Títol: DESENVOLUPAMENT D'UN FRONT END PER RECORD LINKAGE BASAT EN JAVA

Volum: 1/1

Alumne: ALBERTO LAITA MANE

Director/Ponent: JOSEP LARRIBA I PEY

Departament: ARQUITECTURA DE COMPUTADORS

Data: 26/01/09

Índice

1. Introducción

Este es el documento del proyecto final de carrera de Ingeniería Técnica de Sistemas, cursada en la Facultad de Informática de Barcelona de la Universidad Politecnica de Catalunya.

El proyecto consiste en realizar una interfaz de usuario que permita la configuración de una fusión estadística o *Record Linkage*.

1.1. Contexto inicial

El proyecto surge a partir de la necesidad por parte de algunos clientes de DAMA-UPC de añadir nuevas funcionalidades o modificar las ya existentes al software de fusión estadística Dalink 4.2.

Debido a que añadir nuevas funcionalidades a Dalink 4.2 era muy costoso, ya que el software ha sido desarrollado por muchos programadores de los cuales ya pocos quedan en el grupo, además hay una mezcla de tecnologías en la implementación de la interfaz gráfica (bibliotecas gráficas wxWidgets y MFC), y que apenas existe documentación del desarrollo, se decidió por parte del grupo desarrollar una nueva versión del software en Java, ya que a la vez el grupo DAMA-UPC había tomado la decisión de desarrollar en este lenguaje las aplicaciones futuras. A esta nueva versión de la aplicación de fusión estadística se le pondría el nombre de Daurum.

Por lo tanto Daurum tiene como base Dalink 4.2. Esto quiere decir que debe poder realizar las mismas funcionalidades que Dalink 4.2. y además con una interfaz parecida, sino igual, para que a los usuarios les resulte cómodo adaptarse a la nueva versión. En lo referente a este proyecto estas necesidades influyen en dos puntos. El primero, que la interfaz para realizar la configuración de una fusión estadística tiene como plantilla la interfaz de Dalink 4.2. Y el segundo que se han aprovechado las partes más complejas del código para la configuración y ejecución de una fusión estadística. Este segundo punto esta explicado con más detalle en el capítulo 4, donde se habla del diseño de la aplicación.

1.2. El problema de Record Linkage

En este capitulo se da una visión general del problema que soluciona *Record Linkage*. El motivo por el cual se origina, que solución propone y las diferentes etapas de que consta.

1.2.1. Origen del problema

El problema de *Record Linkage* surge en entornos donde se trabaja con gran cantidad de datos, los cuales contienen errores de muchos tipos y están distribuidos en varias bases de datos donde las entidades no se encuentran identificadas universalmente. Esto impide detectar las distintas apariciones de una entidad en el conjunto de registros. En tales casos no se pueden utilizar los métodos tradicionales de cruce de bases de datos, ya que, como ya hemos dicho no hay ningún valor que los identifique universalmente y los métodos tradicionales de comparación no son útiles ya que los datos pueden contener errores o ser incompletos y deja a estos comparadores exactos en un segundo plano. También hay que tener en cuenta que el volumen de datos que se manejan en estos entornos puede tener un volumen considerable y comparar cada registro con el resto es, en algunos casos, computacionalmente inviable. Por lo tanto surge la necesidad de un método eficaz que permita identificar las apariciones de cada una de las entidades.

1.2.2. Solución al problema

El R*ecord Linkage* consiste en la definición de una estrategia de comparación que permita la identificación de diferentes registros provenientes de diversos ficheros consiguiendo un mínimo coste de ejecución i una máxima precisión de los registros emparejados encontrados.

En esta estrategia de comparación se han sustituido los comparadores exactos por comparadores probabilísticos. De esta forma se consigue solucionar el problema de los errores en los datos, ya que cuando

dos datos no son exactamente iguales se indica en tanto por ciento el grado de similitud entre ellos, ya sea texto, números o fechas.

Entre las posibles estrategias para comparar diferentes registros y así evitar la comparación exhaustiva entre todos los registros., encontramos el *Sliding Window* y el *Blocking* entre otros. El *Sliding Window* se basa en ordenar y deslizar una ventana de un tamaño predeterminado W, por encima de los registros de un fichero, comparando cada uno de ellos con los últimos W registros. El *Blocking* consiste en construir bloques de registros que contienen los mismos valores para un atributo o los mismos valores para un subconjunto de un atributo y comparar cada registro con el resto de registros del bloque al que pertenece.

1.2.3. El proceso de Record Linkage

El proceso de *Record Linkage* consta de varias fases. Estas se realizan de manera secuencial y ordenada y cada una se ocupa de una parte de la solución. A continuación vemos un diagrama con la secuencia de las fases.

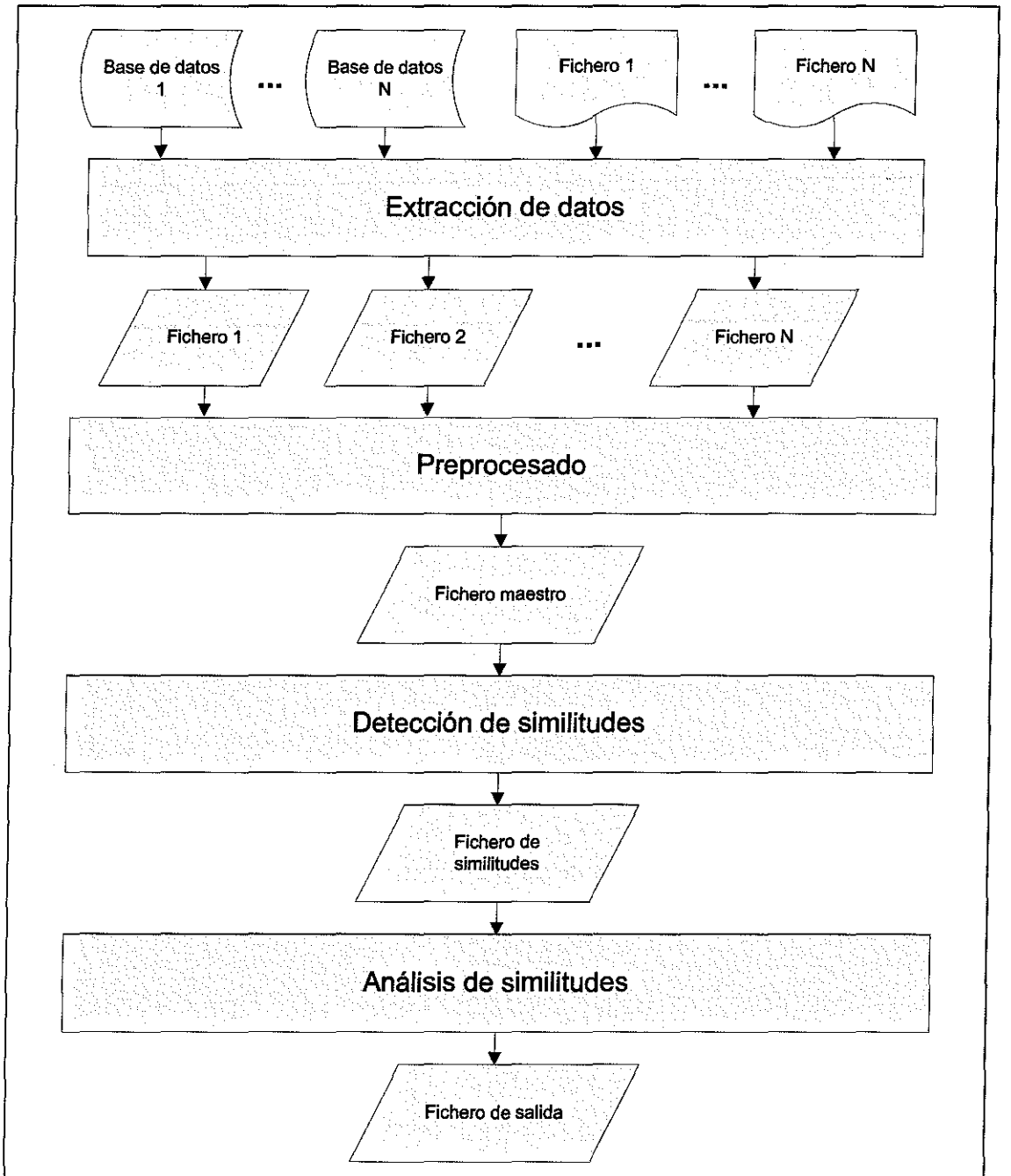

Figura 1: Diagrama de secuencia de procesos de Record Linkage

La primera fase, **extracción de datos**, consiste en extraer los datos de las fuentes de datos, seleccionar los campos relevantes para la comparación y transformarlos a un formato común que se utiliza en el resto del proceso. Una vez tenemos los datos se pasa a la fase de **preproceso**, donde se homogenizan los datos de entrada de forma que maximice la eficacia de su comparación. En esta etapa es donde se eliminan valores que no aportan información y se enriquecen los datos. En la siguiente fase,

detección de similitudes, se compara parejas de registros y se decide, para cada pareja, si estos registros son buenos candidatos a formar parte de una misma entidad. En esta etapa se aplican los métodos que permiten evitar la comparación exhaustiva de registros. En la última fase, **análisis de similitudes,** se decide entre las distintas parejas cuales son potencialmente iguales y por lo tanto pertenecen a una misma entidad.

1.3. El documento

El presente documento esta estructurado de la misma forma que se estructura un proyecto en la metodología establecida en la ingeniería del software. En esta metodología el primer paso es analizar los requisitos que el sistema debe cumplir. A continuación realizar la especificación detallada de las funcionalidades. Y por último su diseño e implementación.

A continuación se proporciona una breve descripción del contenido de cada capitulo:

- En el capítulo 2 se definen formalmente los objetivos del proyecto y sus requerimientos.
- En el capítulo 3 se especifican las funcionalidades.
- En el capítulo 4 se describe la arquitectura del programa y una explicación de las partes más relevantes.
- En el capítulo 5 se explican los puntos más importantes de la implementación.
- En el capítulo 6 se presentan las conclusiones obtenidas en la realización del proyecto, repasando los objetivos conseguidos.
- En el capítulo 7 se comentan las diferentes líneas de trabajo futuro para que la aplicación siga en su desarrollo.
- Finalmente, en el capítulo 8 se indican cuales han sido las referencias consultadas para el proyecto.

2. Definición del proyecto

El principal objetivo de este proyecto es realizar una interfaz de usuario que permita establecer cruces entre uno o varios conjuntos de datos mediante la técnica de fusión estadística de *Record Linkage* de forma sencilla.

Dentro de las diferentes fases del problema de *Record Linkage* (véase figura 1), este proyecto tiene como objetivo dar solución a las de extracción de datos, preproceso y detección de similitudes, que a partir de ahora se hará referencia a ellas como **configuración de *Record Linkage***. La fase en la cual se hace el análisis de las parejas encontradas indicando si realmente son la misma entidad o no ya pertenece a otro módulo de Daurum.

Además de las funcionalidades necesarias para realizar una configuración de *Record Linkage*, se han requerido de las funcionalidades básicas de Daurum con el fin de dar una base sobre la que ejecutar el módulo de configuración de *Record Linkage*, así como englobar a otros módulos en un mismo entorno y la configuración de estos.

En la figura 2 podemos ver un diagrama con los grupos de funcionalidades que se han necesitado para llevar a cabo los objetivos del proyecto.

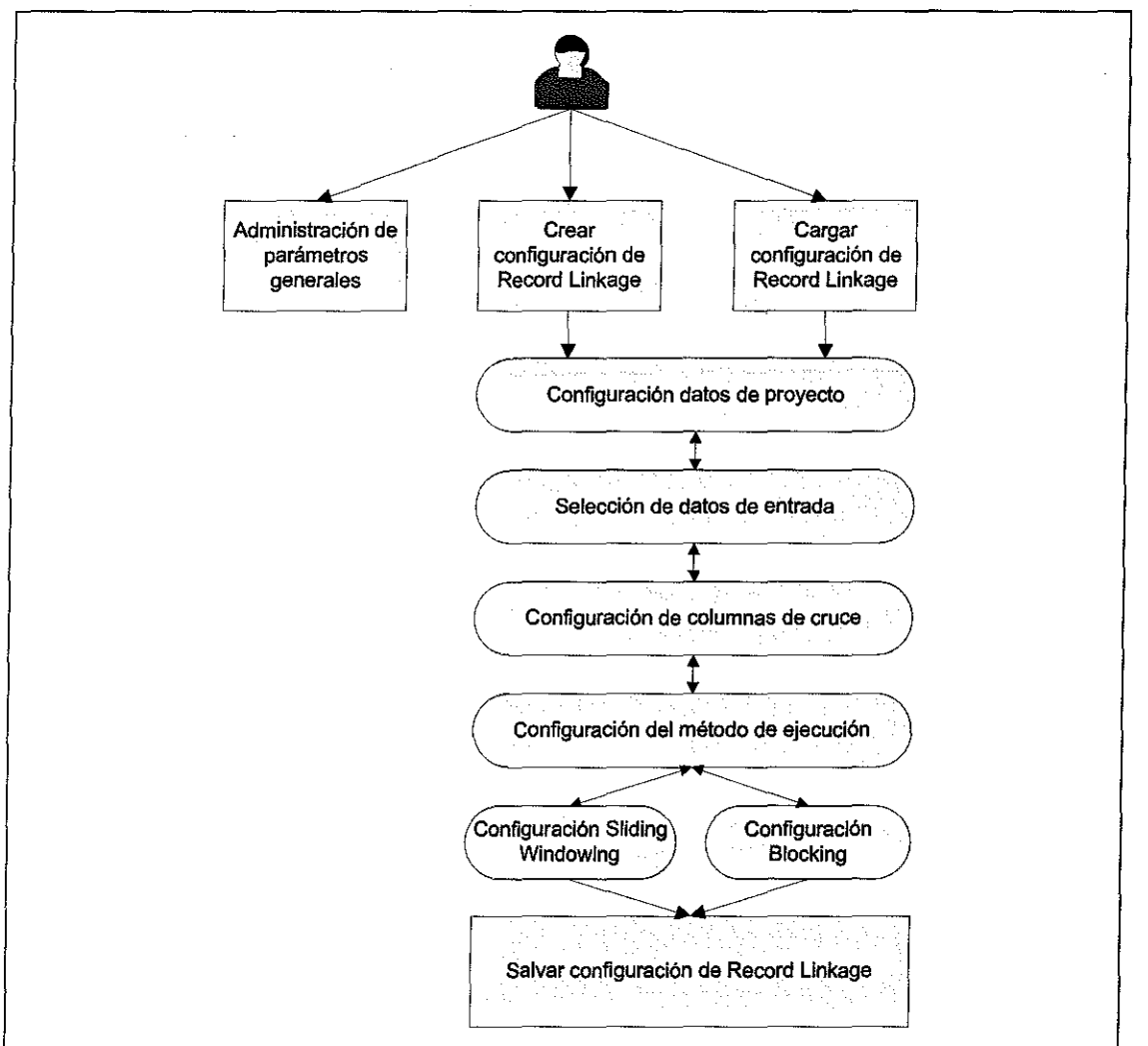

Figura 2: Diagrama de funcionalidades

2.1. Funcionalidades de una configuración de Record Linkage

En este apartado se explican con detalle todas las funcionalidades necesarias para que un usuario, a través de la interfaz, pueda realizar una configuración de *Record Linkage*.

Estas funcionales han sido extraídas de la aplicación Dalink 4.2 ya que se ha visto en la práctica que la interfaz y los parámetros que esta permite configurar son suficientemente intuitivas y flexibles como para que el usuario pueda realizar una configuración de *Record Linkage* los más ajustado posible a sus necesidades.

Estas han sido agrupadas siguiendo la estructura que guardan las pantallas que se le van presentado al usuario. Estas son: configuración de datos de proyecto, selección de los datos de entrada, configuración de las columnas de cruce, configuración del método de ejecución y por último la ejecución o detección de similitudes. Todas ellas son explicadas con detalle en los siguientes apartados.

Por último, se han de poder guardar y recuperar configuración de *Record Linkage*, para una posterior utilización o una modificación de configuraciones ya existentes.

Se han añadido imágenes de las pantallas que el usuario ve a medida que realiza una configuración de *Record Linkage*, con el fin de dar una visión más clara de las funcionalidades.

1. Configuración de datos de proyecto

Estas funcionalidades son necesarias para identificar la configuración a posteriori y para que la aplicación sepa donde a de depositar los resultados.

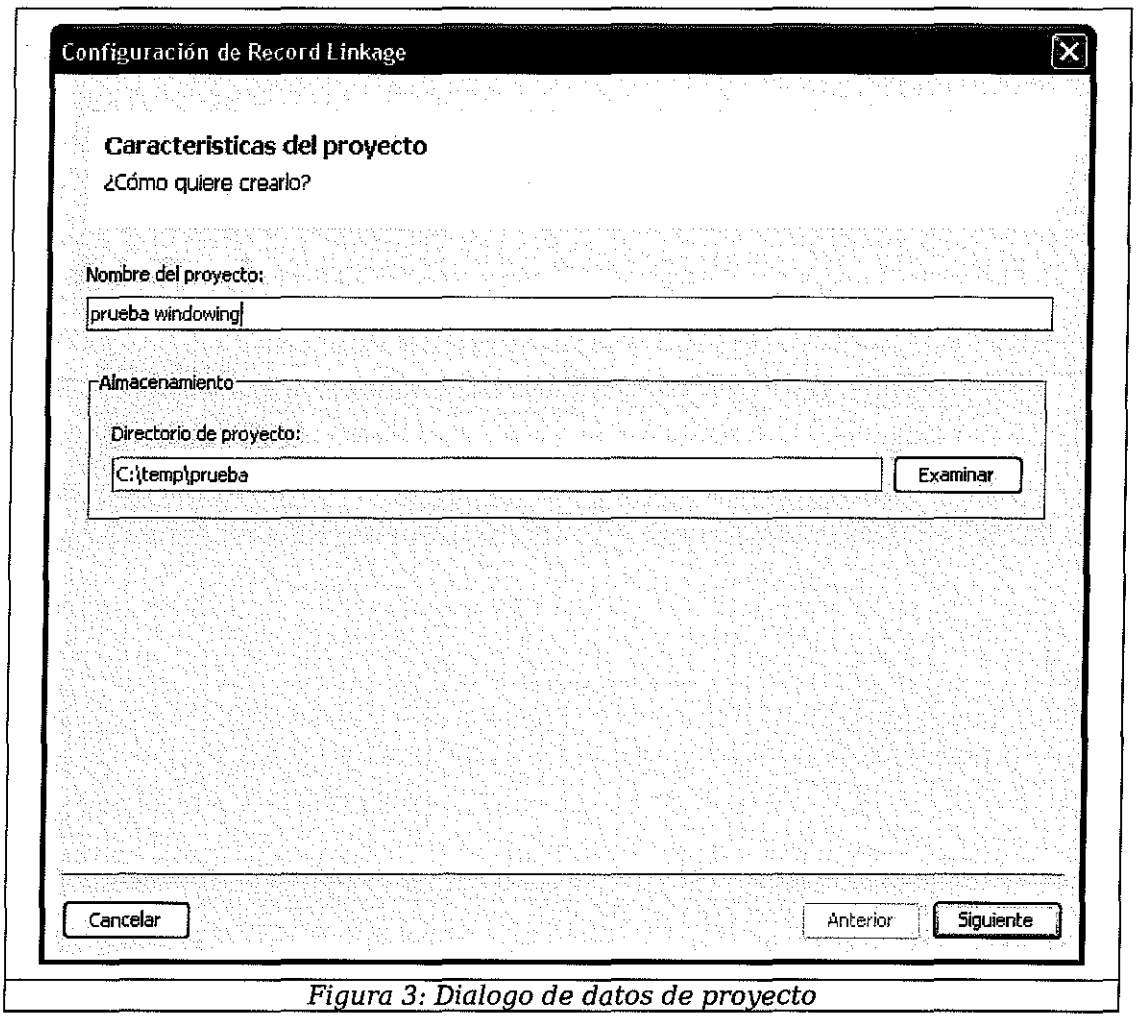

Figura 3: Dialogo de datos de proyecto

Nombre del proyecto: permite al usuario identificar la configuración por el nombre cuando desea recuperarla para su reutilización.

Directorio de proyecto: es el directorio donde se depositaran los resultados, es decir, el fichero con las similitudes. En este directorio también se guardan los ficheros de configuración.

2. Selección de los datos de entrada

Este conjunto de funcionalidades equivale a la fase de extracción de datos en el proceso de *Record Linkage* (véase figura 1). Permiten al usuario indicar que datos van a ser usados para el cruce y configurarlos.

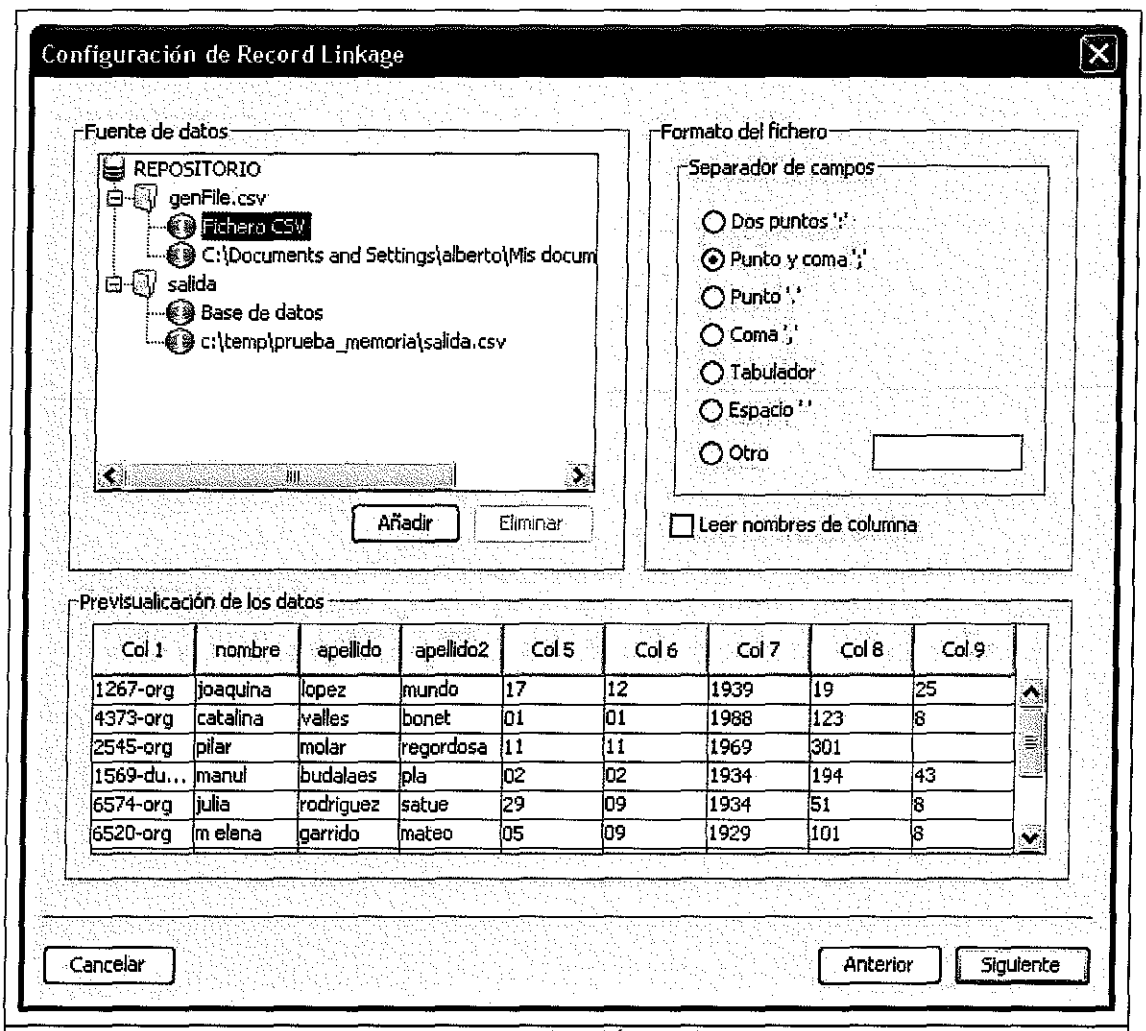

Figura 4: Dialogo de selección de fuente de datos

Los datos pueden provenir de un fichero de texto o de una base de datos. Para cada tipo se requiere de una configuración propia.

Fuente de datos tipo texto: Las configuraciones que se le permiten al usuario para este tipo de fuente de datos son: indicar el delimitador de columna y editar el nombre de las columnas.

- **Indicar el delimitador de columna**: en la figura 4 se puede ver que se sugieren los más comunes, pero también se permite indicar uno diferente. Cada vez que se cambia el delimitador se muestran los resultados en una tabla para que el usuario sepa si es correcta o no la separación de los campos.

- **Editar el nombre de las columnas:** Hay dos formas de editarlas. Si el fichero contiene el nombre de las cabeceras en la primera línea, existe la función para indicar que las recoja. La otra forma es escribirlas manualmente, esto es necesario ya que a la hora de configurar las columnas de cruce es muy útil conocer por un nombre de que columna se trata.

Fuente de datos tipo base de datos: La configuración de una base de datos es un poco más compleja que la de un fichero de texto. Una vez el usuario indica que quiere acceder a una base de datos para indicar que datos se usarán en la fusión, se le muestran todas las conexiones ODBC que existen en el sistema. Una vez ha seleccionado la conexión ODBC que desea y la ha configurado, si lo requiere, se le muestran todas las tablas y vistas de la base de datos (véase figura 5). Es en esta pantalla donde seleccionará las columnas que desee de una tabla o vista. Por último, como el núcleo de la aplicación necesita trabajar con ficheros de texto, se pide al usuario que indique un fichero en el cual se exportaran los datos de la base de datos. Además de esta forma dejamos libre la base de datos y evitamos conflictos con otras aplicaciones que puedan hacer uso.

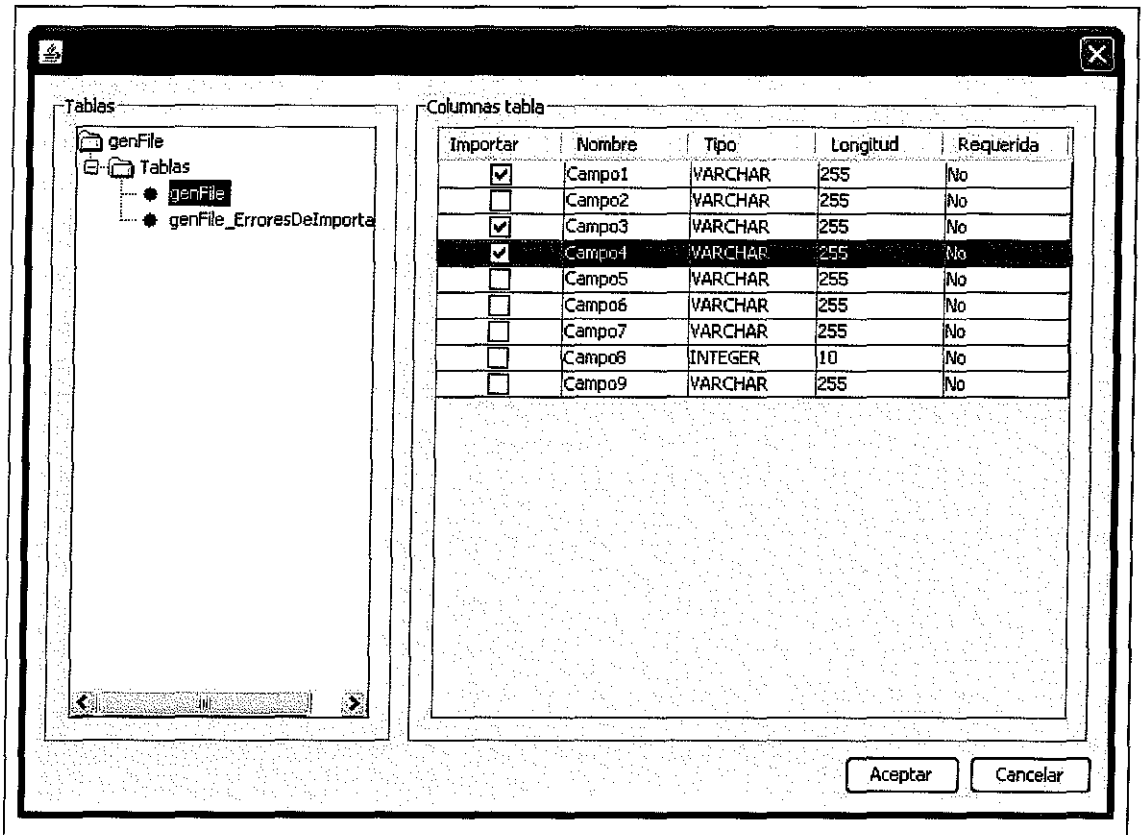

Figura 5: Dialogo de selección de columnas de una base de datos

3. Configuración de las columnas de cruce

Una vez tenemos configurados los datos de entrada con los que vamos a trabajar, el siguiente paso es permitir al usuario indicar que columnas son las que se utilizarán para el cruce de datos y que tipo de homogenización se les realizará. Dados dos registros sólo se compararán las columnas aquí definidas.

También se permite al usuario seleccionar el tipo de comparación, ya que como ya hemos dicho, es una comparación probabilística y existen varios tipos, por ejemplo, para datos alfanuméricos, fechas o números.

Por último, a cada columna de cruce, se le asigna un peso o tanto por ciento que posteriormente será usado para calcular el porcentaje de similitud entre dos registros.

Este conjunto de datos (columnas que se usan para el cruce de datos, tipo de homogenización de cada columna, tipo de comparador y peso) es lo que se conoce como **columna de cruce**.

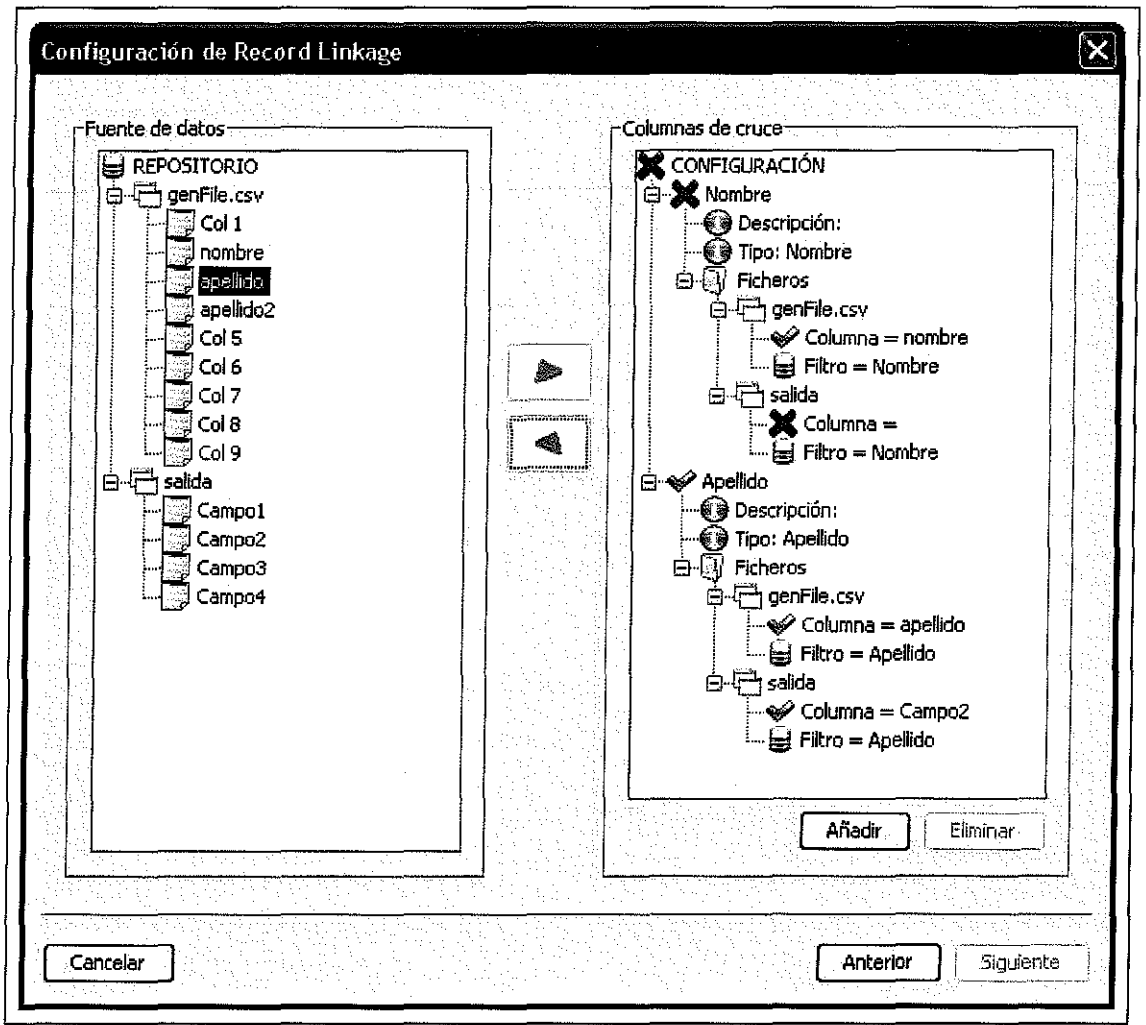

Figura 6: Dialogo de configuración de columnas de cruce

Nombre de columna de cruce: Nombre que se le da a la columna de cruce para identificarla.

Selección de columnas para el cruce: Permite indicar que columnas de las fuentes de datos se usarán para cruzarse entre ellas.

Selección de tipo de comparación: En el momento en que se crea una columna de cruce se indica que tipo de datos tratará, ya que en función de estos se aplicará un tipo concreto de comparación.

Selección de filtro: Esto permite al usuario seleccionar el tipo de homogenización para cada columna de las fuentes de datos usada en alguna columna de cruce. Los filtros se encargan de borrar caracteres extraños, borrar datos que no aportan nada, añadir valores, etc.

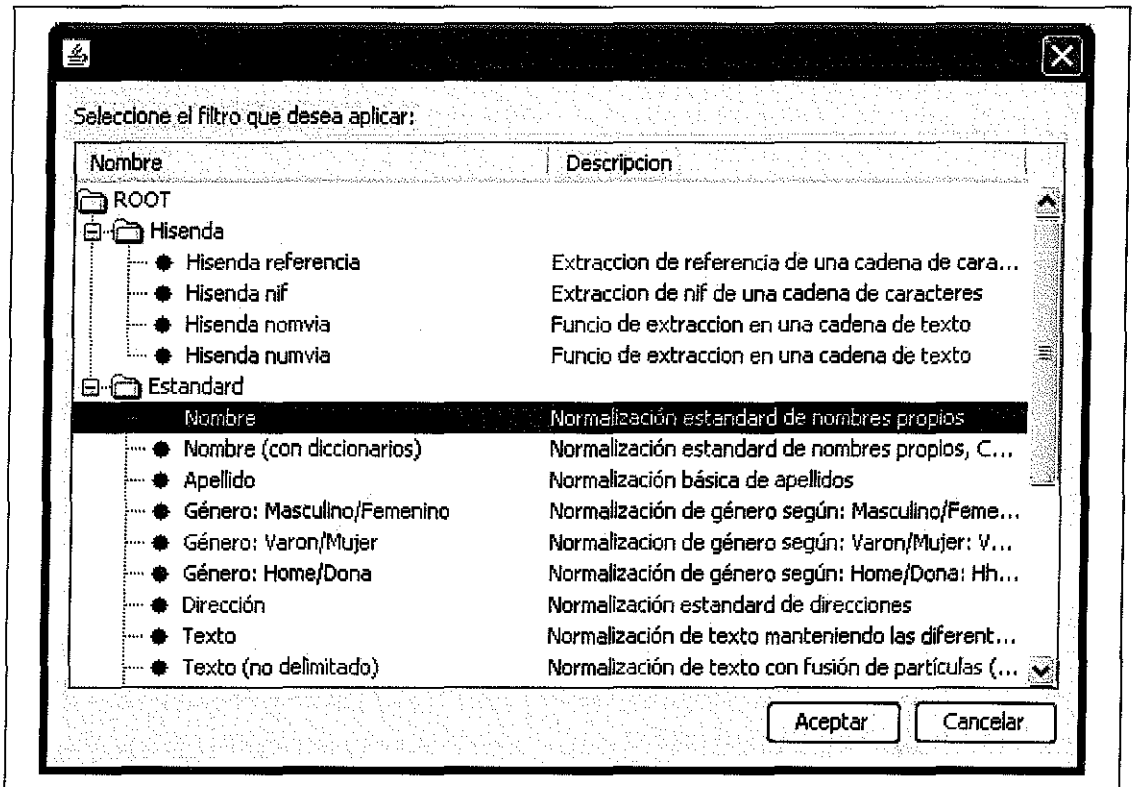

Figura 7:Dialogo de selección de filtro

Configuración de pesos de columnas: Se utiliza para que el usuario pueda indicar que importancia tiene una columna de cruce comparada con el resto de columnas de cruces, ya que una columna de cruce con mucho peso tendrá mayor relevancia en el porcentaje total que indica si dos registros son potencialmente la misma entidad. Es decir, si los datos de una columna de cruce con mucho peso se parecen esto se verá reflejado en que el grado de similitud entre estos dos registros será muy elevado.

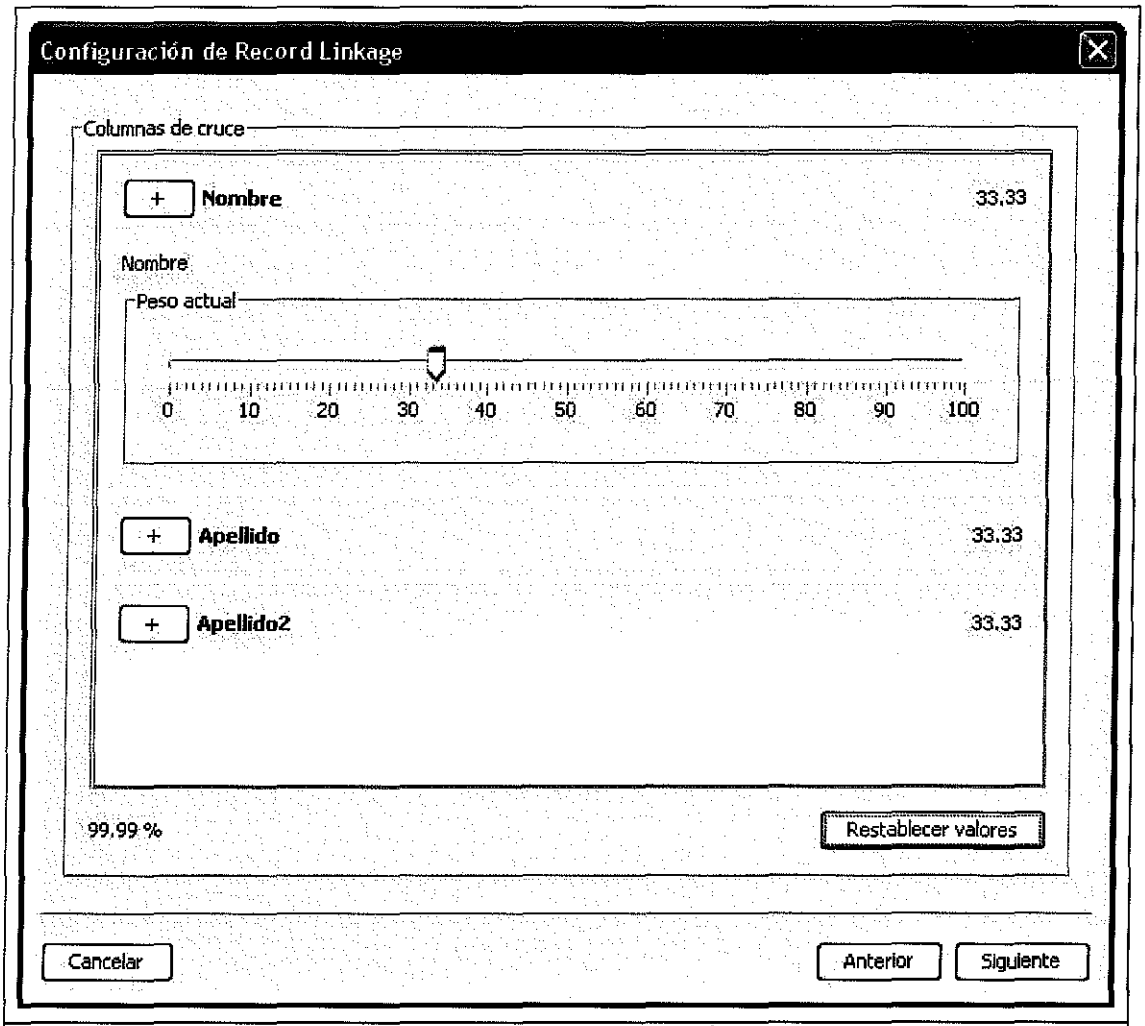

Figura 8: Dialogo de configuración de pesos

4. Selección del método de ejecución

El usuario puede escoger que método se usará para evitar la comparación exhaustiva de cada registro con el resto. En las veces posteriores se nombrará como **método de ejecución** a este tipo de métodos. Los tipos de métodos de ejecución que la aplicación permite escoger son *Sliding Window* y *Blocking*.

A continuación son explicadas con detalle las funcionalidades necesarias para la configuración de cada método de ejecución.

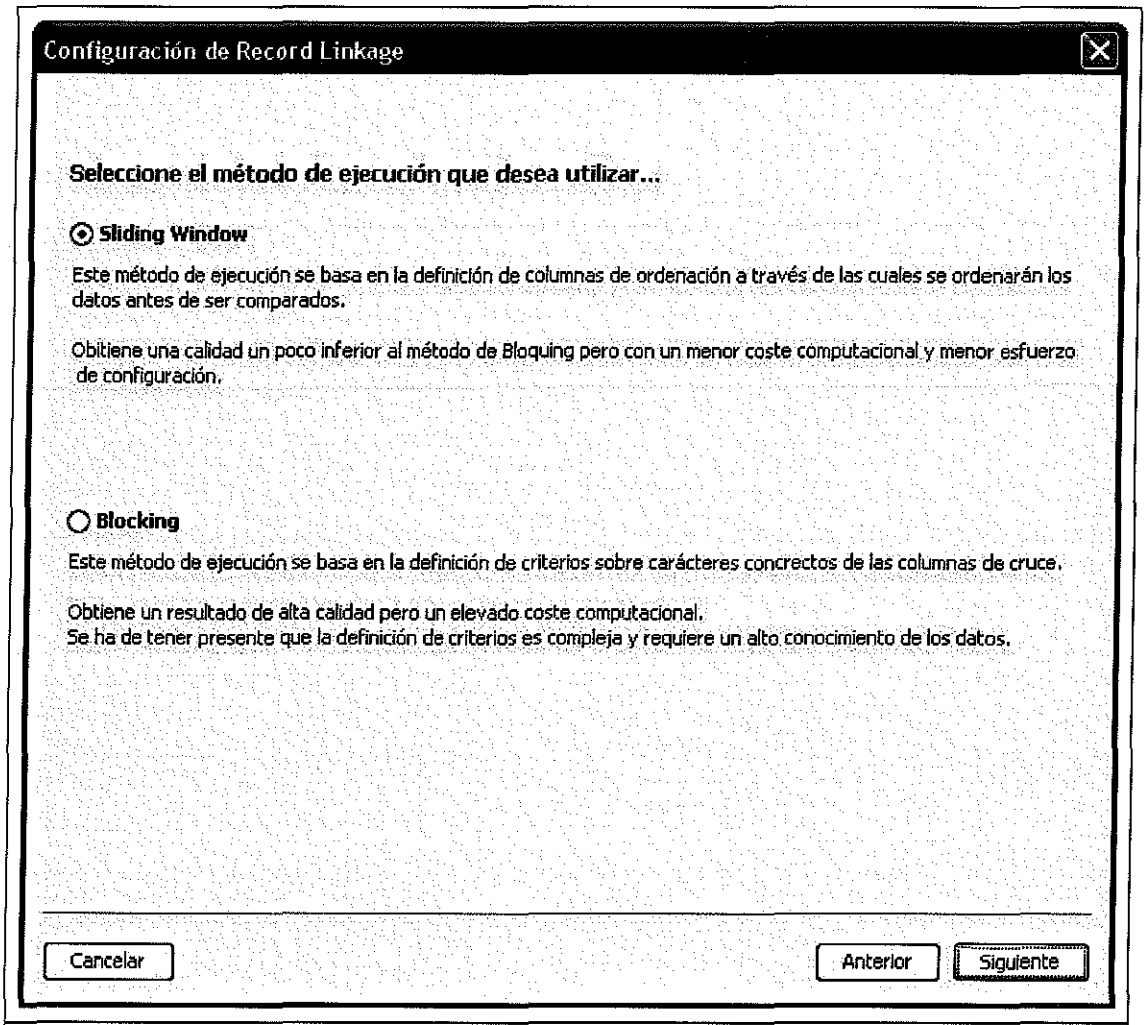

Figura 9: Dialogo de selección de método

5. Configuración Sliding Window

Este método de ejecución consiste en la ordenación de los registros por una columna de cruce y comparar cada registro con los registros posteriores, el número de registros con los que se compara cada registro viene indicado por el tamaño de la ventana. Este proceso se repite tantas veces como columnas de cruce se hayan asignado como columnas de ordenación.

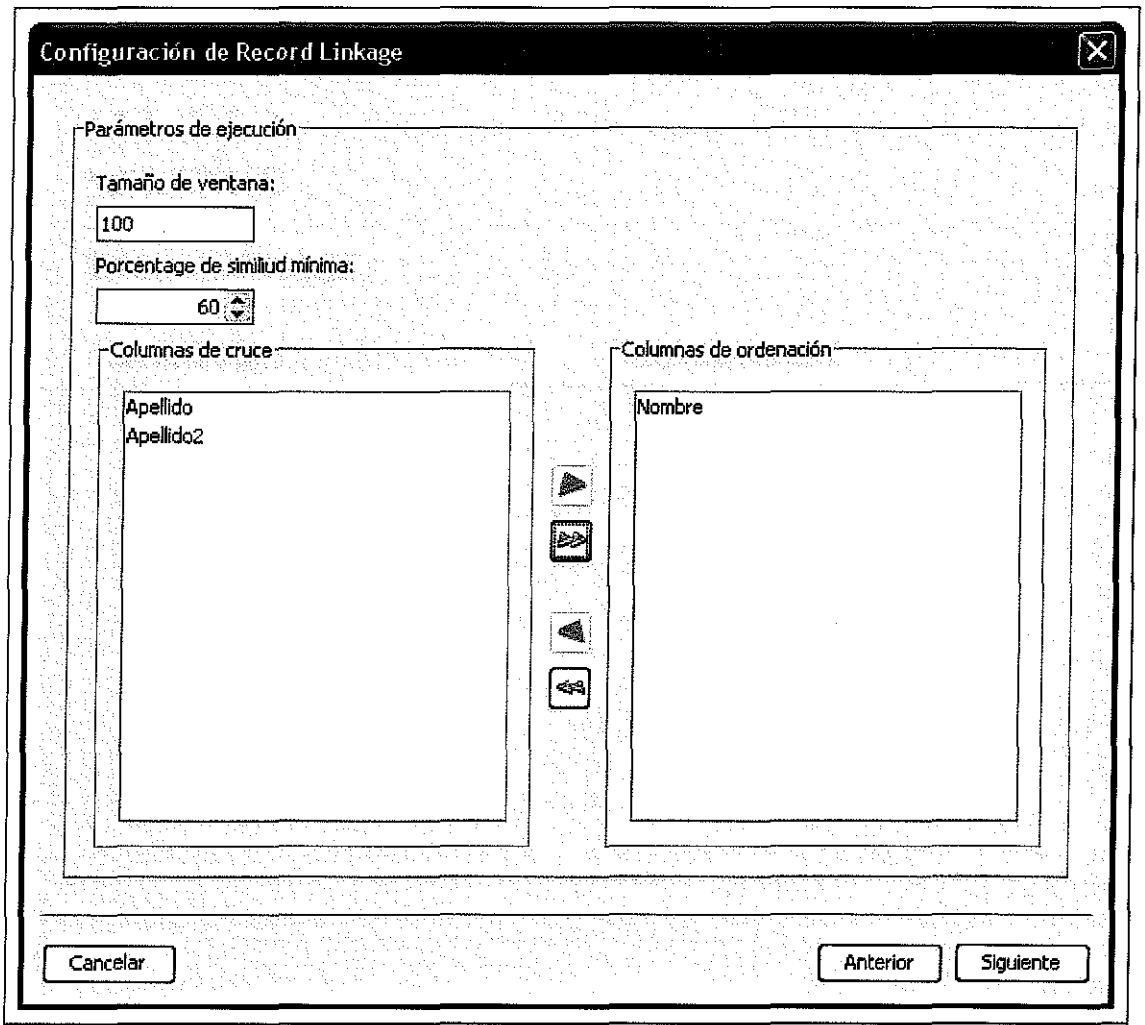

Figura 10: Dialogo de configuración de método de ejecución Sliding Window

Tamaño de ventana: Sirve para indicar el número de registros con los que se comparará cada registro.

Porcentaje de similitud mínima: Indica a partir de que porcentaje se considera que la pareja de registros es potencialmente la misma entidad.

Columnas de ordenación: Indica que columnas de cruce se usarán para ordenar los registros y realizar la comparación.

6. Configuración Blocking

En el método de ejecución *Blocking* la técnica usada para evitar la comparación exhaustiva de cada registro es la de construir bloques de

registros que contienen los mismos valores para un atributo o los mismos valores para un subconjunto de un atributo y comparar cada registro con el resto de registros del bloque al que pertenece.

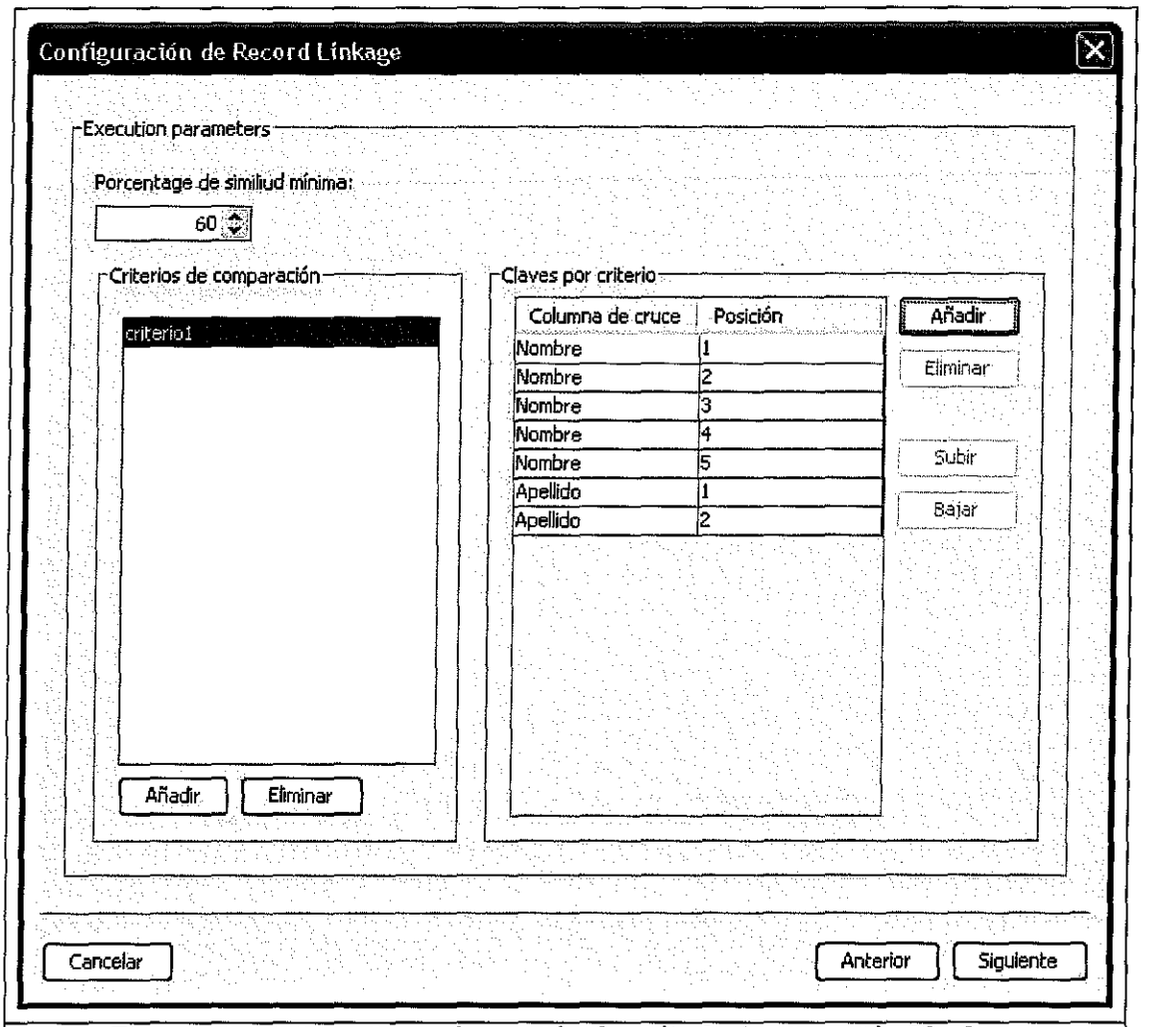

Figura 11: Dialogo de configuración de método de ejecución Blocking

Porcentaje de similitud mínima: Indica a partir de que porcentaje se considera que la pareja de registros es potencialmente la misma entidad.

Criterio de comparación: Cada criterio de comparación hace referencia a un conjunto de bloques. Estos bloques son creados a partir de las claves del criterio.

Claves de criterio: Cada criterio contiene unas claves de criterio, que son las que indican que pauta se usará para crear los bloques.

7. Ejecución de configuración de Record Linkage

Una vez se ha terminado de configurar el método de ejecución ya se ha completado toda la configuración de *Record Linkage,* es entonces cuando pasamos a detectar las posibles similitudes siguiendo los parámetros configurados anteriormente, que para referirnos a esta parte más fácilmente se nombrará como **ejecución de *Record Linkage*.**

Durante la ejecución de Record Linkage se ha añadido la funcionalidad de cancelar ejecución, ya que en algunas ocasiones la ejecución puede tardar más de lo deseado o darnos cuenta de que la configuración de *Record Linkage* esta mal.

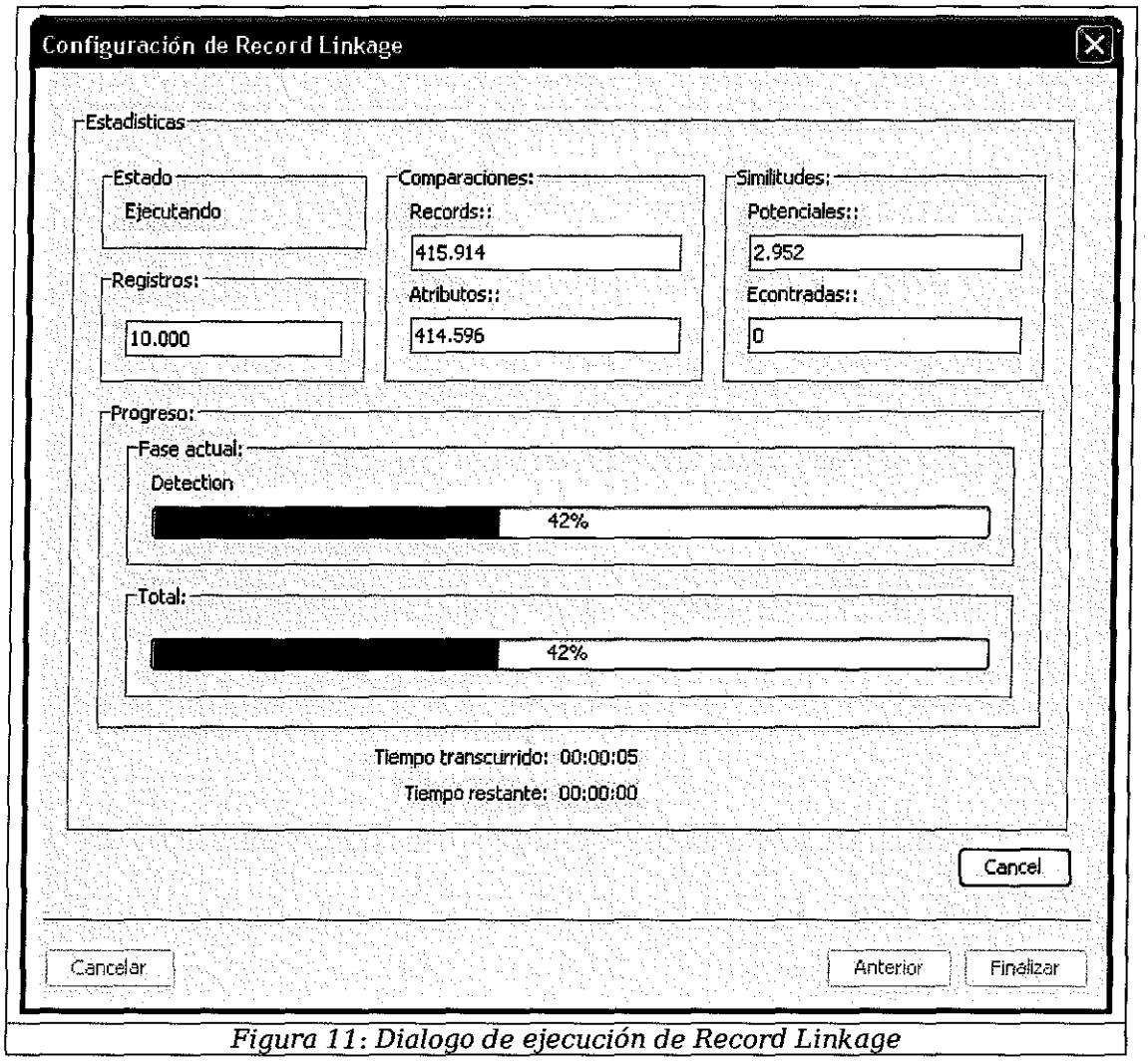

Figura 11: Dialogo de ejecución de Record Linkage

7. Cargar configuraciones de Record Linkage

Esta funcionalidad se utiliza para poder reutilizar configuraciones de *Record Linkage*. A partir de un medio de almacenamiento, en este caso un fichero XML, se recogen todos los parámetros de una configuración de *Record Linkage*.

2.2. Funcionalidades de Daurum

Al tratarse de una aplicación completa también incluye otras características que hacen referencia a la configuración general de la aplicación y es en esta donde se incluye el módulo de configuración de *Record Linkage*. Estas son administrar directorio de filtros y selección de idioma.

Selección de idioma: permite al usuario cambiar el idioma de la aplicación entre el catalán, castellano e ingles.

Administración de directorio de filtros: en esta funcionalidad se permite al usuario añadir o quitar directorios en los cuales se van a buscar ficheros que contienen filtros que posteriormente son cargados para usarlos en la configuración del cruce.

3. Análisis

3.1 Diagrama de casos de uso

El diagrama de casos de uso permite conocer gráficamente los casos de uso, los usuarios del sistema y la forma en la que estos usuarios utilizan el sistema.

Como la aplicación esta destinada a usuarios que no son expertos en la configuración de *Record Linkage*, se ha distinguido solo a un actor, en este caso lo llamamos "Usuario", que puede ser un experto o no.

Para una más fácil clasificación se han agrupado los diferentes casos en grupos y subgrupos. Estos son:

- Administración de parámetros generales
- Configuración de *Record Linkage*
- Configuración de datos de entrada
- Configuración de método de ejecución

3.1.1. Administración de parámetros generales

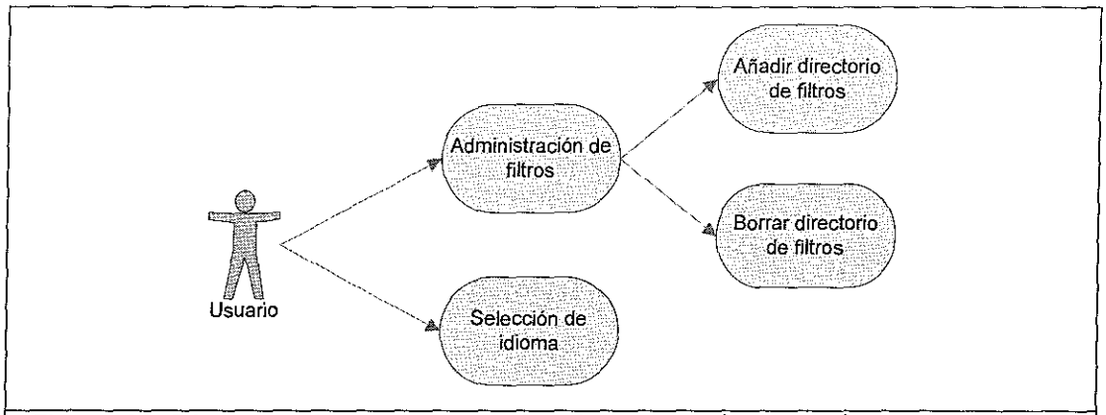

Figura 12: Diagrama de casos de uso de administración de parámetros generales

3.1.2. Configuración de Record Linkage

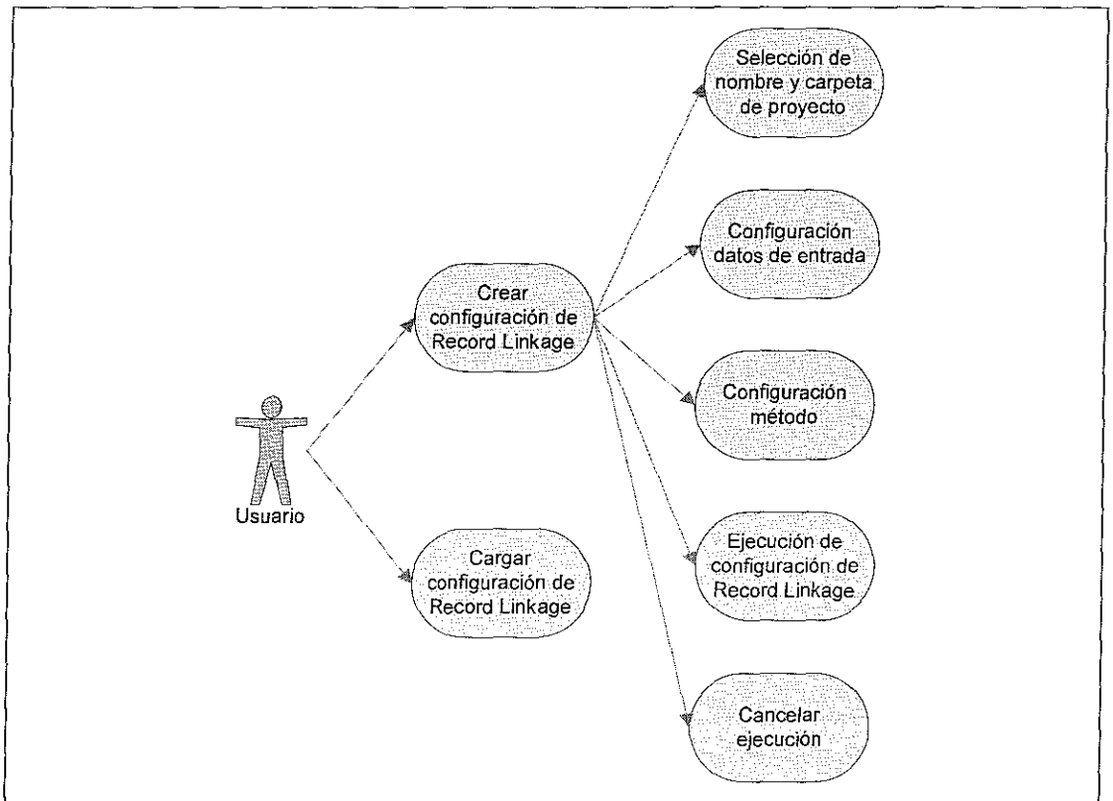

Figura 13: Diagrama de casos de uso de configuración de Record Linkage

3.1.3. Configuración de datos de entrada

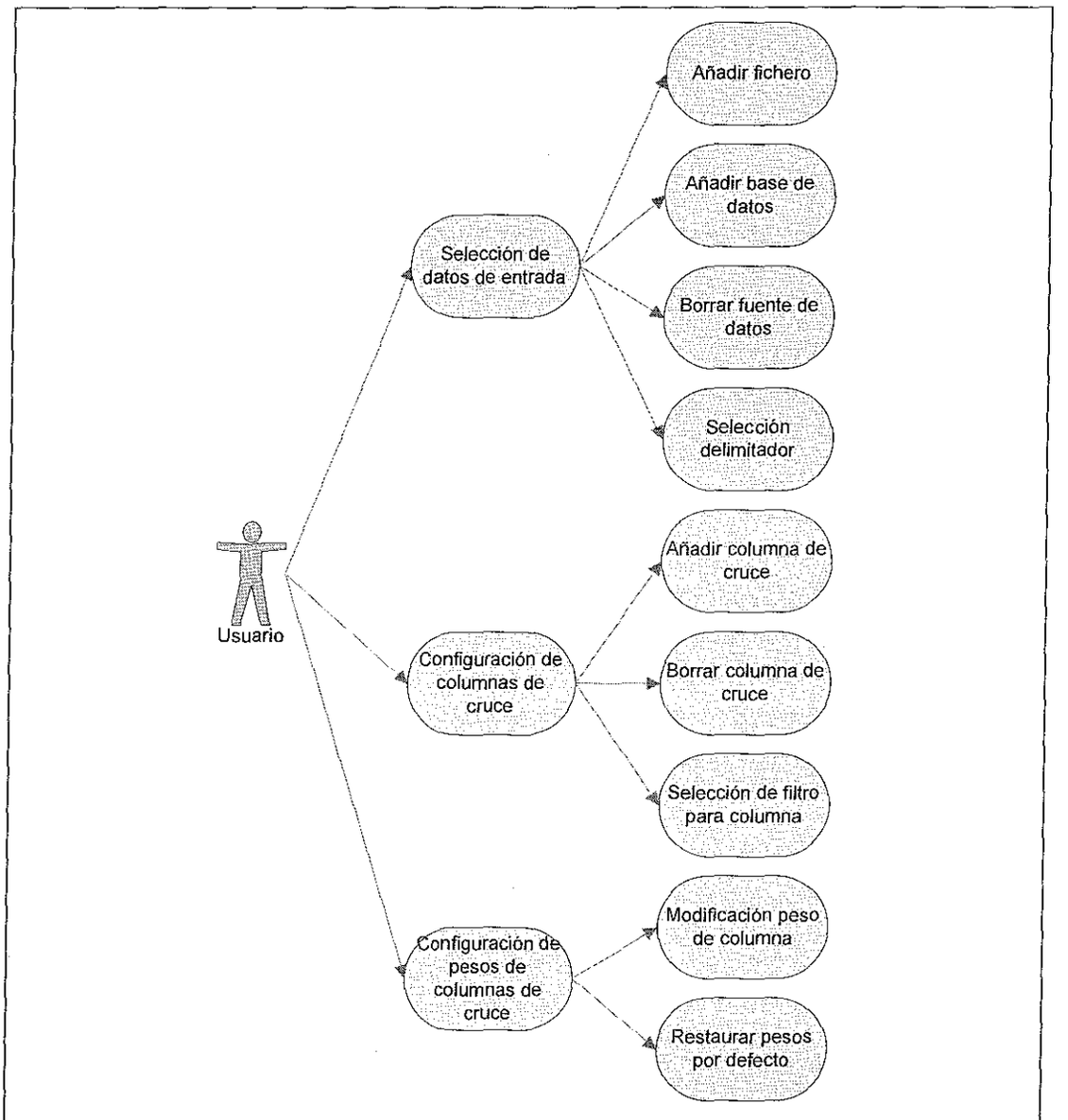

Figura 14: Diagrama de casos de uso de configuración de datos de entrada

3.1.4. Configuración de método de ejecución

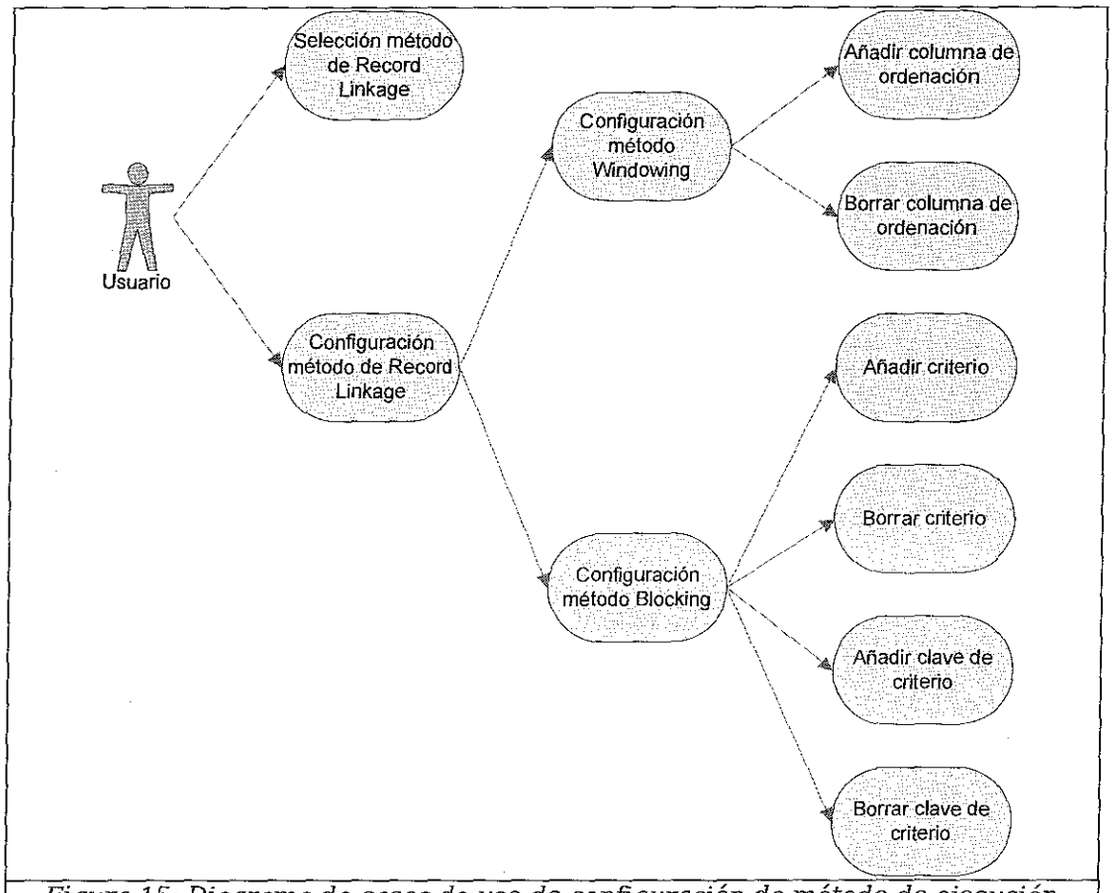

Figura 15: Diagrama de casos de uso de configuración de método de ejecución

3.2. Descripción de los casos de uso

En este apartado se explica con detalle la secuencia de interacción entre los usuarios y el sistema para cada caso de uso.

3.2.1 Administración de parámetros generales

Añadir directorio de filtros

Caso de uso: Añadir directorio de filtros
Actores: Usuario
Resumen: Añadir todos los ficheros de filtros que hay en el directorio
Curso normal de los eventos:

Acción de los actores:	Respuesta del sistema:
1. Solicita dialogo de administración.	
	2. Obtiene la información y la muestra en el diálogo.
3. Añade directorio.	
	4. Guarda la nueva información.

Borrar directorio de filtros

Caso de uso: Borrar directorio de filtros

Actores: Usuario

Resumen: Borrar directorio de la lista de directorios que la aplicación usa para buscar ficheros de filtros.

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Solicita dialogo de administración.	
	2. Obtiene la información y la muestra en el diálogo.
3. Selecciona y borra directorio deseado.	
	4. Guarda la nueva información.

Selección de idioma

Caso de uso: Selección de idioma

Actores: Usuario

Resumen: Cambiar el idioma de la aplicación

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selecciona idioma.	
	2. Aplica cambios y muestra aviso.
3. Reinicia la aplicación.	

3.2.2. Configuración de Record Linkage

Selección de nombre y carpeta de proyecto

Caso de uso: Selección de nombre y carpeta de proyecto

Actores: Usuario

Resumen: Selecciona el nombre y la ubicación de la configuración y de los resultados

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. El usuario introduce el nombre y la ubicación del proyecto	
	2. La aplicación guarda los datos introducidos.

Cursos alternativos:

1. El nombre está vacío. El sistema muestra aviso.

2. La ubicación no existe. El sistema muestra aviso y da opción para crearlo.

Ejecución de configuración de Record Linkage

Caso de uso: Ejecución de configuración de *Record Linkage*

Actores: Usuario

Resumen: Realizar la detección de similitudes con la configuración de *Record Linkage* creada

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Pulsa opción "Ejecutar".	
	2. Guarda configuración del método de Record Linkage.
	3. Ejecuta configuración de Record Linkage y va mostrando el estado en que se encuentra la ejecución.

Cancelar ejecución

Caso de uso: Cancelar ejecución

Actores: Usuario

Resumen: Cancelación de la detección de similitudes antes de que esta haya finalizado

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección opción "Cancelar".	
	2. Cancela los procesos que se están ejecutando.
	3. Actualiza interfaz.

Cargar configuración de Record Linkage

Caso de uso: Cargar configuración de *Record Linkage*

Actores: Usuario

Resumen: Cargar una configuración de *Record Linkage,* para modificarla o ejecutarla.

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección opción "Abrir proyecto".	
	2. Muestra dialogo para seleccionar fichero.
3. Selecciona fichero.	
	4. Muestra dialogo con los datos de la configuración.

Casos alternativos:

1. Si el sistema no puede cargar la configuración del fichero. Muestra un aviso y acaba el caso de uso.

3.2.3. Configuración de datos de entrada

3.2.3.1. Selección de los datos de entrada

Añadir fichero

Caso de uso: Añadir un fichero como fuente de datos
Actores: Usuario
Resumen: Añadir fichero para utilizarlos como datos en la fusión estadística
Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección de opción dato de entrada fichero	
	2. Habilita y deshabilita los elementos gráficos correspondientes.
3. Selecciona fichero	
	4. Guarda los cambios y actualiza la interfaz gráfica.

Añadir base de datos

Caso de uso: Añadir una base de datos como fuente de datos

Actores: Usuario

Resumen: Elegir una base de datos como fuente de datos para la fusión estadística

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección de opción dato de entrada base de datos.	
	2. Habilita y deshabilita los elementos correspondientes de la interfaz.
3. Selecciona la opción de configurar.	
	4. Muestra todas las conexiones ODBC del sistema.
5. Selecciona conexión a base de datos e introduce usuario y contraseña.	
	6. Muestra las tablas y vistas de la base de datos.
7. Selecciona columnas de una tabla o vista.	
	8. Muestra pantalla de selección de fichero donde se extraerán los datos.
9. Selecciona fichero.	
	10. Guarda los cambios y actualiza la interfaz gráfica.

Casos alternativos:

1. No se ha seleccionado ninguna columna. El sistema muestra un aviso.

2. El fichero seleccionado existe. En este caso el sistema muestra un mensaje dando la opción de sobrescribirlo o seleccionar otro fichero.

Borrar fuente de datos

Caso de uso: Borrar fuente de datos

Actores: Usuario

Resumen: Borrar una fuente de datos que ya existe en la configuración.

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección de la fuente de datos que se desea borrar.	
	2. Borra la fuente de datos y guarda los cambios.
	3. Actualiza la interfaz gráfica.

Casos alternativos:

1. Si existe una configuración sobre los pesos o el método de ejecución el sistema avisa que se borrarán estas configuraciones y pide confirmación.

Selección de delimitador

Caso de uso: Selección de delimitador para fuente de datos tipo fichero

Actores: Usuario

Resumen: Cambia el delimitador de columna del fichero

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selecciona fuente de datos tipo fichero.	
	2. Habilita interfaz referente a los delimitadores.
3. Selecciona delimitador.	
	4. Guarda cambios y los muestra en la interfaz gráfica.

3.2.3.2. Configuración de las columnas de cruce

Añadir columna de cruce

Caso de uso: Añadir columna de cruce

Actores: Usuario

Resumen: Añadir nueva columna de cruce a la configuración de la fusión estadística.

Curso normal de los eventos:

Acciones de los actores:	**Respuesta del sistema:**
1. Selección opción "Añadir".	
	2. Muestra dialogo con los parámetros de una columna de cruce
3. Introduce los parámetros de la columna (nombre, tipo y descripción).	
	6. Almacena la configuración de la columna de cruce y actualiza la interfaz gráfica.

Casos alternativos:

1. El nombre de la columna está vacío. El sistema muestra un aviso.

2. El nombre de la columna de cruce ya existe en la configuración. El sistema muestra aviso.

Borrar columna de cruce

Caso de uso: Borrar columna de cruce

Actores: Usuario

Resumen: Eliminar de la configuración una columna de cruce que ya existe.

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selecciona columna de cruce.	
	2. Habilita el botón "Borrar".
3. Pulsa el botón.	
	4. Elimina de la configuración la columna de cruce y actualiza la interfaz gráfica.

Casos alternativos:

1. Si existe configuración sobre los pesos de la columna o el método de ejecución, el sistema avisa que se borrarán estas configuraciones y pide confirmación.

Selección de filtro para columna

Caso de uso: Selección de filtro para columna

Actores: Usuario

Resumen: Cambiar el filtro de la columna de cruce

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Realiza la acción para cambiar el filtro de la columna de cruce (doble-clic).	
	2. Muestra dialogo con todos los filtros, seleccionado el filtro que tiene asignado la columna
3. Selecciona el nuevo filtro y pulsa aceptar.	
	4. Guarda el nuevo filtro para la columna de cruce y muestra los cambios en la interfaz gráfica.

Modificación peso de columna

Caso de uso: Modificación de peso de una columna de cruce

Actores: Usuario

Resumen: Cambiar el peso de una columna de cruce

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Modifica el peso de una columna de cruce	
	2. Valida el nuevo peso, devuelve el peso introducido o si es mayor que el máximo valor aplicable devuelve el máximo valor. **3.** Muestra nuevo peso de la columna y habilita o deshabilita la opción para pasar al siguiente paso.

Restaurar pesos por defecto

Caso de uso: Restaurar pesos por defecto

Actores: Usuario

Resumen: Cambia los pesos de las columnas de cruce por los que se calculan automáticamente por la aplicación

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Pulsa botón "Restablecer valores".	
	2. Calcula los pesos de nuevo, los almacena y actualiza la interfaz gráfica.

3.2.4. Configuración método de ejecución

Selección método de Record Linkage

Caso de uso: Selección método de *Record Linkage*

Actores: Usuario

Resumen: Selección del método de ejecución que se quiere usar para hacer la fusión estadística

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selecciona método de ejecución.	
	2. Crea estructura y valores por defecto para el método seleccionado.
	3. Muestra interfaz gráfica para configurar el método seleccionado.

3.2.4.1. Configuración método Sliding Window

Añadir columna de ordenación

Caso de uso: Añadir columna de ordenación

Actores: Usuario

Resumen: Añadir columna de cruce para que sea usada como columna de ordenación en la fusión estadística

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selecciona columna de cruce.	
	2. Habilita opción de añadir columna de cruce.
3. Pulsa botón añadir.	
	4. Guarda la configuración y actualiza la interfaz.

Borrar columna de ordenación

Caso de uso: Borrar columna de ordenación

Actores: Usuario

Resumen: Quita de las columnas de ordenación la columna de cruce seleccionada

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selecciona columna de ordenación.	
	2. Habilita opción de quitar columna de ordenación.
3. Pulsa botón quitar columna de ordenación.	
	4. Guarda la nueva configuración y actualiza la interfaz.

3.2.4.2. Configuración método Blocking

Añadir criterio

Caso de uso: Añadir criterio

Actores: Usuario

Resumen: Añadir criterio para la configuración *Blocking*

Curso normal de los eventos:

Acciones de los actores:	**Respuesta del sistema:**
1. Solicita añadir nuevo criterio.	
	2. Muestra el diálogo.
3. Introduce nombre criterio.	
	4. Guarda criterio y actualiza interfaz gráfica.

Casos alternativos:

1. El nombre del criterio ya existe en la configuración. Muestra un aviso indicando que no puede haber dos criterios con el mismo nombre.

Borrar criterio

Caso de uso: Borrar criterio

Actores: Usuario

Resumen: Borrar criterio existente en la configuración *Blocking*

Curso normal de los eventos:

Acciones de los actores:	**Respuesta del sistema:**
1. Selecciona criterio.	
	2. Habilita botón "Borrar".
3. Pulsa botón.	
	4. Borra criterio de la configuración y actualiza interfaz gráfica.

Añadir clave de criterio

Caso de uso: Añadir clave de criterio

Actores: Usuario

Resumen: Añadir clave para un criterio

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección de criterio.	
	2. Habilita opción "Añadir".
3. Pulsa opción "Añadir".	
	4. Muestra dialogo.
5. Introduce valores.	
	6. Almacena valores y actualiza interfaz gráfica.

Casos alternativos:

1. Los valores introducidos de las claves no cumplen el formato que se pide. Muestra un mensaje avisando que el valor introducido no es correcto.

Borrar clave de criterio

Caso de uso: Borrar clave de criterio

Actores: Usuario

Resumen: Borrar clave de un criterio

Curso normal de los eventos:

Acciones de los actores:	Respuesta del sistema:
1. Selección de criterio.	
	2. Muestra claves del criterio y habilita opción "Borrar".
3. Selecciona clave y pulsa botón "Borrar".	
	4. Guarda cambios y actualiza interfaz gráfica.

4. Diseño

En este capitulo se analiza en detalle cada caso de uso para determinar como se realiza internamente en el sistema. Es decir para cada una de las interacciones entre el usuario y el sistema se definen los objetos que recogen el mensaje del actor, lo procesan y responden, realizando así la interacción descrita en el escenario del caso de uso.

Para realizar el diseño se ha usado un modelo de arquitectura por capas. Esta consiste en la diferenciación de una aplicación en un conjunto de capas independientes y jerárquicamente ordenadas. Este modelo de arquitectura está explicada con más detalle en el siguiente apartado.

También se da una visión general del núcleo de Daurum y de cómo se ha integrado este en la aplicación, ya que en una configuración de *Record Linkage* se está constantemente accediendo a este.

Por último explicamos cada objeto que hemos necesitado para el diseño de los casos de uso y como interaccionan entre ellos.

4.1. Modelo de arquitectura

El modelo de arquitectura que hemos usado es el de tres capas. Esta se basa en la división en el nivel de presentación o aplicación, nivel de lógica de negocio, y nivel de acceso a datos.

En la capa de presentación se obtienen las diferentes peticiones de los usuarios, se ordenan la ejecución de las acciones y se comunica el resultado de las acciones a los usuarios. En caso de que sea necesario, la capa de presentación hace las comprobaciones de los datos de entrada con el fin de simplificar el proceso. La capa de dominio o lógica recoge las acciones, controla su validez y ejecuta las acciones demandadas, obtiene los resultados y comunica las respuestas. Por último, en la capa de almacenamiento o acceso a datos, se especifica el método de persistencia de los datos.

Este tipo de arquitectura permite aislar cualquier capa de la anterior mediante interfaces bien definidas. De esta manera podríamos desacoplar la parte de presentación de la aplicación y sustituirla por otra. Esto es posible gracias a que las capas tienen el mínimo acoplamientos entre ellas, siendo la interfaz de las capas el único punto de unión.

A continuación, en la figura 16, se muestra visualmente un diagrama explicando el tipo de arquitectura.

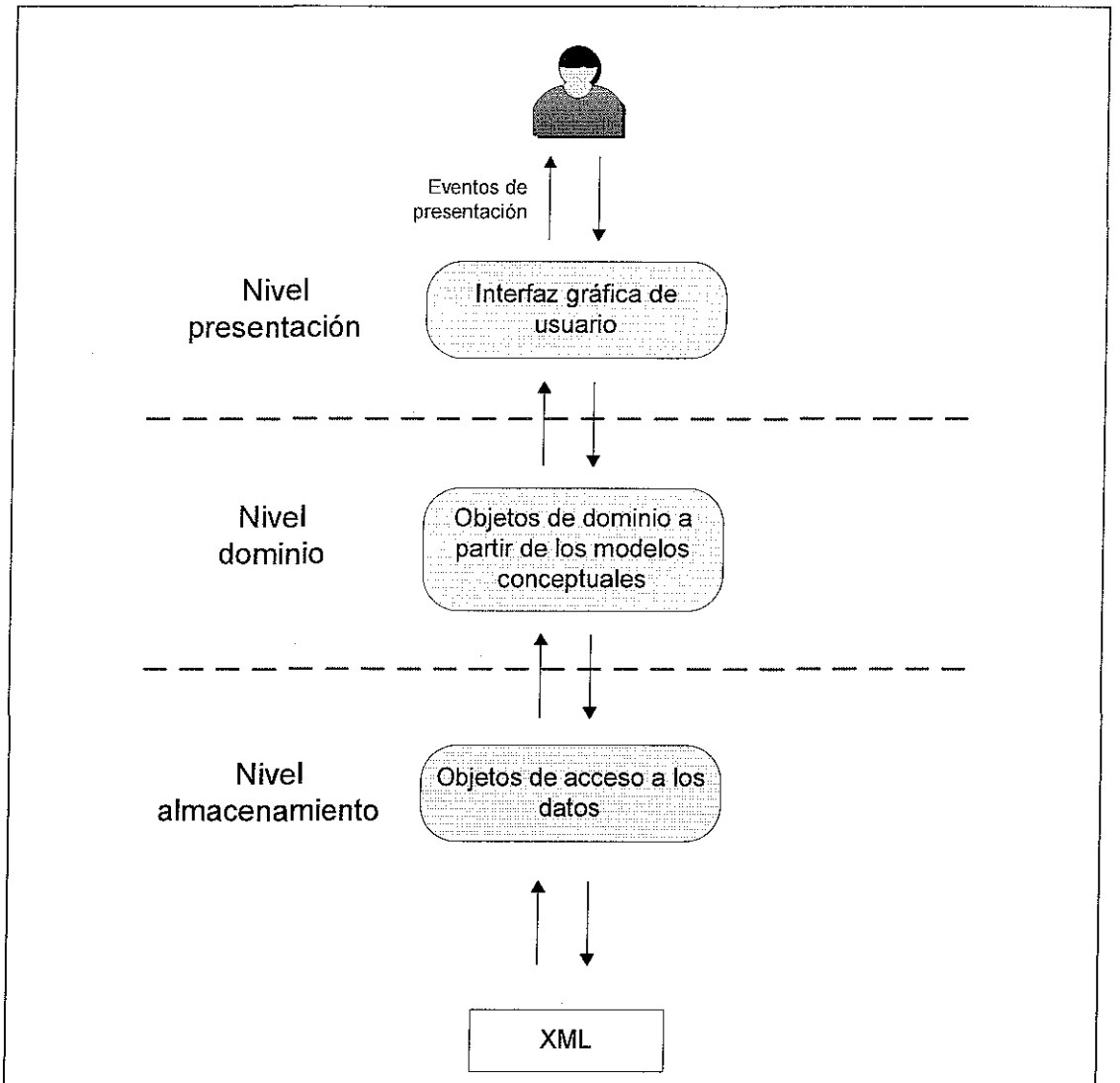

Figura 16: Arquitectura de 3 capas

4.2. Arquitectura global

Para un correcto entendimiento de la aplicación, en este apartado se da una visión general de la arquitectura y las partes que la forman.

En la figura 17 se puede ver los diferentes módulos de que consta la aplicación de Daurum. Este proyecto abarca los módulos de **interfaz principal**, **asistente de configuración de *Record Linkage*** y **administración de parámetros generales**. También se pueden ver los módulos de **rejilla de revisión** y **núcleo Daurum**. En el caso de rejilla de revisión, aunque no forma parte de este proyecto, se ha añadido al diagrama ya que es una parte fundamental en el proceso del problema de *Record* Linkage, equivale a la fase de análisis de similitudes. El módulo núcleo de Daurum es el que se encarga de almacenar y ejecutar una configuración de *Record Linkage.* Debido a la importancia de este último y a su constante uso en una configuración de Record Linkage se da una visión general en el siguiente apartado.

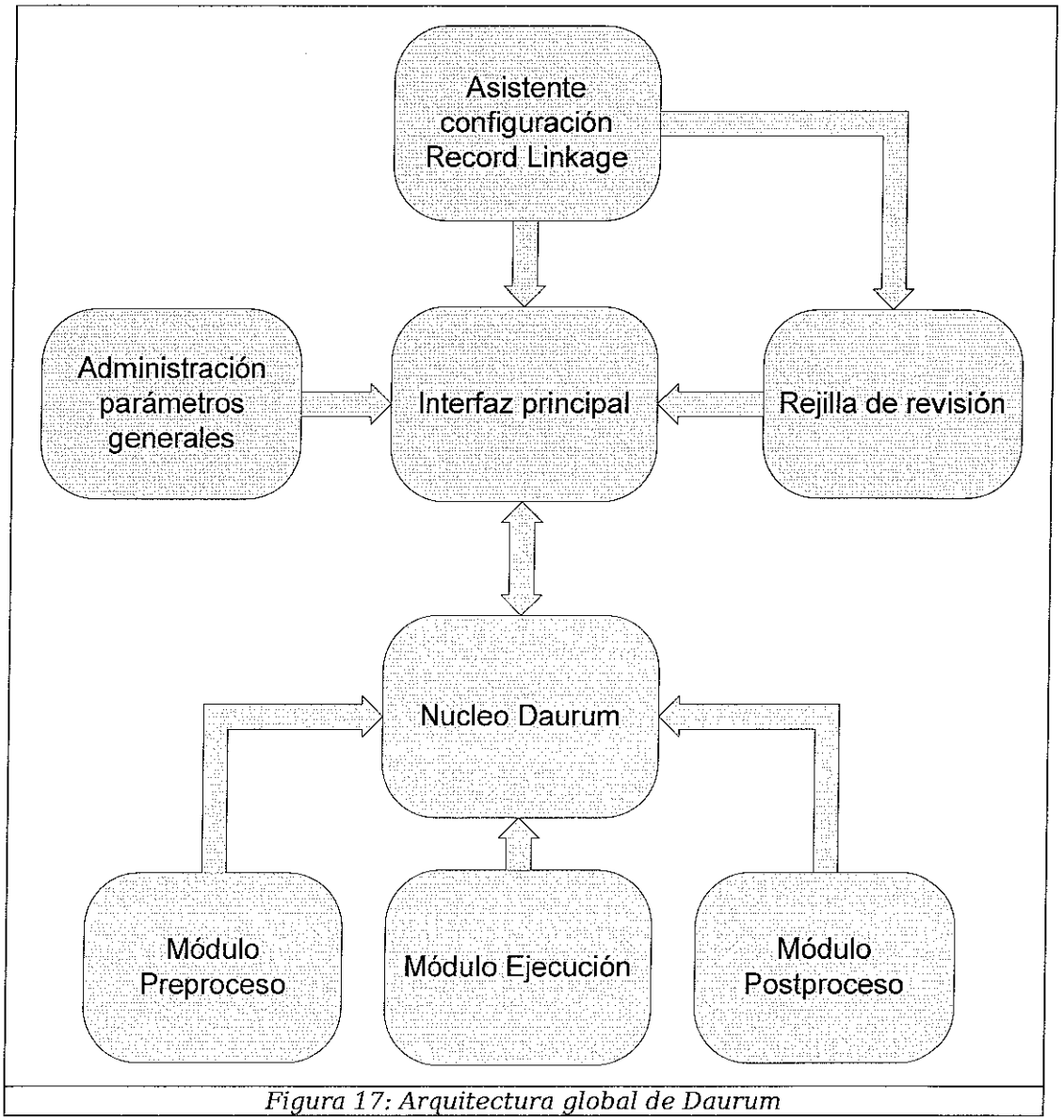

Figura 17: Arquitectura global de Daurum

4.3. Núcleo de Daurum

El módulo núcleo de Daurum es el conjunto de estructuras básicas y operaciones necesarias para el almacenamiento y ejecución de una configuración de Record Linkage. Como podemos ver en la figura 17, este, a su vez, esta dividido en tres módulos, donde cada uno se encarga de una parte de la configuración de Record Linkage. Estos módulos están explicados con más detalle en los siguientes subapartados.

El núcleo de Daurum es la parte de código que se ha reutilizado de la aplicación Dalink 4.2. La aplicación Dalink4.2. está implementada con el

lenguaje de programación C++. Debido a que no es posible comunicar directamente código C++ con código Java, se ha utilizado la tecnología JNI (*Java Native Inteface*) que permite llamar desde Java a código implementado en otros lenguajes. Por lo tanto el núcleo de Daurum esta compuesto por librerías con el código C++ ya compilado que contienen las estructuras y operaciones de una configuración de *Record Linkage* y una interfaz en Java que comunica estas estructuras y operaciones para que al programador le sea transparente esta comunicación. Podemos ver como queda esta estructura en la figura 18.

Por último, para que la aplicación o la capa de lógica no tenga que diferenciar si se trata de la parte de preproceso, de ejecución o de postproceso, se ha integrado las tres partes en una, así mantenemos una configuración de Record Linkage en una sola estructura.

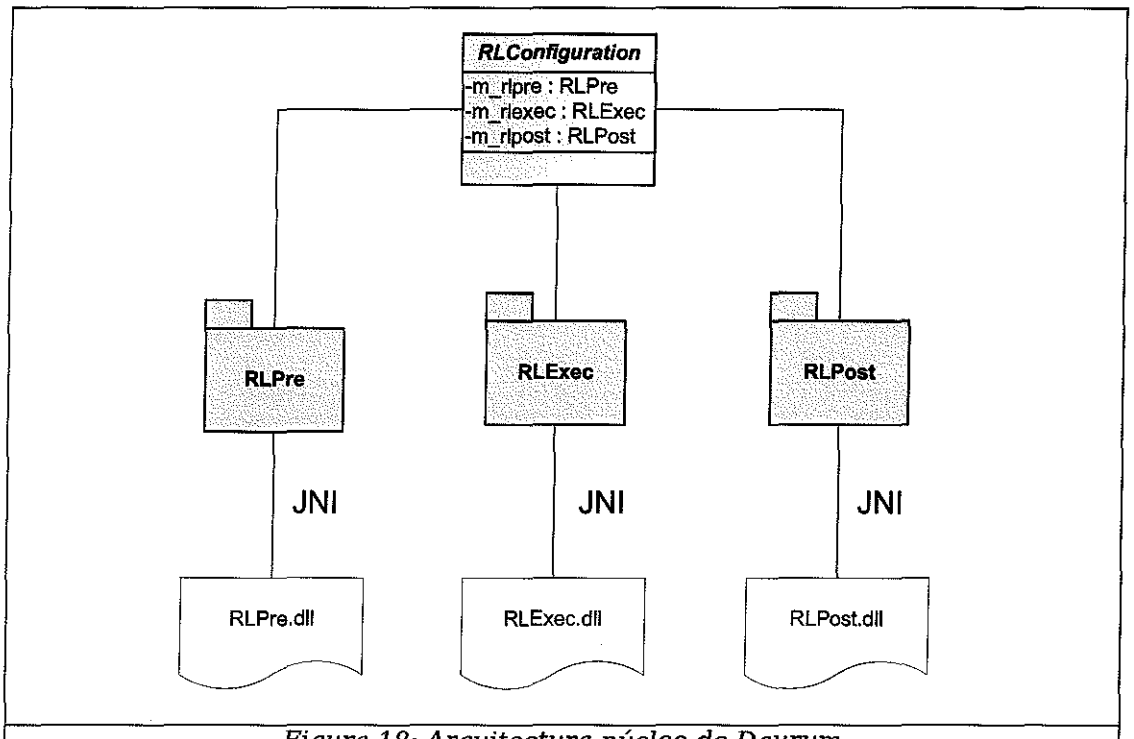

Figura 18: Arquitectura núcleo de Daurum

1. Módulo Preproceso

Este módulo se encarga de almacenar cuales son las fuentes de datos y que filtrado se les aplicará. También se encarga de la configuración de

las columnas de cruce, el comparador que se usará y el peso de cada columna.

2. Módulo Ejecución

Este módulo es donde encontramos la configuración del método para reducir el número de comparaciones. Permite la configuración del método de ejecución Sliding Window y el de Blocking.

3. Módulo Postproceso

El módulo de postproceso es el encargado de trabajar con el fichero de similitudes una vez obtenido este último. Permite odenar las similitudes por el campo que se le indique y modificar el estado de una similitud.

4.4. Descripción detallada de las clases y sus operaciones

A continuación vemos que clases y operaciones son necesarias para el diseño de los casos de uso. Esta apartado esta dividido de la misma forma que se ha divido el apartado de análisis de requisitos.

1. Administración de parámetros generales

Para llevar a cabo la funcionalidad de administración de parámetros generales, la cual incluye la funcionalidad de selección de lenguaje y la de administrar los directorios de filtros, hemos necesitado de una clase de presentación especifica, *AdminDialog*, para que el usuario pueda interaccionar con la configuración. También vemos que la clase del dialogo principal, *MainFrame*, también se comunica con las clases de dominio ya que la modificación del lenguaje se hace desde esta y no desde *AdminDialog*.

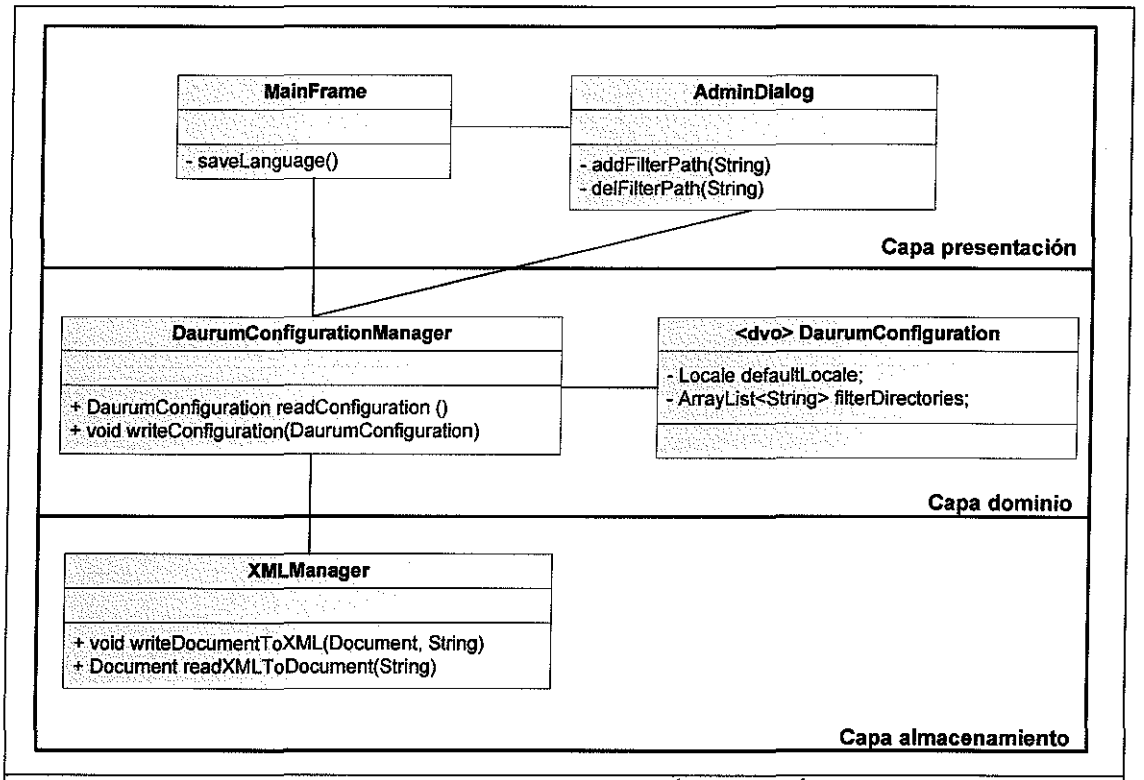

Figura 19: Diagrama clases de administración de parámetros generales

MainFrame: Dialogo principal de la aplicación. Esta clase presentación es la que instancia al resto de clases presentación principales. En este caso instancia a *AdminDialog*.

AdminDialog: Dialogo desde el cual se permite al usuario añadir y quitar directorio de filtros.

DaurumConfigurationManager: Clase de lógica que hace de interfaz entre las clases de la capa de presentación y la clase *DaurumConfiguration*. También se encarga de convertir la entidad *DaurumConfiguration* a formato XML y viceversa.

DaurumConfiguration: Clase entidad. Su función es almacenar los valores de la configuración de Daurum. No tiene ninguna operación de lógica.

- *defaultLocale*: Almacena el idioma de la aplicación.
- *filterDirectories*: Contiene todos los directorios desde donde la aplicación cargará los ficheros de filtros.

XMLManager: Esta clase es la encargada de cargar y guardar ficheros XML.

2. Configuración datos de proyecto Record Linkage

La configuración de los datos de proyecto es la primera pantalla que le aparece al usuario cuando realiza una configuración de *Record Linkage*. Vemos que la primera clase que hay en la capa de presentación es *MainFrame*, esta es el dialogo principal de la aplicación, desde donde el usuario llama al dialogo que hace de asistente para una configuración de *Record Linkage*, este es *RLWizardDialog*, esta clase de presentación es la principal en una configuración de *Record Linkage* y persiste durante toda la configuración. Las pantallas que van variando son las que dependen de *RLWizardDialog*, en este caso es *ProjectSettingsPanel*, que es un componente del dialogo *RLWizardDialog* y contiene todos los componentes gráficos que un usuario necesita para configurar los parámetros generales de un proyecto de *Record Linkage*.

Esta última clase de presentación, *ProjectSettingsPanel*, se comunica con la clase *RLWizardFacade*, que hace de interfaz entre las clases de presentación y las clases entidad.

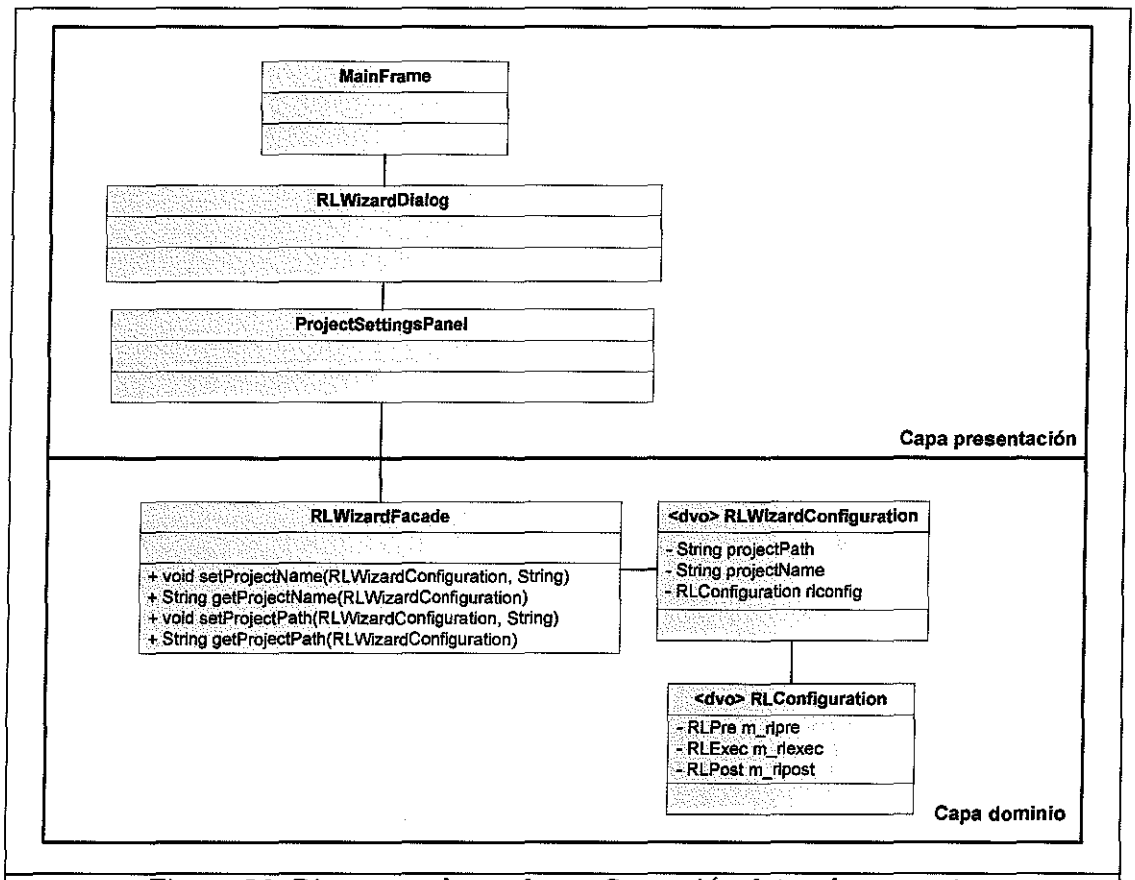

Figura 20: Diagrama clases de configuración datos de proyecto

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal de una configuración *Record Linkage*. Esta es común entre todas las opciones de una configuración.

ProjectSettingsPanel: Componente gráfico que contiene todos los elementos necesarios para configurar los datos de un proyecto (el nombre y la ruta donde se almacenarán los resultados).

RLWizardFacade: Interfaz entre *RLWizardDialog* y todas las subclases de la capa de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Se encarga de almacenar todos los datos de un proyecto de *Record Linkage*. No tiene operaciones de lógica.

RLConfiguration: Clase entidad. Contiene todos los valores de una configuración de *Record Linkage*. Esta configuración está dividida en la parte de preproceso, la parte de método de ejecución y la parte de postproceso.

3. Selección de datos de entrada

En este apartado vemos las clases implicadas a la hora de añadir una fuente de datos para la configuración. En la figura 21 vemos las clases que intervienen en la inclusión de una fuente de datos tipo fichero y en la figura 22 las que intervienen en una fuente de datos tipo base de datos. Se ha hecho la diferenciación para que se observe mas claramente, ya que en las fuentes de datos tipo base de datos intervienen varias clases que no intervienen a la hora de incluir ficheros ya que su configuración es más compleja.

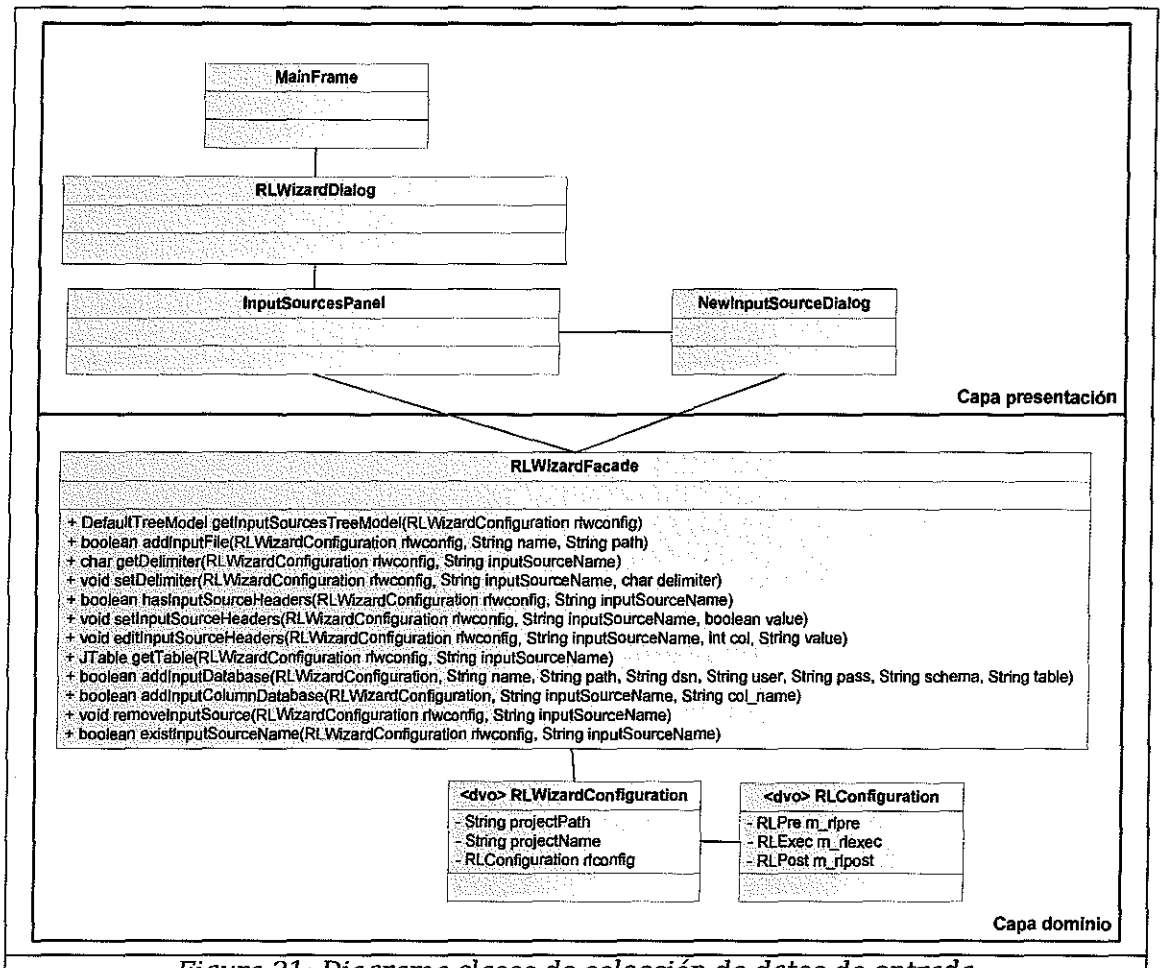

Figura 21: Diagrama clases de selección de datos de entrada

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal en una configuración de un proyecto *Record Linkage*. Esta persiste durante toda la configuración.

InputSourcePanel: Es el conjunto de componentes visuales que permiten al usuario añadir y eliminar fuentes de datos. Esta clase de presentación está integrada en el dialogo *RLWizardDialog* como un componente gráfico más.

NewInputSourceDialog: Dialogo que permite al usuario añadir una fuente de datos. Permite configurar el nombre, el tipo y en el caso de una fuente de datos tipo fichero su ruta.

RLWizardFacade: Clase de dominio que sirve como interfaz entre las clases de la capa de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene el nombre y la ruta de un proyecto *Record Linkage* y la configuración de este.

RLConfiguration: Clase entidad. Contiene todos los valores de una configuración *Record Linkage*.

A continuación, vemos el diagrama de las clases que intervienen en el proceso de añadir una fuente de datos tipo base de datos.

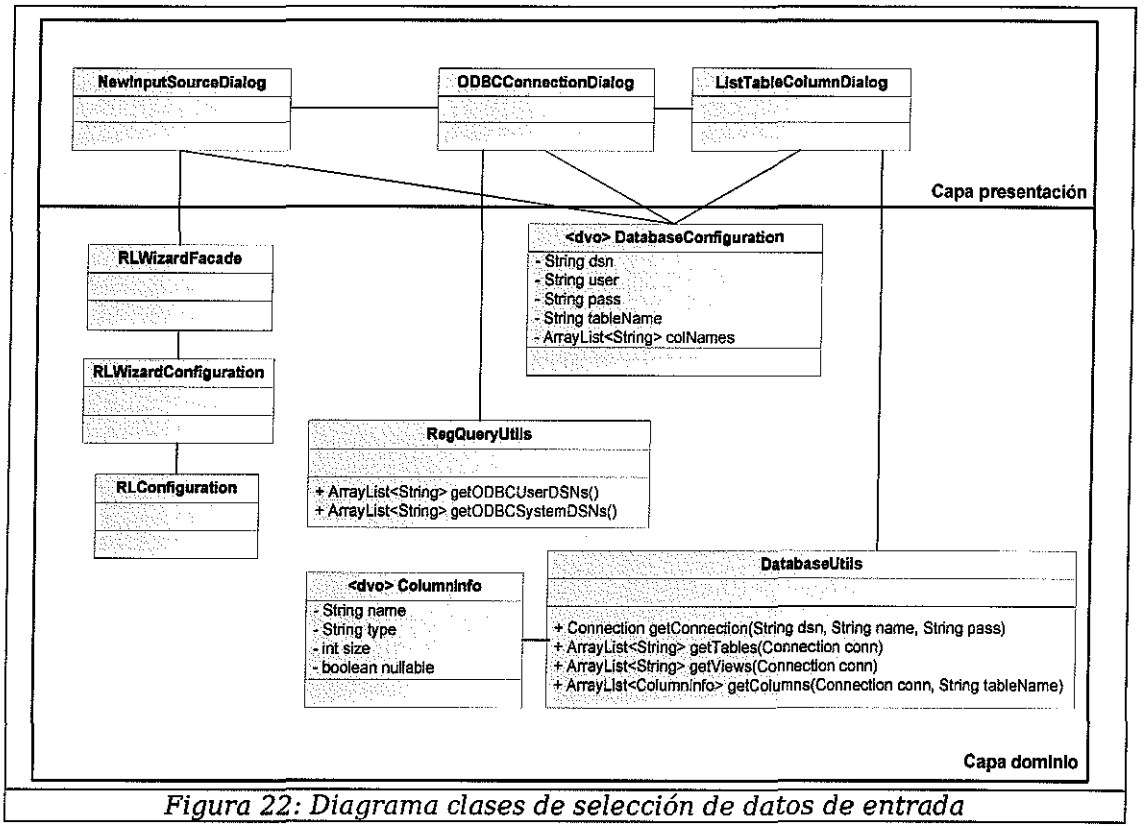

Figura 22: Diagrama clases de selección de datos de entrada

NewInputSourceDialog: Dialogo que permite al usuario añadir una fuente de datos. Permite configurar el nombre y el tipo. Desde este es donde se instancia al dialogo para configurar la conexión a la base de datos.

ODBCConnectionDialog: Dialogo que permite configurar los datos para una conexión ODBC.

ListTableColumnDialog: Dialogo que muestra todas las tablas y sus columnas de la conexión ODBC seleccionada.

DatabaseConfiguration: Clase entidad. Almacena temporalmente los datos de la conexión y las columnas seleccionadas hasta añadirlas a la clase entidad *RLConfiguration*.

RegQueryUtils: Clase de lógica que contiene funciones que llaman al registro. Tiene dos funciones, una para pedir la lista de los nombres de

las conexiones ODBC de sistema y otra para pedir las conexiones ODBC de usuario.

DatabaseUtils: Clase de lógica que contiene funciones relacionadas con las bases de datos. Estas son crear una conexión, recibir una lista con las tablas o recibir información de las columnas de una tabla.

ColumnInfo: Clase entidad. Sirve para agrupar todos los datos de una columna de una base de datos.

RLWizardFacade: Clase de lógica que sirve de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage*.

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage*.

4. Configuración columnas de cruce

En este apartado vemos todas las clases y las operaciones que intervienen en la configuración de las columnas de cruce para una configuración de *Record Linkage*. Este apartado esta dividido en dos partes que equivalen a dos pantallas diferentes. La primera es la configuración de las columnas de cruce, esto implica la selección de las columnas de los datos de entrada que se usarán para hacer el cruce y que tipo de preprocesado se le aplicará a cada columna de las fuentes de datos implicada en alguna columna de cruce. La segunda parte se trata de la configuración del peso de cada columna de cruce creada.

A continuación vemos, en el diagrama de la figura 23, las clases que intervienen para la configuración de las columnas de cruce.

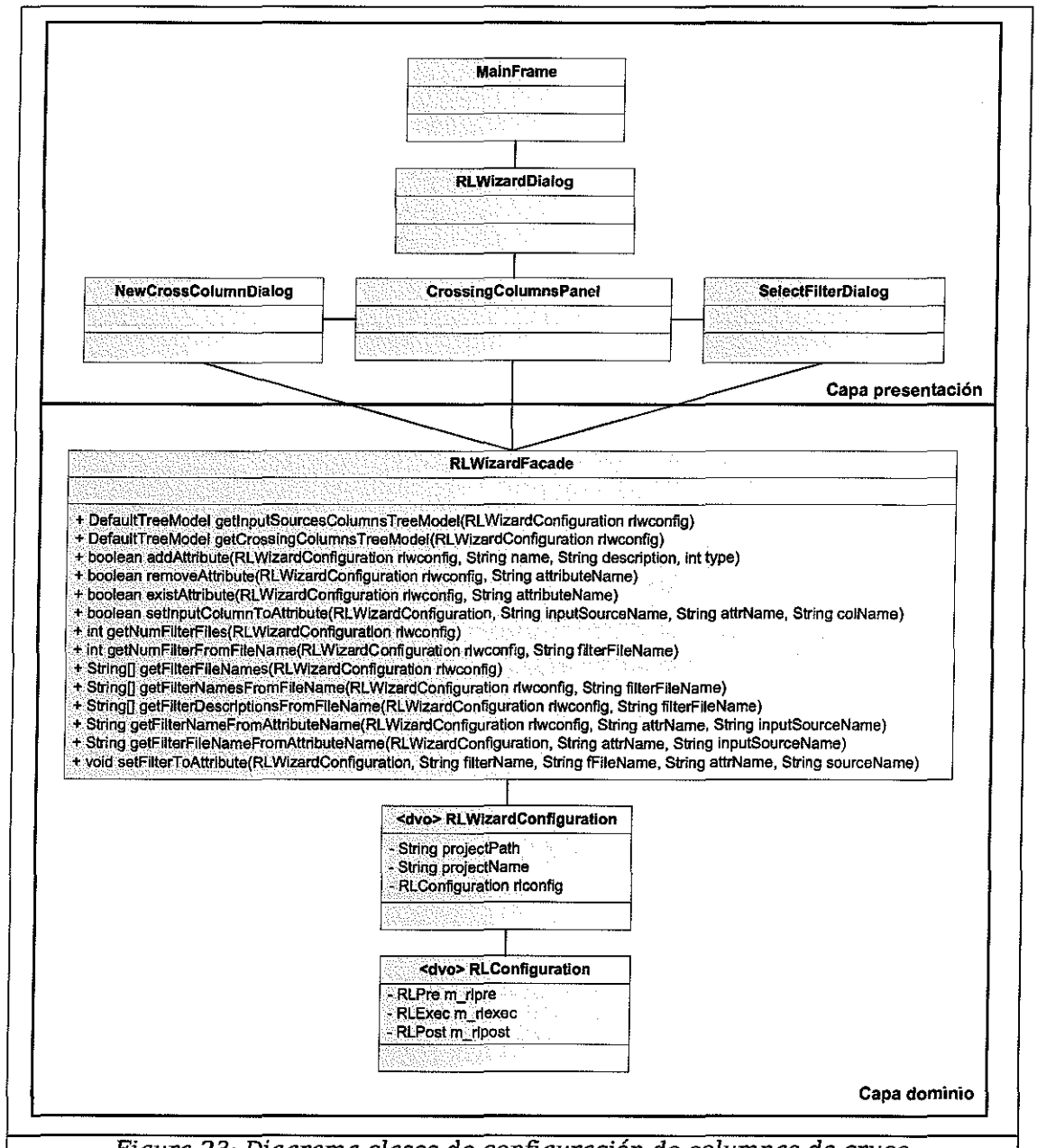

Figura 23: Diagrama clases de configuración de columnas de cruce

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal de una configuración de un proyecto de *Record Linkage*. Este persiste durante toda la configuración.

CrossingColumnsPanel: Conjunto de componentes necesarios para que el usuario pueda configurar las columnas de cruce. Entre estas opciones están añadir, configurar y modificar columnas de cruce.

NewCrossColumnDialog: Dialogo que permite al usuario configurar el nombre, la descripción y el tipo de una nueva columna de cruce.

SelectFilterDialog: Dialogo que muestra todos los filtros de todos los ficheros de filtros y permite al usuario seleccionar uno.

RLWizardFacade: Clase de lógica que hace de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage*.

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage*.

A continuación vemos las clases que interviene en la configuración de los pesos de las columnas de cruce.

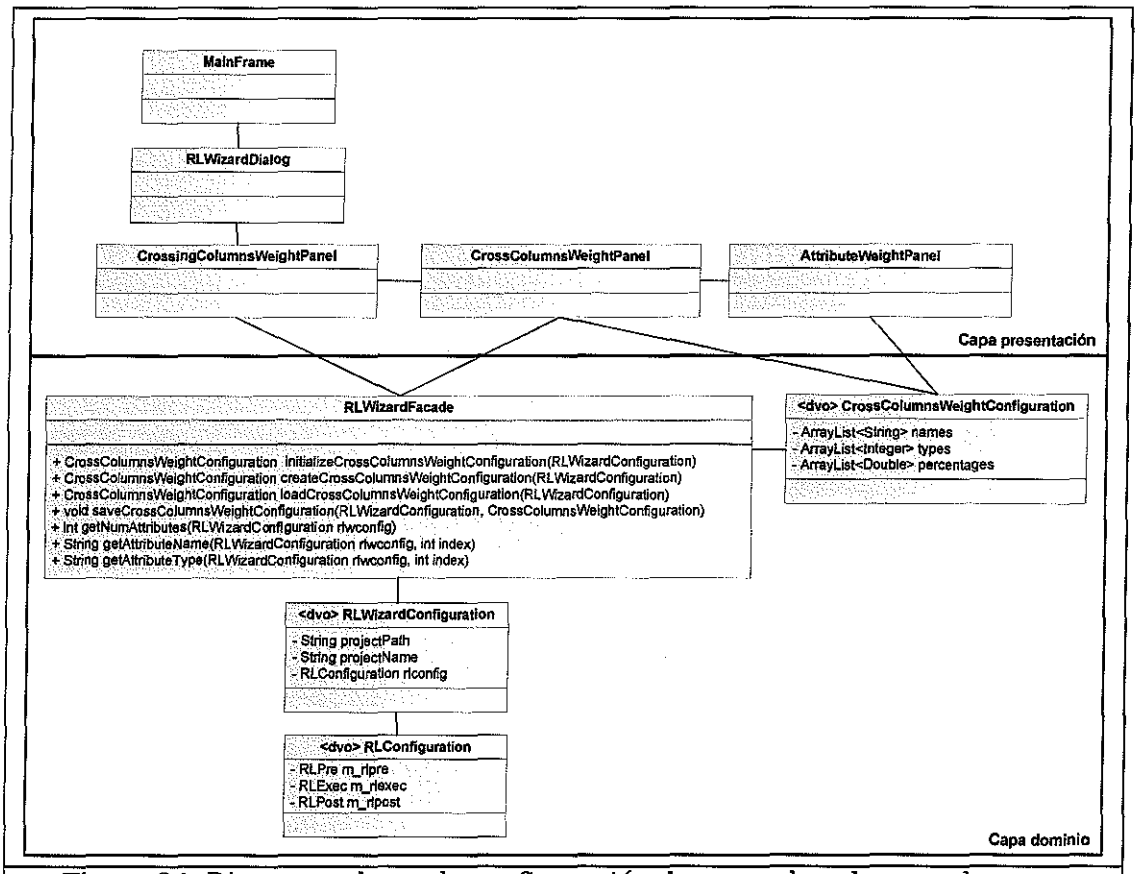

Figura 24: Diagrama clases de configuración de pesos de columnas de cruce

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal en una configuración de un proyecto de *Record Linkage*. Este persiste durante toda la configuración.

CrossingColumnsWeightPanel: Conjunto de componentes que permiten al usuario configurar los pesos de las columnas de cruce. Es un componente gráfico integrado en *RLWizardDialog*.

CrossColumnWeightPanel: Componente gráfico que agrupa el conjunto de componentes que permiten la configuración del peso de cada columna.

AttributeWeightPanel: Componente gráfico que muestra el nombre, el tipo y el peso de una columna de cruce y permite al usuario modificar el peso.

CrossColumnWeightConfiguration: Clase entidad. Almacena los datos de los pesos de las columnas temporalmente hasta que son introducidos en la clase entidad *RLConfiguration.*

RLWizardFacade: Clase de lógica que hace de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage.*

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage.*

5. Selección método de ejecución

En la figura 25 se ve el diagrama de clases que intervienen en la selección del método de ejecución.

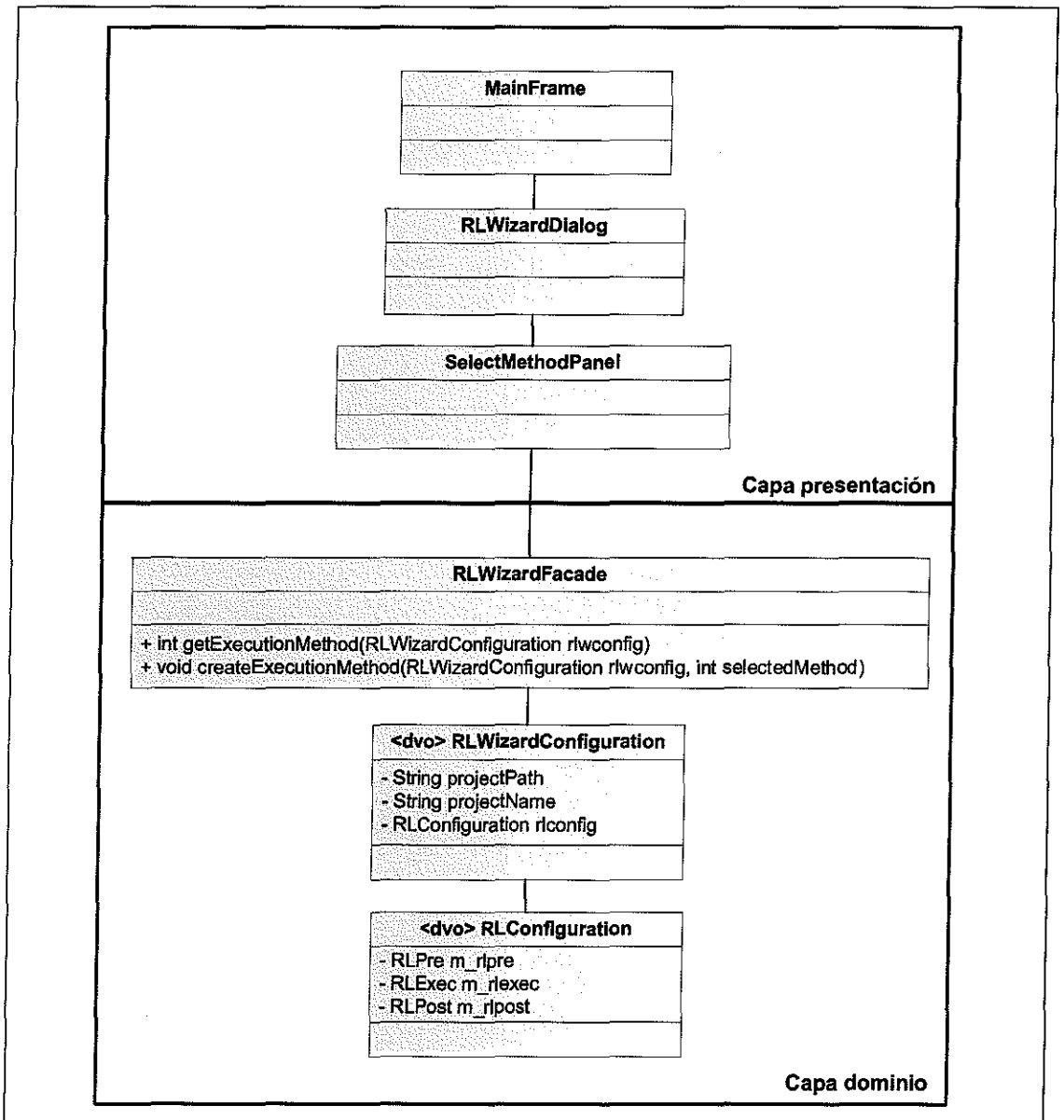

Figura 25: Diagrama clases de selección de método de ejecución

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal en una configuración de un proyecto de *Record Linkage*. Persiste durante toda la configuración.

SelectMethodPanel: Conjunto de componentes que permiten al usuario configurar el método de ejecución.

RLWizardFacade: Clase de lógica que hace de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage*.

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage*.

6. Configuración método Sliding Window

A continuación, en la figura 26, vemos el conjunto de clases que intervienen en la configuración del método de ejecución *Sliding Window*.

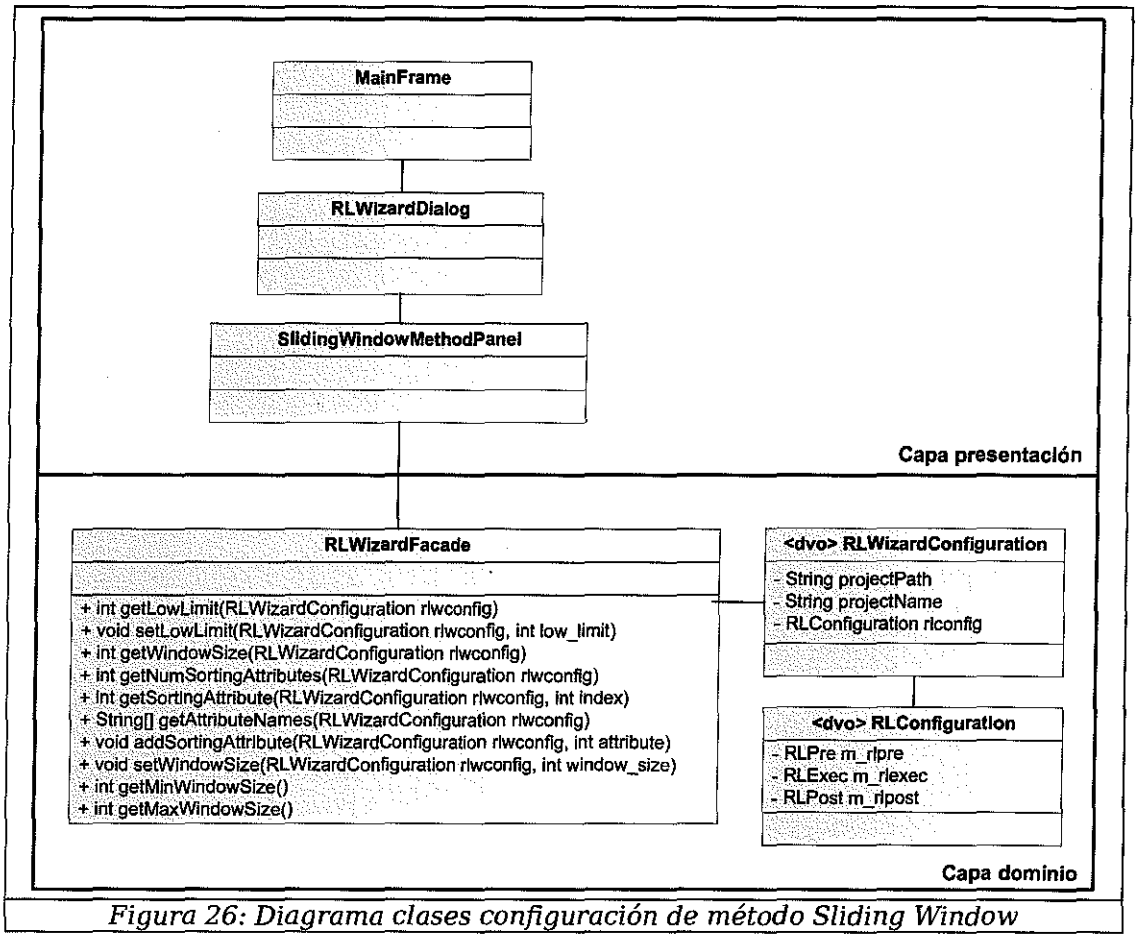

Figura 26: Diagrama clases configuración de método Sliding Window

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal en una configuración de un proyecto de *Record Linkage*. Persiste durante toda la configuración.

SlidingWindowMethodPanel: Conjunto de componentes que permiten al usuario configurar el método de ejecución *Sliding Window*. Estas opciones son las de porcentaje de similitud mínima, tamaño de la ventan y columnas de ordenación. Es un componente gráfico de *RLWizardDialog*.

RLWizardFacade: Clase de lógica que hace de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage*.

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage*.

7. Configuración método Blocking

A continuación, en la figura 27, vemos el conjunto de clases necesarias para la configuración del método de ejecución *Blocking*.

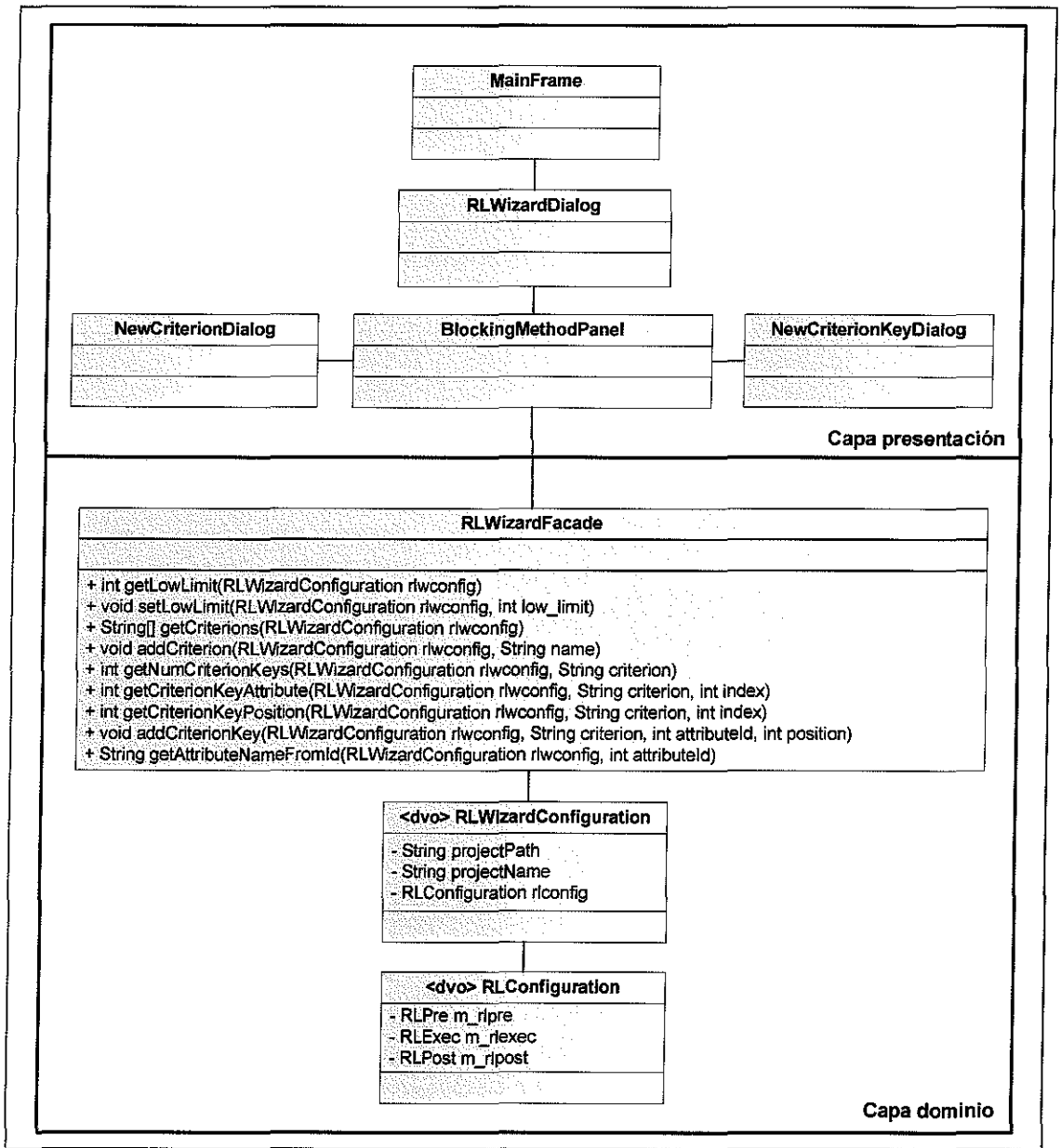

Figura 27: Diagrama clases configuración de método Blocking

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal en una configuración de un proyecto de *Record Linkage*. Persiste durante toda la configuración.

BlockingMethodPanel: Conjunto de componentes gráficos que permiten al usuario configurar el método de ejecución *Blocking*. Es un componente de *RLWizardDialog*.

NewCriterionDialog: Dialogo que permite al usuario añadir un nuevo criterio a la configuración *Blocking*.

NewCriterionKeyDialog: Dialogo que permite añadir claves a un criterio existente en la configuración *Blocking*.

RLWizardFacade: Clase de lógica que hace de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage*.

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage*.

8. Ejecución de Record Linkage

Una vez realizada todos los pasos de una configuración de *Record Linkage* se realiza el paso ejecución de la configuración o lo que es lo mismo detección de similitudes. Es también en este punto donde se guarda la configuración para una posterior utilización.

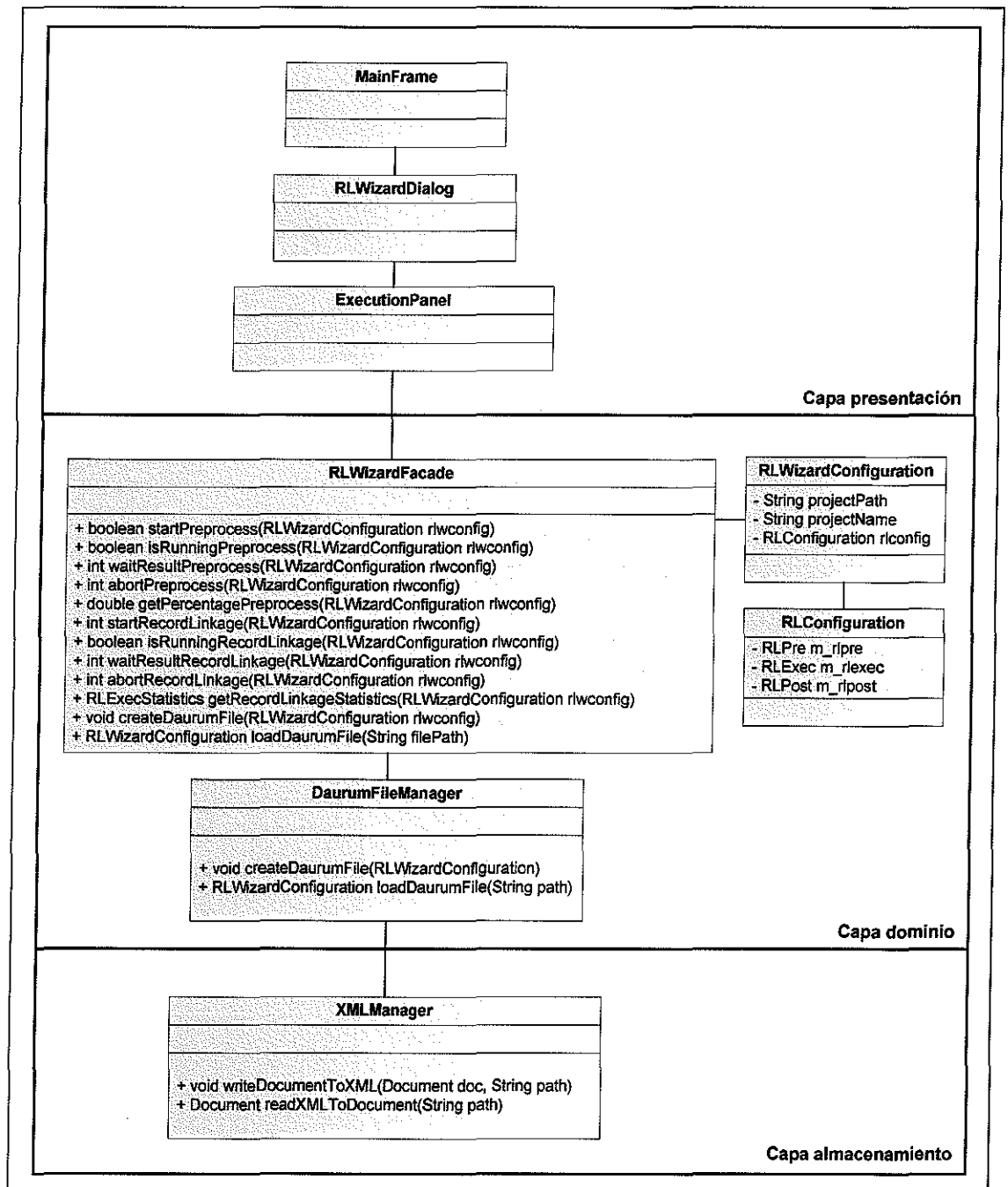

Figura 28: Diagrama clases de ejecución

MainFrame: Dialogo principal, encargado de instanciar al dialogo principal de una configuración de *Record Linkage*.

RLWizardDialog: Dialogo principal en una configuración de un proyecto de *Record Linkage*. Persiste durante toda la configuración.

ExecutionPanel: Conjunto de componentes que muestran el estado de la ejecución mientras esta transcurre. Es un componente de *RLWizardDialog*.

RLWizardFacade: Clase de lógica que hace de interfaz entre las clases de presentación y las clases entidad.

RLWizardConfiguration: Clase entidad. Contiene todos los datos de una configuración de un proyecto de *Record Linkage*.

RLConfiguration: Clase entidad. Contiene todos los datos de una configuración de *Record Linkage*.

DaurumFileManager: Clase de lógica que convierte los datos de una configuración de un proyecto de *Record Linkage* en estructura XML.

XMLManager: Clase que carga y guarda ficheros XML.

5. Implementación

En este apartado se explican los puntos más importantes a la hora de realizar la implementación del diseño.

El lenguaje programación que se ha usado en la implementación ha sido Java 6. El primer motivo es que es uno de los requerimientos del proyecto, pero también se trata de un lenguaje muy versátil ya que dispone desde clases que implementan estructuras básicas como son las listas, colas o tablas de hash hasta paquetes para el diseño de interfaces gráficas. Sus principales características son que es un lenguaje orientado a objetos, que hace una autogestión de la memoria y que es independiente de la plataforma sobre la que se ejecuta.

La herramienta que se ha usado para la implementación ha sido NetBeans 6. Esta se trata de un entorno de programación muy completo. De todas las características que proporciona NetBeans quizás la más relevante para este proyecto, al tratarse de realizar una interfaz de usuario es que dispone de un módulo que puedes diseñarlas de manera gráfica.

Los puntos más importantes a la hora de realizar la implementación han sido: la explotación de las características que ofrece la librería gráfica Swing para la interfaz, el diseño de la persistencia de los datos y la gestión de errores.

5.1. Librería gráfica Swing

Java 6 proporciona de base dos librerías gráficas. Estas son AWT y Swing. Para este proyecto se ha utilizado Swing ya que dispone de una mayor independencia de la plataforma que AWT y los componentes son más flexibles a la hora de extenderlos para personalizar su diseño o su comportamiento.

Las extensiones que se han necesitado para llevar a cabo las funcionalidades requeridas son:

- Cabeceras de tabla editables.
- Personalizar iconos de árbol de expansión.
- Componente híbrido entre un árbol de expansión y una tabla.

1. Cabeceras editables

Para llevar a cabo la funcionalidad de poder editar los nombres de las columnas de una tabla y hacerlo de la misma forma en que se hacia en Dalink 4.2., se ha necesitado extender la clase JHeader a EditableHeaders, para permitir que al hacer doble-clic en la cabecera el texto se pueda editar.

A continuación vemos las clases necesarias para llevar a cabo esta funcionalidad.

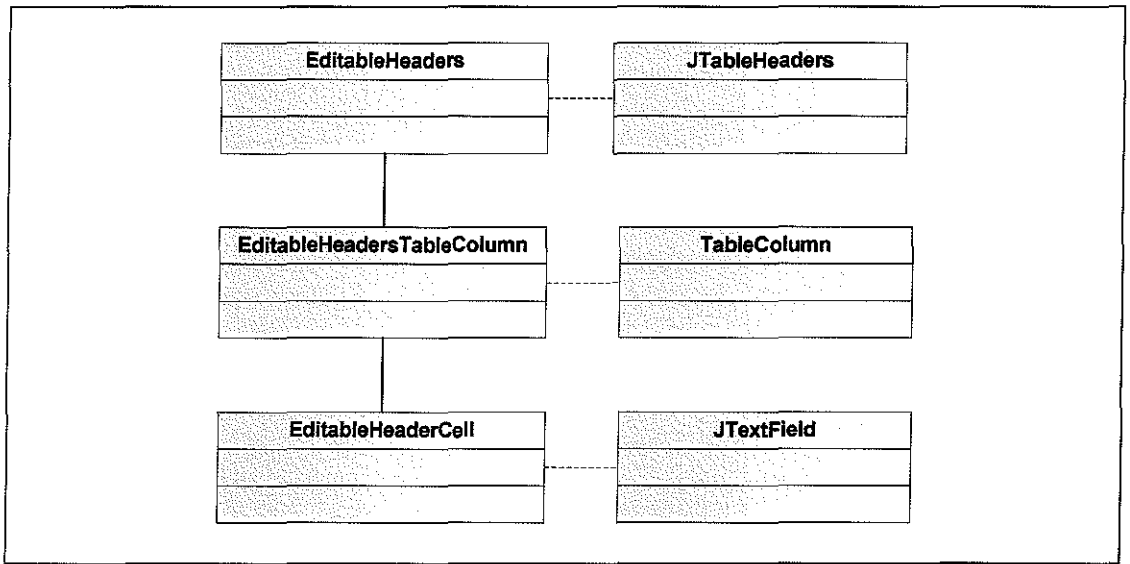

Figura 29: Diagrama clases de cabeceras editables

La clase *EditableHeaders* es una extensión de *JTableHeaders*. Es la clase principal para este nuevo tipo de cabeceras que se añaden a una *JTable*. La clase *EditableHeadersTableColumn* representa a una columna de las cabeceras, como se ve en la figura 29 es una extensión de *TableColum*. La diferencia es que recoge el evento cuando hacemos doble-clic sobre la columna y permite editar el nombre de la columna. Esta

característica está implementada en la clase *EditableHeaderCell* que es una extensión de *JTextField*.

2. Personalizar iconos de árbol de expansión

Por defecto Swing muestra unos iconos en el árbol de expansión o *JTree*, pero estos iconos no corresponden a la entidad que el icono representa por lo que se ha necesitado de expandir el componente o una parte de el conjunto de clases que intervienen en la renderización de un *JTree* para que estos iconos correspondan con el nodo que se esta mostrando.

Vemos en la figura 30 la clases que hemos extendido para llevar a cabo esta funcionalidad de la interfaz gráfica.

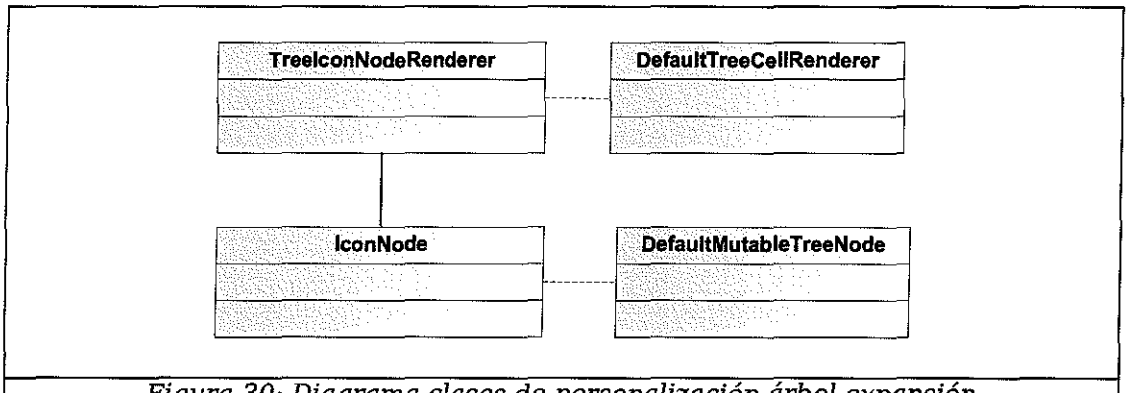

Figura 30: Diagrama clases de personalización árbol expansión

Se ha extendido la clase que por defecto renderizar un *JTree*, esta nueva clase es *TreeIconNodeRenderer*. En función del nodo que tenga que representar seleccionar un icono u otro para mostrar. También vemos que está la clase *IconNode* que es una extensión de un nodo por defecto en un *JTree*, la única diferencia con este es que contiene una variable que almacena el tipo de nodo que es para que *TreeIconNodeRenderer* sepa que icono ha de mostrar.

3. Componente híbrido entre árbol y tabla

Para representar el componente que se ve en la figura 31 se ha usado la clase de ofrece Sun llamada *JTreeTable*. Pero para que esta cumpla con las

funcionalidades que se necesitan para mostrar los ficheros de filtros y los filtros que estos contienen se ha necesitado extender alguna de las clases que intervienen en el componente *JTreeTable*.

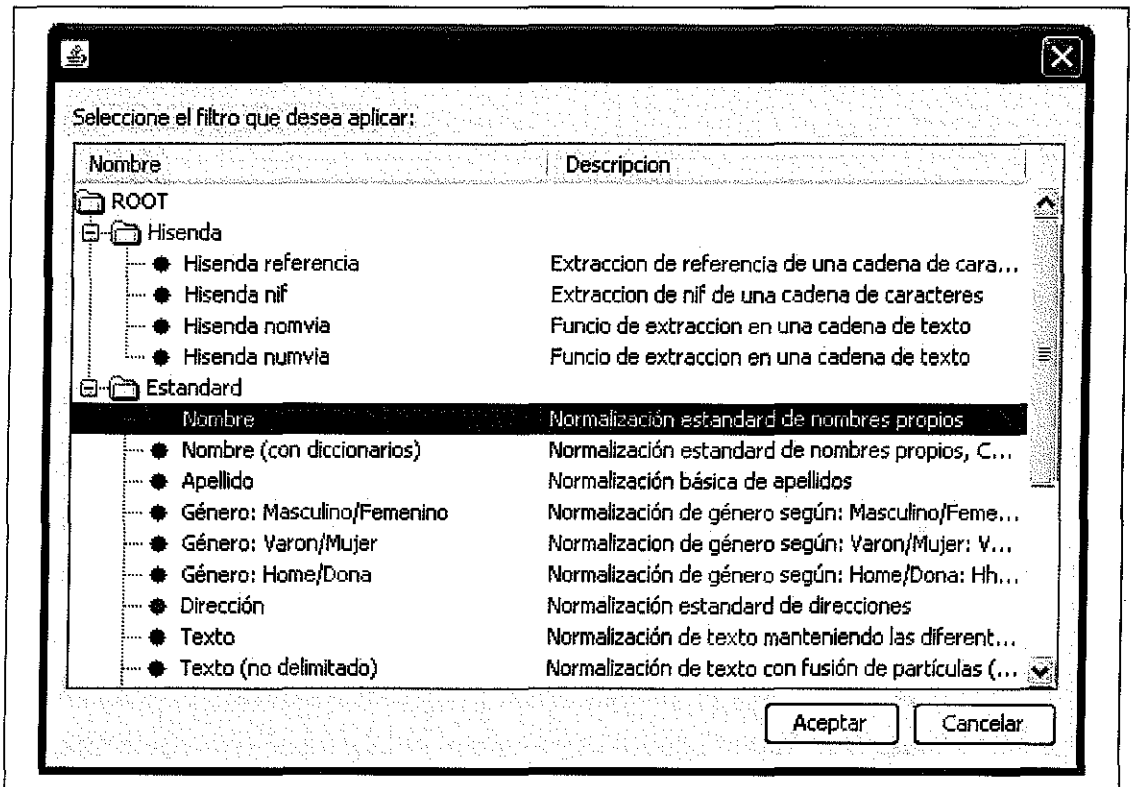

Figura 31: Pantalla de selección de filtro

A continuación vemos que clases hemos necesitado extender del componente *JTreeTable*.

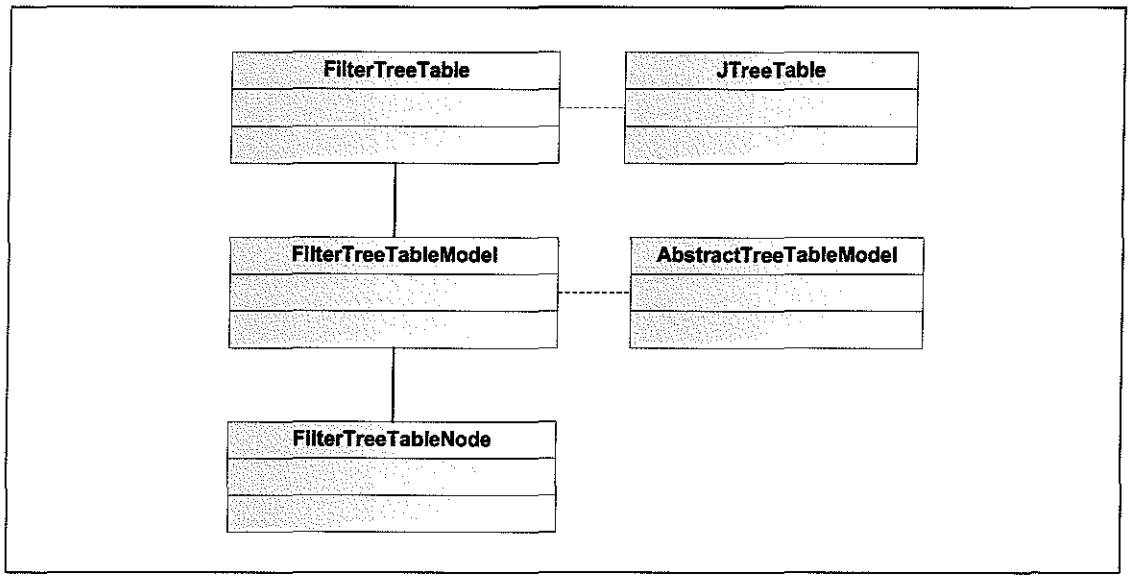

Figura 32: Diagrama clases de cabeceras editables

La clase principal de este nuevo componente es *FilterTreeTable*, que es una extensión de *JTreeTable*. Esta nueva clase hace uso de un modelo del contenido personalizado. Este modelo es *FilterTreeTableModel*, este muestra todos los ficheros de filtros y los filtros que contienen en forma de árbol. Por último, *FilterTreeTableNode*, es una clase entidad que contiene toda la información de un filtro (nombre, descripción y fichero al que pertenece).

5.2. Implementación del sistema de persistencia

Para implementar el sistema de persistencia de los datos hemos utilizado ficheros XML. Gracias a XML podemos estructurar la información para trabajar con ella utilizando librerías específicas.

Existen dos manera de cargar los datos de un fichero XML: SAX y DOM. Con SAX el documento XML se lee de forma secuencial y se van lanzando distintos eventos dependiendo del elemento que se haya leído. Esta forma permite cargar documentos XML parcialmente. Con DOM el documento XML es leído integro y almacenado en memoria. Una vez cargado, se puede navegar por la estructura fácilmente.

Para este proyecto se ha usado SAX ya que de esta forma podemos ir indetificando los elementos a medida que los vamos leyendo y así realizar la implementación de la carga de datos de manera más intuitiva.

En la aplicación almacenamos dos tipos de información: la configuración de parámetros generales y las configuraciones de Record Linkage.

La estructura del fichero XML que almacena la configuración de los parámetros generales es la siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<DaurumConfiguration>
    <Locale>es_ES</Locale>
    <FilterDirectories>
        <FilterDirectory>.\filters</FilterDirectory>
    </FilterDirectories>
</DaurumConfiguration>
```

Figura X: Estructura fichero configuración parámetros generales

La estructura del fichero XML que almacena una configuración de *Record Linkage* es la siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<DaurumProject name="prueba">
    <RLPre path="c:\prueba\prueba.rlp"/>
    <RLExec method="0" path="c:\prueba\prueba.rle"/>
    <RLPost path="c:\prueba\prueba.rlt"/>
</DaurumProject>
```

Figura X: Estructura fichero configuración de Record Linkage

Vemos que el fichero de configuración de *Record Linkage* hace referencia a otros dos ficheros. Estos ficheros son generados por el núcleo de Daurum a partir de la configuración almacenada de cada una de las partes del núcleo (RLPre, RLExec, RLPost).

5.4. Gestión de errores

Una parte importante a la hora de realizar una aplicación es la gestión de errores y avisos para evitar que el programa tenga comportamientos inesperados.

En este caso se ha creado una jerarquía de clases que identifican a cada uno de los errores para que en función del tipo de error la aplicación sepa como actuar. También se a creado una clase que gestiona todos los errores de la cual hacen uso todas las clases de la capa de presentación, así se centraliza la gestión de los errores.

A continuación vemos un diagrama donde podemos ver la jerarquía de clases que forman parte de una excepción:

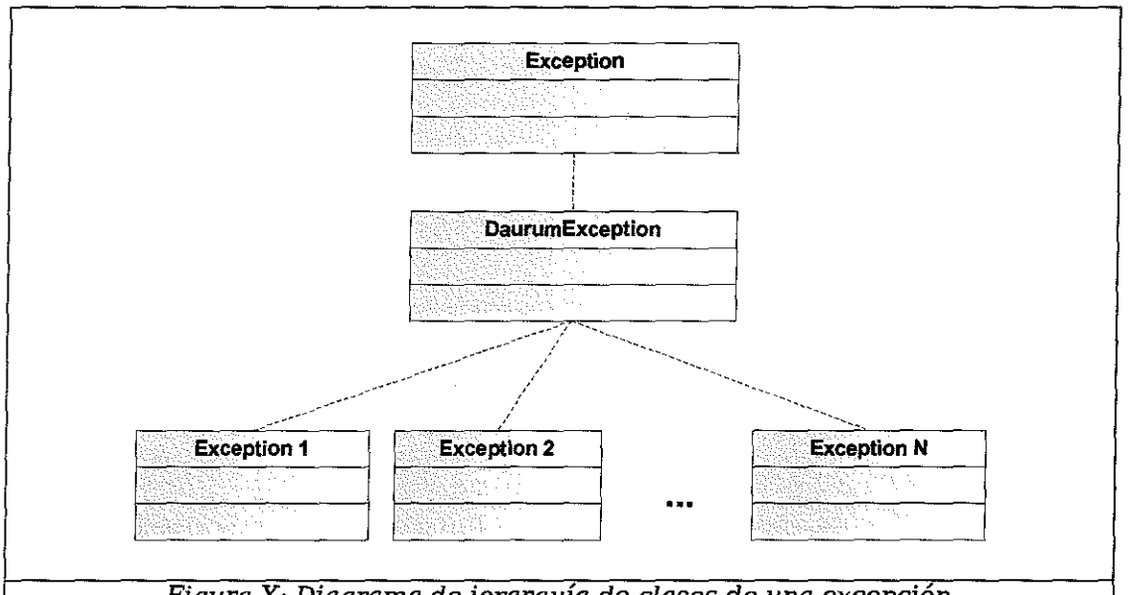

Figura X: Diagrama de jerarquía de clases de una excepción

En la siguiente figura vemos un diagrama donde se muestra como se lanza una excepción por la aplicación y quien la gestiona.

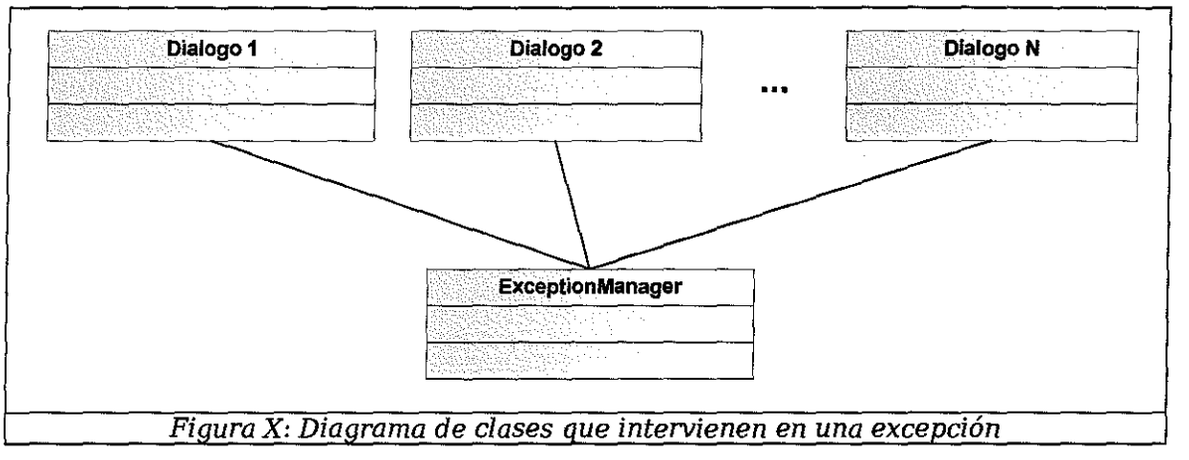

Figura X: Diagrama de clases que intervienen en una excepción

6. Futuras mejoras

En este proyecto se ha desarrollado, dentro de todo el proceso para solucionar el problema de *Record Linkage* (véase figura 1), la parte de extracción de datos, la fase de preprocesado y la fase de detección de similitudes. Por otra parte ya existe un módulo para Daurum que resuelve la fase de análisis de similitudes. Dentro de las partes que afectan a este proyecto, las mejoras que podrían hacerse son:

- Un editor de filtros completo. Actualmente los filtros están programados en *JavaScript*. Esto hace que para un usuario no experto se le haga muy complicado modificar o crear filtros para el preprocesado de los datos. Esta ampliación consistiría en crear un asistente que permitiera a un usuario no experto crear filtros a medida.
- Otra mejora sería la de dejar de usar para el acceso a las bases de datos ODBC. De esta forma la aplicación sería completamente independiente de la plataforma.
- Por último, sería hacer la aplicación modularizable, es decir, que cada uno de los pasos del problema de Record Linkage fuese independiente. Por ejemplo, podríamos crear un proyecto que solo fuese el preprocesado de unos datos o la extracción de una base de datos. O usar dos proyectos de preprocesado para realizar la detección de similitudes, con una interfaz fácil y agradable para el usuario. De esta forma se podría explotar cada una de las funcionalidades de cada fase y no solo para una configuración de *Record Linkage*.

7. Conclusiones

Durante los capítulos anteriores se han descrito los objetivos del proyecto, como se han especificado y como se ha realizado su diseño e implementación.

De lo expuesto en este documento, las fases de identificar los objetivos del proyecto y especificarlos, al partir estos de una aplicación que ya existe, ha sido una tarea mecánica, ya que ha consistido en ir replicando cada una de las funcionalidades de Dalink 4.2. en lo que se refiere a una configuración de *Record Linkage.*

Para realizar el diseño e implementación se ha hecho de tal forma que cada una de sus partes sea independiente. Esto quiere decir que se ha desarrollado una base sobre la que se puedan incluir módulos que solucionen partes del problema del *Record Linkage* y sobre esta se ha desarrollado un pequeño módulo para su administración y el módulo de asistente de configuración de *Record Linkage.* Este último resuelve las fases de extracción de datos, preproceso y detección de similitudes.

Por lo tanto, este proyecto forma parte de una aplicación más extensa que resuelva todas las fases del problema de *Record Linkage*, de manera que se ha desarrollado de modo que puedan añadirse otras partes, como la que resuelve la fase de análisis de similitudes, o que el módulo de asistente de configuración de *Record Linkage* pueda integrarse en otra aplicación.

Realizar este proyecto me ha permitido desarrollar una aplicación de principio a fin, donde se han aplicado técnicas de ingeniería del software. Esto me ha aportado vivir cada una de sus fases y ver cuales son los posibles problemas que pueden ir apareciendo durante su evolución. Como el replanteo del diseño para que cada una de las partes sean independientes y así poder reutilizarlas en otras aplicaciones. Así como llevar un control de versiones y un control de errores.

Para concluir, convendría que este proyecto fuese ampliado con el resto de módulos que faltan, como son el análisis de similitudes o la extracción de las similitudes a bases de datos o ficheros, de tal forma que la migración de Dalink 4.2. al lenguaje de programación Java estuviese completa y así poder ir añadiendo mejoras a medida que los clientes que usen esta herramienta las vayan necesitando.

8. Bibliografía

[1] Foro Sun Java http://forums.sun.com

[2] Ejemplos Java http://www.esus.com

[3] Tutorial Swing http://java.sun.com/docs/books/tutorial/uiswing/

[4] API de Java SE 6 http://java.sun.com/javase/6/docs/api

[5] Proyecto ingeniería informática Configuración de métodos de Record Linkage para dos ficheros (2003) Jordi Gomez Bao