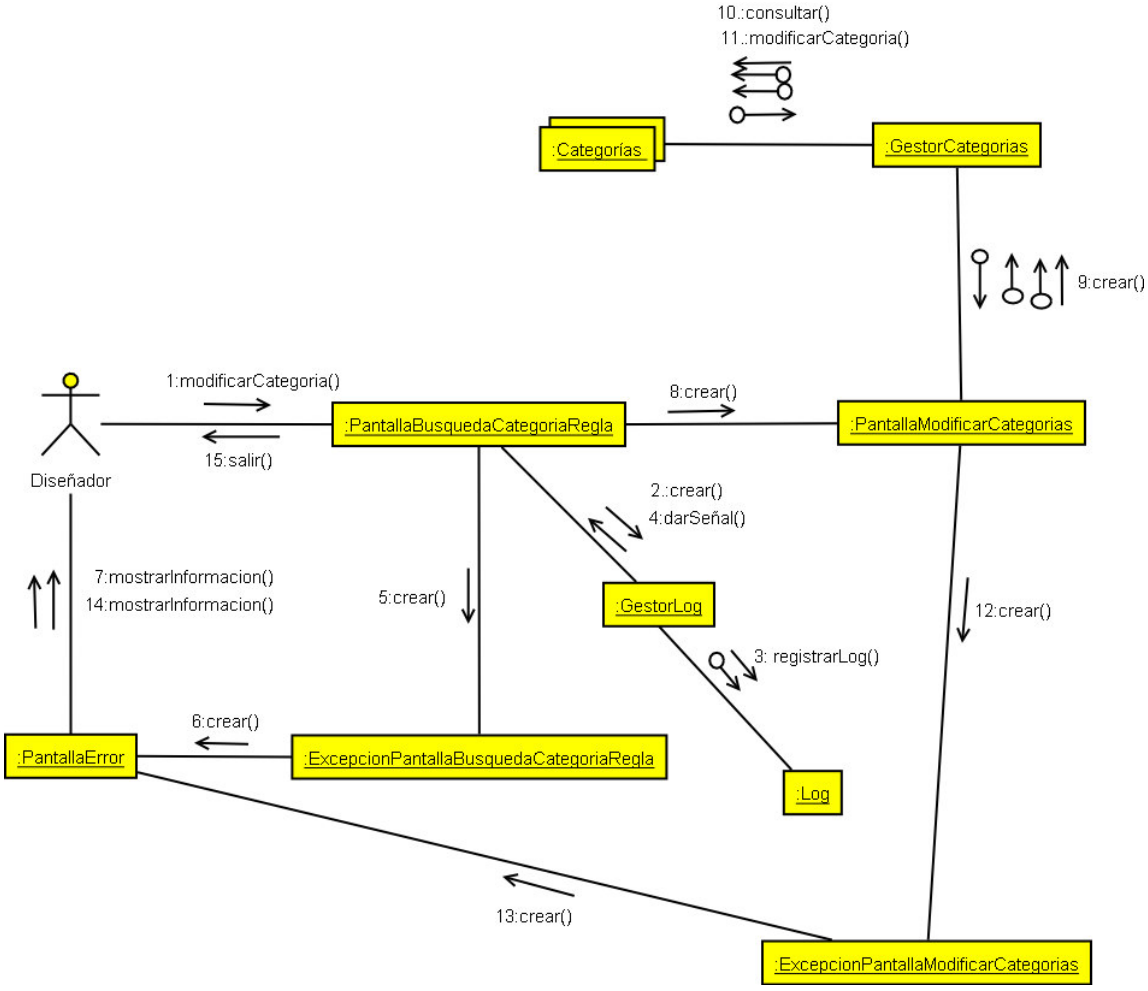

Figura 100. Diagrama de colaboración del modelo de diseño de UC-9

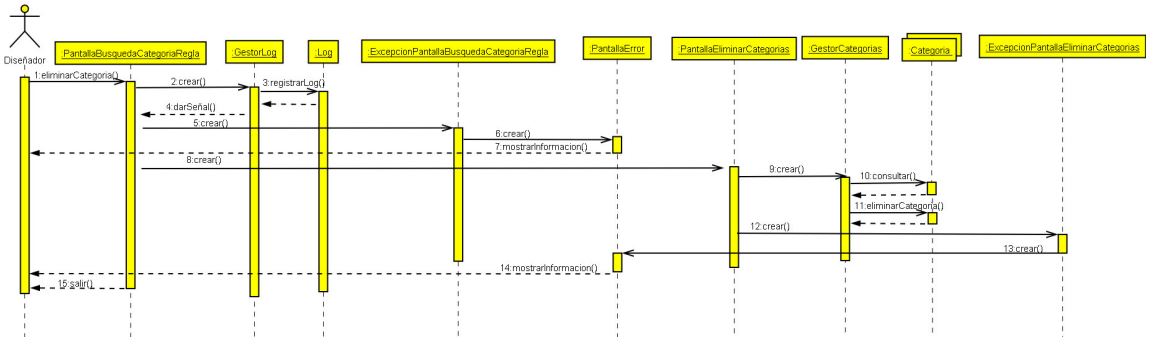

Figura 101. Diagrama de secuencia del modelo de diseño de UC-10

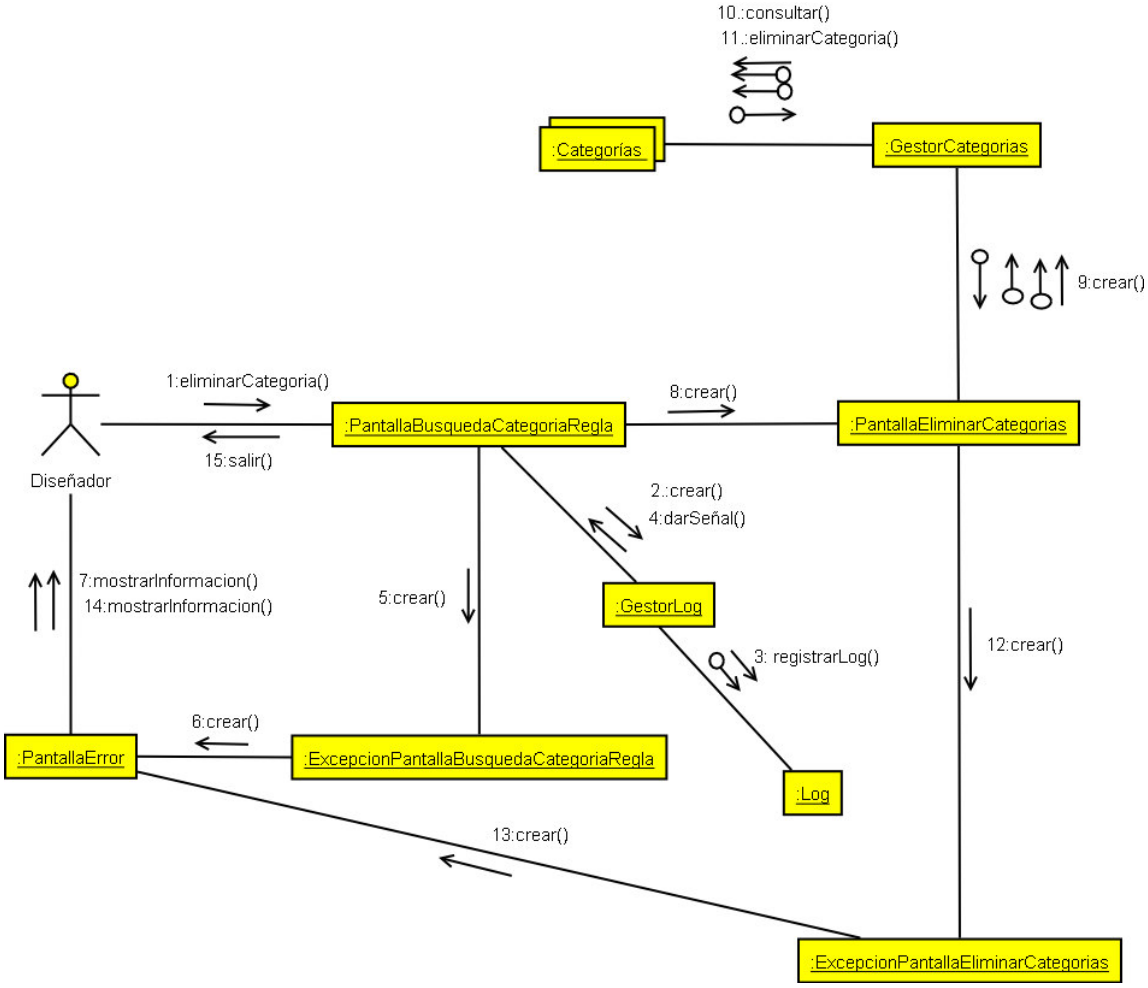

Figura 102. Diagrama de colaboración del modelo de diseño de UC-10

5.2.4. Modelo de diseño de gestión de reglas

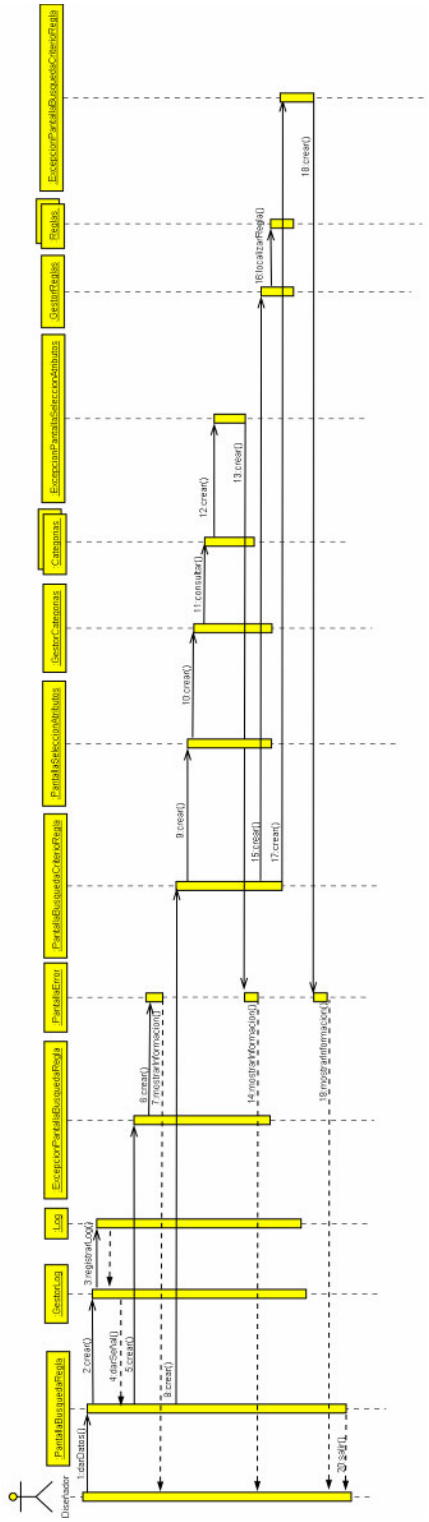

Figura 103. Diagrama de secuencia del modelo de diseño de UC-11

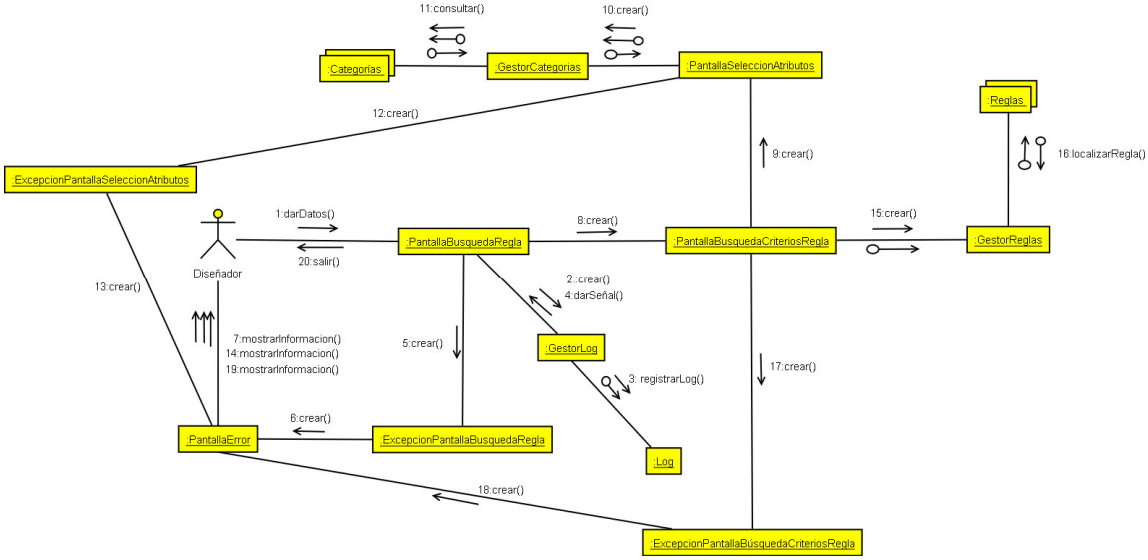

Figura 104. Diagrama de colaboración del modelo de diseño de UC-11

Las modificaciones o eliminaciones de los antecedentes, consecuentes, relaciones y categorías, que recogen la Figura 108 y la Figura 110, se llevarán a efecto siempre que no afecten a otras reglas existentes.

Luego habrá que especificar la cláusula RESTRICT en las operaciones en SQL sobre las entidades especificadas, contemplándose dicha restricción en la implementación final.

Por otro lado, las modificaciones puede producir la creación de una nueva instancia de la entidad que se está modificando.

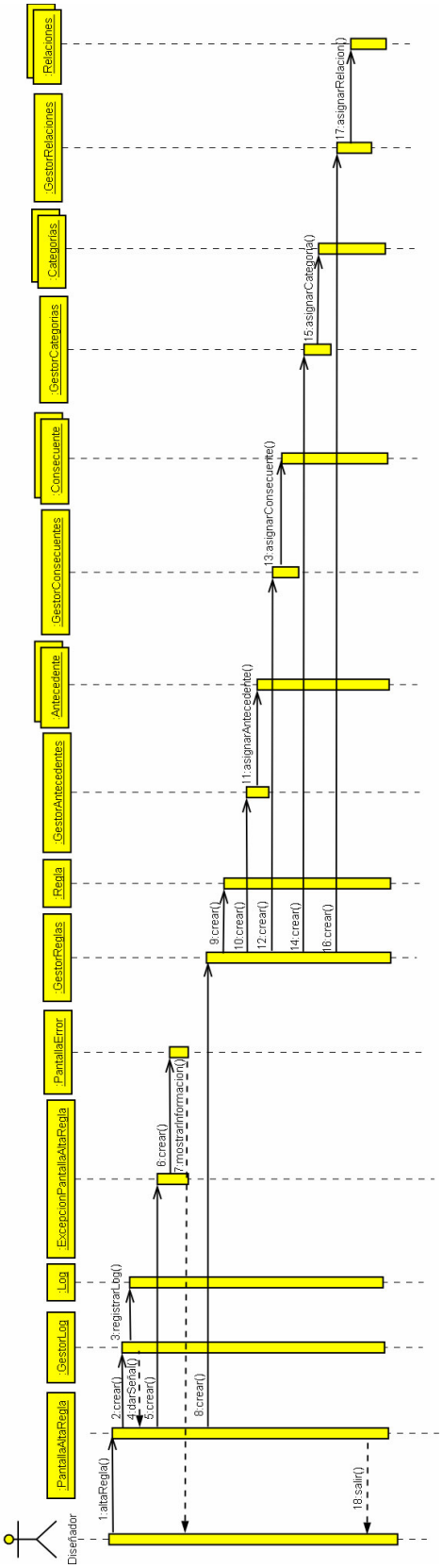

Figura 105. Diagrama de secuencia del modelo de diseño de UC-12

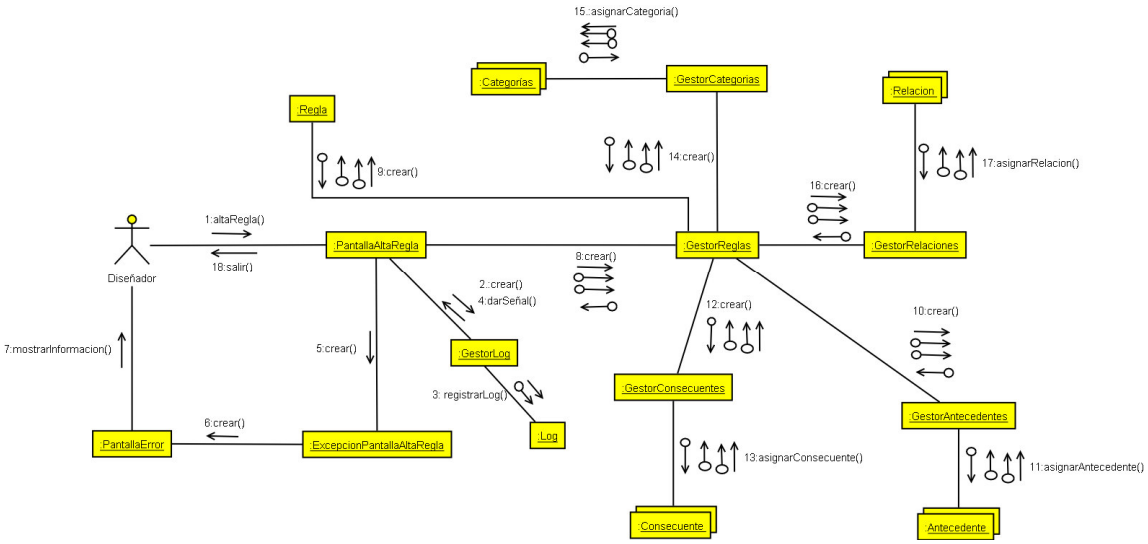

Figura 106. Diagrama de colaboración del modelo de diseño de UC-12

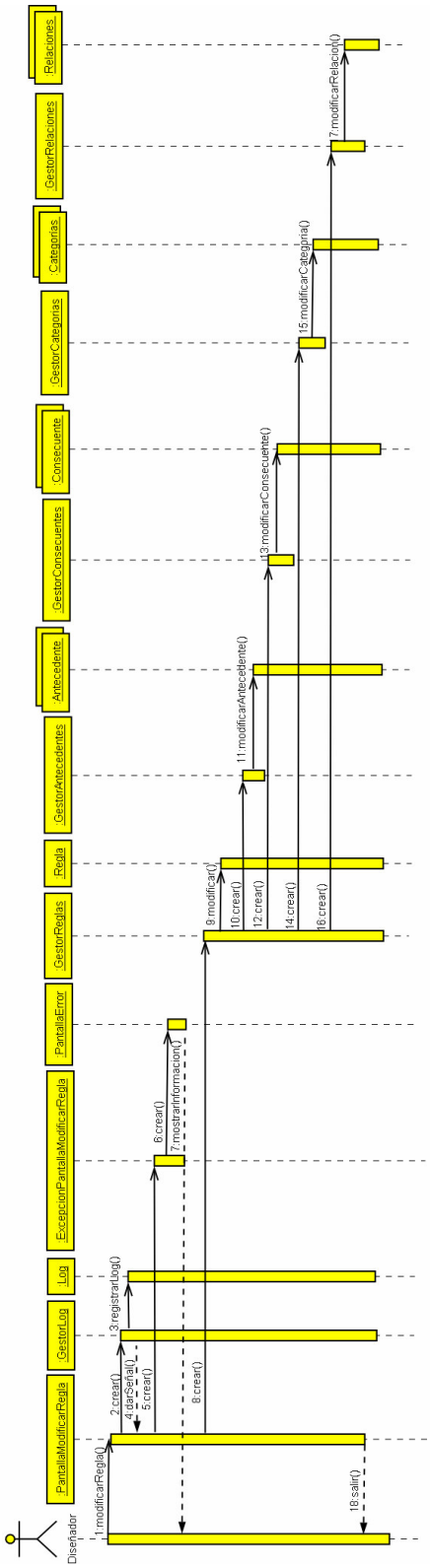

Figura 107. Diagrama de secuencia del modelo de diseño de UC-13

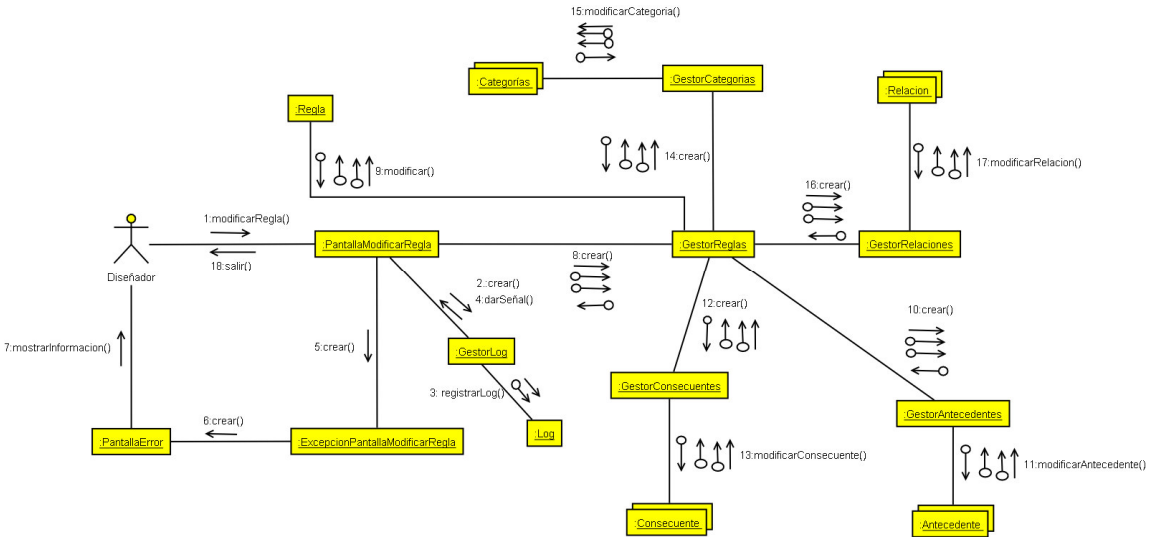

Figura 108. Diagrama de colaboración del modelo de diseño de UC-13

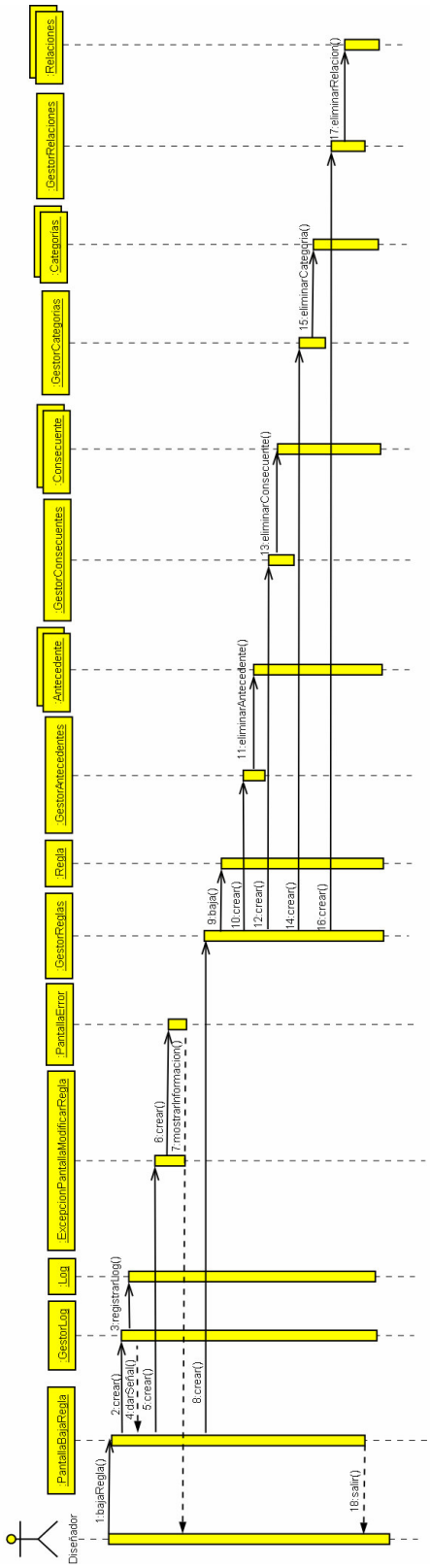

Figura 109. Diagrama de secuencia del modelo de diseño de UC-14

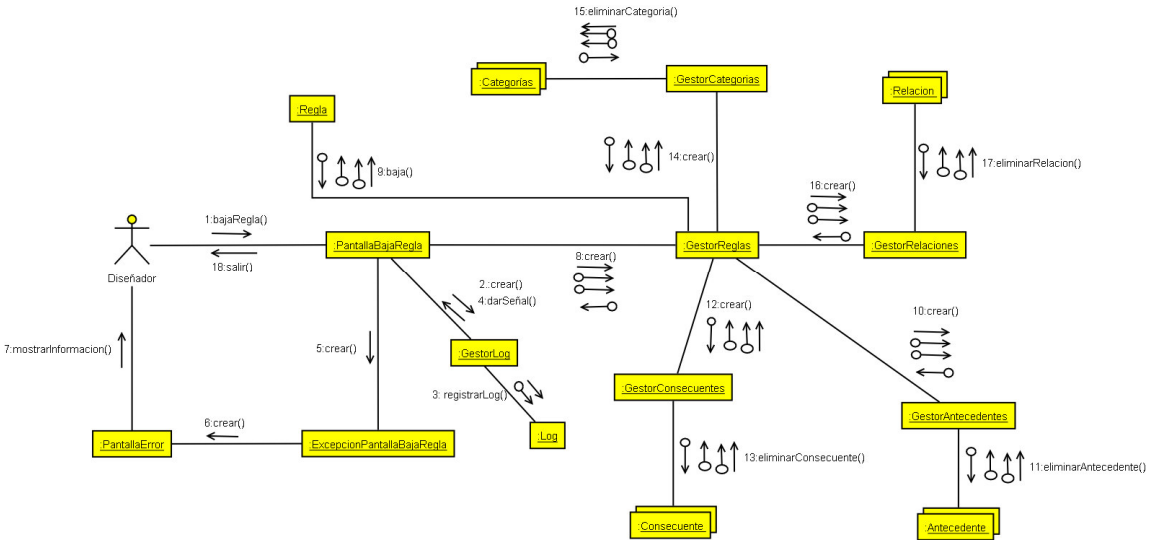

Figura 110. Diagrama de colaboración del modelo de diseño de UC-14

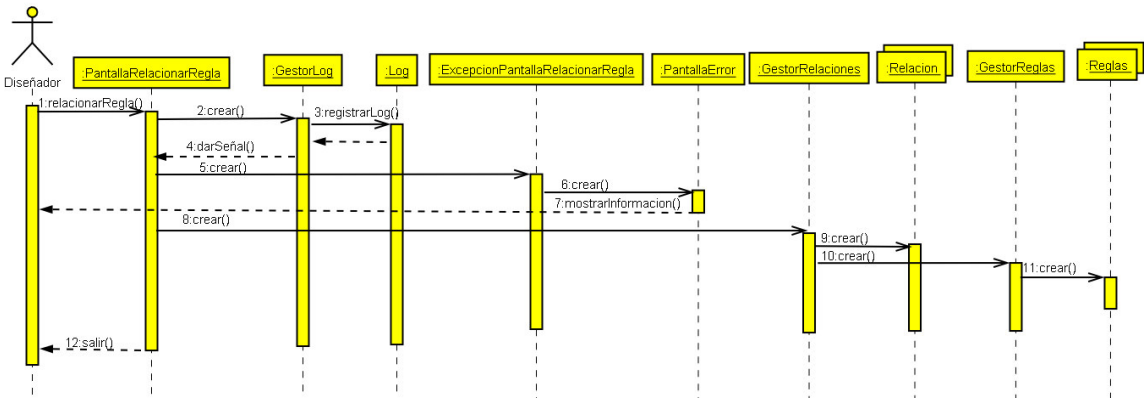

Figura 111. Diagrama de secuencia del modelo de diseño de UC-15

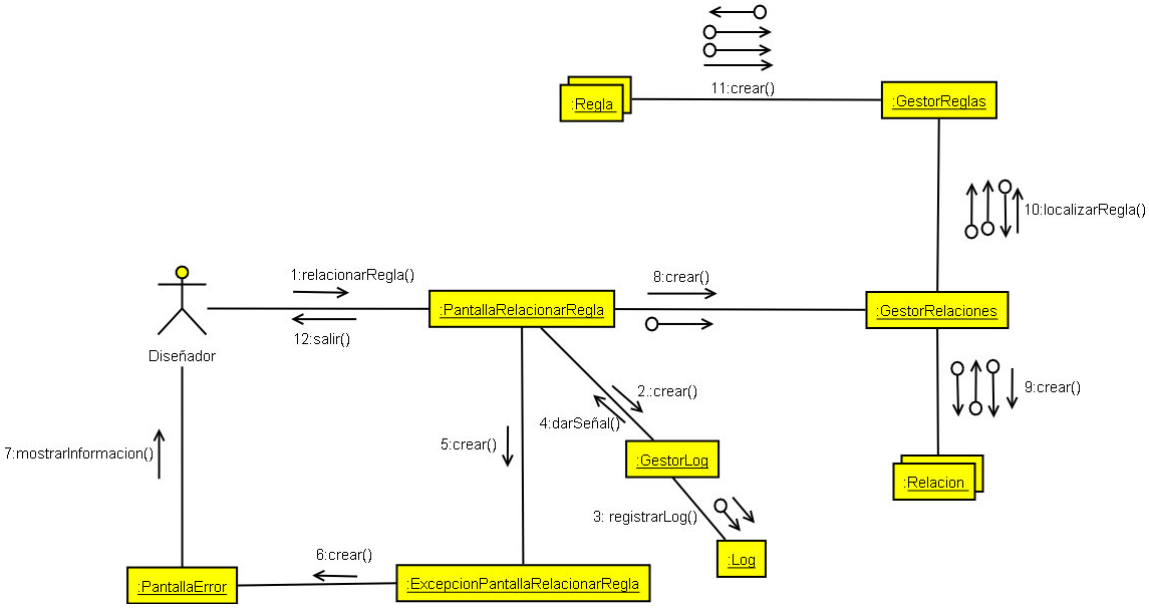

Figura 112. Diagrama de colaboración del modelo de diseño de UC-15

Los diagramas de notificaciones (Figura 113 y Figura 114) no recogen instancias de clases de pantallas, ya que el sistema se limita a notificar a los destinatarios de las notificaciones que se han especificado en el alta de los diferentes elementos susceptibles de notificación.

El actor únicamente aparece como receptor de la notificación. Por este motivo no se contemplan instancias de clases de excepciones, las cuales hemos vinculado a las clases de ventanas. Tampoco se presentan instancias de clases relacionadas con Logs, ya que la secuencia se lleva a cabo independientemente de que un usuario disponga de una sesión abierta o no.

Otro punto a destacar reside en que el actor no es el diseñador, como ocurre con el resto de los actores de este paquete, sino que se trata del usuario. Ello se debe a que todos los actores pueden recibir algún tipo de notificación del sistema.

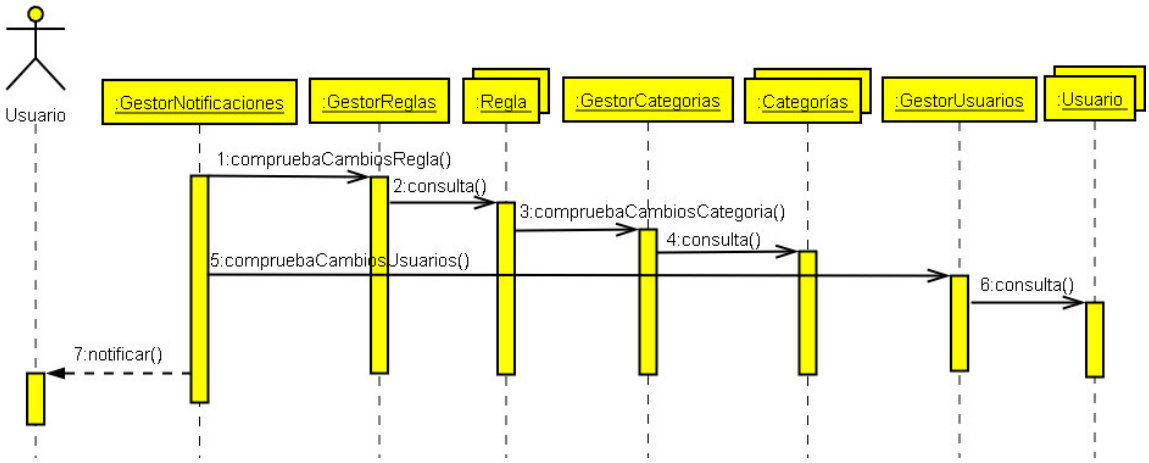

Figura 113. Diagrama de secuencia del modelo de diseño de UC-16

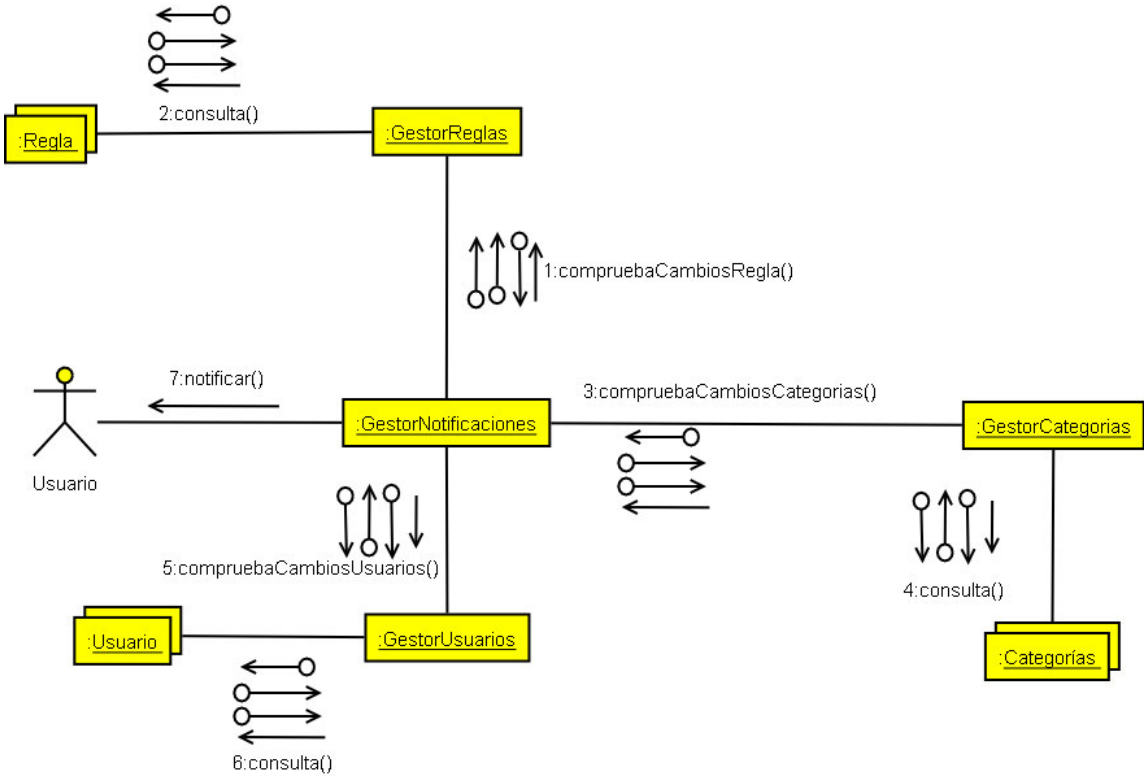

Figura 114. Diagrama de colaboración del modelo de diseño de UC-16

5.2.5. Modelo de diseño de gestión de usuarios

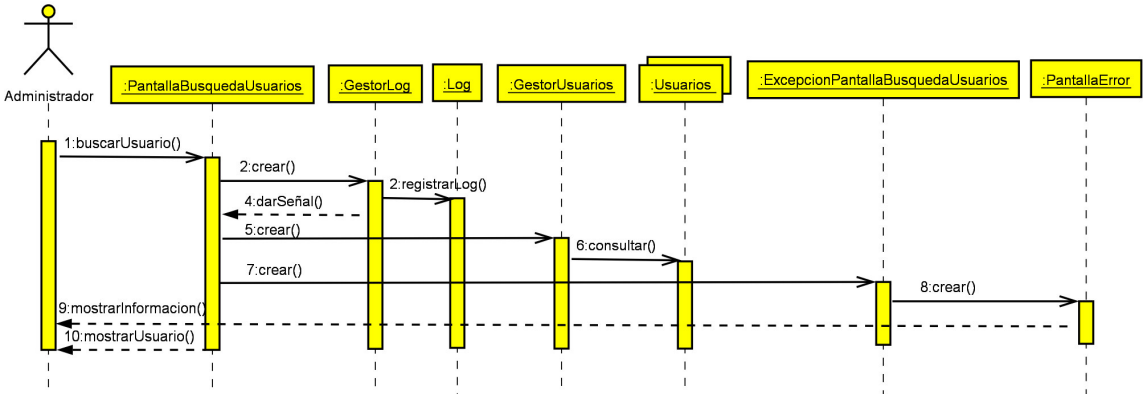

Figura 115. Diagrama de secuencia del modelo de diseño de UC-17

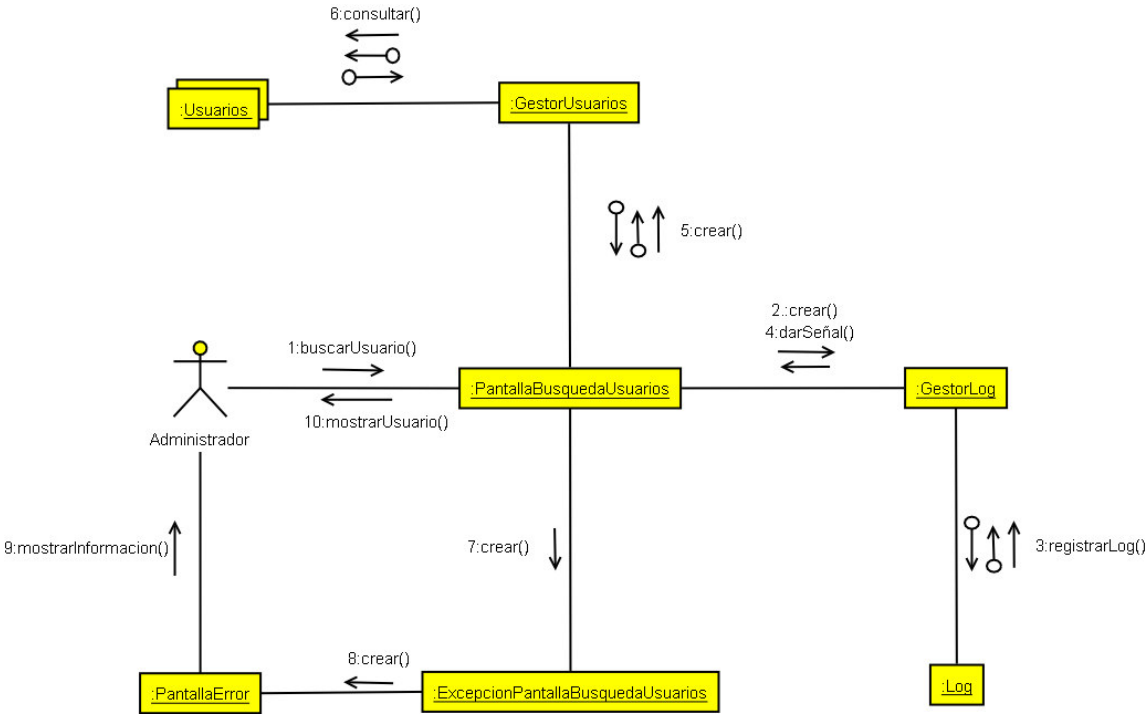

Figura 116. Diagrama de colaboración del modelo de diseño de UC-17

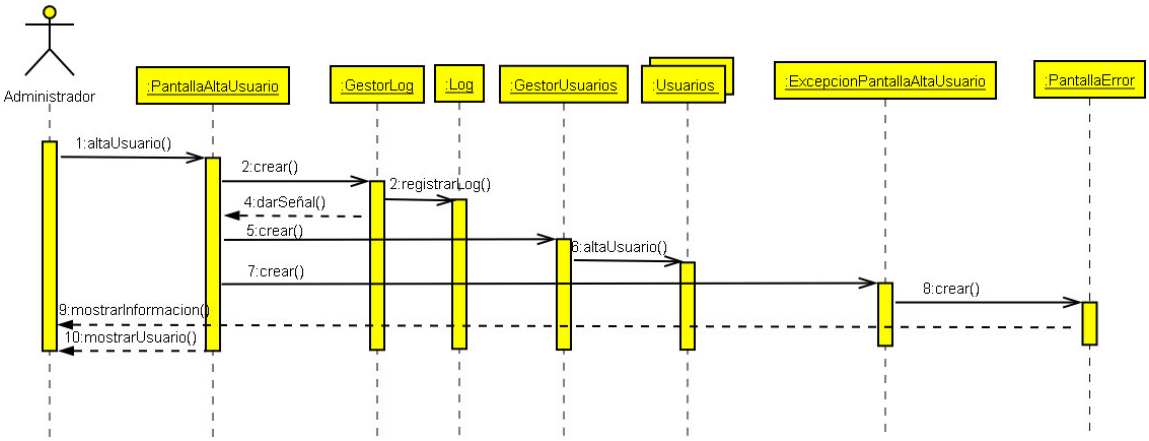

Figura 117. Diagrama de secuencia del modelo de diseño de UC-18

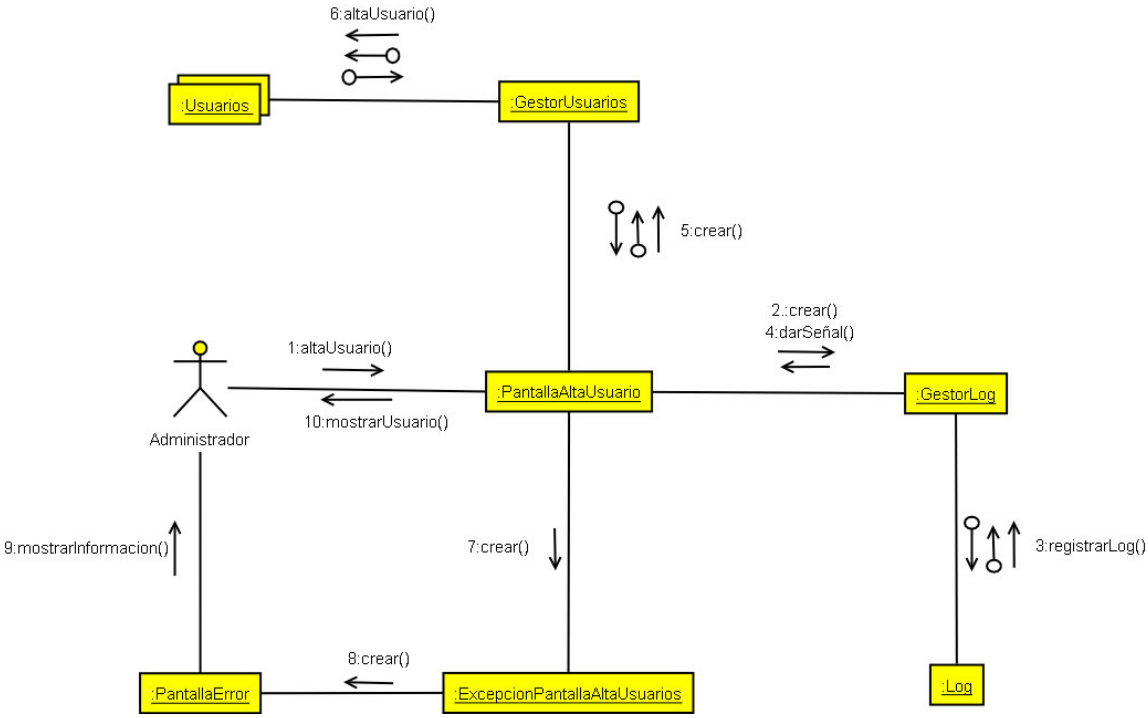

Figura 118. Diagrama de colaboración del modelo de diseño de UC-18

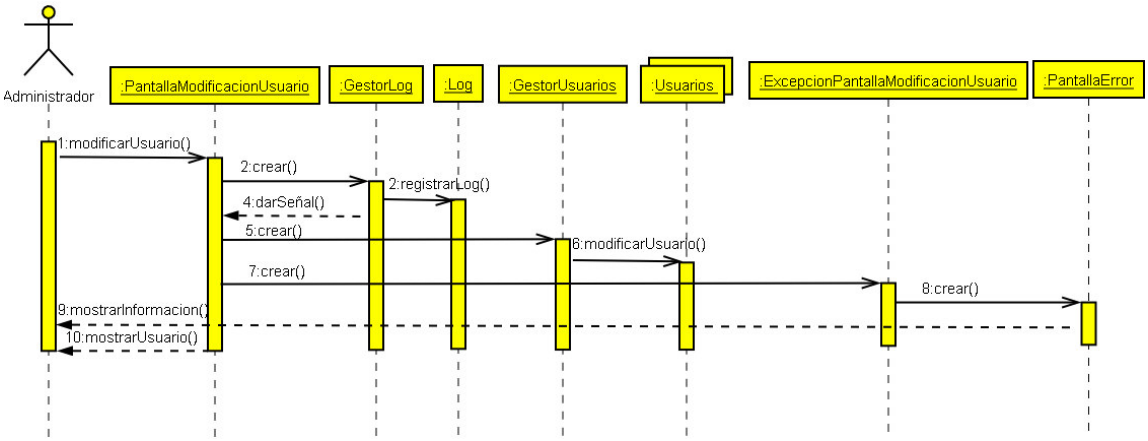

Figura 119. Diagrama de secuencia del modelo de diseño de UC-19

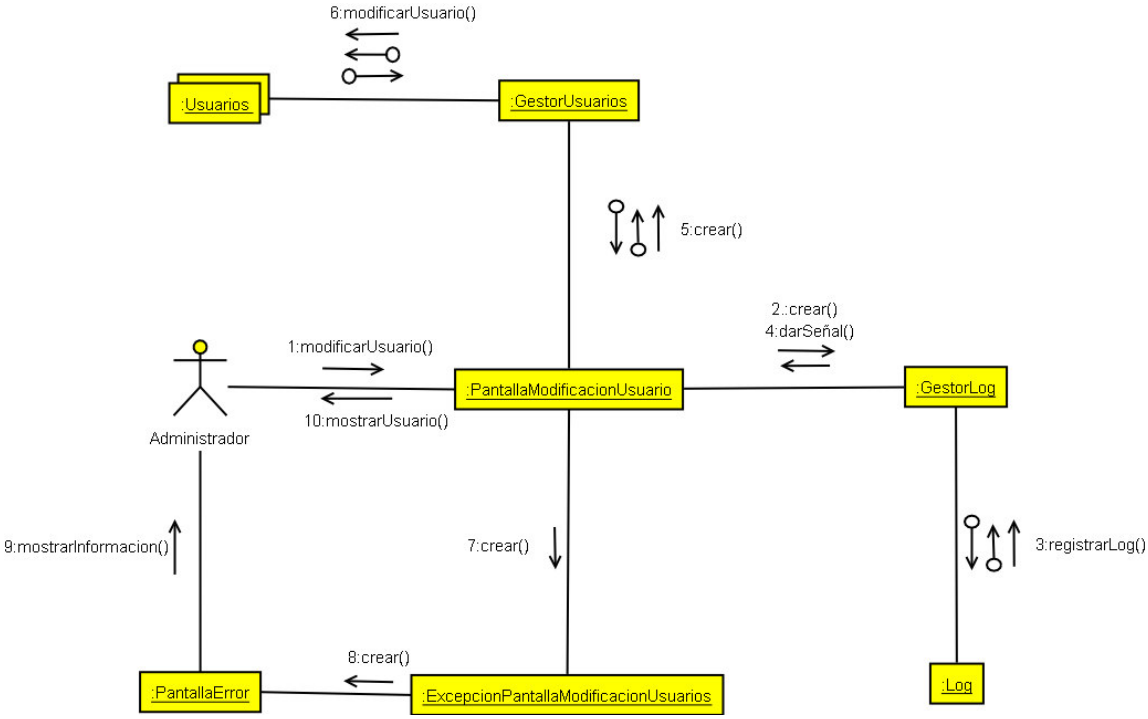

Figura 120. Diagrama de colaboración del modelo de diseño de UC-19

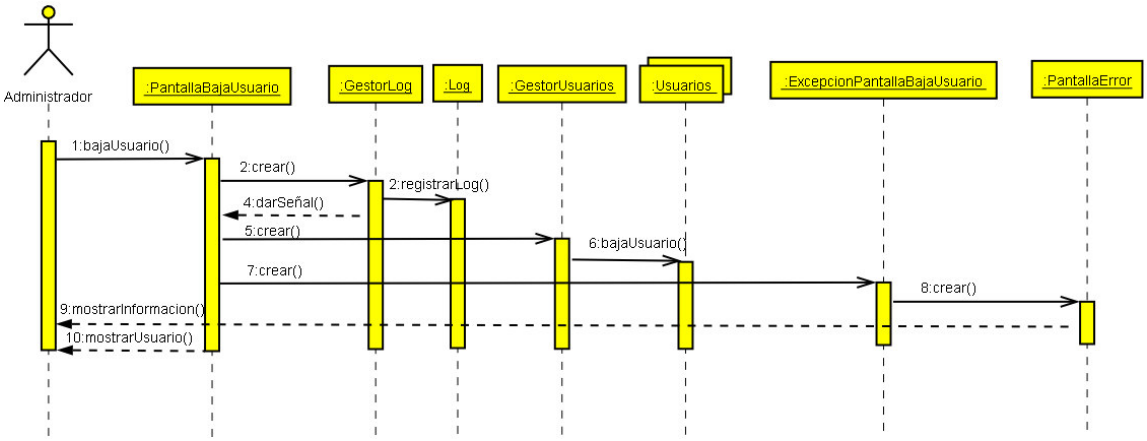

Figura 121. Diagrama de secuencia del modelo de diseño de UC-20

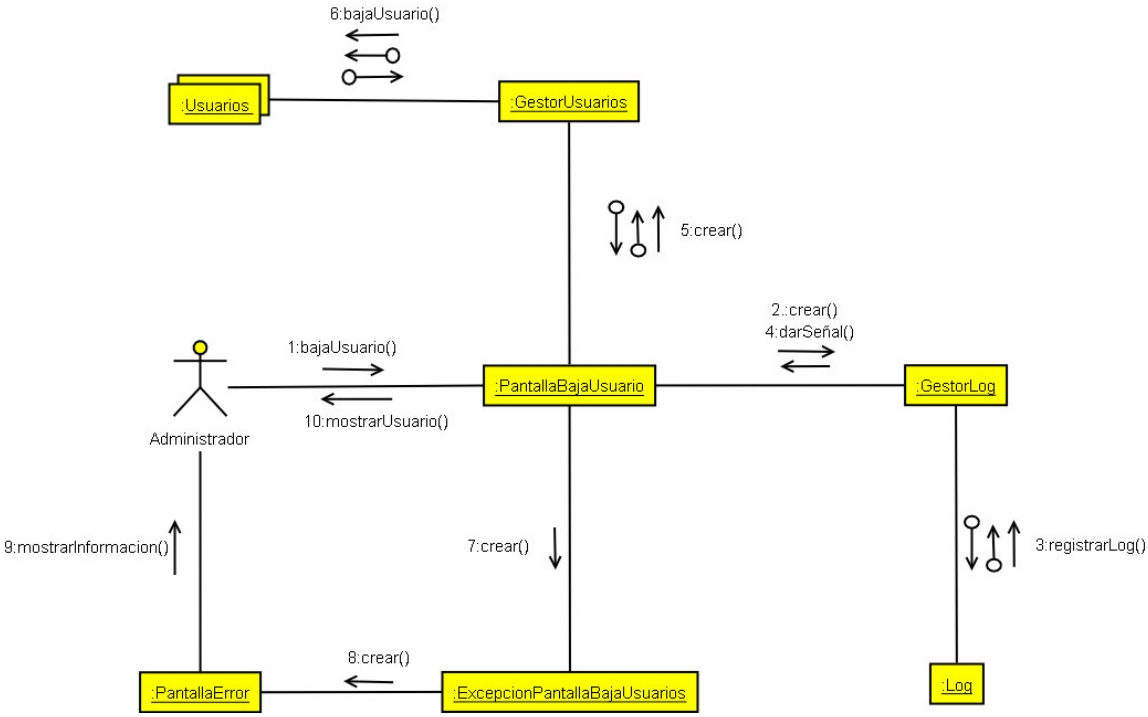

Figura 122. Diagrama de colaboración del modelo de diseño de UC-20

5.3. Diseño de la persistencia

En este apartado se especifica la persistencia contenida en el paquete Persistencia del diagrama de paquetes (Figura 82). El paquete Especificación persistencia no se especifica, ya que contiene básicamente la definición del mecanismo de persistencia mediante JDBC y el intercambio de ficheros desde el dispositivo móvil al servidor. Ambas cuestiones pertenecen al dominio de la construcción, por lo que se escapa al objetivo de este TFC.

Para la especificación de la persistencia se utiliza el modelo Entidad-Relación (ER). El modelo ER [COS05] es uno de los enfoques de modelado de datos que más se utiliza actualmente por su simplicidad y legibilidad. Su legibilidad se ve favorecida porque proporciona una notación diagramática muy comprensiva.

El diagrama ER se recoge en la Figura 123. Concretamente las entidades detectadas se detallan en la Tabla 32.

Tabla 32. Entidades persistentes

Entidad	Descripción
Persona	Superclase de Usuario, Administrador y Diseñador. Recoge los atributos y operaciones comunes a todas las personas en el sistema.
Usuario	Subclase de Persona. Encapsula al usuario básico del motor.
Administrador	Subclase de Persona. Encapsula a la persona que administra los usuarios básicos del motor.
Diseñador	Subclase de Persona. Encapsula a la persona que administra la reglas, categorías, relaciones y notificaciones.
Departamento	Entidad que agrupa usuarios.
Categoria	Entidad que agrupa reglas.
Componente	Superclase de Antecedente y Consecuente. Agrupa elementos que componen parte de la regla IF-THEN. Pueden existir componentes repetidos en ambas subclases.
Antecedente	Subclase de componente. Engloba al componente inicial de una regla.
Consecuente	Subclase de componente. Engloba al componente final de una regla.
Regla	Se ha modelado como una entidad asociativa entre los componentes, las categorías, y las notificaciones utilizadas por los usuarios y gestionadas por los diseñadores. La entidad que resulta de considerar una interrelación entre entidades como si fuese una entidad es una entidad asociativa, y tendrá el mismo nombre que la interrelación sobre la que se define. La utilidad de una entidad asociativa consiste en que se puede interrelacionar con otras entidades y, de forma indirecta, nos permite tener interrelaciones en las que intervienen interrelaciones. Asimismo, se modela una interrelación recursiva para recoger las relaciones entre Reglas.
Notificación	Se ha modelado como una entidad asociativa entre las reglas y las personas. La notificación es opcional para las reglas, que pueden ser notificadas o no. No obstante, si se notifica la presencia de una entidad Persona es ineludible.

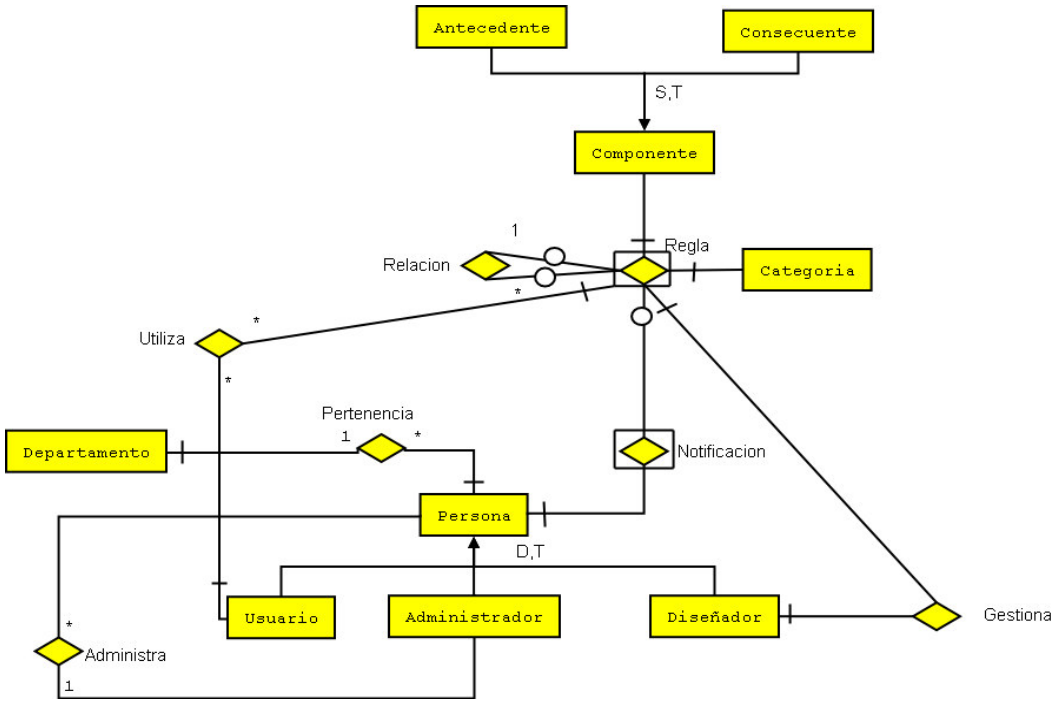

Figura 123. Diseño de la persistencia

5.4. Diseño de Interfaces Gráficas de Usuario

En la Figura 124 se recoge la jerarquía de pantallas. Las clases abstractas no se diseñarán, ya que no tendrán instancias.

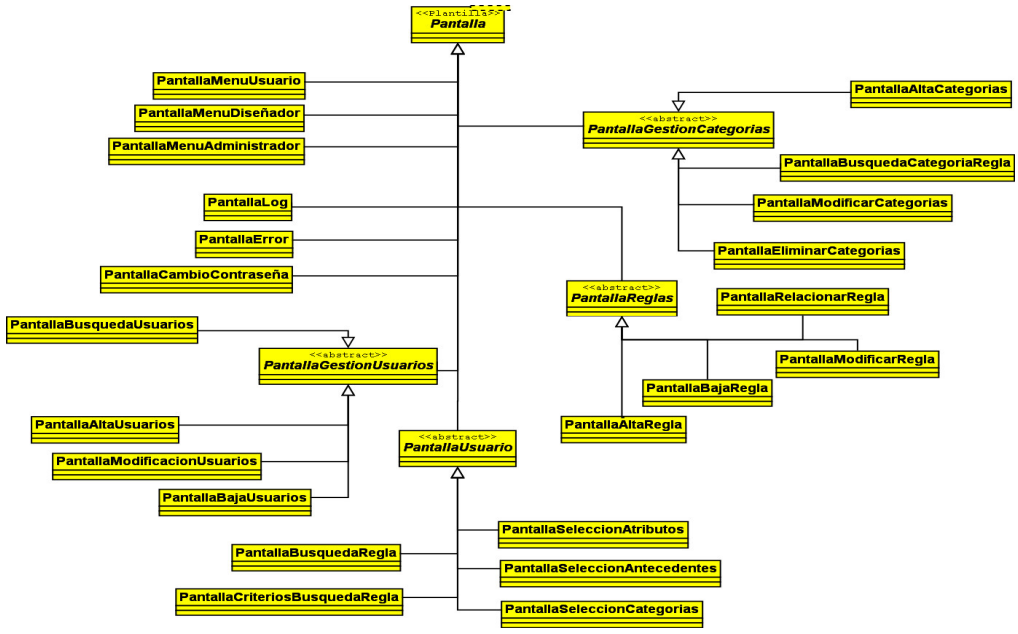

Figura 124. Jerarquía de pantallas

El diseño de las interfaces gráficas de usuario se realizará usando el Netbeans mobility pack, que es un plug-in de Netbeans 5.0, el cual se recoge en la Figura 125. En el citado entorno aparecen a la derecha del diseño de la pantalla los comandos asignados. Un aspecto a tener en cuenta es que, en los dispositivos móviles existen comandos asignados a botones físicos del dispositivo. Por ello, no es preciso incluirlos en el GUI.

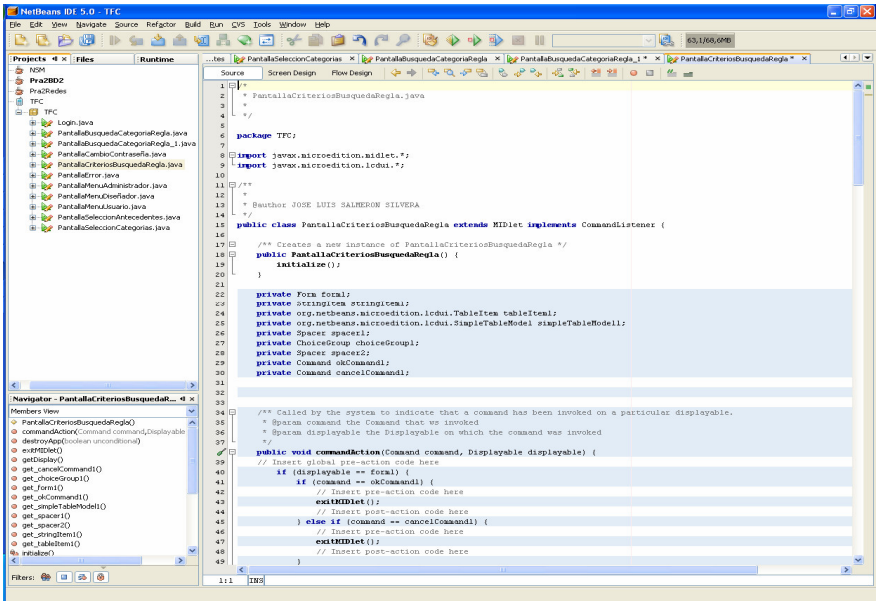

Figura 125. Netbeans 5.0 con mobility pack 5.0 para CDLC

En los siguientes epígrafes se recoge el diseño del GUI, el diseño del flujo entre pantallas de un mismo MIDlet y el código fuente de éste. No obstante, el código que se ha incluido únicamente hace referencia a los GUIs y a su flujo. No se ha completado ningún código relacionado con la construcción.

5.4.1. PantallaLog

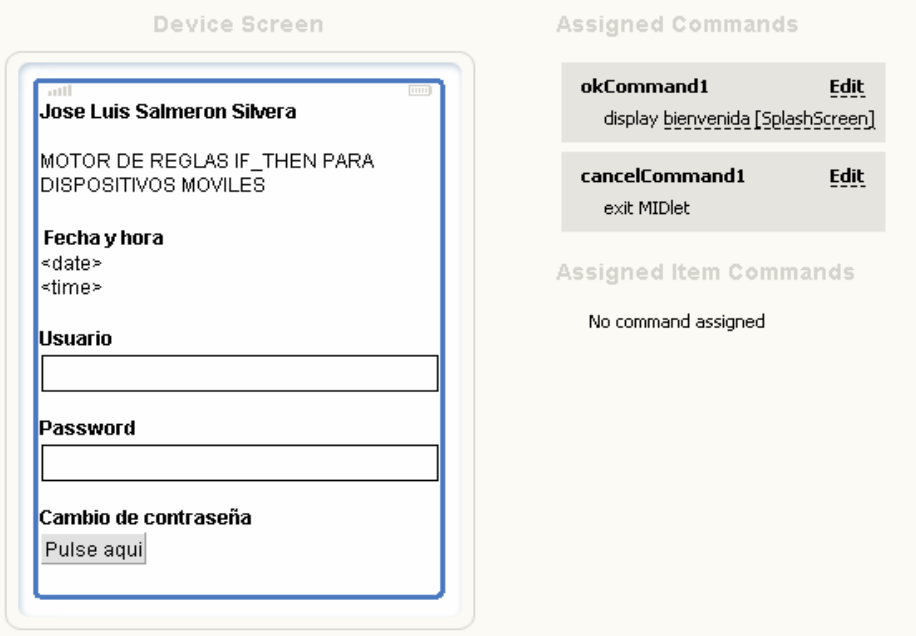

Figura 126. Diseño de GUI PantallaLog

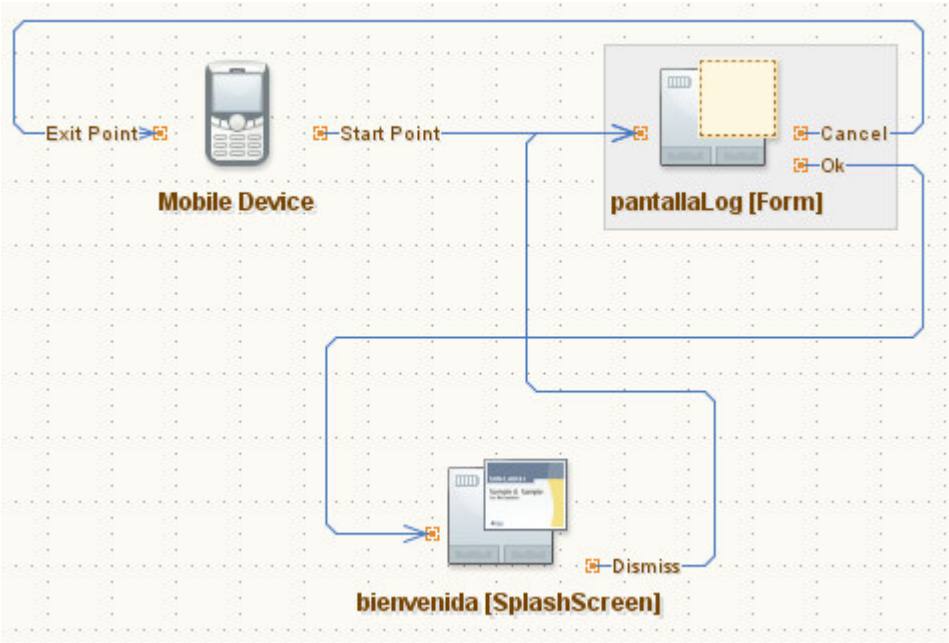

Figura 127. Diseño de flujo de PantallaLog

Tabla 33. Código fuente del MIDlet de PantallaLog

```
/*
 * PantallaLog.java
 *
 */

package TFC;
```

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaLog extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaLog */
    public PantallaLog () {
        initialize();
    }

    private Form pantallaLog;
    private TextField textField1;
    private TextField textField2;
    private StringItem stringItem1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Command screenCommand1;
    private Command itemCommand1;
    private DateField dateField1;
    private Image image1;
    private StringItem stringItem2;
    private org.netbeans.microedition.lcdui.SplashScreen bienvenida;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == pantallaLog) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_bienvenida());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_pantallaLog());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
```

```
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for pantallaLog component and should be called
instead of accessing pantallaLog field directly.
     * @return Instance for pantallaLog component
     */
    public Form get_pantallaLog() {
        if (pantallaLog == null) {
            // Insert pre-init code here
            pantallaLog = new Form(null, new Item[] {
                get_stringItem1(),
                get_dateField1(),
                get_textField1(),
                get_textField2(),
                get_stringItem2()
            });
            pantallaLog.addCommand(get_okCommand1());
            pantallaLog.addCommand(get_cancelCommand1());
            pantallaLog.setCommandListener(this);
            // Insert post-init code here
        }
        return pantallaLog;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("  \nUsuario", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for textField2 component and should be called
instead of accessing textField2 field directly.
     * @return Instance for textField2 component
     */
    public TextField get_textField2() {
        if (textField2 == null) {
            // Insert pre-init code here
            textField2 = new TextField("   \nPassword", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField2;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Jose Luis Salmeron Silvera", "  \nMOTOR DE
REGLAS IF_THEN PARA DISPOSITIVOS MOVILES\n  ");
            stringItem1.setLayout(Item.LAYOUT_CENTER | Item.LAYOUT_VCENTER |
Item.LAYOUT_NEWLINE_BEFORE | Item.LAYOUT_NEWLINE_AFTER | Item.LAYOUT_SHRINK |
Item.LAYOUT_VSHRINK | Item.LAYOUT_EXPAND | Item.LAYOUT_VEXPAND);
            // Insert post-init code here
        }
        return stringItem1;
    }
```

```
    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }
    /** This method returns instance for itemCommand1 component and should be called
instead of accessing itemCommand1 field directly.
     * @return Instance for itemCommand1 component
     */
    public Command get_itemCommand1() {
        if (itemCommand1 == null) {
            // Insert pre-init code here
            itemCommand1 = new Command("Item", Command.ITEM, 1);
            // Insert post-init code here
        }
        return itemCommand1;
    }

    /** This method returns instance for dateField1 component and should be called
instead of accessing dateField1 field directly.
     * @return Instance for dateField1 component
     */
    public DateField get_dateField1() {
        if (dateField1 == null) {
            // Insert pre-init code here
            dateField1 = new DateField(" Fecha y hora", DateField.DATE_TIME);
            // Insert post-init code here
        }
        return dateField1;
    }

    /** This method returns instance for image1 component and should be called
instead of accessing image1 field directly.
     * @return Instance for image1 component
     */
    public Image get_image1() {
        if (image1 == null) {
```

```
            // Insert pre-init code here
            try {
                image1 = Image.createImage("<No Image>");
            } catch (java.io.IOException exception) {
            }
            // Insert post-init code here
        }
        return image1;
    }

    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("  \nCambio de contrase\u00F1a", "Pulse
aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }

    /** This method returns instance for bienvenida component and should be called
instead of accessing bienvenida field directly.
     * @return Instance for bienvenida component
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_bienvenida() {
        if (bienvenida == null) {
            // Insert pre-init code here
            bienvenida = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            bienvenida.setNextDisplayable(get_pantallaLog());
            // Insert post-init code here
        }
        return bienvenida;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.2. PantallaCambioContraseña

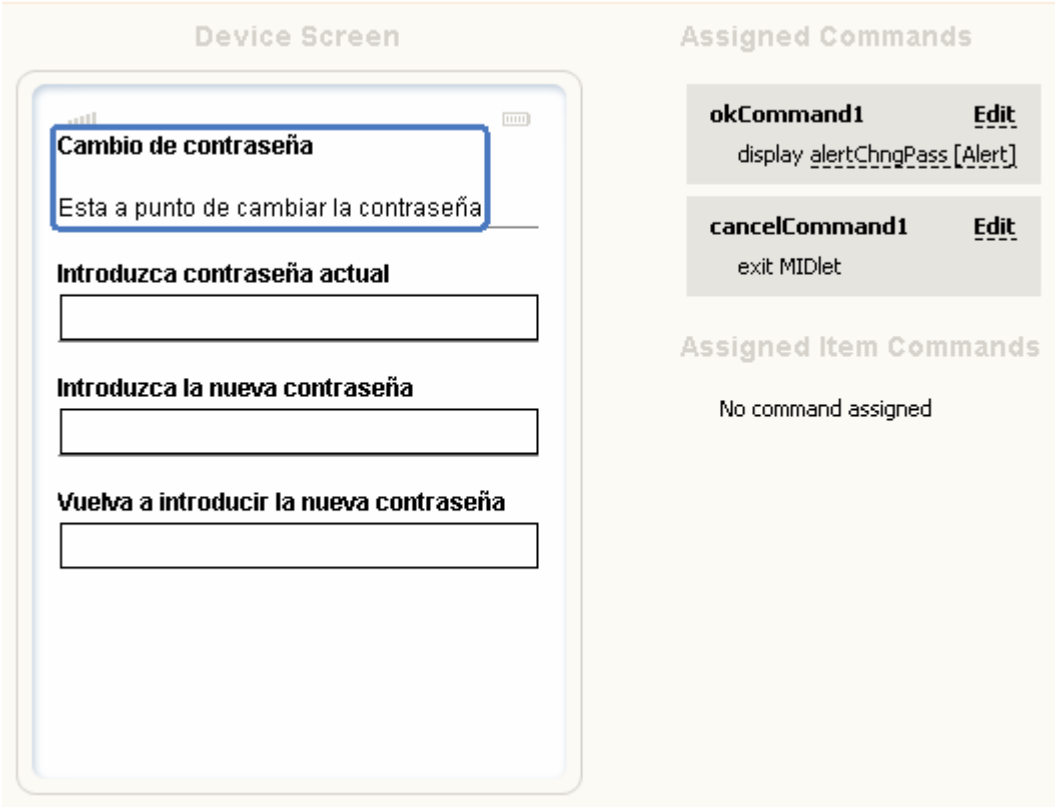

Figura 128. Diseño de GUI PantallaCambioContraseña

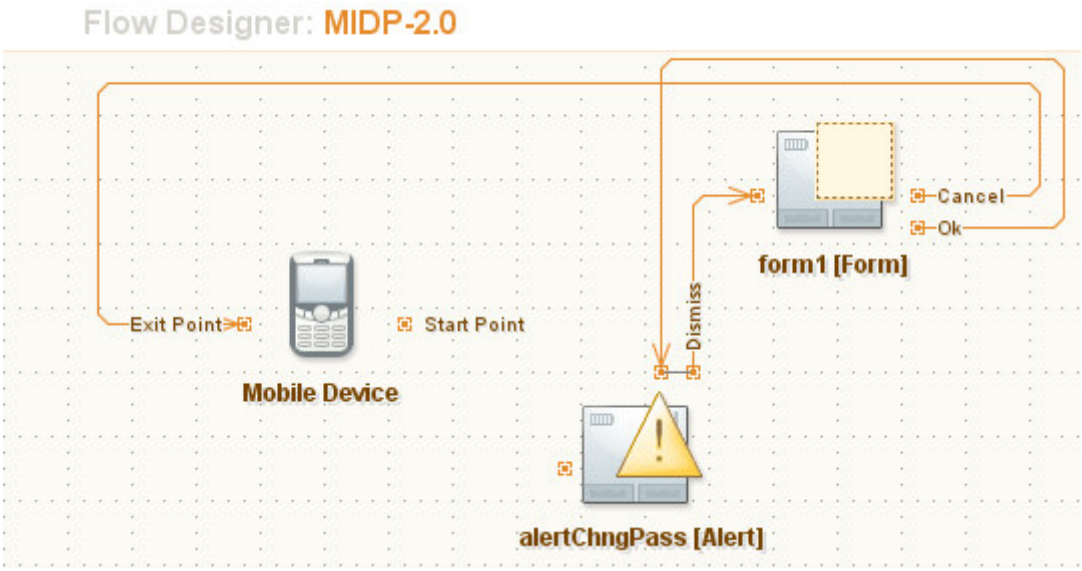

Figura 129. Diseño de flujo de PantallaCambioContraseña

Tabla 34. Código fuente del MIDlet de PantallaCambioContraseña

```
/*
 * PantallaCambioContrasena.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaCambioContrasena extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaCambioContrasena */
    public PantallaCambioContrasena() {
        initialize();
    }

    private Form form1;
    private TextField textField1;
    private TextField textField2;
    private TextField textField3;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer1;
    private Alert alertChngPass;
    private Spacer spacer2;
    private Command itemCommand1;
    private StringItem stringItem1;
    private Spacer spacer3;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_alertChngPass(), get_form1());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here

        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
```

```
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer3(),
                get_textField1(),
                get_spacer1(),
                get_textField2(),
                get_spacer2(),
                get_textField3()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("    \nIntroduzca contrase\u00F1a actual",
null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for textField2 component and should be called
instead of accessing textField2 field directly.
     * @return Instance for textField2 component
     */
    public TextField get_textField2() {
        if (textField2 == null) {
            // Insert pre-init code here
            textField2 = new TextField("    \nIntroduzca la nueva contrase\u00F1a",
null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField2;
    }

    /** This method returns instance for textField3 component and should be called
instead of accessing textField3 field directly.
     * @return Instance for textField3 component
     */
    public TextField get_textField3() {
        if (textField3 == null) {
```

```
            // Insert pre-init code here
            textField3 = new TextField("    \nVuelva a introducir la nueva
contrase\u00F1a", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField3;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for alertChngPass component and should be called
instead of accessing alertChngPass field directly.
     * @return Instance for alertChngPass component
     */
    public Alert get_alertChngPass() {
        if (alertChngPass == null) {
            // Insert pre-init code here
            alertChngPass = new Alert(null, "<Enter Text>", null, AlertType.INFO);
            alertChngPass.setTimeout(-2);
            // Insert post-init code here
        }
        return alertChngPass;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
```

```
        }
        return spacer2;
    }

    /** This method returns instance for itemCommand1 component and should be called
instead of accessing itemCommand1 field directly.
     * @return Instance for itemCommand1 component
     */
    public Command get_itemCommand1() {
        if (itemCommand1 == null) {
            // Insert pre-init code here
            itemCommand1 = new Command("Item", Command.ITEM, 1);
            // Insert post-init code here
        }
        return itemCommand1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Cambio de contrase\u00F1a", "   \nEsta a
punto de cambiar la contrase\u00F1a");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.3. PantallaError

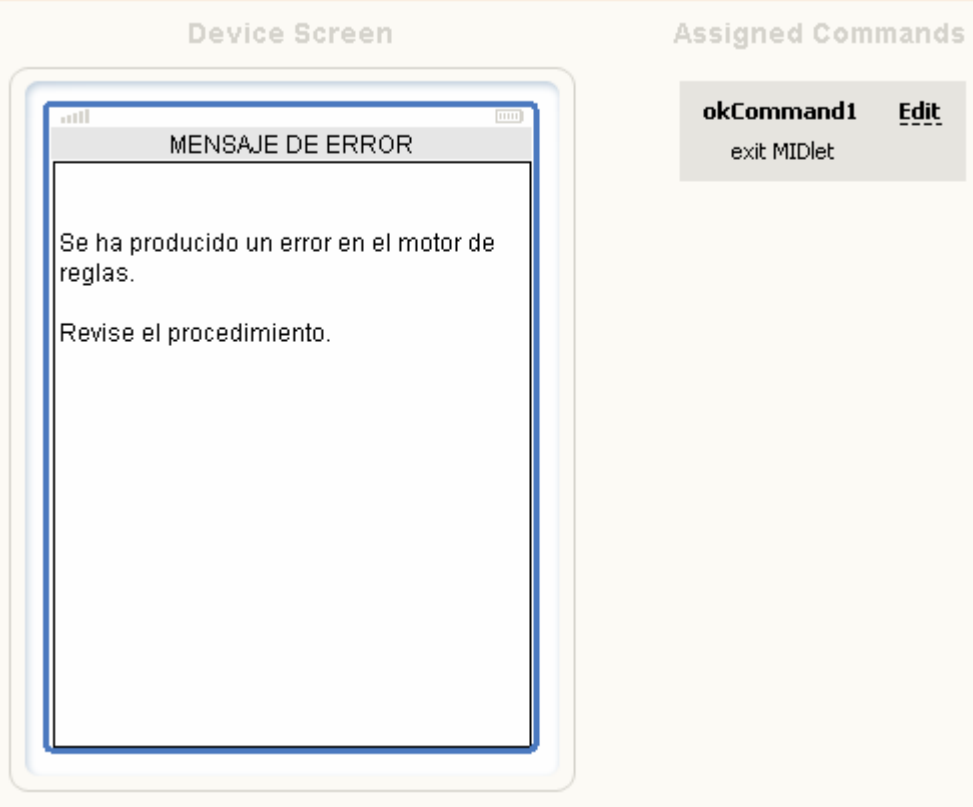

Figura 130. Diseño de GUI de PantallaError

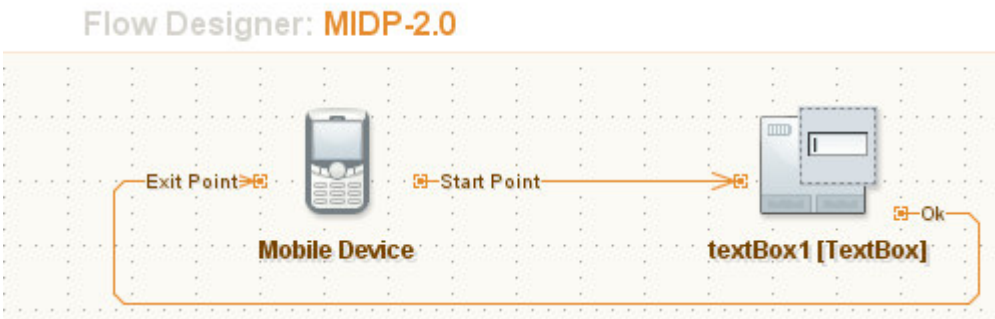

Figura 131. Diseño de flujo de PantallaError

Tabla 35. Código fuente del MIDlet de PantallaError

```
/*
 * PantallaError.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
```

```
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaError extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaError */
    public PantallaError() {
        initialize();
    }

    private TextBox textBox1;
    private Command okCommand1;

    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == textBox1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
    }

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_textBox1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for textBox1 component and should be called
instead of accessing textBox1 field directly.
     * @return Instance for textBox1 component
     */
    public TextBox get_textBox1() {
        if (textBox1 == null) {
            // Insert pre-init code here
            textBox1 = new TextBox("MENSAJE DE ERROR", "\n\nSe ha producido un error
en el motor de reglas.\n\nRevise el procedimiento.", 120, TextField.ANY);
            textBox1.addCommand(get_okCommand1());
            textBox1.setCommandListener(this);
            // Insert post-init code here
        }
        return textBox1;
    }
```

```
    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.4. PantallaMenuUsuario

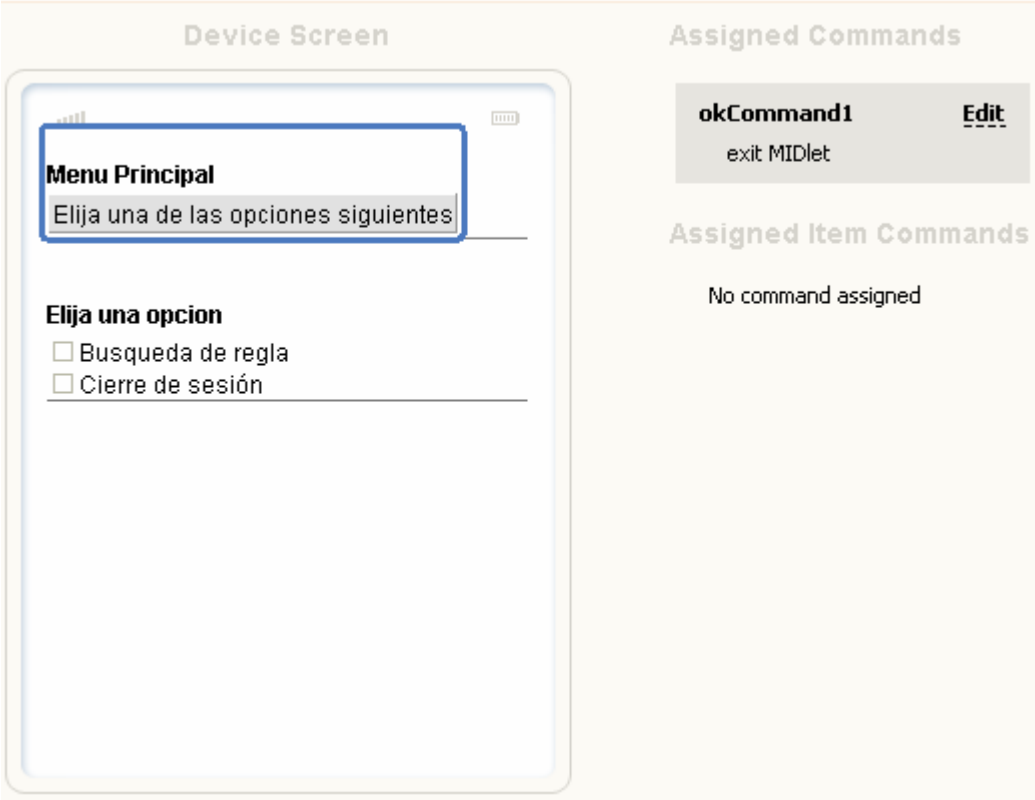

Figura 132. Diseño de GUI de PantallaMenuUsuario

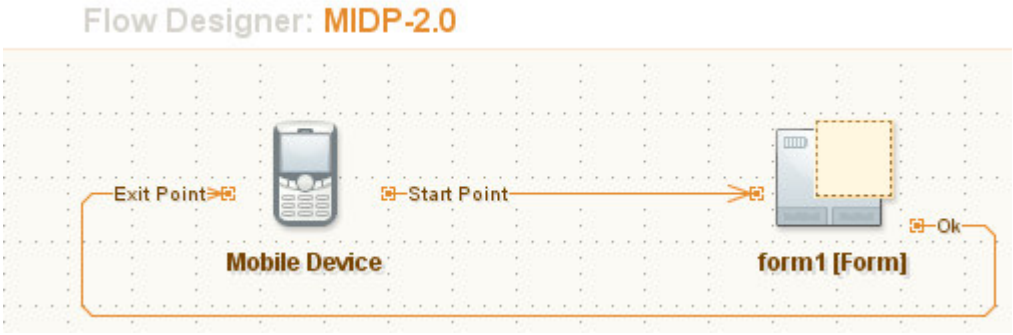

Figura 133. Diseño de flujo de PantallaMenuUsuario

Tabla 36. Código fuente del MIDlet de PantallaMenuUsuario

```
/*
 * PantallaMenuUsuario.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaMenuUsuario extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaMenuUsuario */
    public PantallaMenuUsuario() {
        initialize();
    }

    private Form form1;
    private Command screenCommand1;
    private ChoiceGroup choiceGroup1;
    private Spacer spacer1;
    private StringItem stringItem1;
    private Command okCommand1;
    private Command okCommand2;
    private Command okCommand3;
    private Command okCommand4;
    private Spacer spacer7;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
```

```
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }


    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_choiceGroup1(),
                get_spacer7()
            });
            form1.addCommand(get_okCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }

    /** This method returns instance for choiceGroup1 component and should be called
instead of accessing choiceGroup1 field directly.
     * @return Instance for choiceGroup1 component
     */
    public ChoiceGroup get_choiceGroup1() {
        if (choiceGroup1 == null) {
            // Insert pre-init code here
            choiceGroup1 = new ChoiceGroup("  \n  \nElija una opcion",
Choice.MULTIPLE, new String[] {
```

```
                "Busqueda de regla",
                "Cierre de sesi\u00F3n"
            }, new Image[] {
                null,
                null
            });
            choiceGroup1.setSelectedFlags(new boolean[] {
                false,
                false
            });
            // Insert post-init code here
        }
        return choiceGroup1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            spacer1.setLayout(Item.LAYOUT_DEFAULT | Item.LAYOUT_NEWLINE_BEFORE |
Item.LAYOUT_NEWLINE_AFTER);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("    \nMenu Principal", "Elija una de las
opciones siguientes", javax.microedition.lcdui.Item.BUTTON);
            stringItem1.setLayout(Item.LAYOUT_CENTER | Item.LAYOUT_VCENTER |
Item.LAYOUT_NEWLINE_BEFORE | Item.LAYOUT_NEWLINE_AFTER | Item.LAYOUT_SHRINK |
Item.LAYOUT_VSHRINK | Item.LAYOUT_EXPAND | Item.LAYOUT_VEXPAND);
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for okCommand2 component and should be called
instead of accessing okCommand2 field directly.
     * @return Instance for okCommand2 component
     */
    public Command get_okCommand2() {
        if (okCommand2 == null) {
            // Insert pre-init code here
            okCommand2 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
```

```
        return okCommand2;
    }


    /** This method returns instance for okCommand3 component and should be called
instead of accessing okCommand3 field directly.
     * @return Instance for okCommand3 component
     */
    public Command get_okCommand3() {
        if (okCommand3 == null) {
            // Insert pre-init code here
            okCommand3 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand3;
    }



    /** This method returns instance for okCommand4 component and should be called
instead of accessing okCommand4 field directly.
     * @return Instance for okCommand4 component
     */
    public Command get_okCommand4() {
        if (okCommand4 == null) {
            // Insert pre-init code here
            okCommand4 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand4;
    }

    /** This method returns instance for spacer7 component and should be called
instead of accessing spacer7 field directly.
     * @return Instance for spacer7 component
     */
    public Spacer get_spacer7() {
        if (spacer7 == null) {
            // Insert pre-init code here
            spacer7 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer7;

    }


    public void startApp() {
    }


    public void pauseApp() {
    }


    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.5. PantallaMenuDiseñador

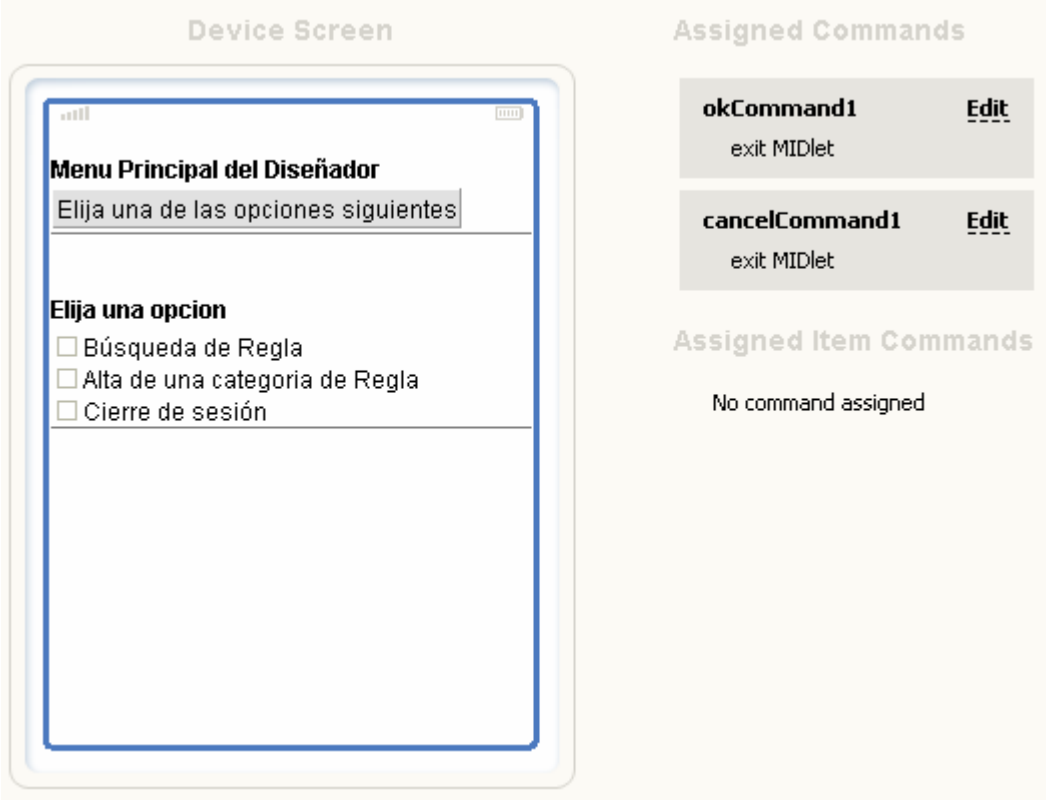

Figura 134. Diseño de GUI de PantallaMenuDiseñador

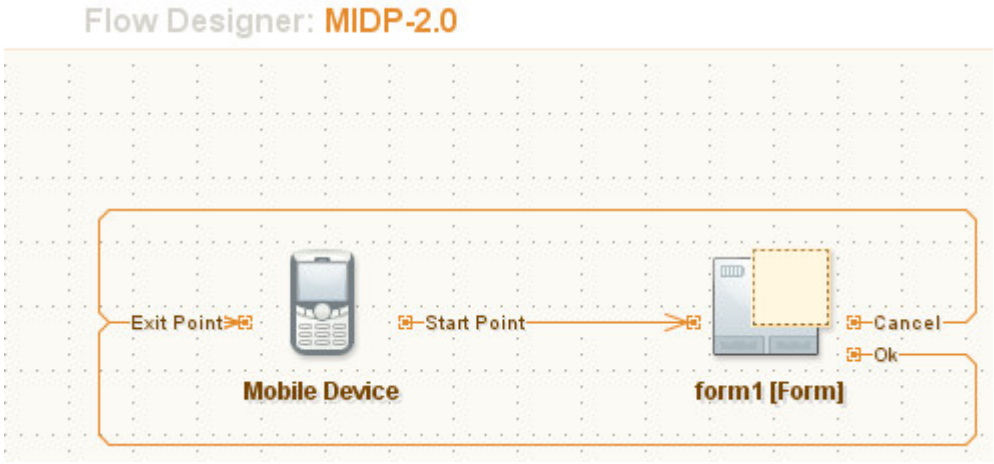

Figura 135. Diseño de flujo de PantallaMenuDiseñador

Tabla 37. Código fuente del MIDlet de PantallaMenuDiseñador

```
/*
 * PantallaMenuDiseñador.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaMenuDisenador extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaMenuDiseñador */
    public PantallaMenuDisenador() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private ChoiceGroup choiceGroup1;
    private Spacer spacer1;
    private Spacer spacer2;
    private Command okCommand1;
    private Command cancelCommand1;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }
```

```
    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_choiceGroup1(),
                get_spacer2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("   \nMenu Principal del
Dise\u00F1ador\n\n", "Elija una de las opciones siguientes\n",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for choiceGroup1 component and should be called
instead of accessing choiceGroup1 field directly.
     * @return Instance for choiceGroup1 component
     */
    public ChoiceGroup get_choiceGroup1() {
        if (choiceGroup1 == null) {
            // Insert pre-init code here
            choiceGroup1 = new ChoiceGroup("   \n  \nElija una opcion",
Choice.MULTIPLE, new String[] {
                "B\u00FAsqueda de Regla",
                "Alta de una categoria de Regla",
                "Cierre de sesi\u00F3n"
            }, new Image[] {
                null,
                null,
                null
            });
            choiceGroup1.setSelectedFlags(new boolean[] {
                false,
                false,
                false
            });
            // Insert post-init code here
        }
```

```
        return choiceGroup1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.6. PantallaMenuAdministrador

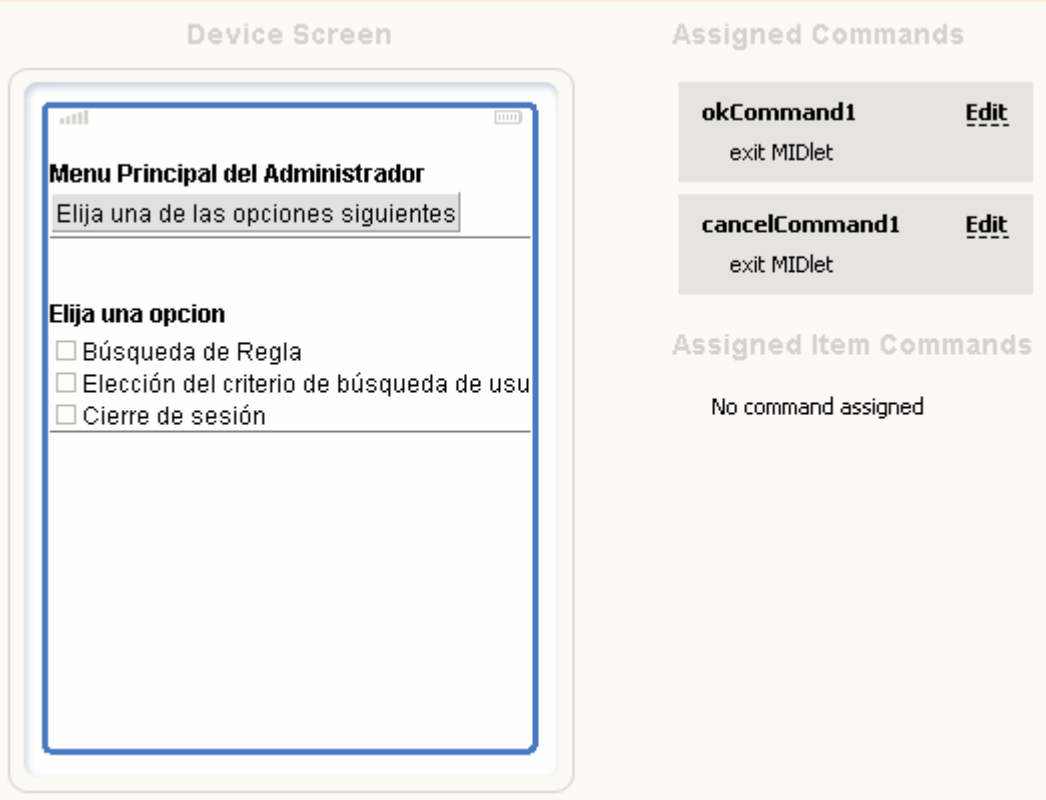

Figura 136. Diseño del GUI de PantallaMenuAdministrador

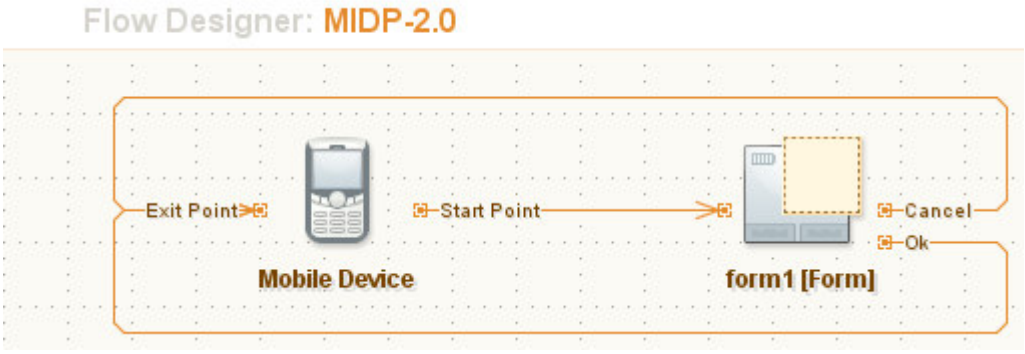

Figura 137. Diseño de flujo de PantallaMenuAdministrador

Tabla 38. Código fuente del MIDlet de PantallaMenuAdministrador

```
/*
 * PantallaMenuAdministrador.java
 *
 */
```

```
package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaMenuAdministrador extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaMenuAdministrador */
    public PantallaMenuAdministrador () {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private ChoiceGroup choiceGroup1;
    private Spacer spacer1;
    private Spacer spacer2;
    private Command okCommand1;
    private Command cancelCommand1;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }
```

```
    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_choiceGroup1(),
                get_spacer2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("   \nMenu Principal del Administrador\n\n",
"Elija una de las opciones siguientes\n", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for choiceGroup1 component and should be called
instead of accessing choiceGroup1 field directly.
     * @return Instance for choiceGroup1 component
     */
    public ChoiceGroup get_choiceGroup1() {
        if (choiceGroup1 == null) {
            // Insert pre-init code here
            choiceGroup1 = new ChoiceGroup("   \n  \nElija una opcion",
Choice.MULTIPLE, new String[] {
                "B\u00FAsqueda de Regla",
                "Elecci\u00F3n del criterio de b\u00FAsqueda de usuario",
                "Cierre de sesi\u00F3n"
            }, new Image[] {
                null,
                null,
                null
            });
            choiceGroup1.setSelectedFlags(new boolean[] {
                false,
                false,
                false
            });
            // Insert post-init code here
        }
        return choiceGroup1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
```

```
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }


    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }


    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }


    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    public void startApp() {
    }


    public void pauseApp() {
    }


    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.7. PantallaSelecciónAntecedentes

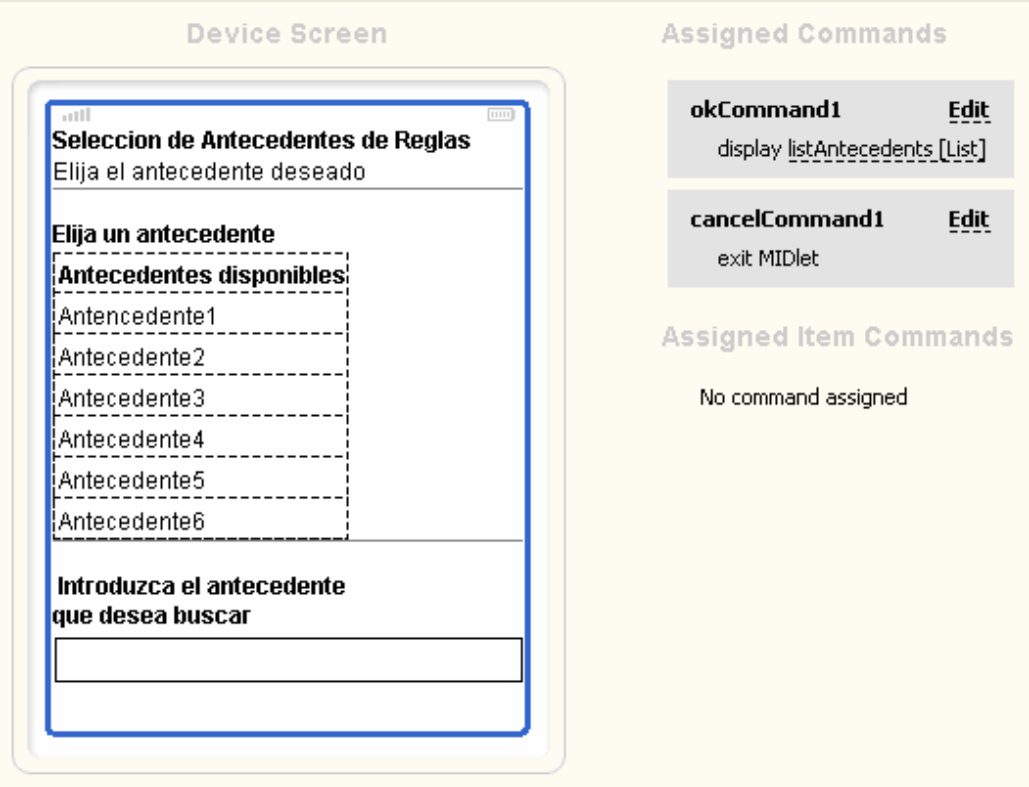

Figura 138. Diseño de GUI de PantallaSeleccionAntecedentes

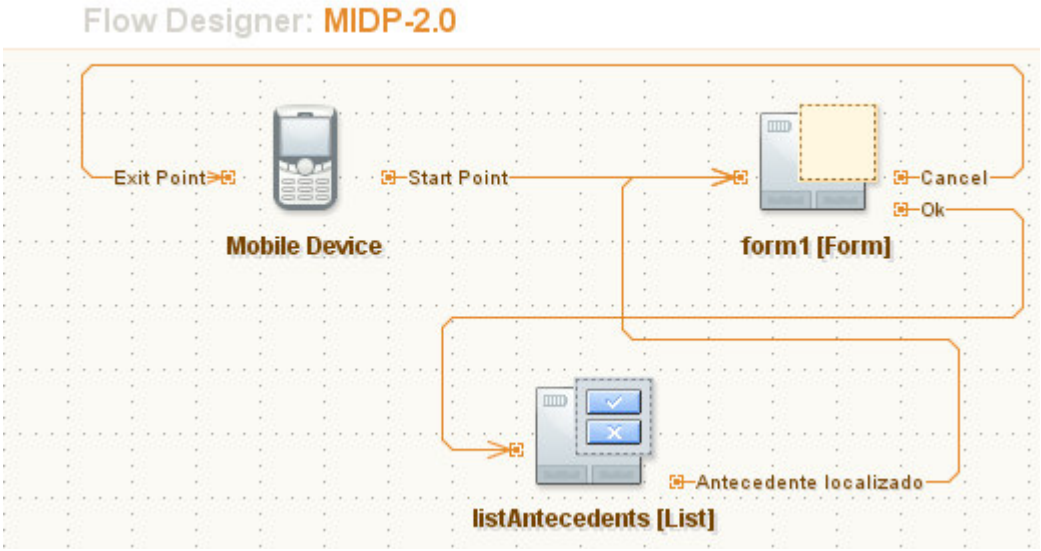

Figura 139. Diseño de Flujo de PantallaSeleccionAntecedentes

Tabla 39. Código fuente del MIDlet de PantallaSeleccionAntecedentes

```
/*
 * PantallaSeleccionAntecedentes.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaSeleccionAntecedentes extends MIDlet implements CommandListener
{

    /** Creates a new instance of PantallaSeleccionAntecedentes */
    public PantallaSeleccionAntecedentes() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Spacer spacer2;
    private Command okCommand1;
    private Command cancelCommand1;
    private TextField textField1;
    private List listAntecedents;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listAntecedents());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == listAntecedents) {
            if (command == listAntecedents.SELECT_COMMAND) {
                switch (get_listAntecedents().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
                        getDisplay().setCurrent(get_form1());
                        // Insert post-action code here
                        break;
                }
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
```

```
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Seleccion de Antecedentes de Reglas",
"Elija el antecedente deseado\n");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
```

```
    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"  \nElija un antecedente\n", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Antencedente1",
                },
                new String[] {
                    "Antecedente2",
                },
                new String[] {
                    "Antecedente3",
                },
                new String[] {
                    "Antecedente4",
                },
                new String[] {
                    "Antecedente5",
                },
                new String[] {
                    "Antecedente6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Antecedentes disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
```

```
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("    \n Introduzca el antecedente \nque desea
buscar", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for listAntecedents component and should be
called instead of accessing listAntecedents field directly.
     * @return Instance for listAntecedents component
     */
    public List get_listAntecedents() {
        if (listAntecedents == null) {
            // Insert pre-init code here
            listAntecedents = new List(null, Choice.IMPLICIT, new String[]
{"Antecedente localizado"}, new Image[] {null});
            listAntecedents.setCommandListener(this);
            listAntecedents.setSelectedFlags(new boolean[] {false});
            // Insert post-init code here
        }
        return listAntecedents;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.8. PantallaSeleccionAtributos

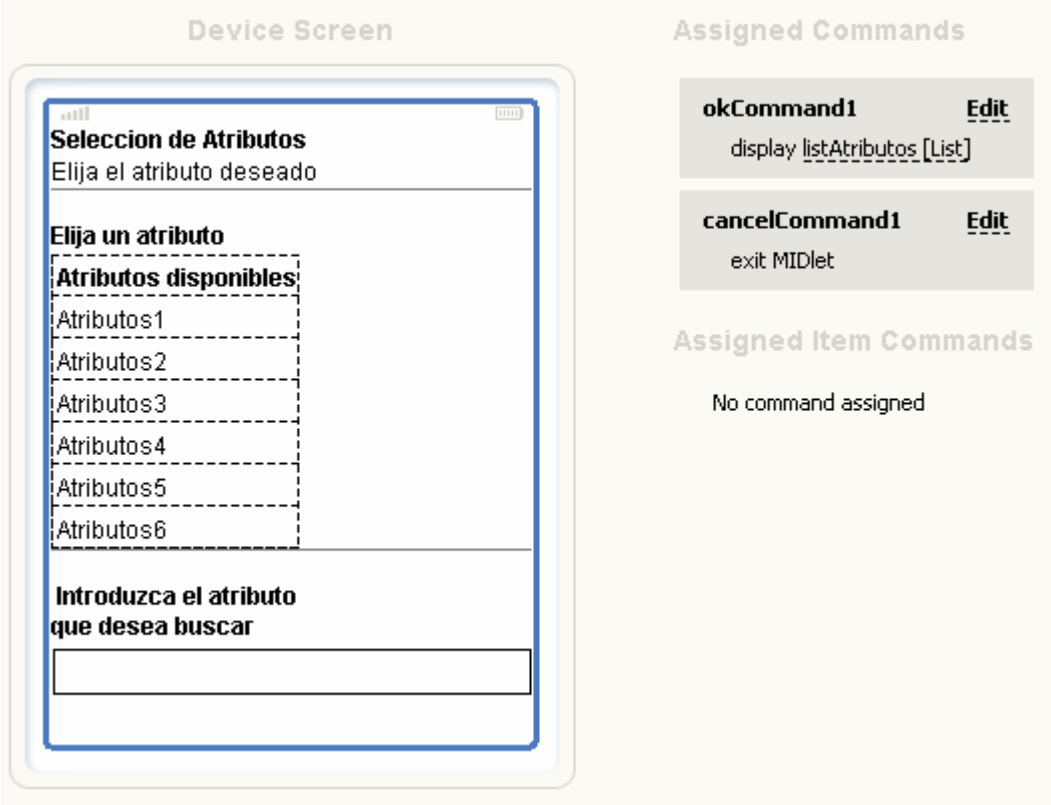

Figura 140. Diseño de GUI de PantallaSeleccionAtributos

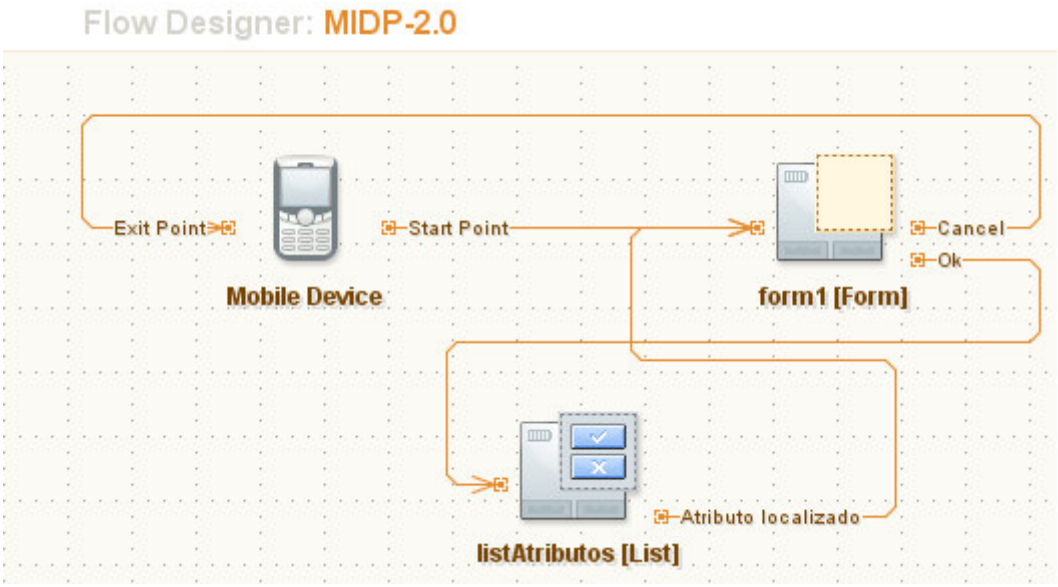

Figura 141. Diseño de flujo de PantallaSeleccionAtributos

Tabla 40. Código fuente del MIDlet de PantallaSeleccionAtributos

```
/*
 * PantallaSeleccionAtributos.java
 *
 * Created on 7 de mayo de 2006, 21:26
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaSeleccionAtributos extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaSeleccionAtributos */
    public PantallaSeleccionAtributos() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Spacer spacer2;
    private Command okCommand1;
    private Command cancelCommand1;
    private TextField textField1;
    private List listAtributos;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listAtributos());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == listAtributos) {
            if (command == listAtributos.SELECT_COMMAND) {
                switch (get_listAtributos().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
                        getDisplay().setCurrent(get_form1());
                        // Insert post-action code here
                        break;
                }
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
```

```
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Seleccion de Atributos", "Elija el atributo
deseado\n");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
```

```
    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"  \nElija un atributo\n", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Atributos1",
                },
                new String[] {
                    "Atributos2",
                },
                new String[] {
                    "Atributos3",
                },
                new String[] {
                    "Atributos4",
                },
                new String[] {
                    "Atributos5",
                },
                new String[] {
                    "Atributos6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Atributos disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
```

```
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("    \n Introduzca el atributo \nque desea
buscar", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for listAtributos component and should be called
instead of accessing listAtributos field directly.
     * @return Instance for listAtributos component
     */
    public List get_listAtributos() {
        if (listAtributos == null) {
            // Insert pre-init code here
            listAtributos = new List(null, Choice.IMPLICIT, new String[] {"Atributo
localizado"}, new Image[] {null});
            listAtributos.setCommandListener(this);
            listAtributos.setSelectedFlags(new boolean[] {false});
            // Insert post-init code here
        }
        return listAtributos;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.9. PantallaCriteriosBusquedaRegla

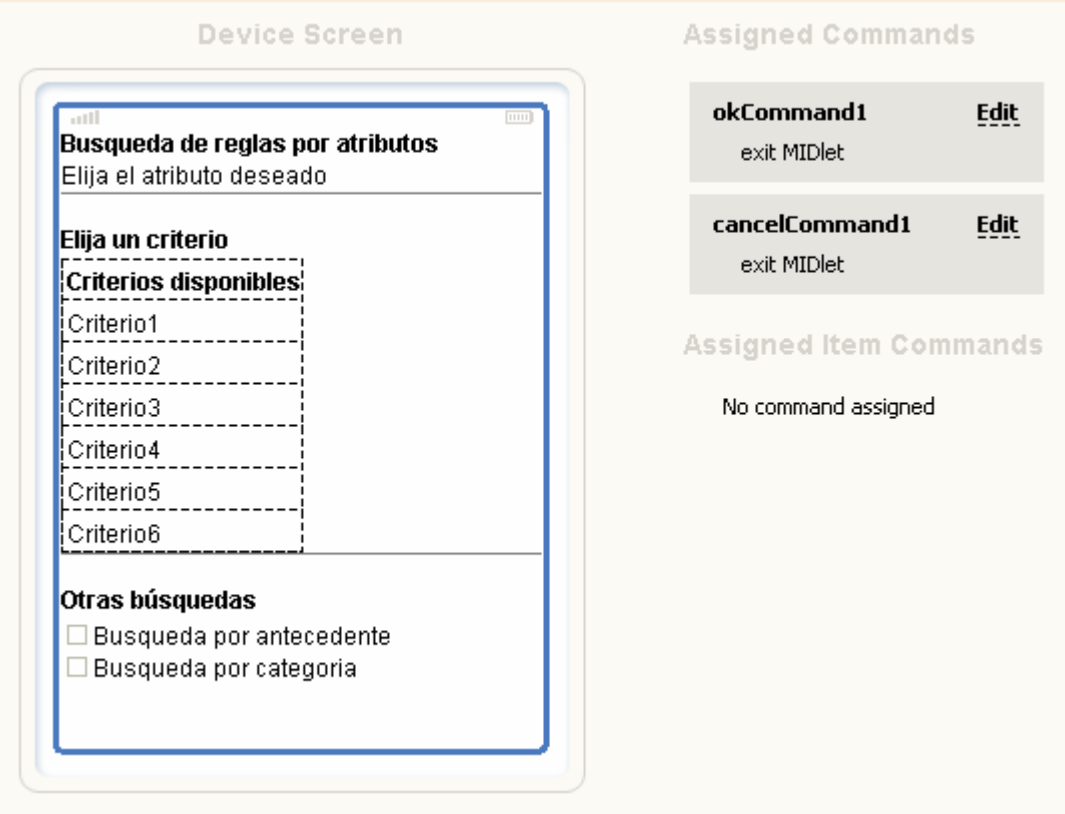

Figura 142. Diseño de GUI de PantallaCriteriosBusquedaRegla

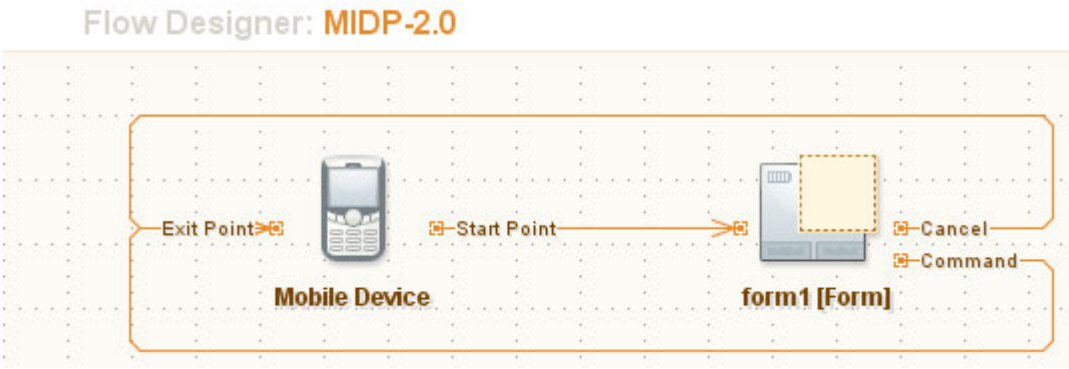

Figura 143. Diseño de flujo de PantallaCriteriosBusquedaRegla

Tabla 41. Código fuente del MIDlet de PantallaCriteriosBusquedaRegla

```
/*
 * PantallaCriteriosBusquedaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaCriteriosBusquedaRegla extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaCriteriosBusquedaRegla */
    public PantallaCriteriosBusquedaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Spacer spacer1;
    private ChoiceGroup choiceGroup1;
    private Spacer spacer2;
    private Command okCommand1;
    private Command cancelCommand1;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }
```

```
    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer2(),
                get_tableItem1(),
                get_spacer1(),
                get_choiceGroup1()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Busqueda de reglas por atributos", "Elija el
atributo deseado");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(), "
\nElija un criterio", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Criterio1",
```

```
                },
                new String[] {
                    "Criterio2",
                },
                new String[] {
                    "Criterio3",
                },
                new String[] {
                    "Criterio4",
                },
                new String[] {
                    "Criterio5",
                },
                new String[] {
                    "Criterio6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Criterios disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for spacer1 component and should be called instead
of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }


    /** This method returns instance for choiceGroup1 component and should be called
instead of accessing choiceGroup1 field directly.
     * @return Instance for choiceGroup1 component
     */
    public ChoiceGroup get_choiceGroup1() {
        if (choiceGroup1 == null) {
            // Insert pre-init code here
            choiceGroup1 = new ChoiceGroup("   \nOtras b\u00FAsquedas",
Choice.MULTIPLE, new String[] {
                "Busqueda por antecedente",
                "Busqueda por categoria"
            }, new Image[] {
                null,
                null
            });
            choiceGroup1.setSelectedFlags(new boolean[] {
                false,
                false
            });
            // Insert post-init code here
        }
        return choiceGroup1;
    }


    /** This method returns instance for spacer2 component and should be called instead
of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
```

```
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }



    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }

        return okCommand1;
    }


    /** This method returns instance for cancelCommand1 component and should be called
instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.10. PantallaAltaRegla

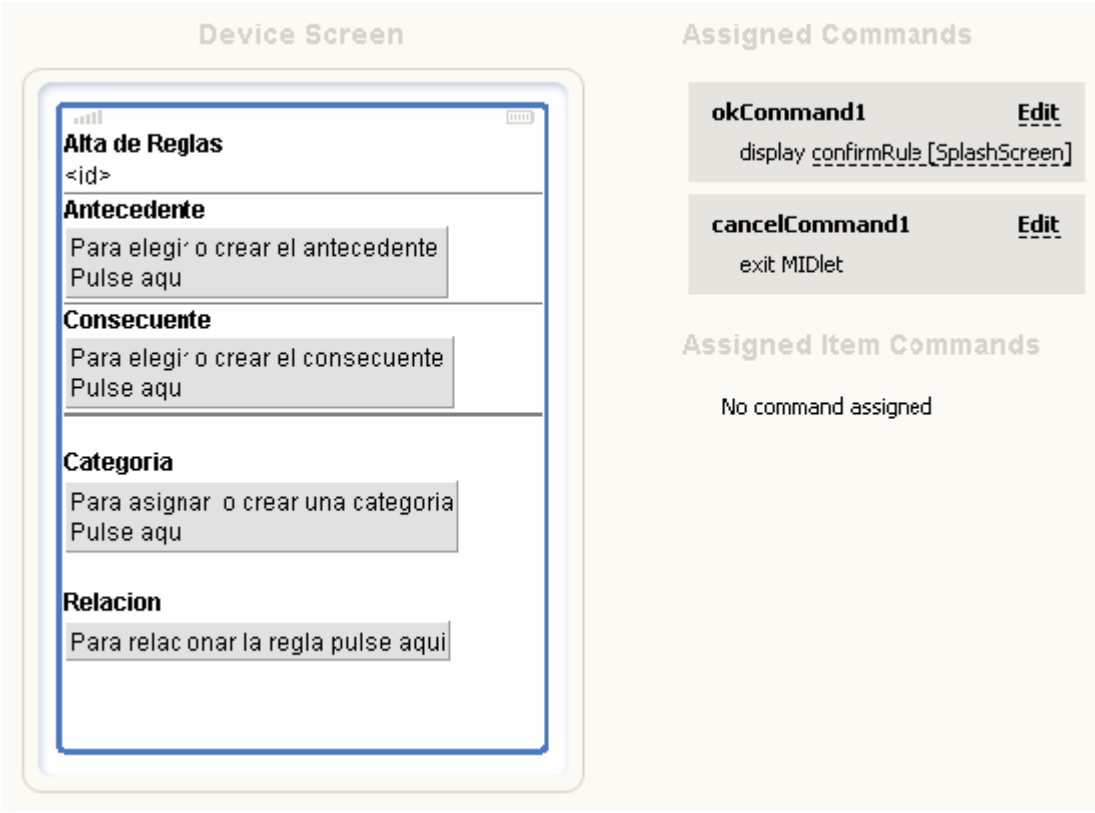

Figura 144. Diseño de GUI de PantallaAltaRegla

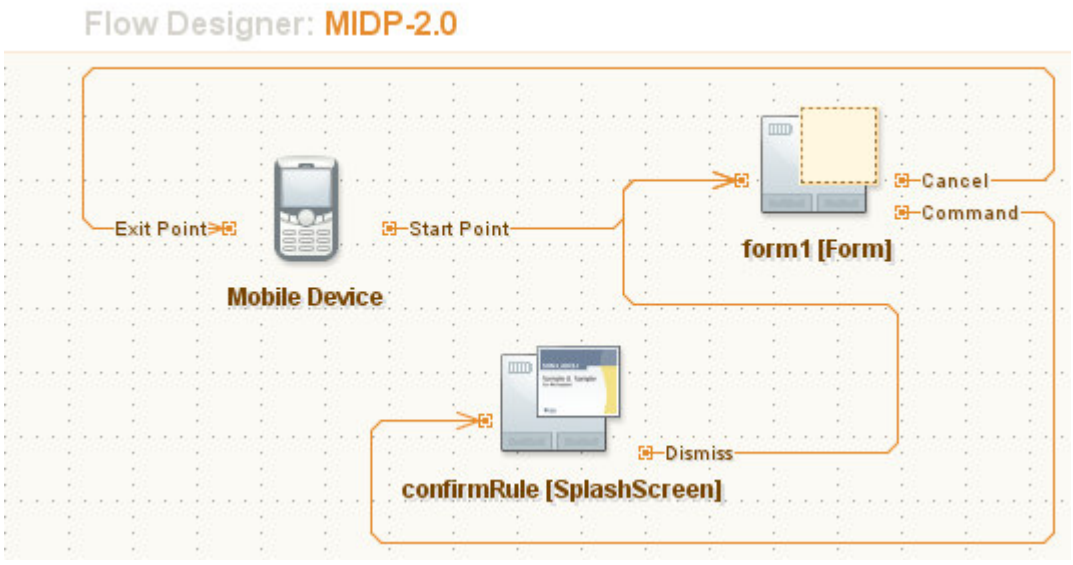

Figura 145. Diseño de flujo de PantallaAltaRegla

Tabla 42. Código fuente del MIDlet de PantallaAltaRegla

```
/*
 * PantallaAltaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaAltaRegla extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaSeleccionCategorias */
    public PantallaAltaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private Spacer spacer4;
    private StringItem stringItem3;
    private StringItem stringItem4;
    private StringItem stringItem5;
    private StringItem stringItem6;
    private org.netbeans.microedition.lcdui.SplashScreen confirmRule;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmRule());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
```

```
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_stringItem3(),
                get_spacer2(),
                get_stringItem4(),
                get_spacer3(),
                get_spacer4(),
                get_stringItem5(),
                get_stringItem6()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Alta de Reglas", "<id>");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
```

```
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Usuario1",
                },
                new String[] {
                    "Usuario2",
                },
                new String[] {
                    "Usuario3",
                },
                new String[] {
                    "Usuario4",
                },
                new String[] {
                    "Usuario5",
                },
                new String[] {
                    "Usuario6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Usuarios creados",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
```

```
    }


    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }
    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer4;
    }

    /** This method returns instance for stringItem3 component and should be called
instead of accessing stringItem3 field directly.
     * @return Instance for stringItem3 component
     */
    public StringItem get_stringItem3() {
        if (stringItem3 == null) {
            // Insert pre-init code here
            stringItem3 = new StringItem("Antecedente", "Para elegir o crear el
antecedente \nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem3;
    }

    /** This method returns instance for stringItem4 component and should be called
instead of accessing stringItem4 field directly.
     * @return Instance for stringItem4 component
     */
    public StringItem get_stringItem4() {
        if (stringItem4 == null) {
            // Insert pre-init code here
            stringItem4 = new StringItem("Consecuente", "Para elegir o crear el
consecuente \nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem4;
    }

    /** This method returns instance for stringItem5 component and should be called
```

```
instead of accessing stringItem5 field directly.
     * @return Instance for stringItem5 component
     */
    public StringItem get_stringItem5() {
        if (stringItem5 == null) {
            // Insert pre-init code here
            stringItem5 = new StringItem("\nCategoria", "Para asignar  o crear una
categoria\nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem5;
    }

    /** This method returns instance for stringItem6 component and should be called
instead of accessing stringItem6 field directly.
     * @return Instance for stringItem6 component
     */
    public StringItem get_stringItem6() {
        if (stringItem6 == null) {
            // Insert pre-init code here
            stringItem6 = new StringItem("\nRelacion", "Para relacionar la regla
pulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem6;
    }

    /** This method returns instance for confirmRule component and should be called
instead of accessing confirmRule field directly.
     * @return Instance for confirmRule component
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_confirmRule() {
        if (confirmRule == null) {
            // Insert pre-init code here
            confirmRule = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            confirmRule.setNextDisplayable(get_form1());
            // Insert post-init code here
        }
        return confirmRule;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.11. PantallaModificarRegla

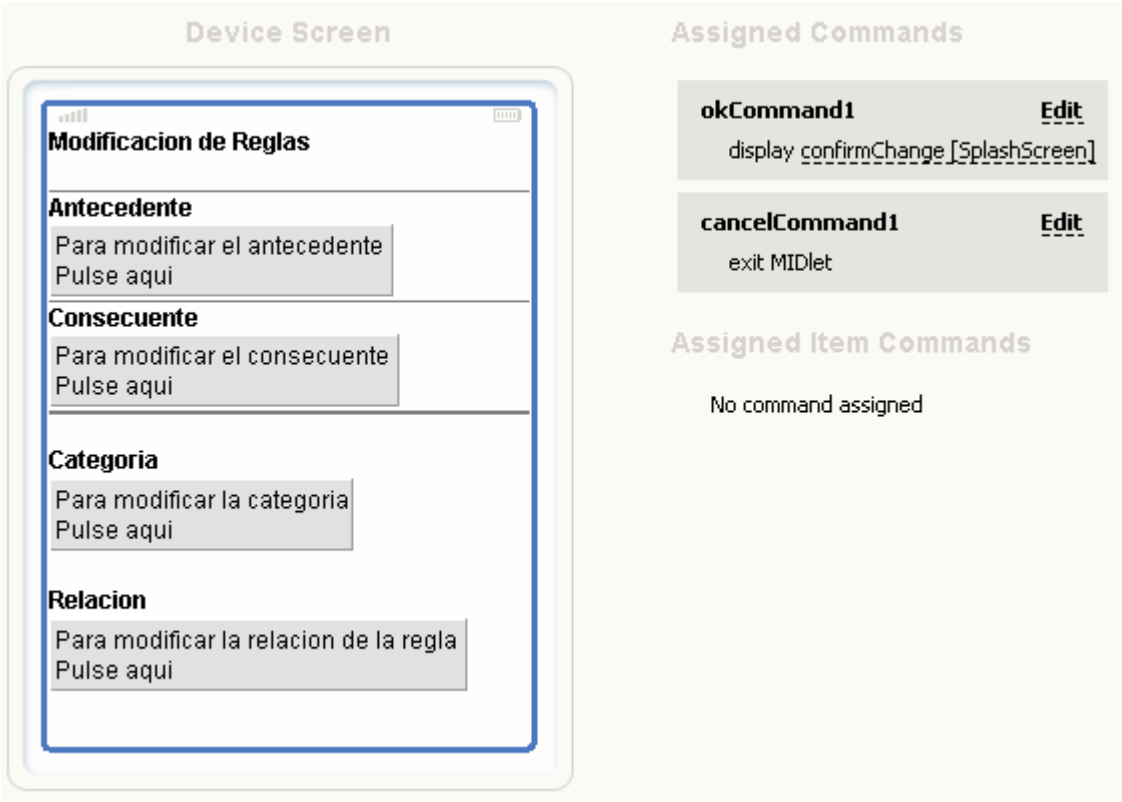

Figura 146. Diseño de GUI de PantallaModificarRegla

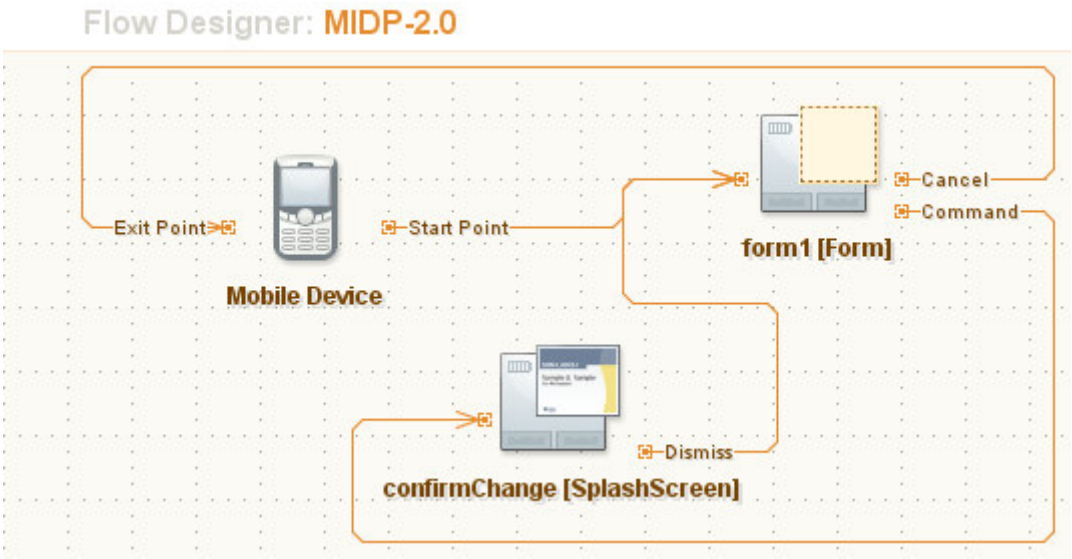

Figura 147. Diseño de flujo de PantallaModificarRegla

Tabla 43. Código fuente del MIDlet de PantallaModificarRegla

```
/*
 * PantallaModificarRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaModificarRegla extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaModificarRegla */
    public PantallaModificarRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private Spacer spacer4;
    private StringItem stringItem3;
    private StringItem stringItem4;
    private StringItem stringItem5;
    private StringItem stringItem6;
    private org.netbeans.microedition.lcdui.SplashScreen confirmChange;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmChange());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
```

```
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_stringItem3(),
                get_spacer2(),
                get_stringItem4(),
                get_spacer3(),
                get_spacer4(),
                get_stringItem5(),
                get_stringItem6()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Modificacion de Reglas", "");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
    /** This method returns instance for simpleTableModel1 component and should be
```

```
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Regla1",
                },
                new String[] {
                    "Regla2",
                },
                new String[] {
                    "Regla3",
                },
                new String[] {
                    "Regla4",
                },
                new String[] {
                    "Regla5",
                },
                new String[] {
                    "Regla6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Usuarios creados",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
```

```
            // Insert post-init code here
        }
        return spacer2;
    }


    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }
    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer4;
    }

    /** This method returns instance for stringItem3 component and should be called
instead of accessing stringItem3 field directly.
     * @return Instance for stringItem3 component
     */
    public StringItem get_stringItem3() {
        if (stringItem3 == null) {
            // Insert pre-init code here
            stringItem3 = new StringItem("Antecedente", "Para modificar el
antecedente \nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem3;
    }

    /** This method returns instance for stringItem4 component and should be called
instead of accessing stringItem4 field directly.
     * @return Instance for stringItem4 component
     */
    public StringItem get_stringItem4() {
        if (stringItem4 == null) {
            // Insert pre-init code here
            stringItem4 = new StringItem("Consecuente", "Para modificar el
consecuente \nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem4;
```

```
    }


    /** This method returns instance for stringItem5 component and should be called
instead of accessing stringItem5 field directly.
     * @return Instance for stringItem5 component
     */
    public StringItem get_stringItem5() {
        if (stringItem5 == null) {
            // Insert pre-init code here
            stringItem5 = new StringItem("  \nCategoria", "Para modificar la
categoria\nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem5;
    }


    /** This method returns instance for stringItem6 component and should be called
instead of accessing stringItem6 field directly.
     * @return Instance for stringItem6 component
     */
    public StringItem get_stringItem6() {
        if (stringItem6 == null) {
            // Insert pre-init code here
            stringItem6 = new StringItem("  \nRelacion", "Para modificar la relacion
de la regla \nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem6;
    }


    /** This method returns instance for confirmChange component and should be called
instead of accessing confirmChange field directly.
     * @return Instance for confirmChange component
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_confirmChange() {
        if (confirmChange == null) {
            // Insert pre-init code here
            confirmChange = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            confirmChange.setNextDisplayable(get_form1());
            // Insert post-init code here
        }
        return confirmChange;
    }


    public void startApp() {
    }


    public void pauseApp() {
    }


    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.12. PantallaBajaRegla

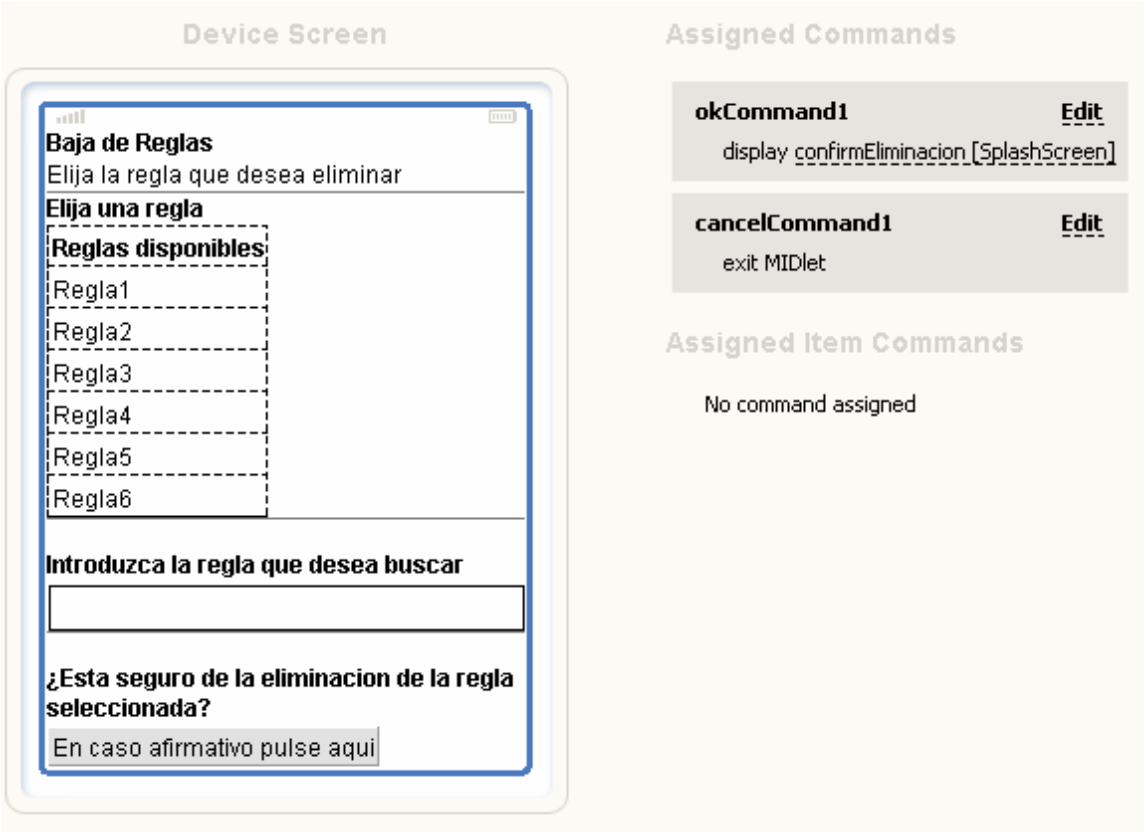

Figura 148. Diseño de GUI de PantallaBajaRegla

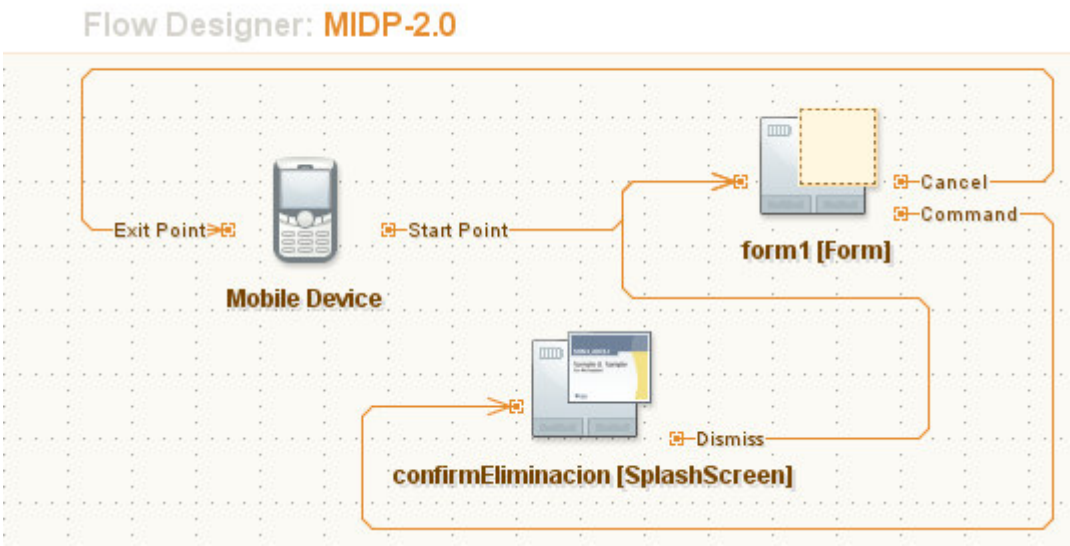

Figura 149. Diseño de flujo de PantallaBajaRegla

Tabla 44. Código fuente del MIDlet de PantallaBajaRegla

```
/*
 * PantallaBajaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaBajaRegla extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaBajaRegla */
    public PantallaBajaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private TextField textField1;
    private StringItem stringItem2;
    private Spacer spacer3;
    private Command screenCommand1;
    private org.netbeans.microedition.lcdui.SplashScreen confirmEliminacion;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmEliminacion());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}


    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }
```

```
    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }


    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }


    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1(),
                get_spacer3(),
                get_stringItem2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Baja de Reglas", "Elija la regla que desea
eliminar");
            // Insert post-init code here
        }
        return stringItem1;
    }


    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
```

```
    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"Elija una regla", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }


    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Regla1",
                },
                new String[] {
                    "Regla2",
                },
                new String[] {
                    "Regla3",
                },
                new String[] {
                    "Regla4",
                },
                new String[] {
                    "Regla5",
                },
                new String[] {
                    "Regla6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Reglas disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }


    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }


    /** This method returns instance for cancelCommand1 component and should be
```

```
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }



    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField(" \nIntroduzca la regla que desea buscar",
null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }



    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("   \n\u00BFEsta seguro de la eliminacion de
la regla seleccionada?", "En caso afirmativo pulse aqui",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }

        return stringItem2;
    }



    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
```

```
        }
        return spacer3;
    }



    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }



    /** This method returns instance for confirmEliminacion component and should be
called instead of accessing confirmEliminacion field directly.
     * @return Instance for confirmEliminacion component
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_confirmEliminacion() {
        if (confirmEliminacion == null) {
            // Insert pre-init code here
            confirmEliminacion = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            confirmEliminacion.setNextDisplayable(get_form1());
            // Insert post-init code here
        }
        return confirmEliminacion;
    }



    public void startApp() {
    }



    public void pauseApp() {
    }



    public void destroyApp(boolean unconditional) {
    }


}
```

5.4.13. PantallaRelacionarRegla

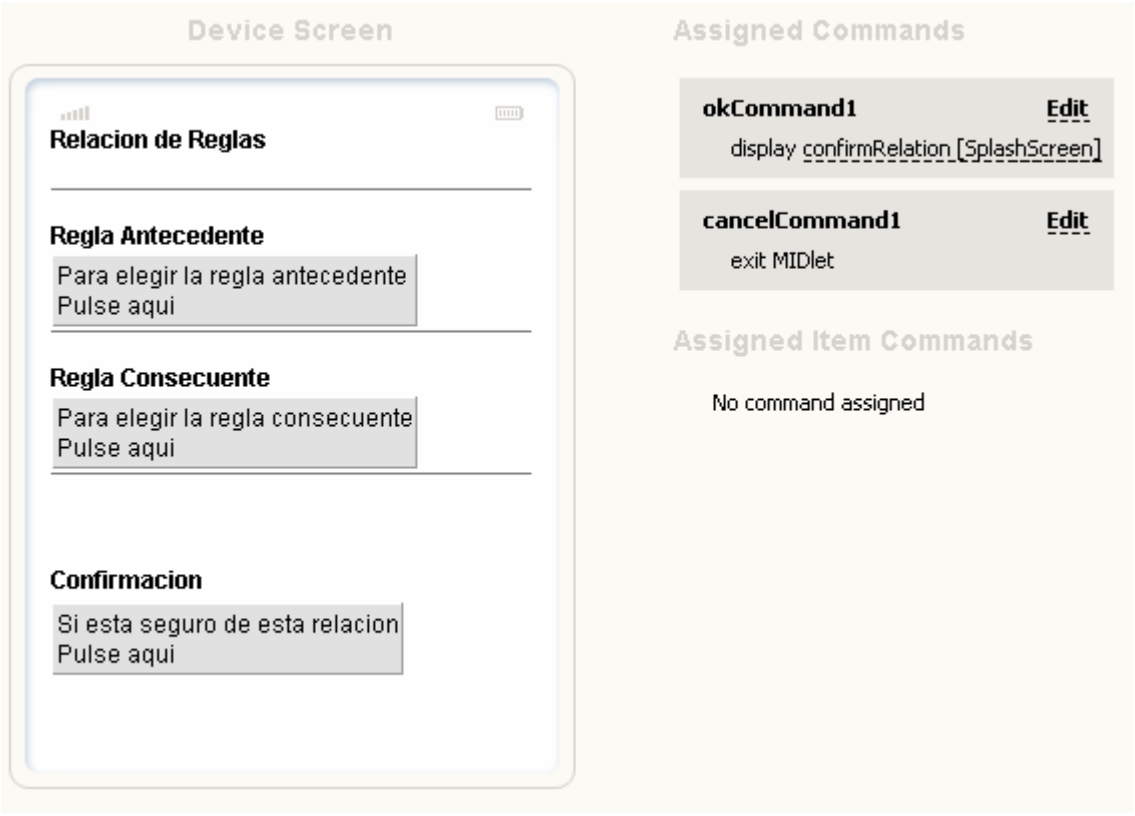

Figura 150. Diseño de GUI de PantallaRelacionarRegla

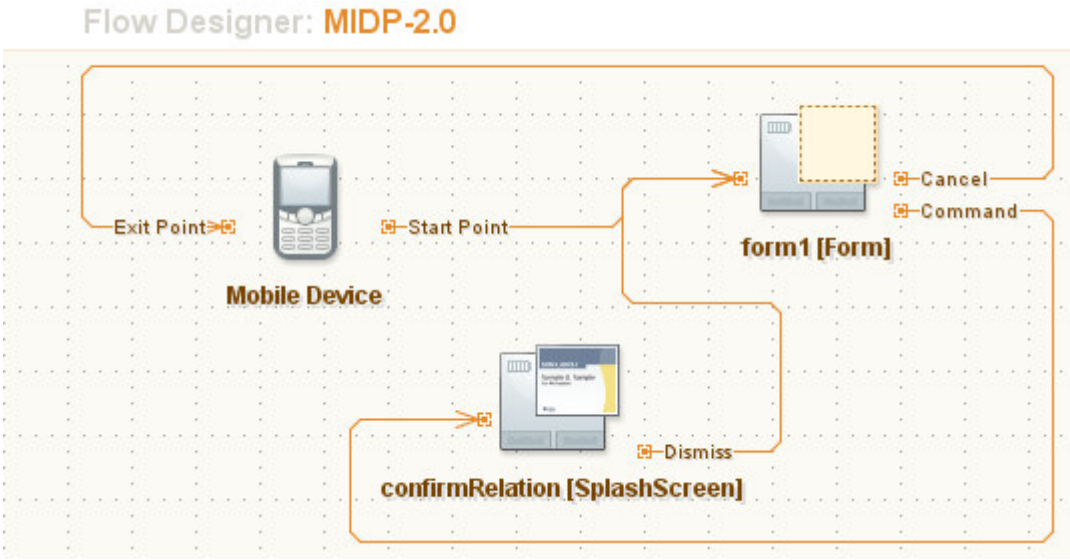

Figura 151. Diseño de flujo de PantallaRelacionarRegla

Tabla 45. Código fuente del MIDlet de PantallaRelacionarRegla

```
/*
 * PantallaRelacionarRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaRelacionarRegla extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaRelacionarRegla */
    public PantallaRelacionarRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private StringItem stringItem3;
    private StringItem stringItem4;
    private StringItem stringItem6;
    private org.netbeans.microedition.lcdui.SplashScreen confirmRelation;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmRelation());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
```

```
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_stringItem3(),
                get_spacer2(),
                get_stringItem4(),
                get_spacer3(),
                get_stringItem6()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Relacion de Reglas", "");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
```

```
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Usuario1",
                },
                new String[] {
                    "Usuario2",
                },
                new String[] {
                    "Usuario3",
                },
                new String[] {
                    "Usuario4",
                },
                new String[] {
                    "Usuario5",
                },
                new String[] {
                    "Usuario6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Usuarios creados",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }


    /** This method returns instance for spacer3 component and should be called
```

```
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }

    /** This method returns instance for stringItem3 component and should be called
instead of accessing stringItem3 field directly.
     * @return Instance for stringItem3 component
     */
    public StringItem get_stringItem3() {
        if (stringItem3 == null) {
            // Insert pre-init code here
            stringItem3 = new StringItem("  \nRegla Antecedente", "Para elegir la
regla antecedente \nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem3;
    }

    /** This method returns instance for stringItem4 component and should be called
instead of accessing stringItem4 field directly.
     * @return Instance for stringItem4 component
     */
    public StringItem get_stringItem4() {
        if (stringItem4 == null) {
            // Insert pre-init code here
            stringItem4 = new StringItem("  \nRegla Consecuente", "Para elegir la
regla consecuente\nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem4;
    }

    /** This method returns instance for stringItem6 component and should be called
instead of accessing stringItem6 field directly.
     * @return Instance for stringItem6 component
     */
    public StringItem get_stringItem6() {
        if (stringItem6 == null) {
            // Insert pre-init code here
            stringItem6 = new StringItem("\n  \n\nConfirmacion", "Si esta seguro de
esta relacion\nPulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem6;
    }

    /** This method returns instance for confirmRelation component and should be
called instead of accessing confirmRelation field directly.
     * @return Instance for confirmRelation component
```

```
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_confirmRelation() {
        if (confirmRelation == null) {
            // Insert pre-init code here
            confirmRelation = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            confirmRelation.setNextDisplayable(get_form1());
            // Insert post-init code here
        }
        return confirmRelation;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.14. PantallaBusquedaUsuarios

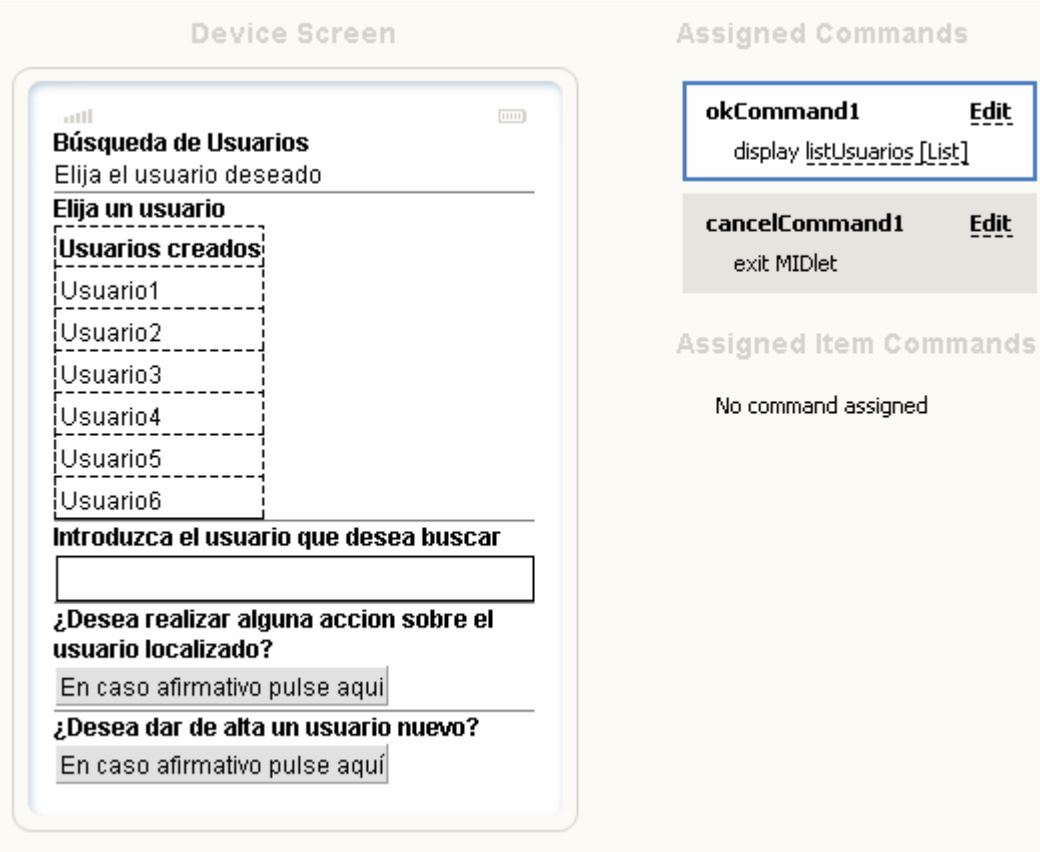

Figura 152. Diseño de GUI de PantallaBusquedaUsuarios

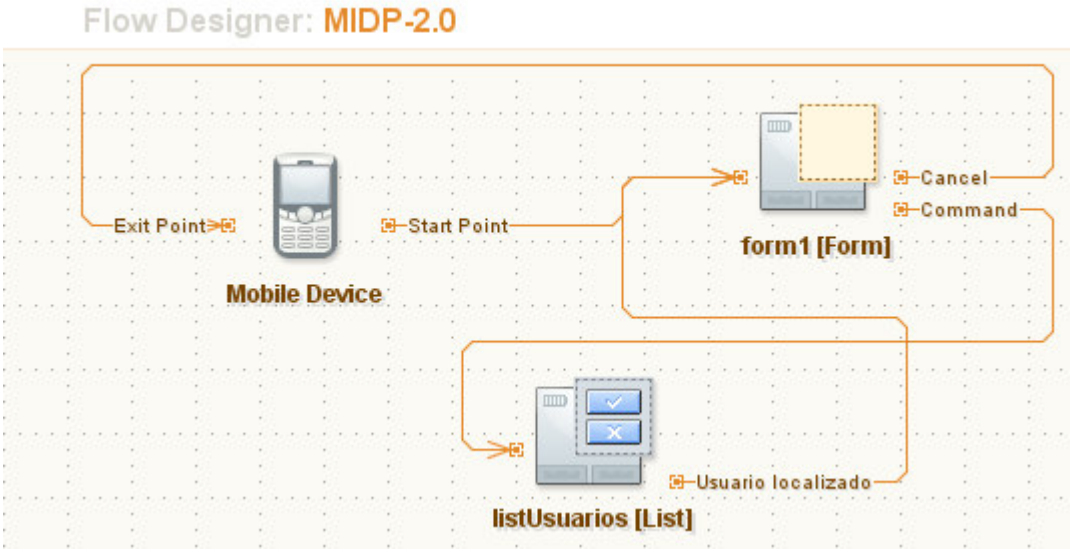

Figura 153. Diseño de flujo de PantallaBusquedaUsuarios

Tabla 46. Código fuente del MIDlet de PantallaBusquedaUsuarios

```
/*
 * PantallaBusquedaUsuarios.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaBusquedaUsuarios extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaBusquedaUsuarios */
    public PantallaBusquedaUsuarios () {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private List listUsuarios;
    private Spacer spacer2;
    private TextField textField1;
    private StringItem stringItem2;
    private Spacer spacer3;
    private Command screenCommand1;
    private StringItem stringItem3;
    private Spacer spacer4;
```

```
    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listUsuarios());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == listUsuarios) {
            if (command == listUsuarios.SELECT_COMMAND) {
                switch (get_listUsuarios().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
                        getDisplay().setCurrent(get_form1());
                        // Insert post-action code here
                        break;
                }
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1(),
                get_spacer3(),
                get_stringItem2(),
```

```
                get_spacer4(),
                get_stringItem3()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("B\u00FAsqueda de Usuarios", "Elija el
usuario deseado");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"Elija un usuario", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Usuario1",
                },
                new String[] {
                    "Usuario2",
                },
                new String[] {
                    "Usuario3",
```

```
                },
                new String[] {
                    "Usuario4",
                },
                new String[] {
                    "Usuario5",
                },
                new String[] {
                    "Usuario6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Usuarios creados",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for listUsuarios component and should be called
instead of accessing listUsuarios field directly.
     * @return Instance for listUsuarios component
     */
    public List get_listUsuarios() {
        if (listUsuarios == null) {
            // Insert pre-init code here
            listUsuarios = new List(null, Choice.IMPLICIT, new String[] {"Usuario
localizado"}, new Image[] {null});
            listUsuarios.setCommandListener(this);
            listUsuarios.setSelectedFlags(new boolean[] {false});
            // Insert post-init code here
        }
        return listUsuarios;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
```

```
        }
        return spacer2;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("Introduzca el usuario que desea buscar",
null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("\u00BFDesea realizar alguna accion sobre el
usuario localizado?", "En caso afirmativo pulse aqui",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }

    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }

    /** This method returns instance for stringItem3 component and should be called
instead of accessing stringItem3 field directly.
     * @return Instance for stringItem3 component
     */
    public StringItem get_stringItem3() {
        if (stringItem3 == null) {
            // Insert pre-init code here
            stringItem3 = new StringItem("\u00BFDesea dar de alta un usuario nuevo?",
"En caso afirmativo pulse aqu\u00ED", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
```

```
        return stringItem3;
    }

    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer4;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.15. PantallaAltaUsuarios

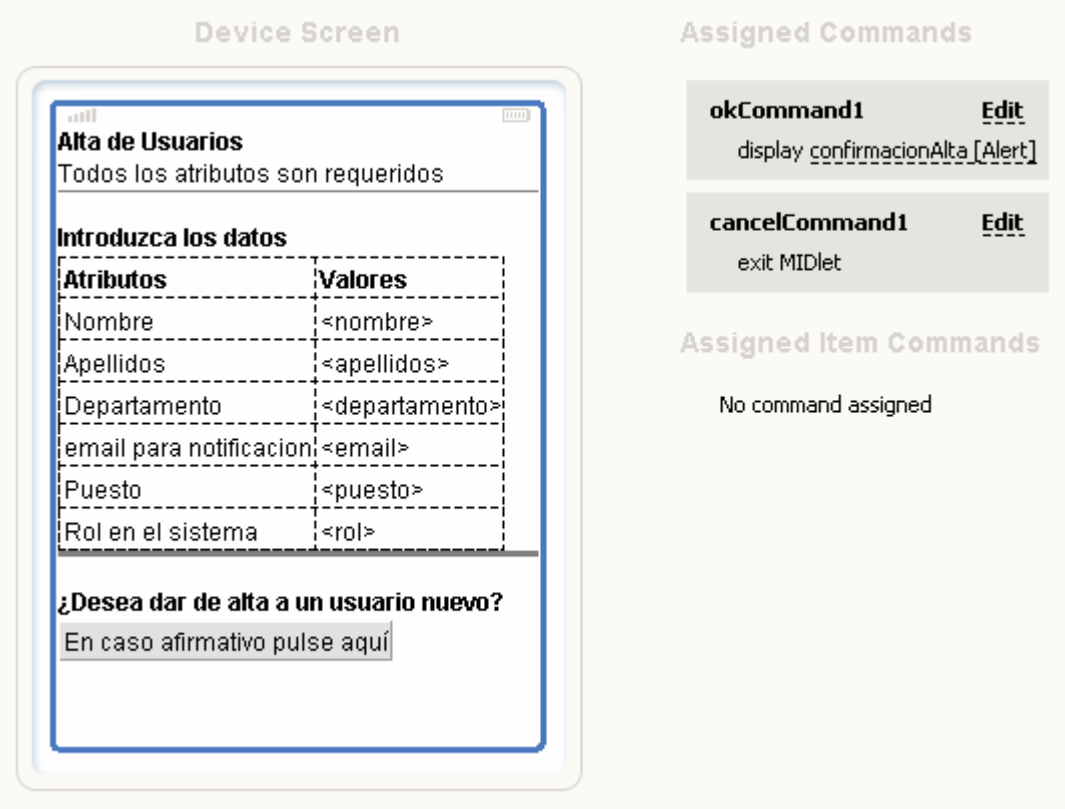

Figura 154. Diseño de GUI de PantallaAltaUsuarios

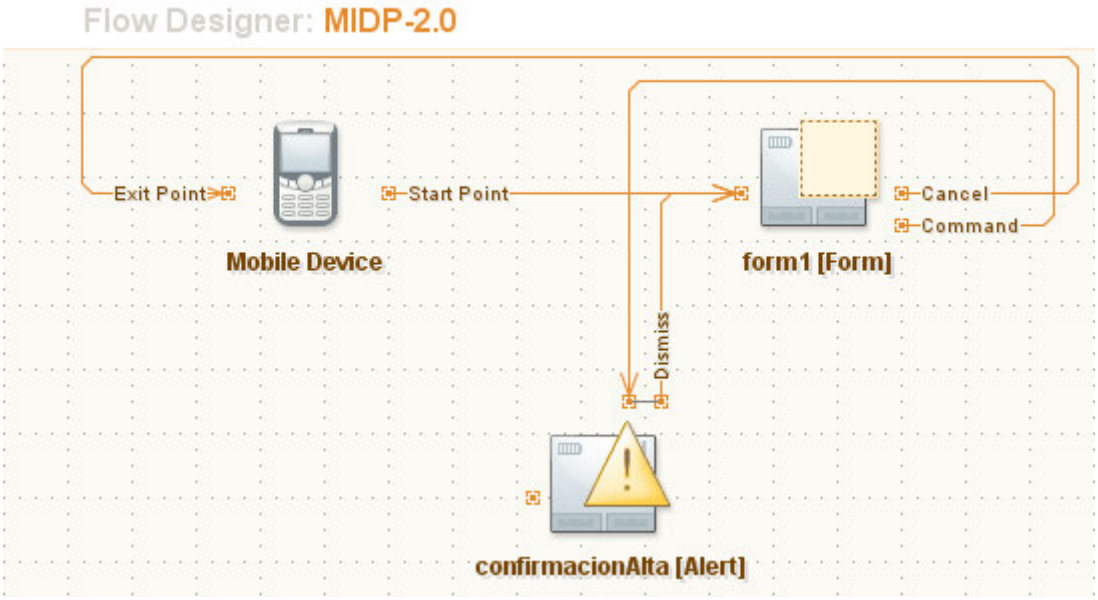

Figura 155. Diseño de flujo de PantallaAltaUsuarios

Tabla 47. Código fuente del MIDlet de PantallaAltaUsuarios

```
/*
 * PantallaAltaUsuarios.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaAltaUsuarios extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaAltaUsuarios */
    public PantallaAltaUsuarios() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private StringItem stringItem3;
    private Spacer spacer4;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private Alert confirmacionAlta;
```

```
    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmacionAlta(), get_form1());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_spacer3(),
                get_spacer4(),
                get_stringItem3()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
```

```
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Alta de Usuarios", "Todos los atributos son
requeridos\n");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"\nIntroduzca los datos\n", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Nombre",
                    "<nombre>",
                },
                new String[] {
                    "Apellidos",
                    "<apellidos>",
                },
                new String[] {
                    "Departamento",
                    "<departamento>",
                },
                new String[] {
                    "email para notificacion",
                    "<email>",
                },
                new String[] {
                    "Puesto",
                    "<puesto>",
                },
```

```
                new String[] {
                    "Rol en el sistema",
                    "<rol>",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Atributos",
                "Valores",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
```

```
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }


    /** This method returns instance for stringItem3 component and should be called
instead of accessing stringItem3 field directly.
     * @return Instance for stringItem3 component
     */
    public StringItem get_stringItem3() {
        if (stringItem3 == null) {
            // Insert pre-init code here
            stringItem3 = new StringItem("  \n\u00BFDesea dar de alta a un usuario
nuevo?", "En caso afirmativo pulse aqu\u00ED", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem3;
    }


    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer4;
    }


    /** This method returns instance for confirmacionAlta component and should be
called instead of accessing confirmacionAlta field directly.
     * @return Instance for confirmacionAlta component
     */
    public Alert get_confirmacionAlta() {
        if (confirmacionAlta == null) {
            // Insert pre-init code here
            confirmacionAlta = new Alert(null, "<Enter Text>", null, AlertType.INFO);
            confirmacionAlta.setTimeout(-2);
            // Insert post-init code here
        }
        return confirmacionAlta;
    }


    public void startApp() {
    }


    public void pauseApp() {
    }


    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.16. PantallaModificacionUsuarios

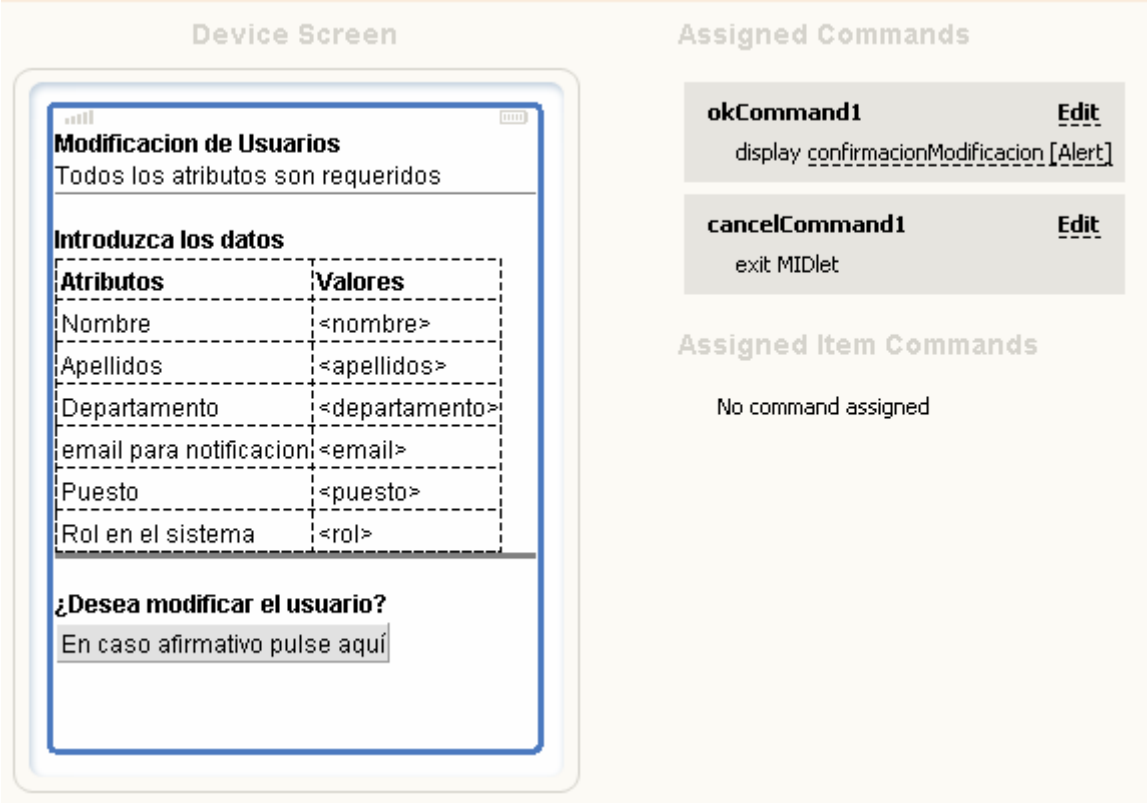

Figura 156. Diseño de GUI de PantallaModificacionUsuarios

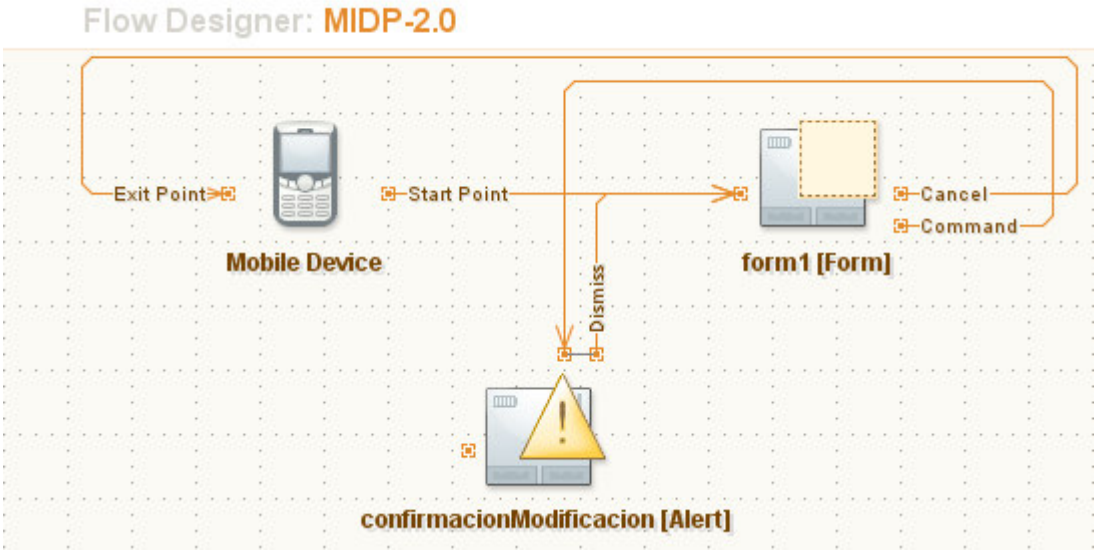

Figura 157. Diseño de flujo de PantallaModificacionUsuarios

Tabla 48. Código fuente del MIDlet de PantallaModificacionUsuarios

```
/*
 * PantallaModificacionUsuarios.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaModificacionUsuarios extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaModificacionUsuarios */
    public PantallaModificacionUsuarios() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private StringItem stringItem3;
    private Spacer spacer4;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private Alert confirmacionModificacion;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmacionModificacion(), get_form1());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
```

```
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_spacer3(),
                get_spacer4(),
                get_stringItem3()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Modificacion de Usuarios", "Todos los
atributos son requeridos\n");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
```

```
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"\nIntroduzca los datos\n", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Nombre",
                    "<nombre>",
                },
                new String[] {
                    "Apellidos",
                    "<apellidos>",
                },
                new String[] {
                    "Departamento",
                    "<departamento>",
                },
                new String[] {
                    "email para notificacion",
                    "<email>",
                },
                new String[] {
                    "Puesto",
                    "<puesto>",
                },
                new String[] {
                    "Rol en el sistema",
                    "<rol>",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Atributos",
                "Valores",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
```

```
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }

    /** This method returns instance for stringItem3 component and should be called
instead of accessing stringItem3 field directly.
     * @return Instance for stringItem3 component
     */
    public StringItem get_stringItem3() {
        if (stringItem3 == null) {
            // Insert pre-init code here
            stringItem3 = new StringItem("  \n\u00BFDesea modificar el usuario?", "En
caso afirmativo pulse aqu\u00ED", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem3;
    }

    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
```

```
        }
        return spacer4;
    }

    /** This method returns instance for confirmacionModificacion component and
should be called instead of accessing confirmacionModificacion field directly.
     * @return Instance for confirmacionModificacion component
     */
    public Alert get_confirmacionModificacion() {
        if (confirmacionModificacion == null) {
            // Insert pre-init code here
            confirmacionModificacion = new Alert(null, "<Enter Text>", null,
AlertType.INFO);
            confirmacionModificacion.setTimeout(-2);
            // Insert post-init code here
        }
        return confirmacionModificacion;
    }
    public void startApp() {
    }
    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.17. PantallaBajaUsuarios

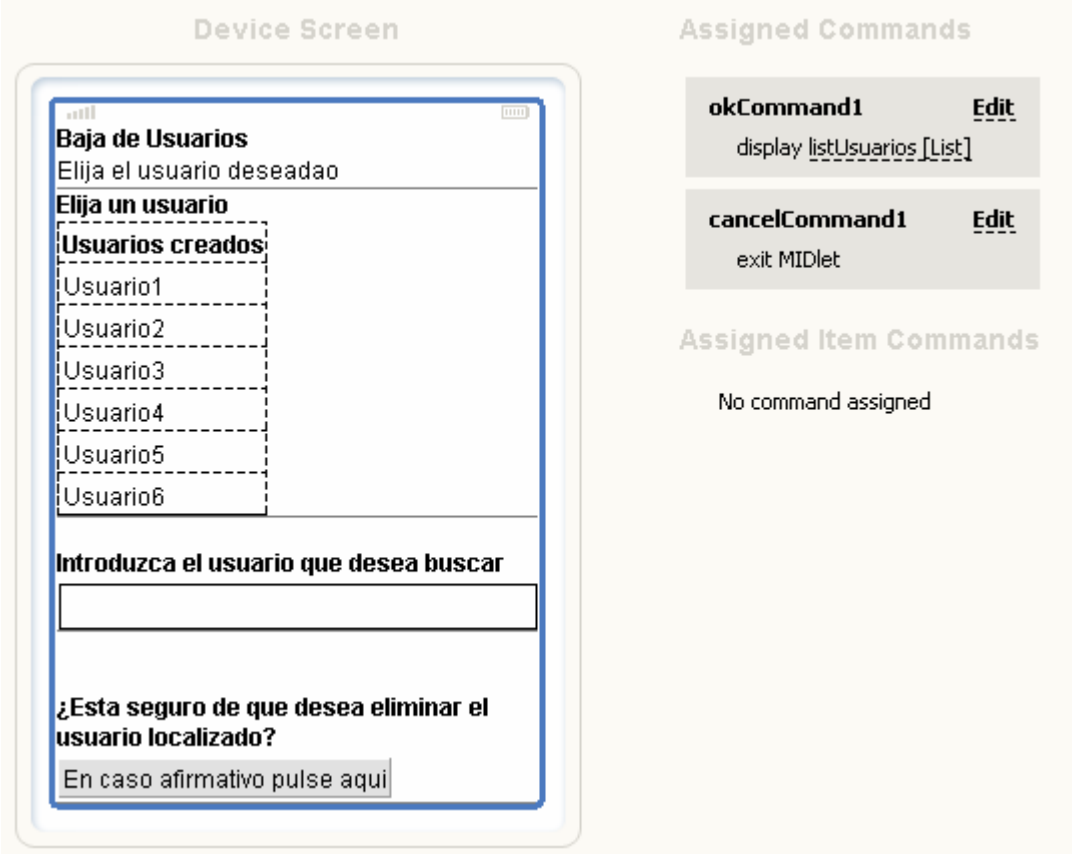

Figura 158. Diseño de GUI de PantallaBajaUsuarios

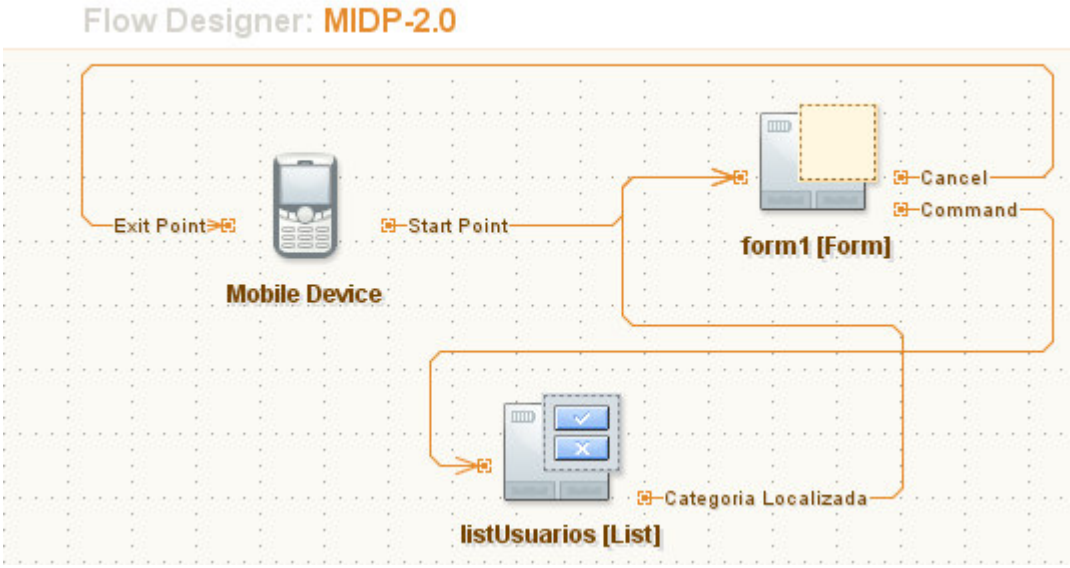

Figura 159. Diseño de flujo de PantallaBajaUsuarios

Tabla 49. Código fuente del MIDlet de PantallaBajaUsuarios

```
/*
 * PantallaBajaUsuarios.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JLS
 */
public class PantallaBajaUsuarios extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaBajaUsuarios */
    public PantallaBajaUsuarios() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private List listUsuarios;
    private Spacer spacer2;
    private TextField textField1;
    private StringItem stringItem2;
    private Spacer spacer3;
    private Command screenCommand1;
    private Spacer spacer4;
```

```
    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listUsuarios());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == listUsuarios) {
            if (command == listUsuarios.SELECT_COMMAND) {
                switch (get_listUsuarios().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
                        getDisplay().setCurrent(get_form1());
                        // Insert post-action code here
                        break;
                }
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1(),
                get_spacer3(),
                get_stringItem2(),
```

```
                get_spacer4()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Baja de Usuarios", "Elija el usuario
deseadao");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"Elija un usuario", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Usuario1",
                },
                new String[] {
                    "Usuario2",
                },
                new String[] {
                    "Usuario3",
                },
```

```
                new String[] {
                    "Usuario4",
                },
                new String[] {
                    "Usuario5",
                },
                new String[] {
                    "Usuario6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Usuarios creados",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for listUsuarios component and should be called
instead of accessing listUsuarios field directly.
     * @return Instance for listUsuarios component
     */
    public List get_listUsuarios() {
        if (listUsuarios == null) {
            // Insert pre-init code here
            listUsuarios = new List(null, Choice.IMPLICIT, new String[] {"Categoria
Localizada"}, new Image[] {null});
            listUsuarios.setCommandListener(this);
            listUsuarios.setSelectedFlags(new boolean[] {false});
            // Insert post-init code here
        }
        return listUsuarios;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
```

```
        return spacer2;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("\nIntroduzca el usuario que desea buscar",
null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("\n\n\u00BFEsta seguro de que desea eliminar
el usuario localizado?", "En caso afirmativo pulse aqui",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }


    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }


    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }


     /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
```

```
            }
            return spacer4;
        }

        public void startApp() {
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

}
```

5.4.18. PantallaSeleccionCategorias

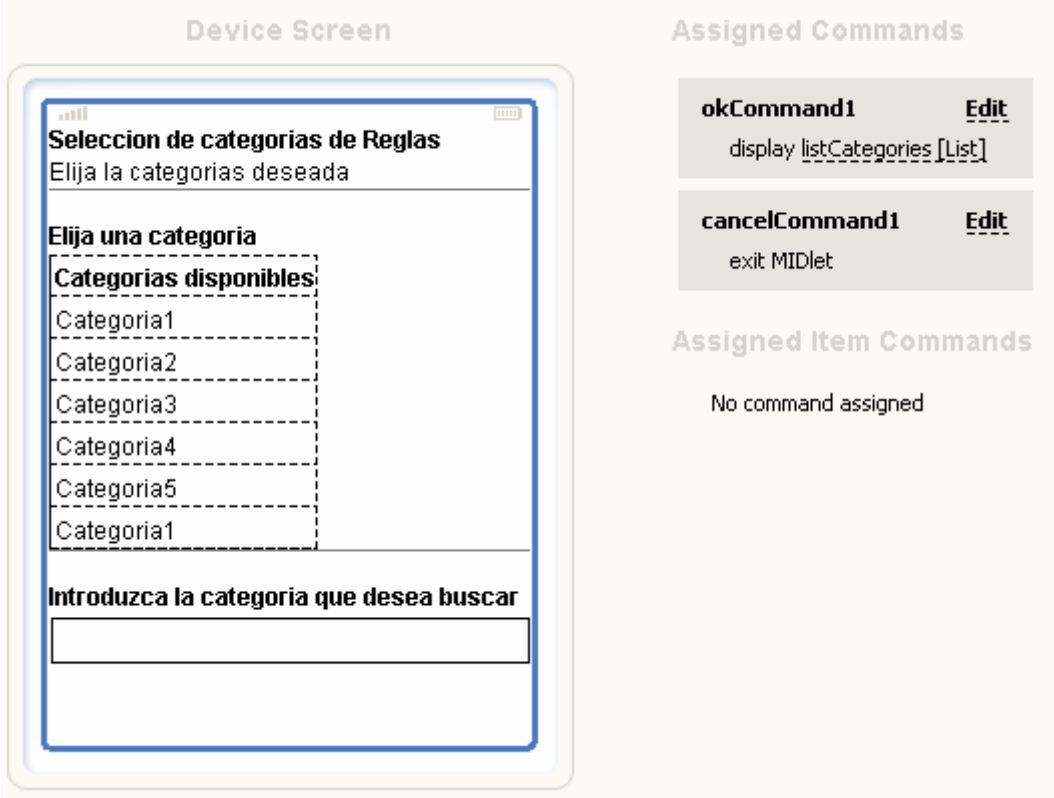

Figura 160. Diseño de GUI de PantallaSeleccionCategorias

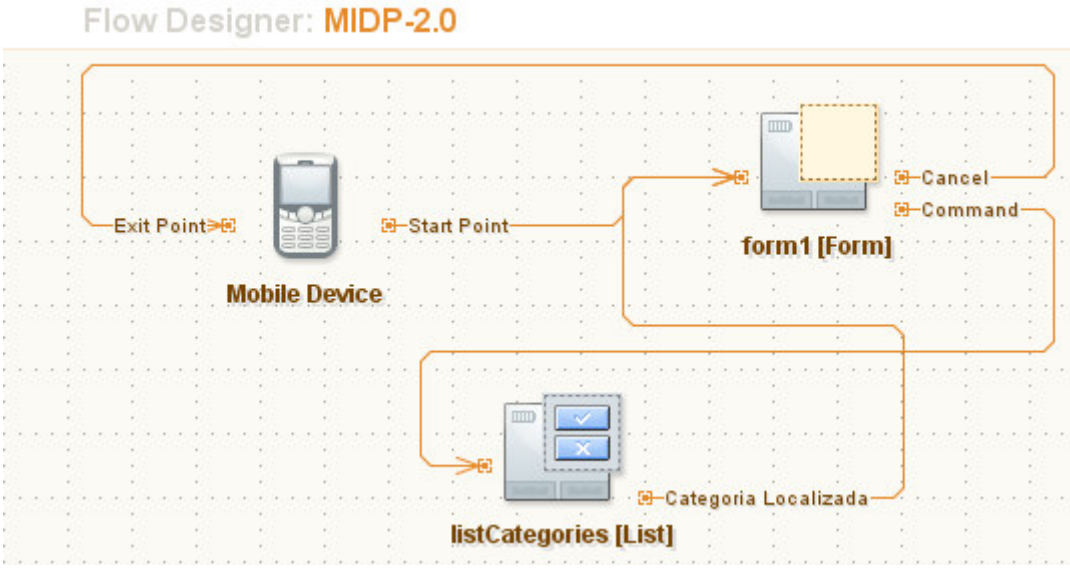

Figura 161. Diseño de flujo de PantallaSeleccionCategorias

Tabla 50. Código fuente del MIDlet de PantallaSeleccionCategorias

```
/*
 * PantallaSeleccionCategorias.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaSeleccionCategorias extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaSeleccionCategorias */
    public PantallaSeleccionCategorias() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private List listCategories;
    private Spacer spacer2;
    private TextField textField1;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
```

```
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listCategories());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == listCategories) {
            if (command == listCategories.SELECT_COMMAND) {
                switch (get_listCategories().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
                        getDisplay().setCurrent(get_form1());
                        // Insert post-action code here
                        break;
                }
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
```

```
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Seleccion de categorias de Reglas", "Elija
la categorias deseada ");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"    \nElija una categoria", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Categoria1",
                },
                new String[] {
                    "Categoria2",
                },
                new String[] {
                    "Categoria3",
                },
                new String[] {
                    "Categoria4",
                },
                new String[] {
                    "Categoria5",
                },
                new String[] {
```

```
                    "Categoria1",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Categorias disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for listCategories component and should be
called instead of accessing listCategories field directly.
     * @return Instance for listCategories component
     */
    public List get_listCategories() {
        if (listCategories == null) {
            // Insert pre-init code here
            listCategories = new List(null, Choice.IMPLICIT, new String[] {"Categoria
Localizada"}, new Image[] {null});
            listCategories.setCommandListener(this);
            listCategories.setSelectedFlags(new boolean[] {false});
            // Insert post-init code here
        }
        return listCategories;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
```

```
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("   \nIntroduzca la categoria que desea
buscar", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.19. PantallaBusquedaCategoriaRegla

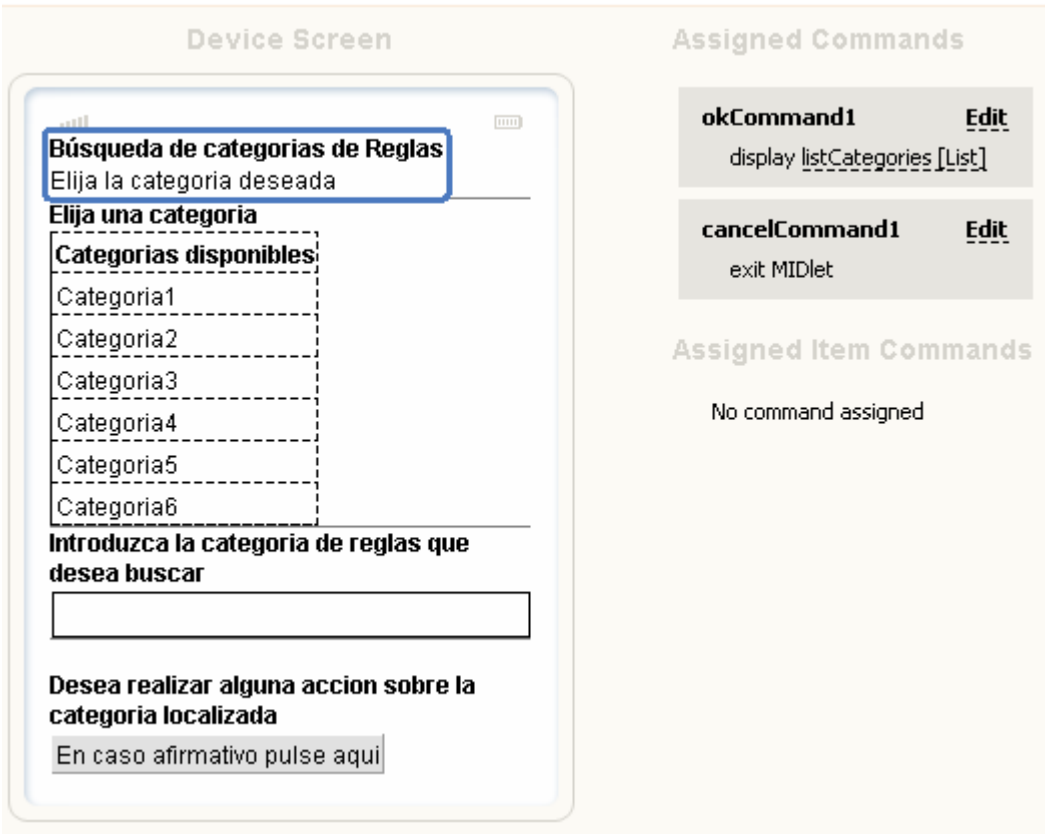

Figura 162. Diseño de GUI de PantallaBusquedaCategoriaRegla

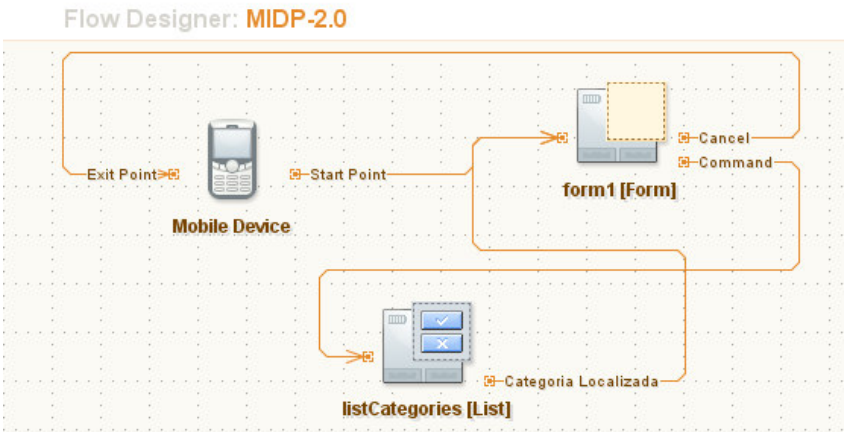

Figura 163. Diseño de flujo de PantallaBusquedaCategoriaRegla

Tabla 51. Código fuente del MIDlet de PantallaBusquedaCategoriaRegla

```
/*
 * PantallaBusquedaCategoriaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaBusquedaCategoriaRegla extends MIDlet implements CommandListener
{

    /** Creates a new instance of PantallaBusquedaCategoriaRegla */
    public PantallaBusquedaCategoriaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private List listCategories;
    private Spacer spacer2;
    private TextField textField1;
    private StringItem stringItem2;
    private Spacer spacer3;
    private Command screenCommand1;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
```

```
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listCategories());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == listCategories) {
            if (command == listCategories.SELECT_COMMAND) {
                switch (get_listCategories().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
                        getDisplay().setCurrent(get_form1());
                        // Insert post-action code here
                        break;
                }
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1(),
                get_spacer3(),
                get_stringItem2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
```

```
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("B\u00FAsqueda de categorias de Reglas",
"Elija la categoria deseada ");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"Elija una categoria", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Categoria1",
                },
                new String[] {
                    "Categoria2",
                },
                new String[] {
                    "Categoria3",
                },
                new String[] {
                    "Categoria4",
                },
                new String[] {
                    "Categoria5",
                },
                new String[] {
```

```
                    "Categoria6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Categorias disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for listCategories component and should be
called instead of accessing listCategories field directly.
     * @return Instance for listCategories component
     */
    public List get_listCategories() {
        if (listCategories == null) {
            // Insert pre-init code here
            listCategories = new List(null, Choice.IMPLICIT, new String[] {"Categoria
Localizada"}, new Image[] {null});
            listCategories.setCommandListener(this);
            listCategories.setSelectedFlags(new boolean[] {false});
            // Insert post-init code here
        }
        return listCategories;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
```

```
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("Introduzca la categoria de reglas que desea
buscar", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("   \nDesea realizar alguna accion sobre la
categoria localizada", "En caso afirmativo pulse aqui",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }

    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.20. PantallaAltaCategoriaRegla

Figura 164. Diseño de GUI de PantallaAltaCategoriaRegla

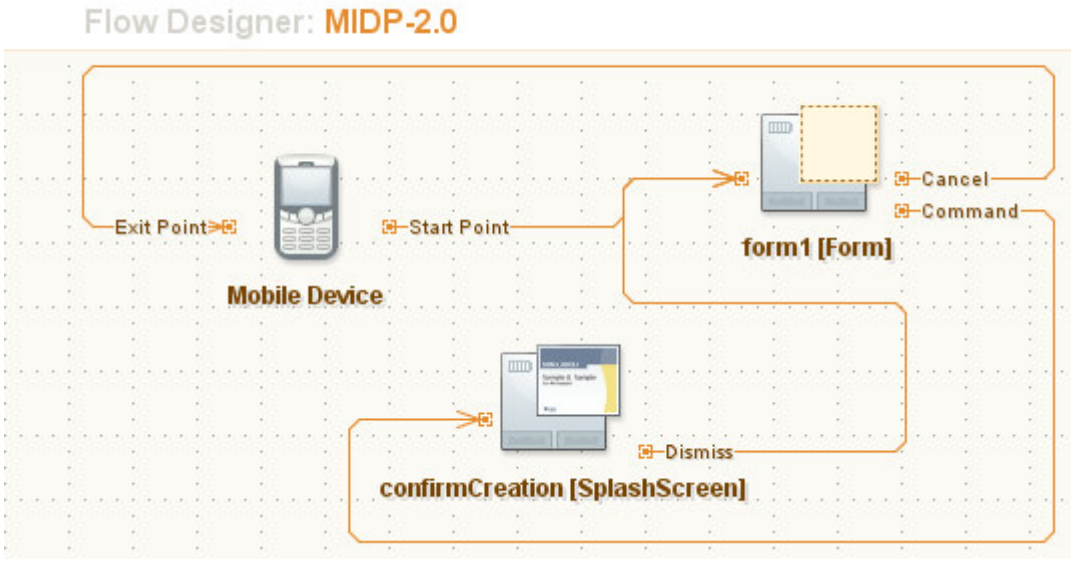

Figura 165. Diseño de flujo de PantallaAltaCategoriaRegla

Tabla 52. Código fuente del MIDlet de PantallaAltaCategoriaRegla

```
/*
 * PantallaAltaCategoriaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaAltaCategoriaRegla extends MIDlet implements CommandListener {

    /** Creates a new instance of PantallaAltaCategoriaRegla */
    public PantallaAltaCategoriaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private Spacer spacer4;
    private org.netbeans.microedition.lcdui.SplashScreen confirmCreation;
    private TextField textField1;
    private TextField textField2;
    private StringItem stringItem2;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmCreation());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}


    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
```

```
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }


    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }


    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_textField1(),
                get_spacer2(),
                get_textField2(),
                get_spacer3(),
                get_spacer4(),
                get_stringItem2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Alta de Categoria", "<id>");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
```

```
    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Usuario1",
                },
                new String[] {
                    "Usuario2",
                },
                new String[] {
                    "Usuario3",
                },
                new String[] {
                    "Usuario4",
                },
                new String[] {
                    "Usuario5",
                },
                new String[] {
                    "Usuario6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Usuarios creados",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }


    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }


    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }


    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
```

```
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }



    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }
    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer4;
    }



    /** This method returns instance for confirmCreation component and should be
called instead of accessing confirmCreation field directly.
     * @return Instance for confirmCreation component
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_confirmCreation() {
        if (confirmCreation == null) {
            // Insert pre-init code here
            confirmCreation = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            confirmCreation.setNextDisplayable(get_form1());
            // Insert post-init code here
        }
        return confirmCreation;
    }


    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
```

```
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("  \nDenominacion", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for textField2 component and should be called
instead of accessing textField2 field directly.
     * @return Instance for textField2 component
     */
    public TextField get_textField2() {
        if (textField2 == null) {
            // Insert pre-init code here
            textField2 = new TextField(" \nCaracteristicas", null, 120,
TextField.ANY);
            // Insert post-init code here
        }
        return textField2;
    }


    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("  \n\u00BFEsta seguro de la creacion de
esta \ncategoria?", "En casi afirmativo\nPulse aqui",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }


    public void startApp() {
    }


    public void pauseApp() {
    }


    public void destroyApp(boolean unconditional) {
    }

}
```

5.4.21. PantallaModificacionCategoriaRegla

Figura 166. Diseño de GUI de PantallaModificacionCategoriaRegla

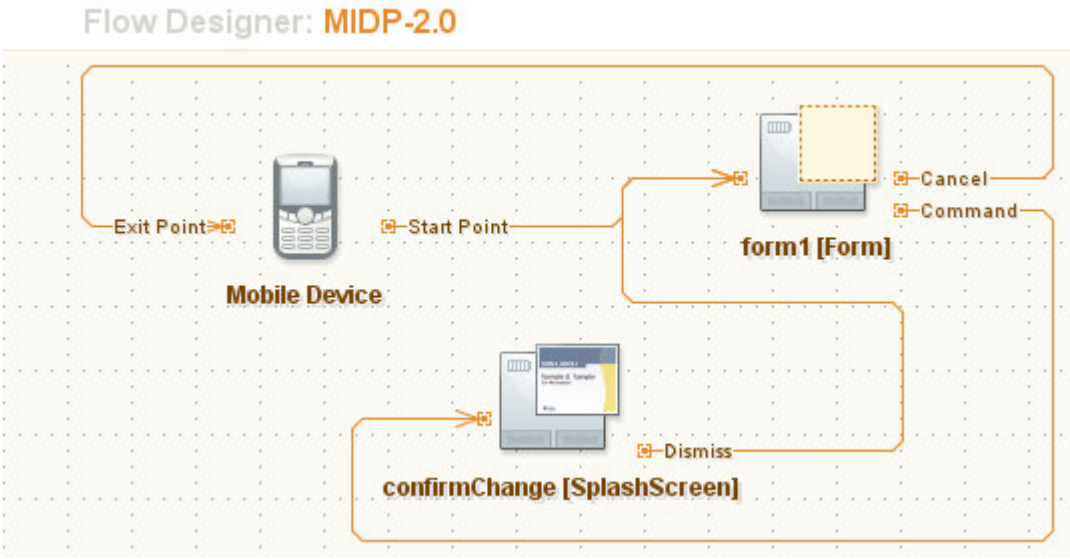

Figura 167. Diseño de flujo de PantallaModificacionCategoriaRegla

Tabla 53. Código fuente del MIDlet de PantallaModificacionCategoriaRegla

```
/*
 * PantallaModificacionCategoriaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JOSE LUIS SALMERON SILVERA
 */
public class PantallaModificacionCategoriaRegla extends MIDlet implements
CommandListener {

    /** Creates a new instance of PantallaModificacionCategoriaRegla */
    public PantallaModificacionCategoriaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private Spacer spacer3;
    private Command screenCommand1;
    private Spacer spacer4;
    private org.netbeans.microedition.lcdui.SplashScreen confirmChange;
    private TextField textField1;
    private TextField textField2;
    private StringItem stringItem2;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmChange());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }
```

```
    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_textField1(),
                get_spacer2(),
                get_textField2(),
                get_spacer3(),
                get_spacer4(),
                get_stringItem2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Eliminacion de Categoria", "<id>");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }
    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
```

```
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Categoria1",
                },
                new String[] {
                    "Categoria2",
                },
                new String[] {
                    "Categoria3",
                },
                new String[] {
                    "Categoria4",
                },
                new String[] {
                    "Categoria5",
                },
                new String[] {
                    "Categoria6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Categorias disponibles",
            });
            // Insert post-init code here
        }
        return simpleTableModel1;
    }
    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }
```

```
    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }
    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }
    /** This method returns instance for spacer4 component and should be called
instead of accessing spacer4 field directly.
     * @return Instance for spacer4 component
     */
    public Spacer get_spacer4() {
        if (spacer4 == null) {
            // Insert pre-init code here
            spacer4 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer4;
    }
    /** This method returns instance for confirmChange component and should be called
instead of accessing confirmChange field directly.
     * @return Instance for confirmChange component
     */
    public org.netbeans.microedition.lcdui.SplashScreen get_confirmChange() {
        if (confirmChange == null) {
            // Insert pre-init code here
            confirmChange = new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            confirmChange.setNextDisplayable(get_form1());
            // Insert post-init code here
        }
        return confirmChange;
    }
    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("  \nDenominacion", "<DenominacionCategoria>",
120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for textField2 component and should be called
instead of accessing textField2 field directly.
     * @return Instance for textField2 component
     */
```

```
    public TextField get_textField2() {
        if (textField2 == null) {
            // Insert pre-init code here
            textField2 = new TextField(" \nCaracteristicas", "<Caracteristicas>",
120, TextField.ANY);
            // Insert post-init code here
        }
        return textField2;
    }
    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("  \n\u00BFEsta seguro de la modificacion de
esta \ncategoria?", "En casi afirmativo\nPulse aqui",
javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }
    public void startApp() {
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
}
```

5.4.22. PantallaEliminarCategoriaRegla

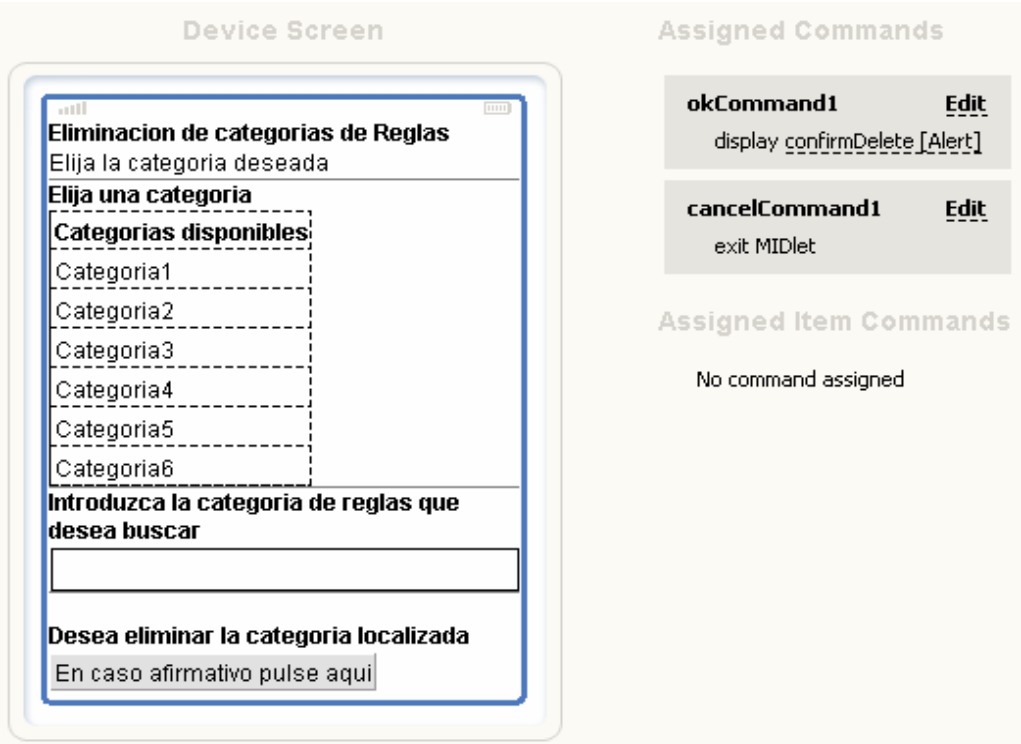

Figura 168. Diseño de GUI de PantallaEliminarCategoriaRegla

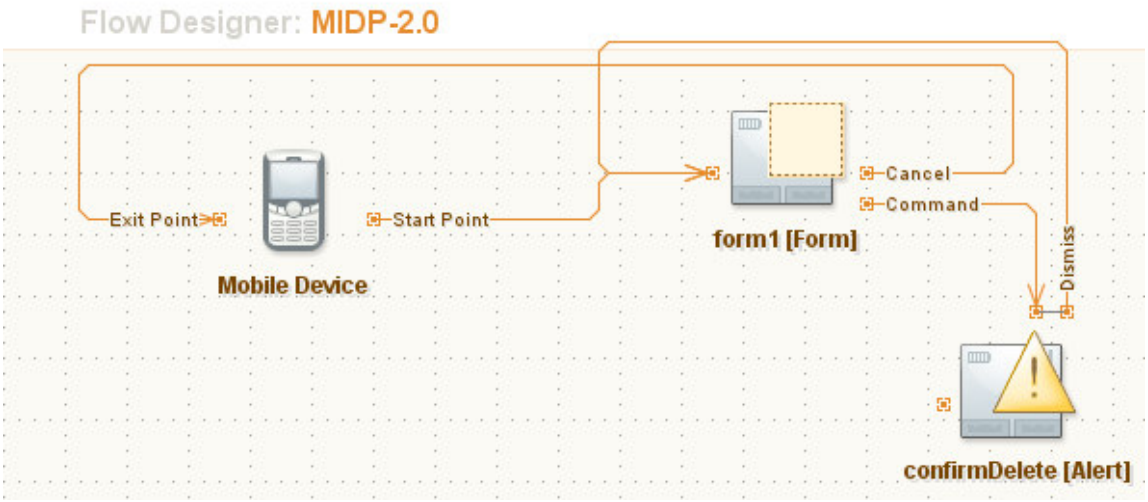

Figura 169. Diseño de flujo de PantallaEliminarCategoriaRegla

Tabla 54. Código fuente del MIDlet de PantallaEliminarCategoriaRegla

```
/*
 * PantallaEliminarCategoriaRegla.java
 *
 */

package TFC;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author JLS
 */
public class PantallaEliminarCategoriaRegla extends MIDlet implements CommandListener
{

    /**
     * Creates a new instance of PantallaEliminarCategoriaRegla
     */
    public PantallaEliminarCategoriaRegla() {
        initialize();
    }

    private Form form1;
    private StringItem stringItem1;
    private Spacer spacer1;
    private org.netbeans.microedition.lcdui.TableItem tableItem1;
    private org.netbeans.microedition.lcdui.SimpleTableModel simpleTableModel1;
    private Command okCommand1;
    private Command cancelCommand1;
    private Spacer spacer2;
    private TextField textField1;
    private StringItem stringItem2;
    private Spacer spacer3;
    private Command screenCommand1;
    private Alert confirmDelete;



    /** Called by the system to indicate that a command has been invoked on a
particular displayable.
     * @param command the Command that ws invoked
     * @param displayable the Displayable on which the command was invoked
```

```
     */
    public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
        if (displayable == form1) {
            if (command == okCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_confirmDelete(), get_form1());
                // Insert post-action code here
            } else if (command == cancelCommand1) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_form1());
        // Insert post-init code here
    }

    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for form1 component and should be called instead
of accessing form1 field directly.
     * @return Instance for form1 component
     */
    public Form get_form1() {
        if (form1 == null) {
            // Insert pre-init code here
            form1 = new Form(null, new Item[] {
                get_stringItem1(),
                get_spacer1(),
                get_tableItem1(),
                get_spacer2(),
                get_textField1(),
                get_spacer3(),
                get_stringItem2()
            });
            form1.addCommand(get_okCommand1());
            form1.addCommand(get_cancelCommand1());
            form1.setCommandListener(this);
            // Insert post-init code here
        }
        return form1;
    }

    /** This method returns instance for stringItem1 component and should be called
instead of accessing stringItem1 field directly.
     * @return Instance for stringItem1 component
     */
    public StringItem get_stringItem1() {
```

```
        if (stringItem1 == null) {
            // Insert pre-init code here
            stringItem1 = new StringItem("Eliminacion de categorias de Reglas",
"Elija la categoria deseada ");
            // Insert post-init code here
        }
        return stringItem1;
    }

    /** This method returns instance for spacer1 component and should be called
instead of accessing spacer1 field directly.
     * @return Instance for spacer1 component
     */
    public Spacer get_spacer1() {
        if (spacer1 == null) {
            // Insert pre-init code here
            spacer1 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer1;
    }

    /** This method returns instance for tableItem1 component and should be called
instead of accessing tableItem1 field directly.
     * @return Instance for tableItem1 component
     */
    public org.netbeans.microedition.lcdui.TableItem get_tableItem1() {
        if (tableItem1 == null) {
            // Insert pre-init code here
            tableItem1 = new org.netbeans.microedition.lcdui.TableItem(getDisplay(),
"Elija una categoria", get_simpleTableModel1());
            // Insert post-init code here
        }
        return tableItem1;
    }

    /** This method returns instance for simpleTableModel1 component and should be
called instead of accessing simpleTableModel1 field directly.
     * @return Instance for simpleTableModel1 component
     */
    public org.netbeans.microedition.lcdui.SimpleTableModel get_simpleTableModel1() {
        if (simpleTableModel1 == null) {
            // Insert pre-init code here
            simpleTableModel1 = new
org.netbeans.microedition.lcdui.SimpleTableModel();
            simpleTableModel1.setValues(new String[][] {
                new String[] {
                    "Categoria1",
                },
                new String[] {
                    "Categoria2",
                },
                new String[] {
                    "Categoria3",
                },
                new String[] {
                    "Categoria4",
                },
                new String[] {
                    "Categoria5",
                },
                new String[] {
                    "Categoria6",
                },
            });
            simpleTableModel1.setColumnNames(new String[] {
                "Categorias disponibles",
            });
            // Insert post-init code here
        }
```

```
        return simpleTableModel1;
    }

    /** This method returns instance for okCommand1 component and should be called
instead of accessing okCommand1 field directly.
     * @return Instance for okCommand1 component
     */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Command", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for cancelCommand1 component and should be
called instead of accessing cancelCommand1 field directly.
     * @return Instance for cancelCommand1 component
     */
    public Command get_cancelCommand1() {
        if (cancelCommand1 == null) {
            // Insert pre-init code here
            cancelCommand1 = new Command("Cancel", Command.CANCEL, 1);
            // Insert post-init code here
        }
        return cancelCommand1;
    }

    /** This method returns instance for spacer2 component and should be called
instead of accessing spacer2 field directly.
     * @return Instance for spacer2 component
     */
    public Spacer get_spacer2() {
        if (spacer2 == null) {
            // Insert pre-init code here
            spacer2 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer2;
    }

    /** This method returns instance for textField1 component and should be called
instead of accessing textField1 field directly.
     * @return Instance for textField1 component
     */
    public TextField get_textField1() {
        if (textField1 == null) {
            // Insert pre-init code here
            textField1 = new TextField("Introduzca la categoria de reglas que desea
buscar", null, 120, TextField.ANY);
            // Insert post-init code here
        }
        return textField1;
    }

    /** This method returns instance for stringItem2 component and should be called
instead of accessing stringItem2 field directly.
     * @return Instance for stringItem2 component
     */
    public StringItem get_stringItem2() {
        if (stringItem2 == null) {
            // Insert pre-init code here
            stringItem2 = new StringItem("   \nDesea eliminar la categoria
localizada", "En caso afirmativo pulse aqui", javax.microedition.lcdui.Item.BUTTON);
            // Insert post-init code here
        }
        return stringItem2;
    }
```

```
    /** This method returns instance for spacer3 component and should be called
instead of accessing spacer3 field directly.
     * @return Instance for spacer3 component
     */
    public Spacer get_spacer3() {
        if (spacer3 == null) {
            // Insert pre-init code here
            spacer3 = new Spacer(1000, 1);
            // Insert post-init code here
        }
        return spacer3;
    }

    /** This method returns instance for screenCommand1 component and should be
called instead of accessing screenCommand1 field directly.
     * @return Instance for screenCommand1 component
     */
    public Command get_screenCommand1() {
        if (screenCommand1 == null) {
            // Insert pre-init code here
            screenCommand1 = new Command("Screen", Command.SCREEN, 1);
            // Insert post-init code here
        }
        return screenCommand1;
    }

    /** This method returns instance for confirmDelete component and should be called
instead of accessing confirmDelete field directly.
     * @return Instance for confirmDelete component
     */
    public Alert get_confirmDelete() {
        if (confirmDelete == null) {
            // Insert pre-init code here
            confirmDelete = new Alert(null, "<Enter Text>", null, AlertType.INFO);
            confirmDelete.setTimeout(-2);
            // Insert post-init code here
        }
        return confirmDelete;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

6. Referencias

Se recogen en este apartado dos tipos de referencias, la documentación formal (epígrafe 6.1.), constituida por libros y artículos científicos publicados en revistas o congresos y otra documentación (epígrafe 6.2.) constituida por referencias a URLs.

La notación elegida para referenciar las citas consta de tres caracteres para identificar al autor principal y dos cifras relativas al año de publicación.

6.1. Documentación formal

[BAJ05] Bajec, M. Krisper, M. A methodology and tool support for managing business rules in organisations, Information Systems 30. 2005. pp. 423-443.

[CAM04] Campderrich, B. Análisis orientado a objetos. Fundació per a la Universitat Oberta de Catalunya. 2004.

[BEY95] Beber, H.R., Holtzblatt, K. Apprenticing with the Customer. Communications of the ACM, 38(5), Mayo 1995.

[BOO99] Booch, G., Rumbaugh, J., Jacobson, I. El Lenguaje Unificado de Modelado. Prentice-Hall. 1999.

[BUB93] Bubenko, J.A., Wangler, B. Objectives driven capture of business rules and of information systems requirements, Proceedings of International Conference on Systems, Man and Cybernetics. Systems Engineering in the Service of Humans, 1, 1993, pp. 670–677.

[COC97] Cockburn, A. Structuring Use Cases with Goals. Journal of Object–Oriented Programming. Nov.-Dic. 1997.

[COS05] Costal, D. Diseño de bases de datos. Fundació per a la Universitat Oberta de Catalunya. 2005.

[DAW02] Dawson, C., Martín, G. El proyecto fin de carrera en informática. Prentice Hall. 2002.

[DUR00] Duran, A. Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información. Tesis Doctoral. Universidad de Sevilla. 2000.

[ERI00] Ericsson, H, Pender, M. Business Modeling with UML. Wiley.

[FER01] Ferreira, M.J., Loucopoulos, P. (2001). Organisation of analysis patterns for effective re-use. Proceedings of the International Conference on Enterprise Information Systems. ICEIS 2001. Setubal, Portugal.

[GAL03] Galvez Rojas, S., Ortega Díaz, L. J2ME. Universidad de Málaga. 2003

[GAU89] Gause, D.C., Weinberg, G.M. Exploring Requirements: Quality Before Design. Dorset House, 1989.

[GOG93] Goguen, J.A., Linde, C. Techniques for Requirements Elicitation. En Proceedings of the First International Symposium on Requirements Engineering, 1993.

[IEE98] IEEE Recommended Practice for Software Requirements Specifications. Software Engineering Standards Committee of the IEEE Computer Society. IEEE Std 830-1998.

[JAC00] Jacobson, I, Booch, G., Rumbaugh, J. El proceso Unificado de Desarrollo de Software. Addison Wesley. 2000.

[JAC93] Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G. Object–Oriented Software Engineering: A Use Case Driven Approach. Addison–Wesley, 4a edición, 1993.

[LAY88] Layzell, P.J., Loucopoulos, P. A rule-based approach to the construction and evolution of business information systems, Proceedings of the Fourth IEEE International Conference on Software Maintenance, Phoenix, Arizona, USA, 1988, pp. 258–264.

[MCO97] McConnell, S. Desarrollo y gestión de proyectos informáticos. McGraw-Hill. 1977.

[MOR02] Morgan, T. Business Rules and Information Systems. Addison-Wesley. 2002.

[OMG06] OMG. Semantics of Business Vocabulary and Business Rules Specification (SBVR). 2006. Disponible en http://www.omg.org/cgi-bin/doc?dtc/06-03-02. Fecha de acceso 23-03-06.

[PET94] Petrounias, I., Loucopoulos, P. A rule based approach for the design and implementation of information systems, in: M. Jarke (Ed.), Proceedings EDBT '94. Springer, Cambridge, UK, 1994.

[PIA96] Piattini, M.G., Calvo-Manzano, J.A., Cervera, J., Fernández, L. Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión. Ra–Ma, 1996.

[RAG94] Raghavan, S., Zelesnik, G., Ford, G. Lecture Notes on Requirements Elicitation. Educational Materials CMU/SEI–94–EM–10, Software Engineering Institute, Carnegie Mellon University, 1994.

[ROL98] Rolland, C. Ben Achour, C. Cauvet, C. Ralyté, J. Sutcliffe, A. Maiden, N.A.M. Jarke, M. Haumer, P. Pohl, K. Dubois, E., Heymans, P. A Proposal for a Scenario Classification Framework. Requirements Engineering Journal, 3(1), 1998.

[ROS03] ROSS, R.G. Principles of Business Rule Approach, Addison-Wesley. 2003.

[RUM00] Rumbaugh, J., Jacobson, I., Booch, G. El Lenguaje Unificado de Modelado. Prentice-Hall. 1999.

[SCH98] Schneider, G., Winters, J.P. Applying Use Cases: a Practical Guide. Addison–Wesley, 1998.

[VAS88] Van Assche, F., Layzell, P.J., Anderson, M. RUBRIC—A rule-based approach to information systems requirements. Proceedings of the First European Conference on Information Technology for Organisational Systems. Athens, May, 1988.

[VHA02] von Halle, B. Business Rules Applied. John Wiley & Sons. 2002.

[WEI98] Weidenhaput, K. Pohl, K. Jarke, M., Haumer, P. Scenarios in System Development: Current Practice. IEEE Software, 15(2):34–45, Marzo/Abril 1998.

6.2. Otra documentación

[BRC06] Business Rules Community. http://www.brcommunity.com/

[BRG00] Business Rule Group. http://www.businessrulesgroup.org/

[BRF06] Business Rules Forum. http://www.businessrulesforum.com/

[J2M06] Java2 Micro Edition. http://java.sun.com/javame/

[RML06] The Rule Markup Language Initiative. http://www.ruleml.org/

[UML04] UML 2.0 Superstructure Specification. http://www.omg.org/cgi-bin/doc?formal/05-07-04.

7. Anexos.

7.1. Anexo I. Planificación con hitos y temporización

La planificación se basa en las fechas de entrega de PEC indicadas en el plan docente del TFC disponible en el aula correspondiente del campus virtual de la UOC, así cómo los productos que deben entregarse cómo resultado del TFC indicados en el mismo documento.

En la planificación seguida en este documento no se ha considerado la realización de iteraciones. Ello se debe a que se dispone de un único recurso humano para su desarrollo.

Por ello, en lugar de seguir la recomendación del Proceso Unificado de Desarrollo de Software [JAC00] consistente en dichas iteraciones (como se detalla en la Figura 170), se elige un ciclo de vida en cascada pura. No resulta adecuado el uso de cascadas modificadas como sugiere McConnell [MCO97] ya que los hitos marcados por las entregas no permiten el solapamiento de etapas.

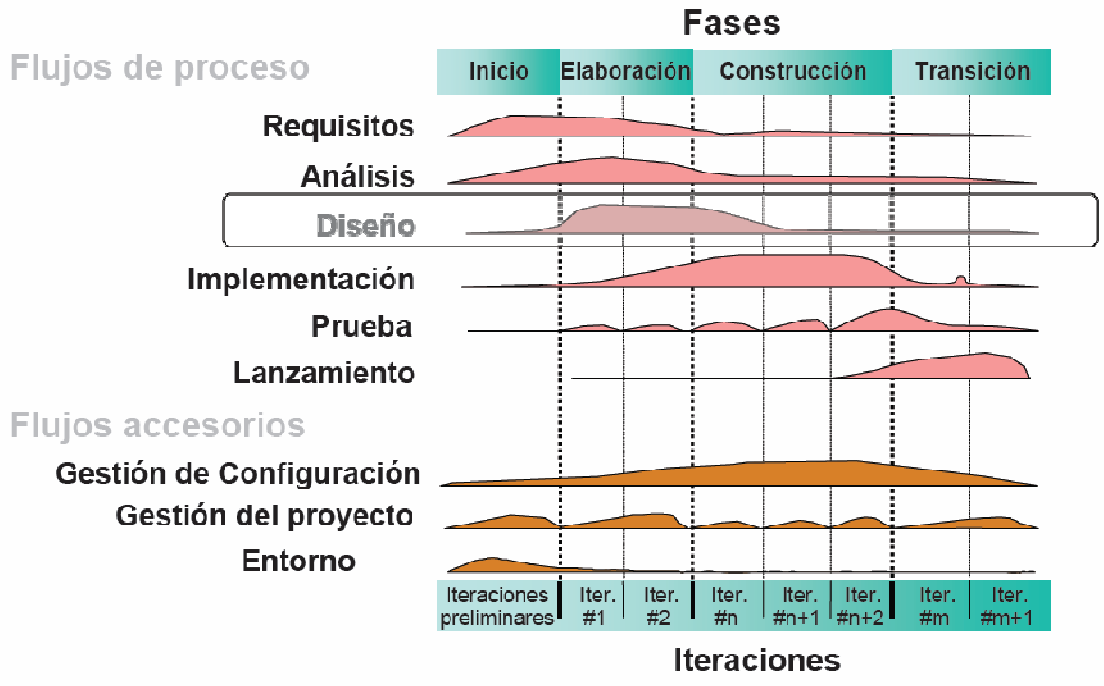

Figura 170. Fases del proceso unificado

7.1.1. Identificación de hitos

Para la realización del TFC se afrontan cuatro hitos, que denominaremos hitos principales, los cuales vienen determinados por la UOC.

Asimismo hemos marcado con carácter previo a los hitos principales una serie de tareas de seguridad, las cuales tiene como misión proporcionarnos una holgura para afrontar posibles retrasos en tareas. Dichas tareas las hemos denominado como *Depuración documentos PECX*, siendo X el número de orden de la Prueba de Evaluación Contínua.

Durante su desarrollo se revisarán las salidas de todas las tareas anteriores, pudiendo reducirse su duración si existe retraso respecto a la planificación establecida. Todos los hitos y tareas de seguridad referidos quedan reflejados en la Tabla 55.

Tabla 55. Hitos y tareas de seguridad del TFC

Hito	Fecha	Categoría
Depuración documentos PEC1	10/03/06-12/03/06	Tarea de seguridad
Entrega PEC1	13/03/06	Hito
Depuración documentos PEC2	19/04/06-20/04/06	Tarea de seguridad
Entrega PEC2	21/04/06	Hito
Depuración documentos PEC3	27/05/06-28/05/06	Tarea de seguridad
Entrega PEC3	29/05/06	Hito
Depuración documentos PEC4	15/06/06	Tarea de seguridad
Entrega PEC4	16/06/06	Hito

7.1.2. Descomposición estructural de actividades (WBS)

Los grafos de descomposición estructural de actividades (Work Breakdown Structure, WBS) muestran la estructura de un proyecto mostrando su organización en lo relativo a fases y tareas, así como el detalle de su estructura en niveles.

Esta clase de descomposición permite ofrecer una visión sintética de la estructura del proyecto, enfocando las actividades que requieren un mayor detalle.

En la Tabla 56 se detallan los diferentes componentes a tres niveles.

Tabla 56. Descomposición estructural de actividades		
Nivel 1	**Nivel 2**	**Nivel 3**
PEC1 - Plan de trabajo	Identificación del TFC	
	Definición de objetivos generales y específicos	
	Depuración documentos PEC1	
PEC2 - Captura de requisitos y análisis	Investigación tema TFC	
	Identificación de los subsistemas	
	Especificación de requisitos	
	Modelo de análisis	Identificación de actores
		Diagrama de casos de uso
		Especificación de casos de uso
	Documento de visión	
	Documento de análisis	
PEC3 - Diseño del motor	Análisis de la arquitectura	
	Diseño de la arquitectura	
	Diseño de los casos de uso	
	Diseño de la persistencia	
	Documento de arquitectura	
	Documento de diseño técnico	
	Diseño de la interfaz gráfica de usuario	
PEC4 - Memoria y presentación-síntesis del proyecto	Depuración entregas parciales	
	Elaboración de la Memoria	
	Elaboración de la presentación visual	

En el siguiente apartado se procede a detallar la estimación temporal del proyecto.

7.1.3. Estimación temporal

Se han estimado los tiempos necesarios para la ejecución de las diferentes tareas que componen el proyecto.

En la Tabla 57 se detalla el nombre de la tarea y su número de orden, la duración en días, así como su fecha de inicio y de finalización previstas. Aparece destacado en negrita, tanto el inicio de las PECS como las fechas de las entregas de éstas salvo la entrega final que aparece en rojo.

Tabla 57. Estimación temporal

	Task Name	Duration	Start	Finish
1	**TFC - Ingenieria del Software**	**108 days?**	**Wed 01/03/06**	**Fri 16/06/06**
2	**PEC1 - Plan de trabajo**	**13 days?**	**Wed 01/03/06**	**Mon 13/03/06**
3	Identificación del TFC	5 days	Wed 01/03/06	Sun 05/03/06
4	Definición de objetivos generales y e	4 days	Mon 06/03/06	Thu 09/03/06
5	Depuración documentos PEC1	2 days	Fri 10/03/06	Sat 11/03/06
6	**Entrega PEC1**	**1 day?**	**Mon 13/03/06**	**Mon 13/03/06**
7	**PEC2 - Captura de requisitos y anális**	**39 days?**	**Tue 14/03/06**	**Fri 21/04/06**
8	Investigación tema TFC	10 days	Tue 14/03/06	Thu 23/03/06
9	Identificación de los subsistemas	3 days	Fri 24/03/06	Sun 26/03/06
10	Especificación de requisitos	6 days	Mon 27/03/06	Sat 01/04/06
11	**Modelo de análisis**	**10 days**	**Sun 02/04/06**	**Tue 11/04/06**
12	Identificación de actores	2 days	Sun 02/04/06	Mon 03/04/06
13	Diagrama de casos de uso	4 days	Tue 04/04/06	Fri 07/04/06
14	Especificación de casos de usc	4 days	Sat 08/04/06	Tue 11/04/06
15	Documento de visión	5 days	Wed 12/04/06	Sun 16/04/06
16	Documento de análisis	7 days	Wed 12/04/06	Tue 18/04/06
17	Depuración documentos PEC2	2 days	Wed 19/04/06	Thu 20/04/06
18	**Entrega PEC2**	**1 day?**	**Fri 21/04/06**	**Fri 21/04/06**
19	**PEC3 - Diseño del motor**	**38 days?**	**Sat 22/04/06**	**Mon 29/05/06**
20	Análisis de la arquitectura	5 days	Sat 22/04/06	Wed 26/04/06
21	Diseño de la arqutectura	7 days	Thu 27/04/06	Wed 03/05/06
22	Diseño de los casos de uso	5 days	Thu 04/05/06	Mon 08/05/06
23	Diseño de la persistencia	4 days	Tue 09/05/06	Fri 12/05/06
24	Documento de arquitectura	3 days	Sat 13/05/06	Mon 15/05/06
25	Documento de diseño técnico	5 days	Tue 16/05/06	Sat 20/05/06
26	Diseño de la interfaz gráfica de usua	5 days	Sun 21/05/06	Thu 25/05/06
27	Depuración documentos PEC3	2 days	Sat 27/05/06	Sun 28/05/06
28	**Entrega PEC3**	**1 day?**	**Mon 29/05/06**	**Mon 29/05/06**
29	**PEC4 - Memoria y presentación-sínte**	**18 days?**	**Tue 30/05/06**	**Fri 16/06/06**
30	Depuración entregas parciales	3 days	Tue 30/05/06	Thu 01/06/06
31	Elaboración de la Memoria	9 days	Fri 02/06/06	Sat 10/06/06
32	Elaboración de la presentación visua	4 days	Sun 11/06/06	Wed 14/06/06
33	Revisión global	1 day?	Thu 15/06/06	Thu 15/06/06
34	**Entrega final del TFC**	**1 day?**	**Fri 16/06/06**	**Fri 16/06/06**

7.1.4. Planificación temporal

En la Figura 171 se presenta la planificación realizada para el desarrollo del presente TFC, la cual se ha plasmado en un gráfico de Gantt realizado con la herramienta MS Project™.

Los cronogramas de barras o gráficos de Gantt fueron concebidos por el ingeniero norteamericano Henry L. Gantt, el cual afrontó el problema de la programación de actividades de manera tal que se pudiese visualizar el periodo de duración de cada actividad, sus fechas de iniciación y terminación, así como el tiempo total requerido para la ejecución de un trabajo.

El instrumento que desarrolló permite también que se siga el curso de cada actividad, al proporcionar información del porcentaje ejecutado de cada una de ellas, así como el grado de adelanto o atraso con respecto al plazo previsto.

ID	Task Name	Duration	Start	Finish
1	**TFC - Ingeniería del Software**	**108 days?**	**Wed 01/03/06**	**Fri 16/06/06**
2	**PEC1 - Plan de trabajo**	**13 days?**	**Wed 01/03/06**	**Mon 13/03/06**
3	Identificación del TFC	5 days	Wed 01/03/06	Sun 05/03/06
4	Definición de objetivos generales y espe	4 days	Mon 06/03/06	Thu 09/03/06
5	Depuración documentos PEC1	2 days	Fri 10/03/06	Sat 11/03/06
6	**Entrega PEC1**	**1 day?**	**Mon 13/03/06**	**Mon 13/03/06**
7	**PEC2 - Captura de requisitos y análisis**	**39 days?**	**Tue 14/03/06**	**Fri 21/04/06**
8	Investigación tema TFC	10 days	Tue 14/03/06	Thu 23/03/06
9	Identificación de los subsistemas	3 days	Fri 24/03/06	Sun 26/03/06
10	Especificación de requisitos	6 days	Mon 27/03/06	Sat 01/04/06
11	**Modelo de análisis**	**10 days**	**Sun 02/04/06**	**Tue 11/04/06**
12	Identificación de actores	2 days	Sun 02/04/06	Mon 03/04/06
13	Diagrama de casos de uso	4 days	Tue 04/04/06	Fri 07/04/06
14	Especificación de casos de uso	4 days	Sat 08/04/06	Tue 11/04/06
15	Documento de visión	5 days	Wed 12/04/06	Sun 16/04/06
16	Documento de análisis	7 days	Wed 12/04/06	Tue 18/04/06
17	Depuración documentos PEC2	2 days	Wed 19/04/06	Thu 20/04/06
18	**Entrega PEC2**	**1 day?**	**Fri 21/04/06**	**Fri 21/04/06**
19	**PEC3 - Diseño del motor**	**38 days?**	**Sat 22/04/06**	**Mon 29/05/06**
20	Análisis de la arquitectura	5 days	Sat 22/04/06	Wed 26/04/06
21	Diseño de la arqutectura	7 days	Thu 27/04/06	Wed 03/05/06
22	Diseño de los casos de uso	5 days	Thu 04/05/06	Mon 08/05/06
23	Diseño de la persistencia	4 days	Tue 09/05/06	Fri 12/05/06
24	Documento de arquitectura	3 days	Sat 13/05/06	Mon 15/05/06
25	Documento de diseño técnico	5 days	Tue 16/05/06	Sat 20/05/06
26	Diseño de la interfaz gráfica de usuario	5 days	Sun 21/05/06	Thu 25/05/06
27	Depuración documentos PEC3	2 days	Sat 27/05/06	Sun 28/05/06
28	**Entrega PEC3**	**1 day?**	**Mon 29/05/06**	**Mon 29/05/06**
29	**PEC4 - Memoria y presentación-síntesis**	**18 days?**	**Tue 30/05/06**	**Fri 16/06/06**
30	Depuración entregas parciales	3 days	Tue 30/05/06	Thu 01/06/06
31	Elaboración de la Memoria	9 days	Fri 02/06/06	Sat 10/06/06
32	Elaboración de la presentación visual	4 days	Sun 11/06/06	Wed 14/06/06
33	Revisión global	1 day?	Thu 15/06/06	Thu 15/06/06
34	**Entrega final del TFC**	**1 day?**	**Fri 16/06/06**	**Fri 16/06/06**

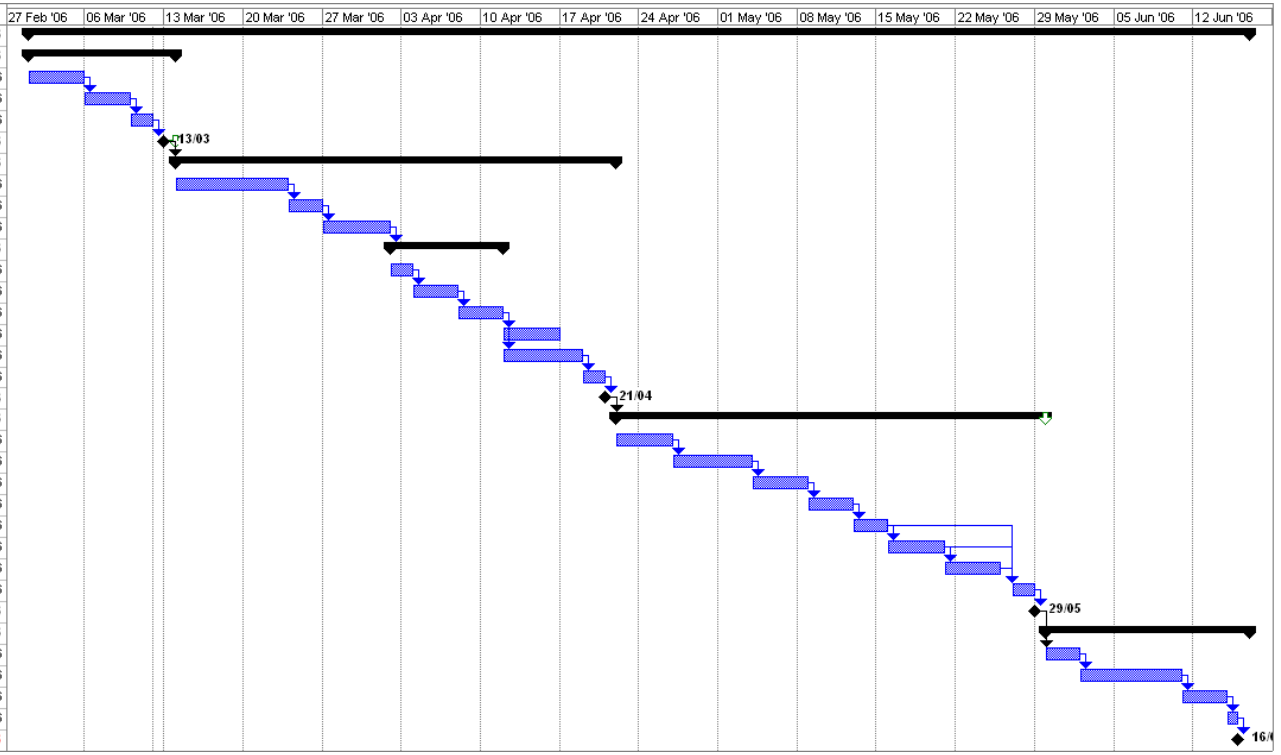

Figura 171. Planificación del TFC

En la Figura 172 se presenta el diagrama de red, que es la versión de los diagramas PERT que ofrece la herramienta con la que se confeccionó el gráfico de Gantt. Los diagramas PERT proporcionan una herramienta para el control del progreso del proyecto. Las actividades de la ruta crítica, permiten por consiguiente, recibir la mayor parte de la atención, debido a que la terminación del proyecto depende fuertemente de ellas. Las actividades no críticas oscilarán en respuesta a la disponibilidad de recursos.

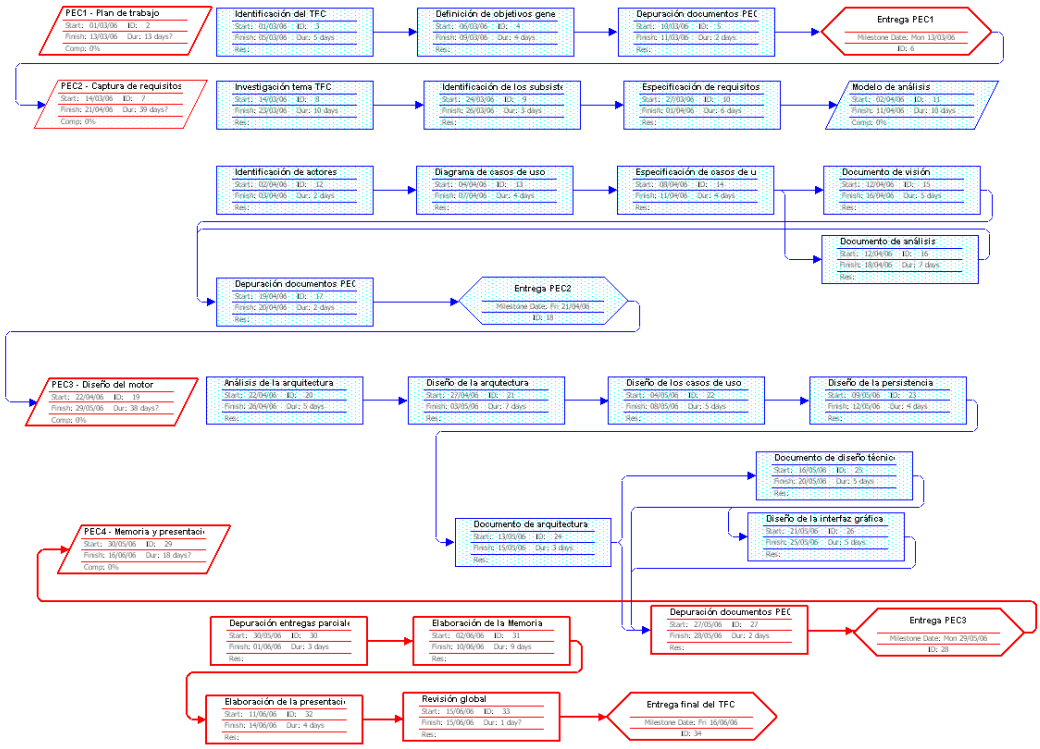

Figura 172. Diagrama de Red

7.2. Anexo II. Medios empleados

Para la realización de este TFC se hará uso de los medios expuestos en la Tabla 58, aportados por el estudiante.

Tabla 58 Medios empleados		
Hardware	Ordenador personal como estación básica de trabajo	
Software	Modelado UML	StarUML 5.0 DIA 0.95
	Otros modelos	OpenOffice Draw 2.0 Netbeans mobility pack
	Diseño GUI	MS Visio 2003 Netbeans 5.0 Netbeans mobility pack
	Planificación	MS Project 2003
	Procesamiento de textos	MS Word 2003 OpenOffice Writer 2.0
Documentación	Toda la documentación empleada se detalla en el epígrafe 5.3. Dicha documentación se compone de libros y artículos en revistas científicas, así como de URLs destacados.	
		

7.3. Anexo III. Glosario de acrónimos

API	Application Programming Interface. Interfaz de programación de aplicaciones.
CDC	Connected Device Configuration. Configuración para dispositivos conectados.
CLDC	Connected Limited Device Configuration. Configuración para dispositivos conectados limitados.
CVM	Compact Virtual Machina. Máquina Virtual Compacta.
ER	Entity - Relationship. Entidad – Relacion.
GPRS	General Packet Radio Service. Servicio General de Radio de Paquetes.
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronical Engineers. Instituto de Ingenieros Eléctricos y Electrónicos.
J2EE	JAVA2 ENTERPRISE EDITION®.
J2ME	JAVA2 MICRO EDITION®.
J2SE	JAVA2 STANDARD EDITION®.
JAD	Joint Application Development. Desarrollo Conjunto de Aplicaciones.
JDBC	Java Database Connectivity. Conectividad a bases de datos de Java.
JVM	Java Virtual Machine. Máquina Virtual de Java.
KVM	Kilobyte Virtual Machine. Máquina Virtual Kilobyte.
OCL	Object Constraint Language. Lenguaje de Restricción de Objeto.
PDA	Personal Digital Assistant. Asistente Personal Digital.
PERT	Program Evaluation and Review Technical. Evaluación de programas y revisión técnica
TFC	Trabajo Fin de Carrera.
UML	Unified Modelling Language. Lenguaje Unificado de Modelado.

UOC Universidad Oberta de Cataluña

WBS Work Breakdown Structure. Descomposición estructural de actividades (literalmente estructura descendente de trabajo).

WYSIWYG What You See Is What You Get. Lo que se ve es lo que se obtiene.