



Centro Integrado de Formación Profesional
AVILÉS
Principado de Asturias

UNIDAD 6: DISEÑO DE INFORMES

DESARROLLO DE INTERFACES

2º CURSO

C.F.G.S. DESARROLLO DE APLICACIONES MULTIPLATAFORMA

REGISTRO DE CAMBIOS

Versión	Fecha	Estado	Resumen de cambios
1.0.	03/01/2024	Aprobado	Primera versión
1.1.	16/02/2024	Aprobado	Se añade apartado de informes sobre CSV
1.2.	03/04/2024	Aprobado	Corregido error en rutas de informes

ÍNDICE

ÍNDICE	2
UNIDAD 6: DISEÑO DE INFORMES	3
6.1. Informes incrustados y no incrustados en la aplicación.....	4
6.2. Herramientas gráficas integradas en el IDE y externas al mismo.....	5
6.2.1. SQL Server Reporting Services	5
6.2.2. JasperReports	5
6.2.3. Otras herramientas de generación de informes	7
6.3. Estructura general. Secciones	8
6.3.1. Encabezados y pies.....	11
6.3.2. Formatos de salida.....	12
6.4. Filtrado de datos.....	13
6.5. Informes con agrupamiento, recuentos parciales y subtotales.....	16
6.6. Numeración de líneas, recuentos y totales	19
6.7. Valores calculados.....	20
6.8. Subinformes	22
6.9. Imágenes. Gráficos	25
6.10. Bibliotecas para generación de informes	29
6.10.1. Clases, métodos, atributos.....	29
6.10.2. Parámetros.....	29
6.11. Conexión con las fuentes de datos. Ejecución de consultas	31
6.11.1. Inclusión de informe en aplicación Java	31
6.11.2. Archivos .csv como origen de datos	32
ÍNDICE DE FIGURAS.....	36
BIBLIOGRAFÍA – WEBGRAFÍA	36

UNIDAD 6: DISEÑO DE INFORMES

Los informes representan un elemento clave cuando se trata de obtener información útil de una base de datos. Estos elementos proporcionan información en un formato que cualquier persona puede entender y, dado que la mayoría de ellos están diseñados para impresión, dan a sus datos un elemento de portabilidad.

La construcción de informes es sin duda un proyecto más complejo que la creación de consultas o formularios. De hecho, es una tarea que requiere algo de planificación. Los informes suelen incluir números de página, fechas y horas en las que se imprimieron, agrupaciones de registros y resúmenes de datos.

Uno de los objetivos más útiles de los informes es mostrar resultados que no pueden representarse directamente en un formato de tabla de base de datos. Un informe debe permitir ver los detalles importantes a primera vista.

Un ejemplo de informe que resume los resultados de una base de datos de subastas no sólo mostraría los registros individuales, sino también grupos de acuerdo con el mejor postor. También mostraría el subtotal, la puja del comprador y el total de la suma para cada ganador. Este informe es un buen ejemplo de cómo un informe puede proporcionar información que simplemente no aparece por sí sola en una tabla de una base de datos.

6.1. INFORMES INCRUSTADOS Y NO INCRUSTADOS EN LA APLICACIÓN

Al planear la creación de una aplicación con un entorno de desarrollo integrado (IDE), una de las consideraciones más importantes es si se utilizan informes incrustados o no incrustados. Saber cuáles son los aspectos fundamentales que afectan a la incrustación de informes ayudará a elegir la mejor estructura para el proyecto.

¿Qué diferencia existe entre los informes incrustados y los informes no incrustados? Un informe incrustado es aquel que se ha importado a un proyecto del IDE o que se ha creado dentro de él. Cuando un informe se incrusta en el proyecto, se genera automáticamente una clase contenedora. Por el contrario, un informe no incrustado es aquel que es externo al proyecto del IDE. Hay muchas formas de acceder al informe para cargarlo en un modelo de objetos a fin de habilitar la interacción mediante programación, pero el informe siempre será externo al proyecto del IDE.

¿Cómo funciona un informe incrustado? Cuando el informe se importa al proyecto o se crea en él, se genera una clase contenedora que normalmente tiene el mismo nombre que el informe. Esta clase contiene o representa el informe en el proyecto. Cuando esto ocurre, todo el código del proyecto interactúa con la clase del informe que se ha creado para representarlo, en lugar de hacerlo con el propio archivo del informe original. Al compilar el proyecto, tanto el informe como su clase contenedora se incrustan en el ensamblado, tal y como ocurriría con cualquier otro recurso del proyecto.

¿Cómo funciona un informe no incrustado? El acceso a un informe no incrustado siempre se obtiene externamente. El software puede acceder de varias formas. Por ejemplo:

- El informe puede estar en una unidad de disco duro en una ruta de directorio de archivos.
- El informe puede estar expuesto como servicio web de informes.

Para simplificar la implementación del proyecto, podría ser preferible usar informes incrustados. Habría que trabajar con menos archivos y no preocuparse de si los informes están colocados correctamente en la ruta de directorio de archivos definida. Además, esta solución es más segura, puesto que los informes no se exponen a modificaciones. Por otra parte, los informes incrustados son más sencillos y seguros, pero conllevan más trabajo; no se pueden modificar sin volver a compilar todo el proyecto. Existen límites en cuanto al tamaño que puede tener un informe incrustado. Un informe muy grande se compila como recurso incrustado.

Si los informes deben modificarse regularmente, es preferible utilizar informes no incrustados. De esta manera, se podrá acceder y modificar más fácilmente, sin tener que preocuparse por la necesidad de volver a compilar los añadidos cada vez. Además, los informes no incrustados tienen ventajas de escalabilidad.

6.2. HERRAMIENTAS GRÁFICAS INTEGRADAS EN EL IDE Y EXTERNAS AL MISMO

6.2.1. SQL SERVER REPORTING SERVICES

Uno de los servicios de generación de informes más conocidos es SQL Server Reporting Services. Se trata de un completo sistema de generación de informes, que cubre todo el proceso de creación de listados, desde el diseño hasta su distribución.

La maquinaria que sustenta Reporting Services se basa en los siguientes pilares:

- Servidor de informes. Es el elemento principal, representa al motor de informes, siendo el proveedor de todas sus características. Se trata de un servicio web encargado del procesamiento y generación de los informes creados con el diseñador visual, que proporciona un sistema de seguridad en el acceso a los informes y expone un API que permite la programación de sus funcionalidades.
- Administrador de informes. Aplicación que facilita la gestión de los informes. Entre las diversas tareas administrativas que permite esta utilidad se encuentra la publicación de los informes en un árbol de directorios, programación de suscripciones, asignación de permisos, creación de directorios, etc. Este componente está disponible desde el entorno de administración general de SQL Server: SQL Server Management Studio, y también como una aplicación web.
- Base de datos del servidor de informes. Se trata de una base de datos SQL Server que almacena la información relativa a los informes publicados, programación de suscripciones, configuración de seguridad y demás metadatos necesarios para la ejecución de los informes.

Otros aspectos relacionados con la gestión del servidor de informes de Reporting Services se indican en los siguientes puntos:

- Orígenes de datos de los informes. La información en la que residen los datos que son visualizados en un informe puede ser de origen muy diverso: SQL Server, Oracle, OLE DB, ODBC, etc.
- Formato de generación. Al visualizar un informe, la salida por defecto es en formato HTML, sin embargo, Reporting Services permite su generación en otros formatos también muy comunes actualmente, tales como PDF, Excel, TIFF, etc.
- Seguridad. Reporting Services incluye un sistema de seguridad basado en la autenticación de Windows, que impide que cualquier usuario pueda acceder libremente a los informes si no dispone de las credenciales adecuadas; esto permite establecer un control de acceso a los informes con datos sensibles.

6.2.2. JASPERREPORTS

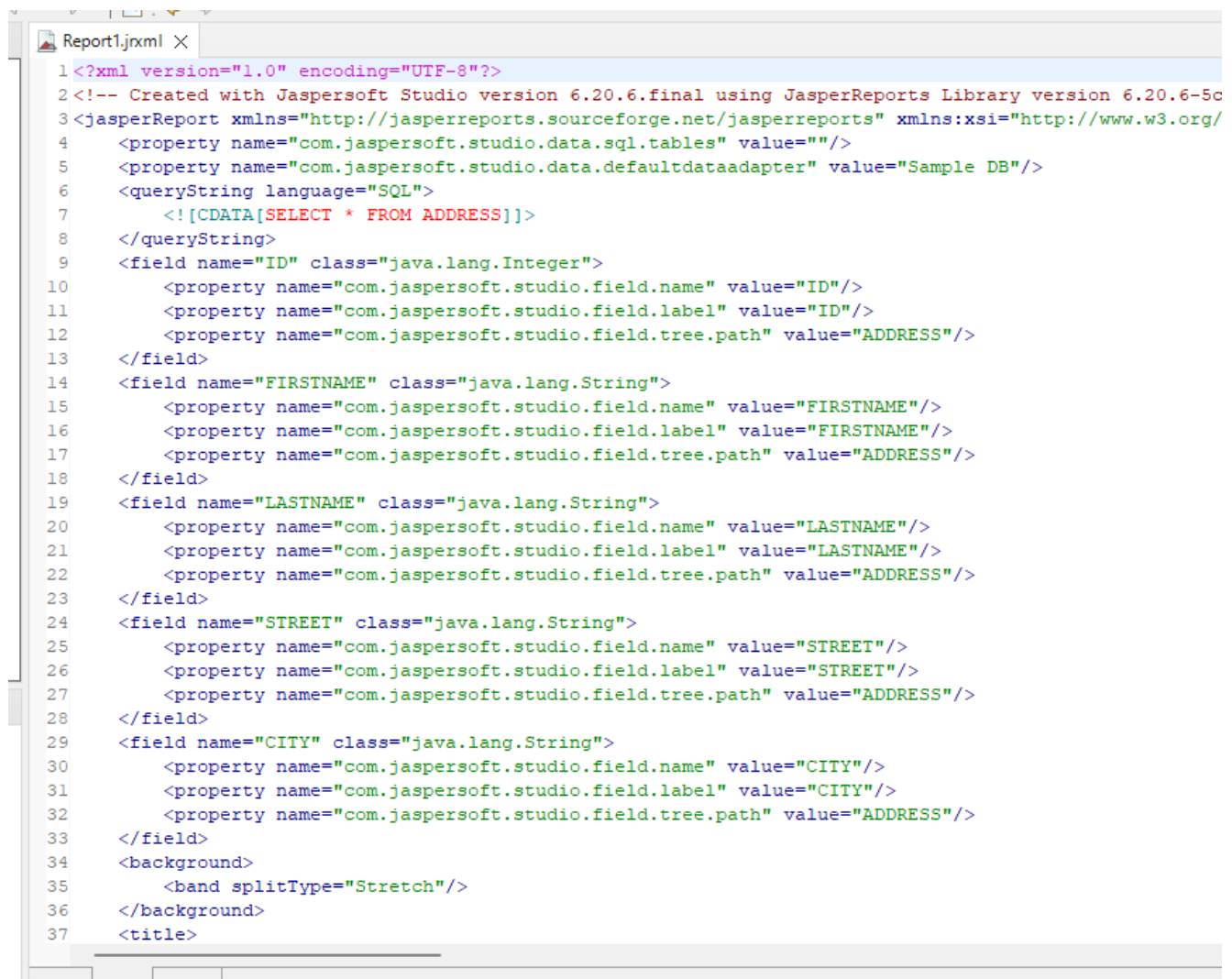
La biblioteca JasperReports de la empresa JasperSoft es un muy buen sistema de Reporting Open Source dedicado a aplicaciones Java. Esta herramienta puede funcionar de manera independiente a Eclipse o en colaboración con este. [Jaspersoft Studio](#) es el editor gráfico

WYSIWYG (What You See Is What You Get) integrado en Eclipse que se encarga de la creación de plantillas de informes.

Para generar informes, hay que pasar por tres etapas: la creación del archivo Jasper describiendo el contenido del informe, la compilación de este archivo en un formato explotable por Jasper en producción y alimentarlo con datos para rellenar el informe.

JasperReports se basa en principio sobre archivos XML (eXtensible Markup Language). No es necesario dominar XML para esta etapa ya que la creación se realiza mediante Jaspersoft Studio, que genera el archivo correspondiente, aunque es recomendable ya que llegados a cierto punto podría ser interesante intervenir directamente en el archivo XML.

Al final de esta etapa, se obtiene un archivo XML con extensión jrxml que representa la plantilla del informe desde el cual se pueden producir varios documentos con distintos formatos.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- Created with Jaspersoft Studio version 6.20.6.final using JasperReports Library version 6.20.6-5c
3 <jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports" xmlns:xsi="http://www.w3.org/
4   <property name="com.jaspersoft.studio.data.sql.tables" value="" />
5   <property name="com.jaspersoft.studio.data.defaultdataadapter" value="Sample DB" />
6   <queryString language="SQL">
7     <![CDATA[SELECT * FROM ADDRESS]]>
8   </queryString>
9   <field name="ID" class="java.lang.Integer">
10     <property name="com.jaspersoft.studio.field.name" value="ID" />
11     <property name="com.jaspersoft.studio.field.label" value="ID" />
12     <property name="com.jaspersoft.studio.field.tree.path" value="ADDRESS" />
13   </field>
14   <field name="FIRSTNAME" class="java.lang.String">
15     <property name="com.jaspersoft.studio.field.name" value="FIRSTNAME" />
16     <property name="com.jaspersoft.studio.field.label" value="FIRSTNAME" />
17     <property name="com.jaspersoft.studio.field.tree.path" value="ADDRESS" />
18   </field>
19   <field name="LASTNAME" class="java.lang.String">
20     <property name="com.jaspersoft.studio.field.name" value="LASTNAME" />
21     <property name="com.jaspersoft.studio.field.label" value="LASTNAME" />
22     <property name="com.jaspersoft.studio.field.tree.path" value="ADDRESS" />
23   </field>
24   <field name="STREET" class="java.lang.String">
25     <property name="com.jaspersoft.studio.field.name" value="STREET" />
26     <property name="com.jaspersoft.studio.field.label" value="STREET" />
27     <property name="com.jaspersoft.studio.field.tree.path" value="ADDRESS" />
28   </field>
29   <field name="CITY" class="java.lang.String">
30     <property name="com.jaspersoft.studio.field.name" value="CITY" />
31     <property name="com.jaspersoft.studio.field.label" value="CITY" />
32     <property name="com.jaspersoft.studio.field.tree.path" value="ADDRESS" />
33   </field>
34   <background>
35     <band splitType="Stretch" />
36   </background>
37   <title>
```

1. Ejemplo de código fuente de informe JasperReports

A partir del archivo jrxml, llamado también plantilla jrxml, Jasper procede a su compilación que permite verificar si toda la plantilla está bien construida, sobre todo si es coherente con la norma XML y si los parámetros son correctos.

Si la compilación finaliza correctamente, se genera un archivo binario con la extensión Jasper. Se puede entonces utilizar este archivo para representar los datos dependiendo de las elecciones tomadas en el momento de la creación del archivo jrxml con Jaspersoft Studio. También es posible compilar en tiempo de ejecución el archivo binario Jasper.

Se pueden utilizar varias fuentes de datos en el archivo Jasper: XML, CSV, un conjunto de registros que provengan de una base de datos SQL, objetos Java con formato JavaBeans, etc. Estas fuentes de datos servirán para generar el informe con un formato apropiado. El documento final está entonces listo para su uso. Solo queda elegir entre todos los tipos de archivo de salida propuestos por Jasper: PDF, DOCX, HTML, ODT...

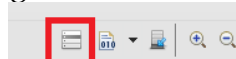
6.2.3. OTRAS HERRAMIENTAS DE GENERACIÓN DE INFORMES

Otras herramientas conocidas para generar informes son:

- [Crystal Reports](#). Esta herramienta de generación de informes se ha utilizado históricamente en IDEs Microsoft como Visual Studio antes de la irrupción de Reporting Services. Ahora pertenece a la empresa SAP, Crystal Solutions. Dispone de SDK's para el desarrollo de aplicaciones .NET, Java y DOM. Es compatible con gran variedad de orígenes de datos, desde motores de base de datos, a hojas de cálculo, archivos XML o SAP.
- [Eclipse BIRT](#). BIRT son las siglas de Business Intelligence Reporting Tools. Es un sistema de generación de informes para aplicaciones web, basadas en Java o en Java EE. Tiene dos componentes principales: un diseñador de informes basado en Eclipse y un componente que se puede agregar al servidor de aplicaciones y que genera informes en tiempo de ejecución. Ofrece un motor de gráficos y es compatible con gran cantidad de orígenes de datos, bien sea SGBD relacionales, archivos con formato, etc.
- [Pentaho Reporting](#). Es un conjunto de herramientas que permiten crear informes relacionales y analíticos de una amplia gama de orígenes de datos. Es capaz de crear informes en PDF, Excel, HTML, XML y CSV entre otros. Puede usarse en Java, C# y Python.
- [R Markdown](#). Forma parte del proyecto R y está principalmente encaminado a la visualización en forma de documentos de la ejecución de programas estadísticos con R. También soporta Python, SQL y otros lenguajes.
- [FastReport Open Source](#). Se trata de un generador de informes para .NET. Se puede utilizar en proyectos MVC y aplicaciones de consola. También puede trabajar con C++, Python y Java.
- [Oracle Reports](#). Herramienta de generación de informes que permite crear informes basados en datos almacenados en Oracle Database.

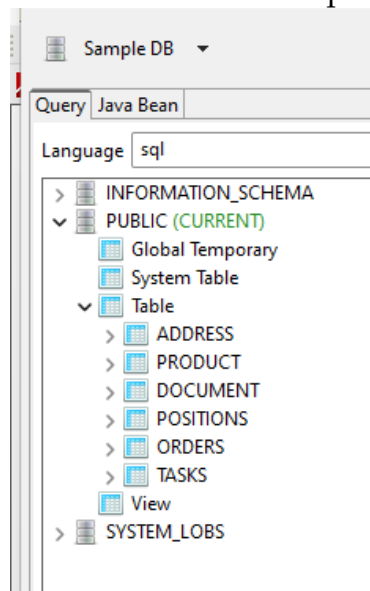
6.3. ESTRUCTURA GENERAL. SECCIONES

Para ilustrar los siguientes apartados, se utilizará JasperReports. En primer lugar, se va a crear un primer informe en JasperReports que muestre la lista de todos los clientes. El generador de informes proporciona una base de datos en HSQLDB llamada SampleDB que está basada en la clásica Northwind presente en gestores como SQL Server o bases de datos como Access. Una vez seleccionada en el apartado **Data Adapters**, se puede ver su estructura en el editor de consultas que tiene la herramienta integrado.



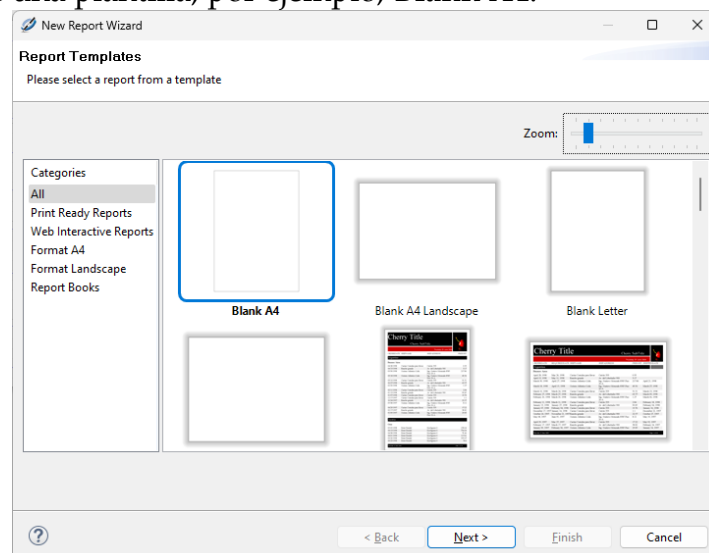
2. Dataset and Query editor dialog

En él puede verse la estructura de la base de datos en el panel a la izquierda:



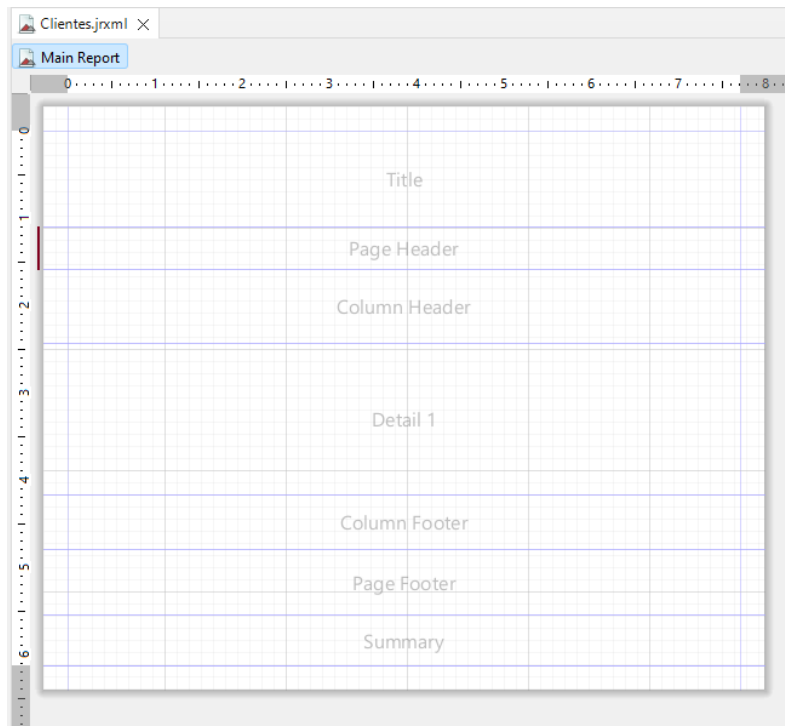
3. Estructura de bases de datos de muestra

Es momento de empezar el primer informe. En el menú, se elige la opción **File / New / Jasper Report**. Aquí se elige una plantilla, por ejemplo, **Blank A4**.



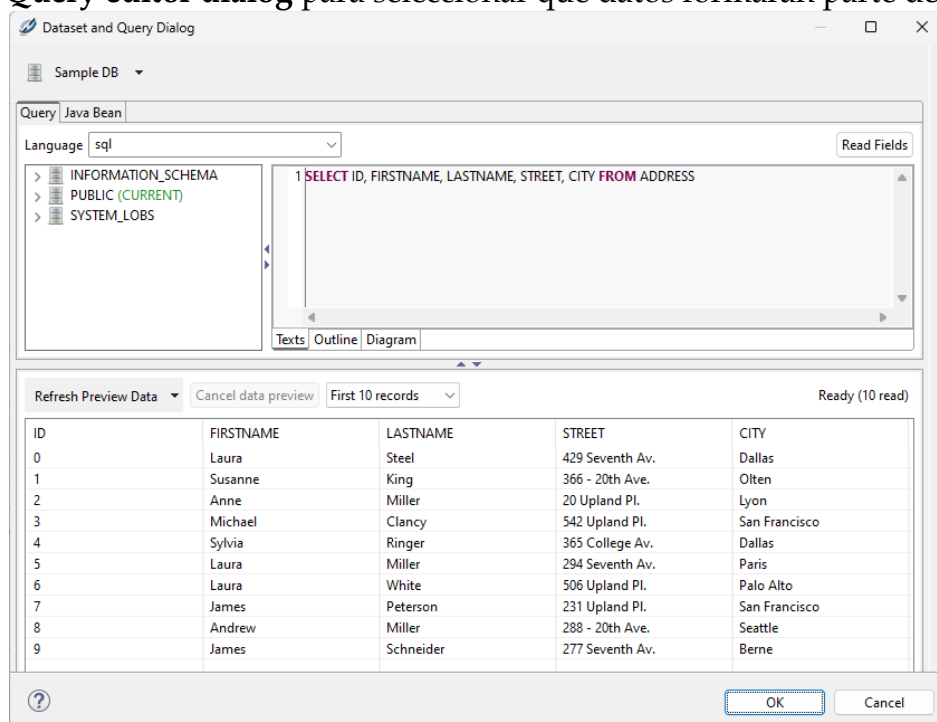
4. Selección de plantilla de informe

Se elige el nombre del informe (Clientes.jrxml) y su ubicación. Ahora se determina cuál será el Data Adapter utilizando la misma opción en la que se vio previamente la estructura de la base de datos. El informe lucirá de esta forma:



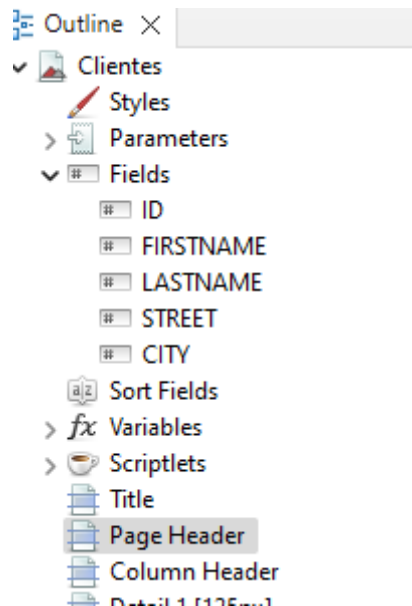
5. Secciones de un informe vacío

Como puede verse, el informe consta de una serie de secciones o, en la nomenclatura JasperReports, bandas. Son las siguientes: Título, Encabezado de página, Encabezado de columna, Detalle 1, Pie de columna, pie de página, resumen. A continuación, hay que volver a **DataSet and Query editor dialog** para seleccionar qué datos formarán parte del informe.



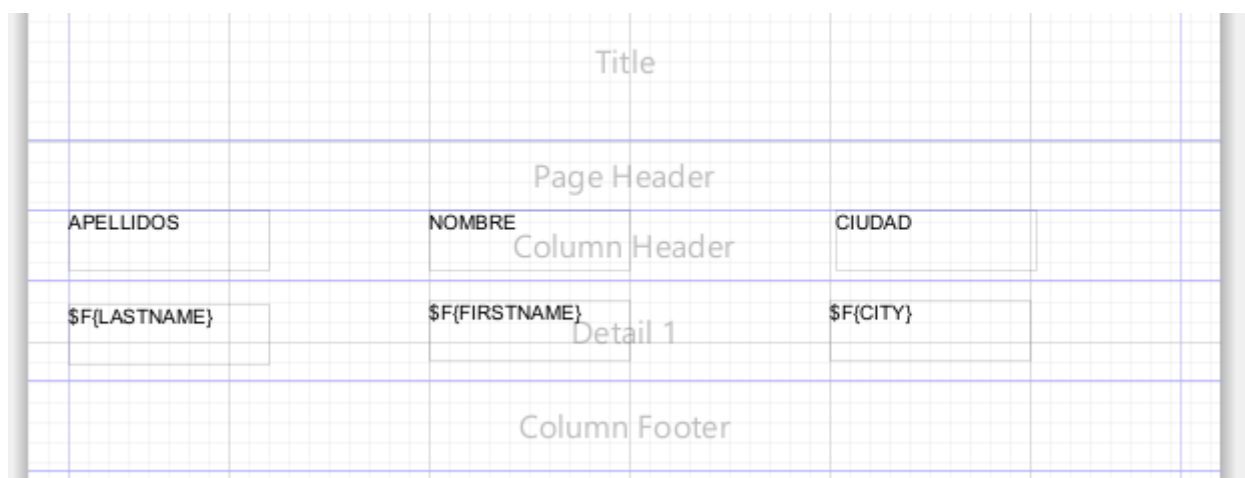
6. Consulta de clientes

Los clientes se encuentran en la tabla ADDRESS, por lo que se lanza una consulta que los devuelva todos. El editor permite mostrar una vista previa de los datos de la consulta. Los campos de la consulta se muestran en el apartado **Fields** dentro del panel **Outline**.



7. Campos de la base de datos en panel Outline

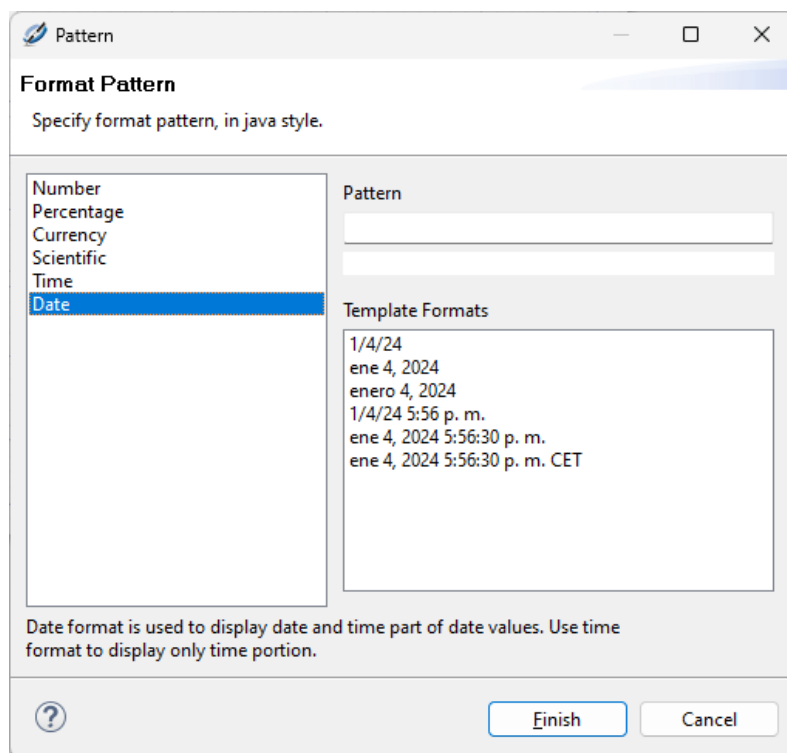
El proceso de añadido de campos al informe es muy simple y se haría en este supuesto en la banda Detail 1. Puesto que va a mostrarlo en forma de tabla, interesa estrechar esta sección para que no haya mucho margen entre un registro y otro. De igual modo, habría que hacer lo mismo con Column Header ya que, al añadir los campos, se agregarán también las etiquetas de columna. Para esto último, basta con arrastrar y soltar cada campo requerido en la vista de detalle. En este ejemplo, serán LASTNAME, FIRSTNAME y CITY. Así mismo, se pueden modificar las etiquetas de columna para que sean más descriptivas además de escribirlas en castellano. Podría quedar de un modo similar a este:



8. Diseño de informe básico de clientes

Para ver cómo queda, se hace clic en la pestaña Preview debajo del editor.

Para modificar el formato de un campo, se selecciona, se hace clic con el botón derecho y se elige **Show Properties** (normalmente el sistema ya las muestra con solo seleccionar el elemento). Aunque no se aplica a este ejemplo, es interesante ir a la pestaña Text Field y hacer clic sobre el botón al lado de **Pattern** para comprobar que se puede cambiar el patrón de presentación del texto, por ejemplo, a distintos tipos numéricos o de fecha.



9. Propiedades de formato

6.3.1. ENCABEZADOS Y PIES

Encabezados y pies son elementos estructurales de un informe que pueden estar asociados a distintos ámbitos. Por ejemplo, se puede tener un encabezado de informe, de página o de columnas como ya se vio en la estructura general. Los elementos complementarios de los encabezados son los pies y de igual modo, están asociados a los mismos elementos que los primeros.

Para añadir la página X de Y en el pie de página de un informe, sólo hay que arrastrar la herramienta **Page X of Y** de la paleta **Composite Elements** en la banda PAGE FOOTER. Como se escribe Page en inglés, se edita la etiqueta sustituyendo la palabra "Page" por "Página". Esta herramienta crea dos campos de texto que muestran la misma variable: PAGE_NUMBER. El campo de texto primero muestra la página actual, el segundo el total de páginas del informe. Esto es posible porque el tiempo de evaluación de cada campo de texto es diferente, en particular, el primer campo de texto tiene el tiempo de evaluación establecido a Now por lo que PAGE_NUMBER contiene el valor de la página actual, el segundo lo tiene establecido a Report (en ese momento de evaluación, JasperReports ha llegado al final del informe, por lo que PAGE_NUMBER contiene el número de la última página).

El tiempo de evaluación de un campo de texto es muy importante porque permite imprimir el valor asumido por una variable en diferentes momentos. Con esta idea, se puede poner la suma total de pedidos en la banda de título y obtener el valor correcto estableciendo el tiempo de evaluación de ese campo de texto a Report (esto se hace automáticamente por JasperReports cuando un campo se arrastra en el título y el usuario elige para mostrar el resultado de una función de agregación).

De igual manera se pueden añadir etiquetas, imágenes a la banda PAGE HEADER para modificar el encabezado de las páginas del informe; por ejemplo, se puede añadir una etiqueta en la que escribir el nombre de la empresa y arrastrar la fecha (Current date) del apartado Composite Elements de la paleta. Cuando se coloca el campo de fecha, el sistema preguntará el formato en el que se desea que aparezca.

Por último, es muy habitual que un informe contenga información de resumen que sirva para aglutinar datos propios de la totalidad del documento como el número de registros, totales generales u otros datos estadísticos como gráficos.

6.3.2. FORMATOS DE SALIDA

Los formatos de salida de un informe también cobran importancia especialmente a la hora de utilizar dicho artefacto en una aplicación. Generalmente, los informes no solo pueden visualizarse en un elemento contenedor dentro de la aplicación (visor en aplicación de escritorio, web, móvil, etc.) sino que también deberían poder exportarse a formatos bien conocidos como PDF, texto enriquecido, texto plano, HTML, etc. También formatos adecuados para hojas de cálculo y procesadores de texto.

Otro aspecto importante en lo que se refiere a los formatos de salida es el tamaño del papel en el que se va a imprimir el informe, en caso de que así se requiera. Habitualmente, su diseño va asociado a un tamaño de hoja concreto, aunque luego pueda modificarse en la impresión. Esto implica que el informe estará optimizado para dicho tamaño.

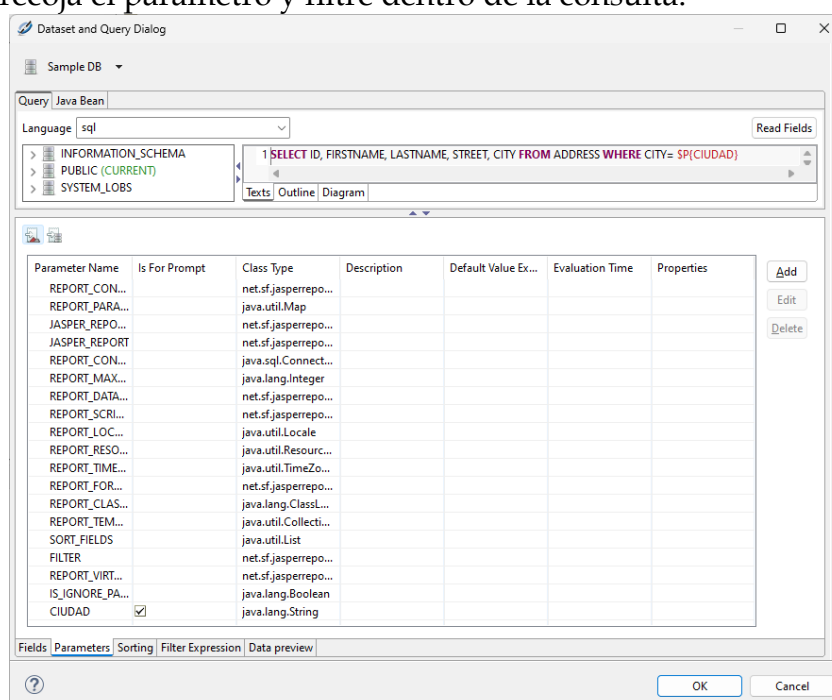
6.4. FILTRADO DE DATOS

Normalmente, los informes se basan en uno o varios criterios proporcionados al lanzar el comando que permite la generación de dicho reporte. Estos criterios pueden venir dados por parámetros, que son elementos variables que pueden servir en la construcción del conjunto de datos propio del informe.

Los parámetros, por tanto, servirán para proporcionar elementos de filtrado dados por el usuario, sea de forma directa, con un prompt en la visualización de un informe ad-hoc, o indirecta, mediante la llamada al informe desde una aplicación de escritorio, web o móvil proporcionando el valor de los parámetros requeridos.

Por ejemplo, si se vuelve al informe que muestra el listado de clientes, se puede incluir un filtrado por ciudad. Para ello, dentro de JasperReports, en la sección **Parameters** (en Outline) se hace clic sobre **Create Parameter** y el generador de informes permitirá dar un nombre al parámetro (fundamental en su uso posterior) una clase Java (por ejemplo, java.lang.String) para definir su tipo y si se quiere preguntar por el parámetro en la llamada al informe mediante una casilla de verificación **Is For Prompting**. En este ejemplo, se le daría el nombre CIUDAD de tipo String y con el prompting activado.

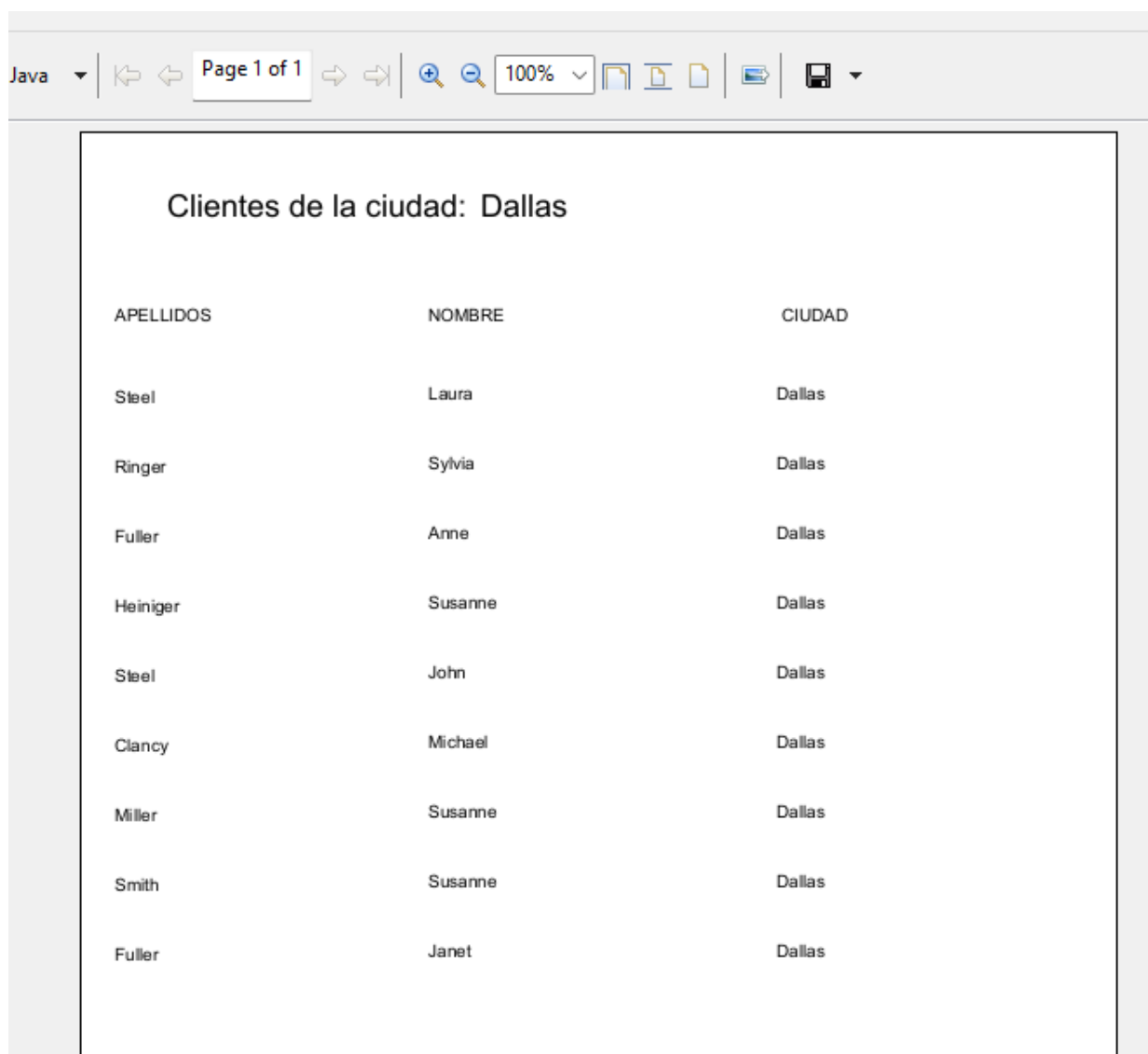
Una vez creado, el parámetro se puede introducir tal cual en el informe al igual que el resto de los que tiene el generador de informes ya integrados y que se verán posteriormente. Por ejemplo, podría ser interesante introducir el parámetro en el título para indicar claramente el criterio de filtrado. Obviamente, el parámetro debe cumplir con su objetivo que no es otro que filtrar la consulta de clientes. Para ello, hay que hacer un retoque al origen de datos e introducir un WHERE que recoja el parámetro y filtre dentro de la consulta.



10. Filtrado con parámetro

Tal como se puede ver en la imagen, ahora la consulta incluye el filtro WHERE y el parámetro se ha introducido arrastrando y soltando desde la pestaña **Parameters** que aparece en la parte inferior y que refleja el elemento recién creado. Como puede comprobarse, JasperReports tiene una nomenclatura propia para sus elementos variables; si los campos se etiquetan como `$F{Campo}`, los parámetros se identificarían con `$P{Parametro}` tal como puede comprobarse con `$P{CIUDAD}`.

Si ahora se lanza la vista previa del informe, lo primero que hará será pedir una ciudad al usuario y, una vez obtenida, expondrá el informe con el filtrado correspondiente. En este ejemplo, se puede probar con la ciudad de Dallas.



APELLIDOS	NOMBRE	CIUDAD
Steel	Laura	Dallas
Ringer	Sylvia	Dallas
Fuller	Anne	Dallas
Heiniger	Susanne	Dallas
Steel	John	Dallas
Clancy	Michael	Dallas
Miller	Susanne	Dallas
Smith	Susanne	Dallas
Fuller	Janet	Dallas

11. Clientes de la ciudad de Dallas

Otro caso de uso se basa en construcciones como

```
SELECT * FROM ORDERS ORDER BY $P!{campos}
```

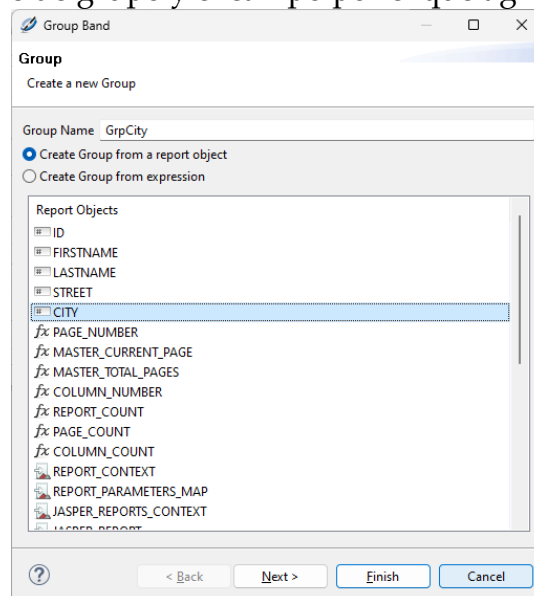
El parámetro será tratado como un campo de SQL. JasperReports lo tendrá en cuenta como una especie de marcador de posición (hay que tener en cuenta la sintaxis especial de `$P!{}`), que será reemplazado por el valor de texto del parámetro (que en este caso puede ser, por ejemplo, "OrderDate DESC"). Con la misma lógica, una consulta puede pasarse íntegramente mediante un parámetro del siguiente modo: `$P!{mi_consulta}`. El número de parámetros en una consulta es arbitraria. Al pasar un valor utilizando la sintaxis `$P!{}`, el valor del parámetro se toma tal cual, el usuario es responsable de la exactitud del valor pasado: la resolución de la sentencia SQL no se realiza por JasperReports en este caso.

De todas maneras, esta opción no es nada conveniente por ser insegura. Si cualquier consulta se ejecuta como parámetro, podría lanzarse una que comprometiera la base de datos o permitiera obtener información no autorizada, es decir, lo que se conoce como inyección SQL.

6.5. INFORMES CON AGRUPAMIENTO, RECuentOS PARCIALES Y SUBTOTALES

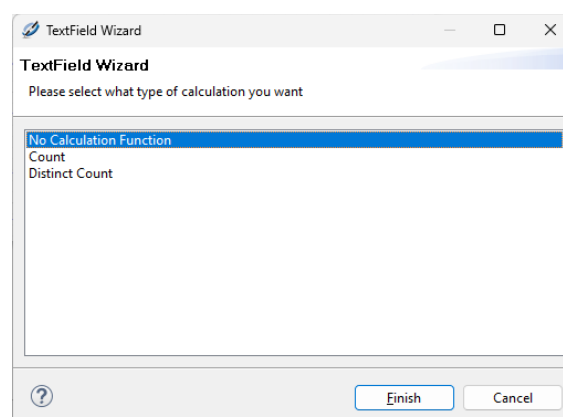
Entre las funcionalidades más importantes que puede aportar un generador de informes está el agrupamiento. Mediante este mecanismo, se puede obtener información de resumen muy útil para el usuario final que aporte principalmente resúmenes concretos sobre datos numéricos, por ejemplo, facturación anual, por puntos de venta, gasto medio, etc.

En el informe de clientes, supóngase que se quiere agrupar por ciudad. En JasperReports, lo primero que hay que hacer es elegir la opción **Create Group** en el menú contextual del informe. El sistema pedirá un nombre de grupo y el campo por el que agrupar (en este caso, CITY).



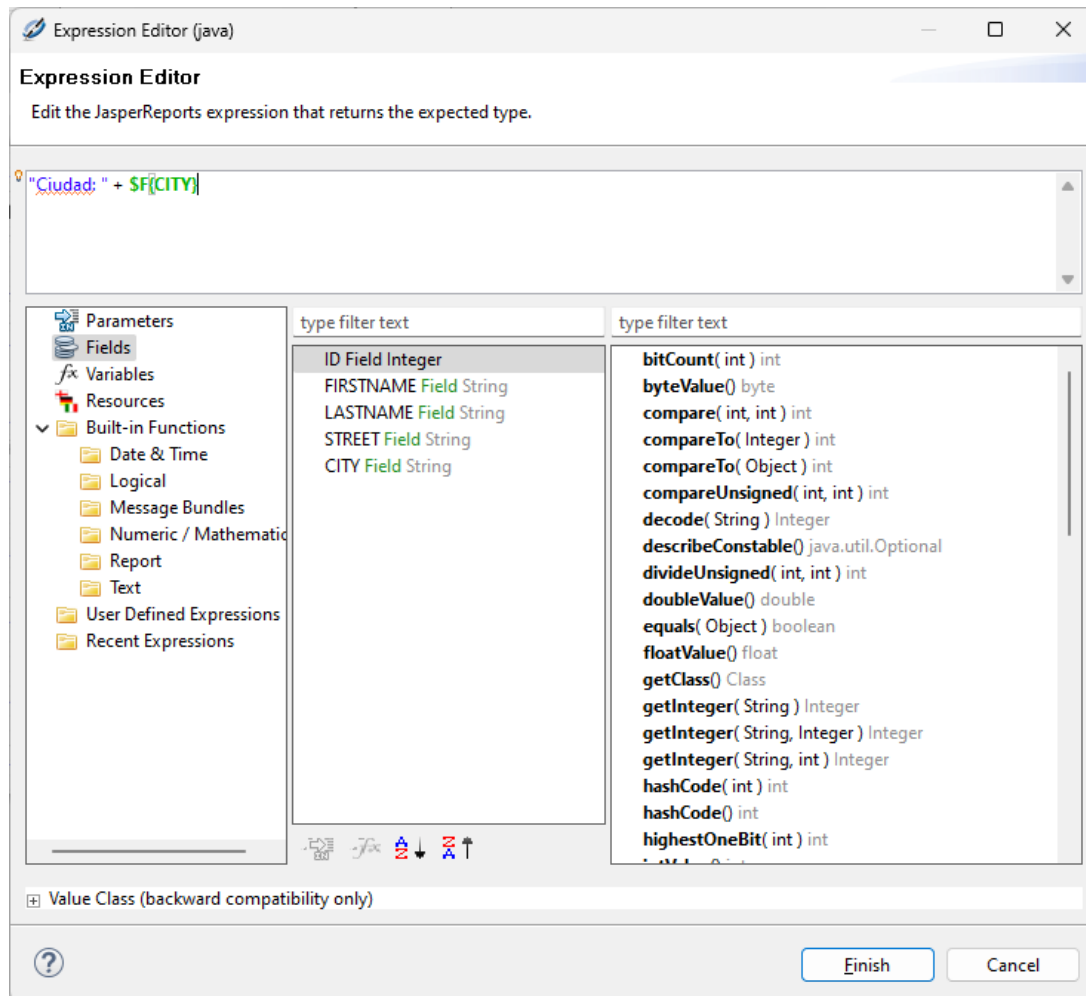
12. Creación de grupo

En la siguiente opción pregunta si se deben crear bandas tanto de encabezado de grupo como pie. Es conveniente dejar ambas opciones marcadas porque son muy útiles ya que se puede incluir información sobre el grupo en el que están los datos. Por ejemplo, si se desea introducir la ciudad en el encabezado de grupo, primeramente, hay que arrastrar el campo CITY a la banda **Group Header**. Al hacerlo, un asistente pedirá cómo se introduce dicha expresión, es decir, si lo hace sin cálculos o si hace falta una información de resumen como puede ser un conteo de registros.



13. Asistente de cuadro de texto en agrupación

En este caso se introduce sin cálculos, pero, además se va a mejorar la expresión como tal, ya que incluye solo la ciudad y en este caso puede interesar aportar más texto informativo. Por tanto, se acude a las propiedades del cuadro de texto y en la pestaña **Text Field**, al lado de **Expression** aparece un pequeño icono el cual lanza el editor de expresiones. Dentro de él aparece el campo tal cual, pero como puede comprobarse, es posible introducir expresiones más complejas que en realidad no dejan de ser escritas en el lenguaje nativo de JasperReports que no es otro que Java. En este caso, se añadirá un texto que indique que el campo se refiere a la ciudad:

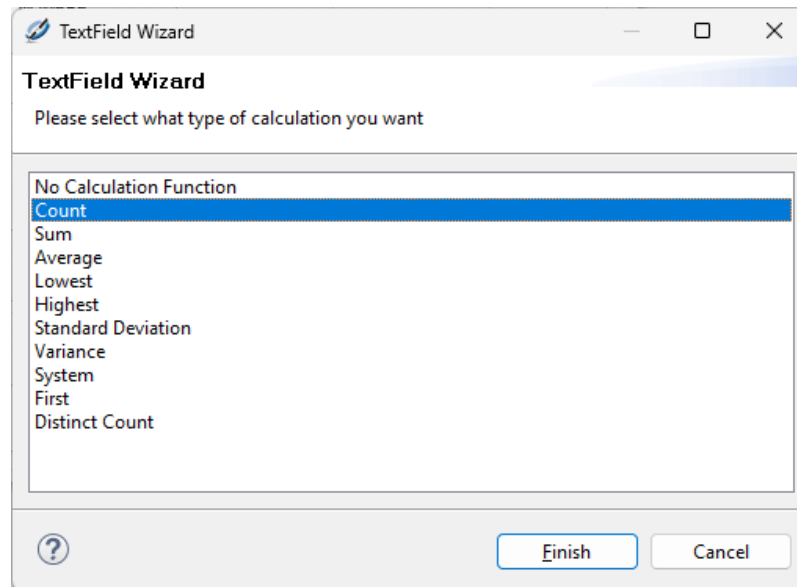


14. Editor de expresiones

Pero si ahora se lanzara la vista previa del informe, el resultado no sería el deseado ya que en la consulta los datos no están ordenados por ciudad, con lo que construye distintos grupos con la misma ciudad. Evidentemente, la solución es tan simple como ordenar por ciudad en la consulta origen: **SELECT** ID, FIRSTNAME, LASTNAME, STREET, CITY **FROM** ADDRESS **ORDER BY** CITY

En cualquier caso, rara vez se hace un agrupamiento de forma aislada sin requerir otros datos. Por ejemplo, podría ser interesante para este caso obtener los recuentos parciales por grupo, es decir, cuántos clientes hay de cada ciudad. De momento, para esa operación ya se tiene una ubicación ideal: el pie del grupo.

Si ahora se arrastra un campo a ese pie de grupo, vuelve a emerger un asistente que esta vez permite ajustar qué función de agregación usar sobre el campo. Para el ejemplo actual, lo que interesa contar son los IDs por ciudad, por lo que se arrastra este campo y como función de agregación se usa Count.



15. Asistente de cálculos en Text Field

Al finalizar, en el pie de grupo se encuentra un nuevo elemento que se conoce como variable y que viene etiquetado como \$V[ID1]. Además, si se consulta el árbol Outline, en la sección variables aparecerá también ID1. También aquí se puede editar la expresión para que sea más amigable y que incluya un texto descriptivo: `"Clientes: " + $V{ID1}`

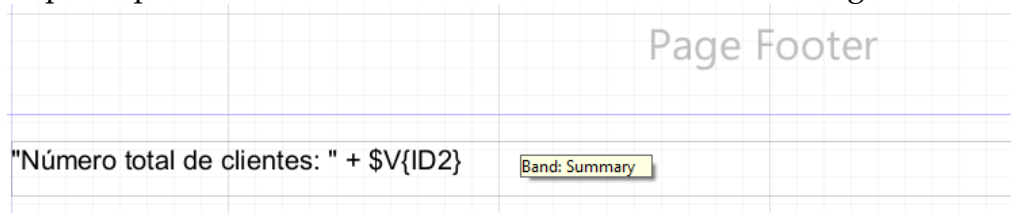
Hay que tener presente que la eliminación de una variable para dejar de usarla definitivamente debe hacerse en el árbol Outline y no dentro del informe ya que, dentro de este, lo que aparece son "instancias" de las variables u otros elementos mientras que en el árbol aparece el elemento en sí mismo.

6.6. NUMERACIÓN DE LÍNEAS, RECuentOS Y TOTALES

Un recuento consiste en aplicar la función de resumen suma a un campo concreto de una consulta. Son útiles para calcular totales y subtotales. En el apartado anterior ya se vieron formas de recuento parcial por grupo, pero no siempre es necesaria la agrupación de datos o, en caso de hacerse, también se necesita disponer de un resumen a nivel de documento que muestre un recuento total o una suma de importes.

A veces puede ser necesario incluir el número de líneas en un informe. Para conseguirlo, se usará una variable creada por el usuario, por ejemplo, una llamada NLineas. Para conseguir que cuente el número de línea para cada registro se modificarán las propiedades de la variable para hacer que sea de tipo entero (`java.lang.Integer`), en Calculation, se indicará que no haga nada (`nothing`), se iniciará la variable a 1 (`Init Value Expression`) y se asignará la expresión `$V{REPORT_COUNT}` (`Value Expression`). Para visualizar los números de línea se arrastra la variable a la banda de detalle.

Volviendo al informe anterior, ahora se puede desear conocer el número total de clientes independientemente de su ciudad. Para este cálculo, vuelve a hacerse uso de variables en JasperReports y de una sección ideal para estos elementos: Summary. La creación de una variable de este tipo es sencilla, basta con agregar a dicha sección el campo sobre el que se desea obtener un total; en este caso, el ID de cliente. La función de agregación en este caso es Count, ya que se desea contar el número de clientes. A partir de ese momento, se crea una variable ID2 que se puede incluir “tuneada” en el informe del modo siguiente:



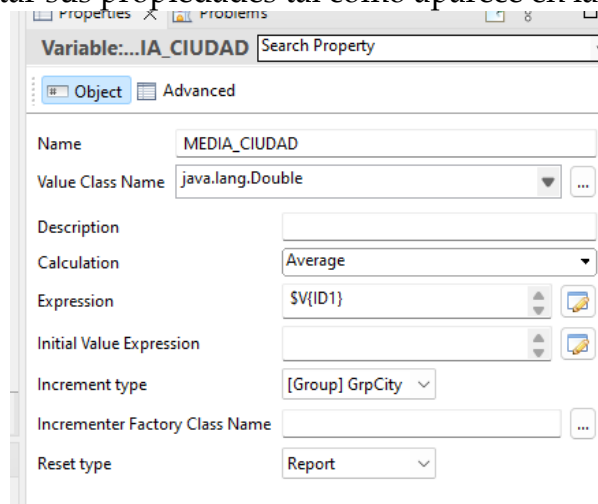
16. Total de clientes en informe

6.7. VALORES CALCULADOS

Existen varias formas de insertar en un informe valores calculados a partir de su información. Una forma muy simple y básica pasa por crear el campo calculado en la consulta. De esto se verá un ejemplo en la parte de subinformes, aunque consiste en crear un campo que dependa de otros y darle un alias para su posterior inclusión, todo ello en la consulta base del informe.

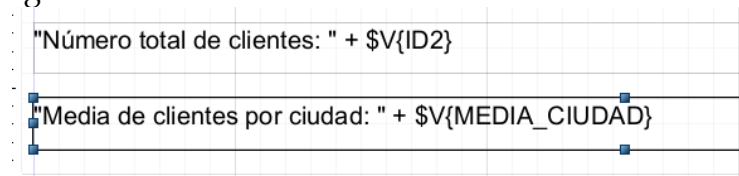
También se pueden utilizar las variables para crear este tipo de valores calculados. Por ejemplo, si en el informe de clientes se desea conocer cuál es la media de clientes por ciudad, podría utilizarse una variable que realizara dicho cálculo.

A continuación, se muestra cómo se debe crear dicha variable. Por supuesto, lo primero que hay que hacer es utilizar **Create Variable** dentro de su apartado en el árbol **Outline**, llamarla **MEDIA_CIUDAD** y ajustar sus propiedades tal como aparece en la imagen.



17. Variable de media por ciudad

Es muy fácil interpretar las distintas propiedades de la variable. El tipo de datos requerido es **Double** ya que va a contener números decimales al ser una media. Como función de cálculo, **Average**, la cual es promedio en inglés. La expresión sobre la que se calcula es la variable ID1 que ya se creó anteriormente. El tipo de incremento es el que se usará para calcular la media; dado que en este caso se trata de una cantidad que aparece referida a cada grupo, se selecciona dicho agrupamiento y, por último, el **Reset type**, que es el momento en que esta variable se va a reinicializar, que en este caso es Report porque se refiere a todo el informe. Si se vuelve a evaluar la utilizada anteriormente para calcular los clientes por ciudad, se verá que la variable se reinicializa por grupo, y es lo lógico teniendo en cuenta que va variando en función de cada agrupamiento. Por último, se inserta en la sección de resumen, aunque con un texto ya preparado para ser legible:



18. Media por ciudad en el informe

Existen variables predefinidas que sirven para hacer recuentos de elementos propios del informe, como `PAGE_NUMBER` que contiene el número total de páginas, o el `REPORT_COUNT` que tiene el número de registros procesados en el momento. Este tipo de variables no pueden ser modificadas o eliminadas.

6.8. SUBINFORMES

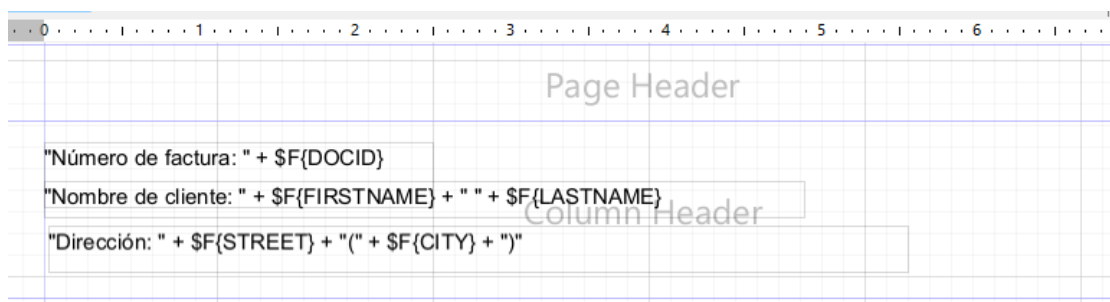
Los subinformes no dejan de ser informes dentro de otros informes. Son muy útiles en vistas de tipo maestro – detalle; por ejemplo, una factura en la que los datos maestros son los principales (fecha, cliente, dirección, número de factura) y el detalle son los productos que forman parte de ella con sus cantidades e importes.

Precisamente, este ejemplo de maestro – detalle con facturas, va a ilustrar la forma de usar los subinformes en JasperReports. Si se visualiza el modelo de datos de la base de datos de muestra, se puede deducir que la factura en sí misma aparecería como DOCUMENT y que POSITIONS sería el detalle de dicha factura. Así mismo, se contiene una referencia al cliente que está en la tabla ADDRESS.

En primer lugar, se va a crear el informe base de la factura y su diseño contendrá los datos básicos de factura y cliente en el encabezado de columna y el subinforme en vista de detalle. El informe contendrá un parámetro IDFACTURA que recogerá su número para filtrar por la que se necesite consultar no olvidando que en este caso se trata de un tipo entero. La consulta base sobre la que trabajará el informe es la siguiente:

```
SELECT D.ID AS DOCID, A.FIRSTNAME, A.LASTNAME, A.STREET, A.CITY
FROM DOCUMENT D
INNER JOIN ADDRESS A ON D.ADDRESSID = A.ID
WHERE D.ID=$P{IDFACTURA}
```

Como se ve, se trabaja sobre el documento con una unión a la tabla de clientes y recogiendo el IDFACTURA como criterio de filtrado. En cuanto al diseño, podría quedar así:



19. Encabezado de factura

Como puede verse, se hace uso de las expresiones para formatear de forma más adecuada los campos.

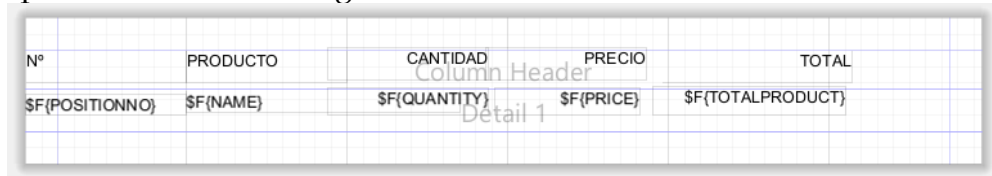
El subinforme se agrega desde el panel **Basic Elements** y se hace desde la vista de detalle. En cuanto se inserta, se lanza un asistente que puede resultar más complejo que la simple configuración del informe. Lo único que se ajustará en ese caso será el nombre del subinforme ya que será un fichero independiente dejando vacía toda la parte de enlace a datos.

Ahora es turno de diseñar este nuevo informe. También en él se ajustará un parámetro que debería llamarse igual que en el informe principal además de ser del mismo tipo y que, como no, vendrá denominado como IDFACTURA.

En este subinforme, la consulta subyacente será la siguiente:

```
SELECT POSITIONNO, PR.NAME, QUANTITY, PRICE, QUANTITY*PRICE AS TOTALPRODUCT
FROM POSITIONS PS
INNER JOIN PRODUCT PR ON PS.PRODUCTID = PR.ID
WHERE DOCUMENTID = ${IDFACTURA}
```

Se ha aprovechado la consulta para obtener un campo calculado directamente desde la base de datos, que en este caso es el importe total por cada producto (TOTALPRODUCT). En cuanto a su diseño, podría ser similar al siguiente:

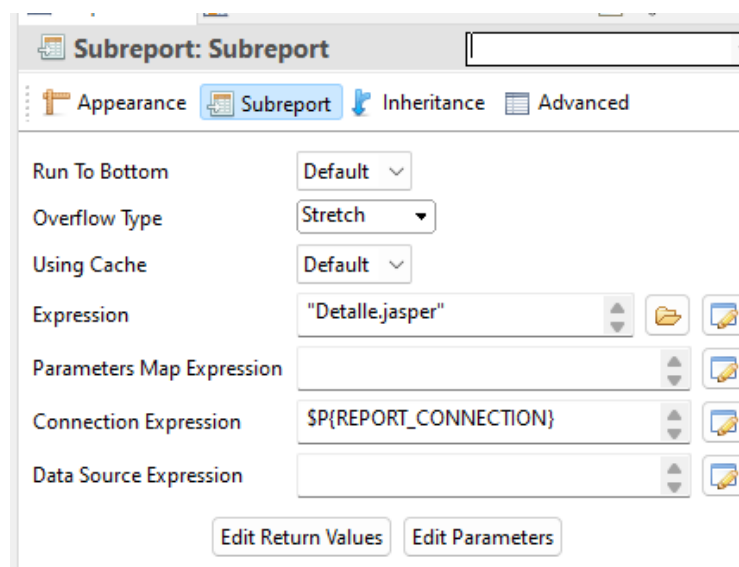


N°	PRODUCTO	CANTIDAD	PRECIO	TOTAL
\$F{POSITIONNO}	\$F{NAME}	\$F{QUANTITY}	\$F{PRICE}	\$F{TOTALPRODUCT}

20. Subinforme con líneas de factura

Como puede verse, se han dejado solo las dos bandas necesarias, aunque según necesidades, podrían requerirse las otras. En ese caso, simplemente bastaría con volverlas a añadir. También hay que tener muy en cuenta los formatos numéricos para cantidad, precio y precio total y su alineación de forma que el informe final quede lo más profesional posible.

El mapeo del parámetro se hace acudiendo a las propiedades del subinforme en el informe principal y editando sus parámetros en el botón que se muestra a continuación (opción **Edit Parameters**):



21. Propiedades de subinforme

Ahora se muestra un diálogo que espera el nombre del parámetro (IDFACTURA) y con qué expresión se corresponde. Si se abre el editor de expresiones, hay que tomar el DOCID, que es el número de la factura en el informe principal para que así sirva de filtrado a las líneas de detalle del subinforme.

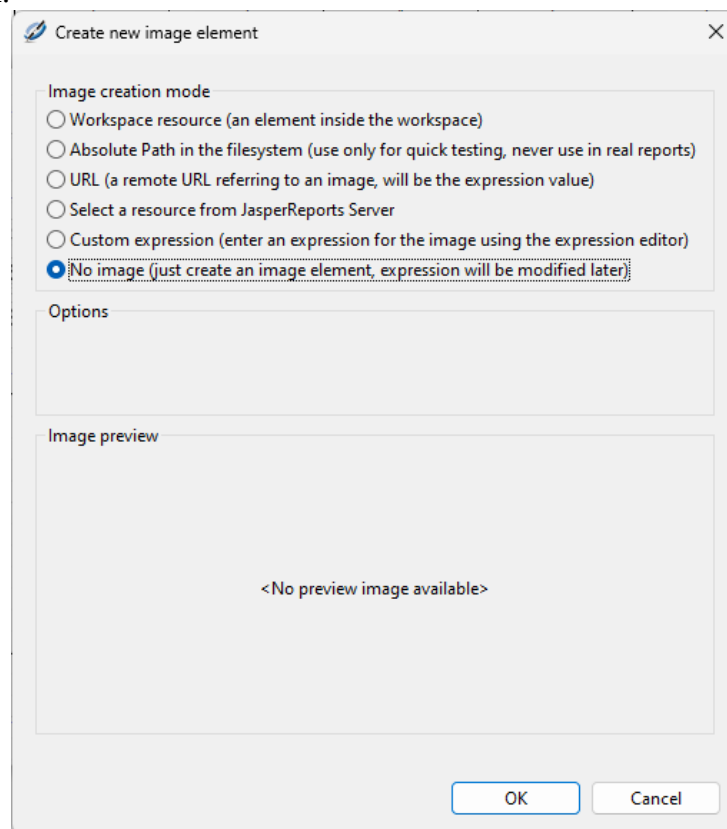
Una vez diseñado, se lanza la vista previa dando un número de factura y el resultado sería más o menos como el de la siguiente imagen.

Java	Page 1 of 1	100%						
FACTURA								
Número de factura: 9								
Nombre de cliente: Andrew Heiniger								
Dirección: 347 College Av.(Lyon)								
Nº	PRODUCTO	CANTIDAD	PRECIO	TOTAL				
0	Iron Ice Tea	8	€27,00	€216,00				
1	Chair Clock	21	€25,80	€541,80				
2	Chair Telephone	12	€16,80	€201,60				
3	Ice Tea Shoe	14	€19,20	€268,80				
4	Ice Tea Iron	16	€4,80	€76,80				
5	Ice Tea Chair	2	€14,70	€29,40				
6	Chair Clock	10	€25,80	€258,00				
7	Telephone Clock	22	€37,20	€818,40				
8	Chair Shoe	11	€10,80	€118,80				

6.9. IMÁGENES. GRÁFICOS

Un generador de informes alcanza sus mayores cotas de potencia y oferta de funcionalidades cuando muestra información lo más visual posible y que implica una buena cantidad de cálculos.

En cuanto a la parte visual, además de poder modificar tipos de letra y formatos de visualización del texto, también es posible insertar imágenes dentro de un informe. Por ejemplo, JasperReports consta de un control llamado Image que se puede insertar en el diseño de un informe. Los distintos orígenes que puede tomar esta imagen se ven en el asistente que se lanza al insertarla:



22. Asistente de creación de imágenes en JasperReports

En definitiva, se puede crear una imagen desde:

- Un recurso dentro del propio espacio de trabajo. En este caso, se puede añadir un fichero de imagen al espacio de trabajo actual e insertarla en el informe cuando se desee.
- Ruta absoluta del sistema de ficheros. Esto solo sirve si se está probando, ya que hay que recordar que se pretende que el informe se inserte en otras aplicaciones o sistemas, con lo que una ruta absoluta no serviría de nada en ese caso.
- URL remota con la imagen. En este caso, dicha URL se incluirá en la expresión subyacente
- Recurso de JasperReports Server. JasperReports, al igual que Reporting Services, también tiene la posibilidad de tener un servidor de informes para que pueda ser usado por distintos sistemas independientemente de su arquitectura. En este caso, podría insertarse el recurso en él y luego tomarlo para su incrustación en informes.

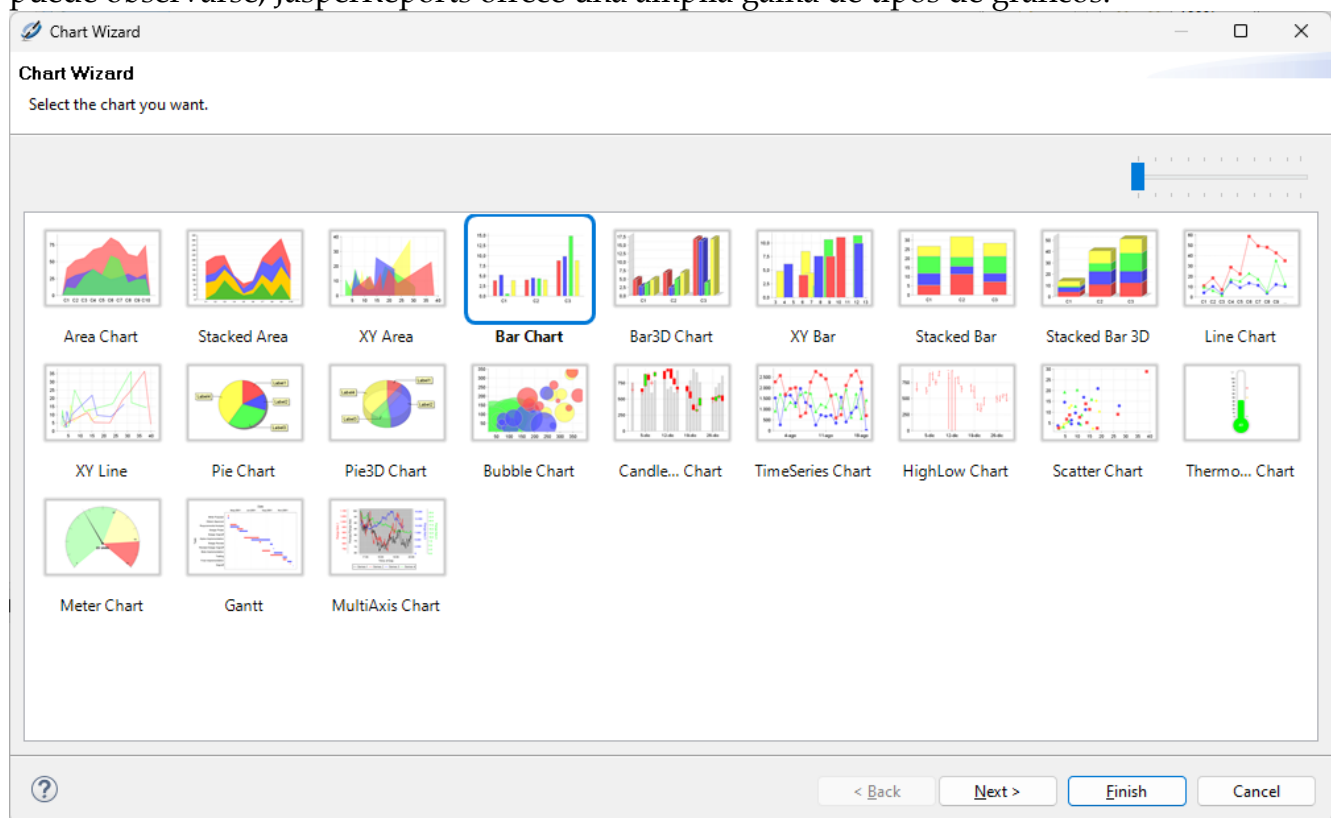
- Expresión personalizada. En este caso, en el diseño se introduce una expresión que permita la carga de dicha imagen.

La última opción simplemente inserta el elemento dejando que posteriormente se decida cuál será su origen.

La primera opción tiene una fácil implementación. Primero se añade un fichero al proyecto, por lo que hay que acudir a su menú contextual y elegir New / File. Se abre un cuadro de diálogo que pide el nombre del fichero, pero como no se conoce la ruta completa, lo mejor es abrir las opciones avanzadas, marcar **Link to file in the file system** y con **Browse** elegir el fichero del disco duro. A partir de ese momento, el sistema incluirá en el Workspace la imagen y ahora se podrá introducir en el informe con la primera opción ya que el diseñador la reconocerá.

En cuanto a la información que pudiera servir para estadísticas, ya se ha visto cómo agrupar y calcular en base al agrupamiento, pero faltaría un elemento clave en cualquier informe empresarial o estadístico: los gráficos. Sobra decir que un buen generador de informes debe forzosamente incluir la posibilidad de incluir gráficos estadísticos en sus diseños.

Volviendo al informe que mostraba los clientes por ciudad, en este caso se introducirá un histograma o informe de barras que muestre el número de clientes por ciudad, pero como puede observarse, JasperReports ofrece una amplia gama de tipos de gráficos.

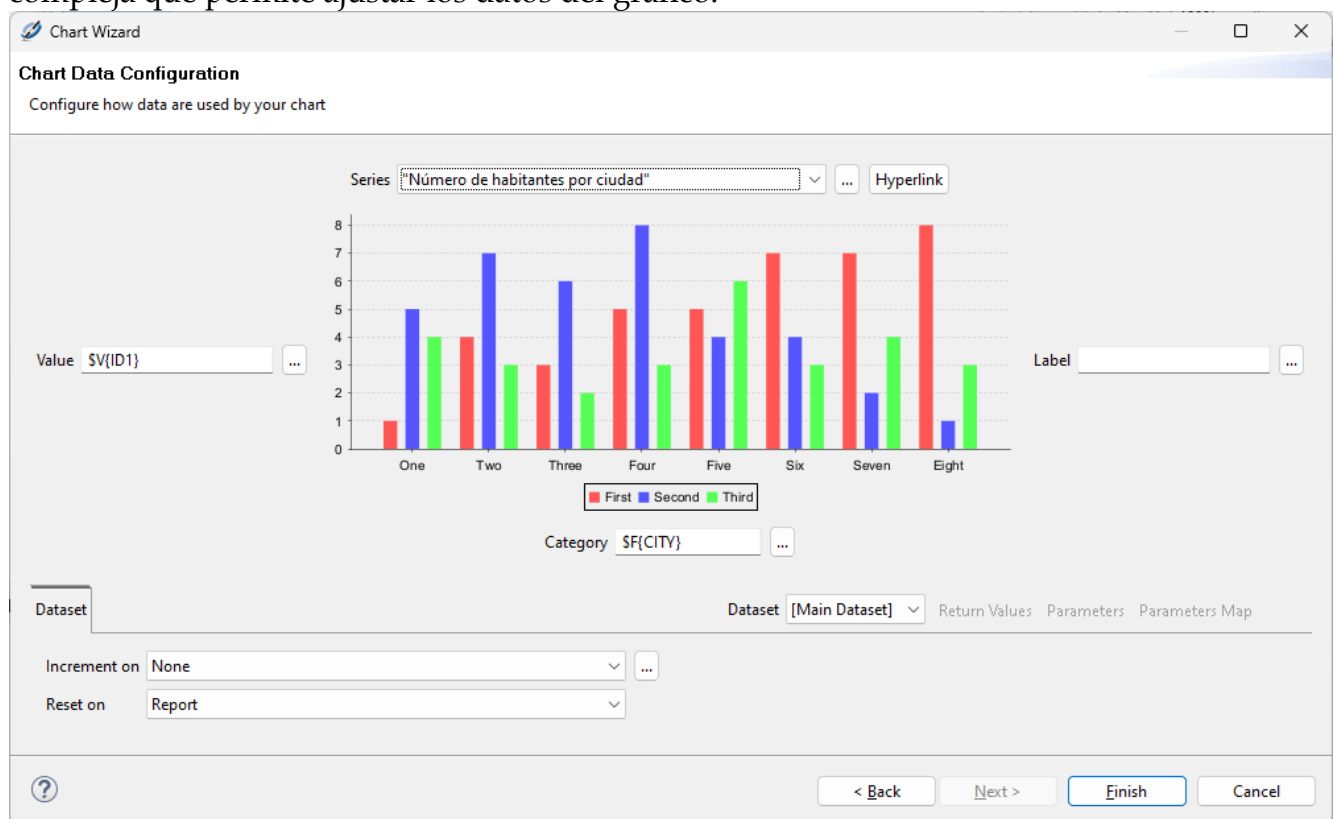


23. Tipos de gráficos en JasperReports

Cuando se trabaja con gráficos conviene aclarar los conceptos de serie y categoría:

- **Serie:** Es el conjunto de datos numéricos a representar. Se puede tener más de una serie en un gráfico, salvo en los circulares que sólo tienen una. A cada serie se le asigna un color diferente o algún otro identificativo que aparece claramente señalado en la leyenda del gráfico. Cada dato de la serie toma valores en un rango, por lo que si se quiere tener más de una serie en el gráfico es conveniente que todas estén dentro del mismo rango.
- **Categoría:** Se corresponde con los datos a representar dentro del eje horizontal de gráfico. Cada dato de la serie toma valores para un dato de la categoría.

La ubicación ideal en este informe de ejemplo para este gráfico sería la sección Summary ya que no dejan de ser unos datos de resumen. Al continuar el asistente, se entra en una zona más compleja que permite ajustar los datos del gráfico.



24. Asistente de gráficos

Cualquier generador de informes va a poder realizar histogramas de varias series, pero en este caso solo se necesitará una, que se llamará "Número de habitantes por ciudad". En cuanto a la categoría, se utiliza el campo CITY ya que se cuenta por ciudades; en lo que se refiere al valor, va a servir la variable anterior que se utilizó para contar los clientes por ciudad y que se identificó como ID1. Para que se puedan visualizar correctamente las ciudades, ahora quedaría agregar un ángulo de rotación al texto de las etiquetas, que podría ser de 75°. Esto puede hacerse en las propiedades del gráfico dentro de la pestaña **Chart Plot** en **Category Axis Tick Label Rotation** ajustando su valor a 75.

El gráfico quedará de esta forma:



25. Histograma de clientes por ciudad

6.10. BIBLIOTECAS PARA GENERACIÓN DE INFORMES

6.10.1. CLASES, MÉTODOS, ATRIBUTOS

Una vez que se ha visto cómo diseñar informes y sus elementos más importantes, los siguientes pasos han de ir encaminados en la integración de dichos informes en las distintas tecnologías de programación y/o lenguajes.

Por ejemplo, ya que JasperReports está escrito en Java, lo primero que se verá es cómo integrar estos informes en una aplicación desarrollada en ese lenguaje. Para ello, obviamente hay que recurrir a la biblioteca que tiene JasperReports para este tipo de arquitecturas y que se conoce como JasperReports Library. Si se desea utilizar en una aplicación implementada con Maven, la dependencia a agregar se añadiría mediante el siguiente código o similar:

```
<dependency>
  <groupId>net.sf.jasperreports</groupId>
  <artifactId>jasperreports</artifactId>
  <version>6.21.0</version>
</dependency>
```

A partir de aquí, ya se pueden usar las clases propias de la biblioteca, en especial dos: JasperPrint y JasperExportManager, la primera para generar el informe y la segunda para exportarlo a un formato concreto, por ejemplo, PDF.

El informe en sí se prepararía con una orden similar a la siguiente:

```
JasperPrint report = JasperFillManager.fillReport("ruta/informe.jasper", parameters,
connection);
```

El método fillReport rellena el informe que ya debe estar compilado como .jasper con la ruta dada. Parameters sería la colección de parámetros (que se verá en el próximo subapartado) y connection la conexión a la base de datos, la cual también se verá posteriormente. La ruta se debe configurar en formato Linux, es decir src/main/java/nombrepaquete/...

Una vez que se “llena” el informe, se puede exportar a un formato a elegir entre XML, HTML o PDF. Dado que este último es el más funcional y profesional, se lanzará su método correspondiente mediante la siguiente codificación:

```
JasperExportManager.exportReportToPdfFile(report, "informe.pdf");
```

Aquí también se le puede dar una ruta concreta del tipo src/main/...

6.10.2. PARÁMETROS

Los parámetros que pueden conformar un informe deben mandarse como una colección de tipo clave-valor que recoja el nombre del parámetro y su valor. El método **fillReport** espera, por tanto, un objeto de tipo **Map<String, Object>** con la colección de parámetros en forma de diccionario. Si no se requiriera ningún parámetro externo, este atributo podría ir simplemente a null o como un diccionario vacío.

Por ejemplo, para el informe de factura que se diseñó anteriormente, habría que proporcionarle un IDFACTURA y su valor, por ejemplo, el número de factura 1.

```
Map<String, Object> parameters = new HashMap<String, Object>();  
parameters.put("IDFACTURA",1);
```

Como se ve, se construye un objeto parameters como HashMap ya que lo que espera fillReport es una colección que interprete Map y aquella lo hace. Una vez inicializado, se le inserta un elemento en el diccionario y ya quedaría preparado para su ejecución.

6.11. CONEXIÓN CON LAS FUENTES DE DATOS. EJECUCIÓN DE CONSULTAS

En el apartado anterior se vio cómo se podría generar un informe desde una aplicación Java, pero sin entrar en detalles en cómo debería ser la conexión. Bien, se parte de que JasperReports soporta conexiones con distintos gestores de bases de datos e incluso con ficheros .csv. Dado que en el ejemplo se utilizó una conexión de muestra, la aplicación Java debería poder replicar esa base de datos y, aunque parece sencillo a priori, requiere de una serie de pasos muy precisos.

6.11.1. INCLUSIÓN DE INFORME EN APLICACIÓN JAVA

Por de pronto, lo primero que hay que hacer es cargar el driver de conexión a HSQLDB, que es la base de datos subyacente que usa JasperReports para la muestra. Una vez más, se recurre a Maven para cargar la dependencia:

```
<dependency>
  <groupId>org.hsqldb</groupId>
  <artifactId>hsqldb</artifactId>
  <version>2.7.2</version>
</dependency>
```

La base de datos de prueba se puede tratar como fichero. HSQLDB tiene varios modos de trabajo: memoria, servidor web, fichero. En este ejemplo, se usará el último modo, pero para ello hay que acudir a la página oficial de la comunidad de JasperReports y descargarse los ficheros de proyecto. En el siguiente enlace se encuentra un archivo jasperreports-x.xx.x-project.zip que hay que descargar:

<https://community.jaspersoft.com/files/file/20-jasperreports%C2%AE-library-community-edition/>

Dentro de él hay un directorio llamado **demo/hsqldb** que contiene cuatro ficheros. Los que interesa obtener son **test.properties** y **test.script**, los cuales representan la base de datos HSQLDB para poder ser utilizada como fichero. Ambos deben integrarse en el proyecto NetBeans sobre el que se desee visualizar el informe.

Ahora ya se puede establecer la conexión mediante código con las siguientes instrucciones:

```
String url = "jdbc:hsqldb:file:rutacompleta/test";
String user = "SA";
String password = "";
Connection connection = null;
Class.forName("org.hsqldb.jdbc.JDBC4Driver");
connection = DriverManager.getConnection(url, user, password);
```

Al igual que en apartados anteriores, la ruta completa se construye con la forma Linux, es decir, separando los directorios por / (por ejemplo, src/main/java/nombrepaquete/...)

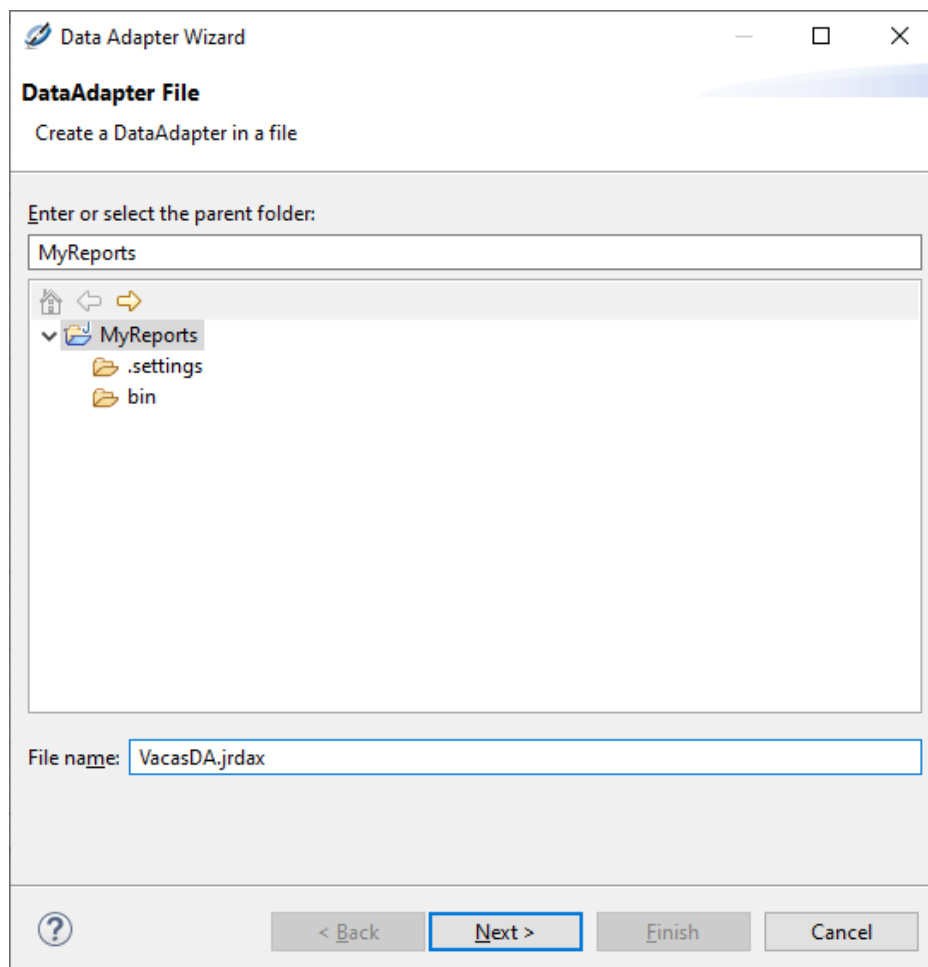
La variable connection es la que se deberá pasar a los métodos de generación y exportación de informes pudiendo ya completarse su obtención completa como PDF.

6.11.2. ARCHIVOS .CSV COMO ORIGEN DE DATOS

Hasta ahora se ha utilizado la base de datos de muestra como origen de datos. Como ya se comentó anteriormente, es posible conectar con distintos orígenes de datos; en este ejemplo se detallará cómo hacerlo con un archivo CSV (Comma Separated Values, Valores Separados por Comas). Para ilustrar este ejemplo, se utilizará el mismo archivo que en la unidad anterior, vacas.csv, el cual se puede encontrar en el siguiente enlace:

<https://github.com/juandbp2dam/VacasWPF/blob/main/VacasWPF/Data/vacas.csv>

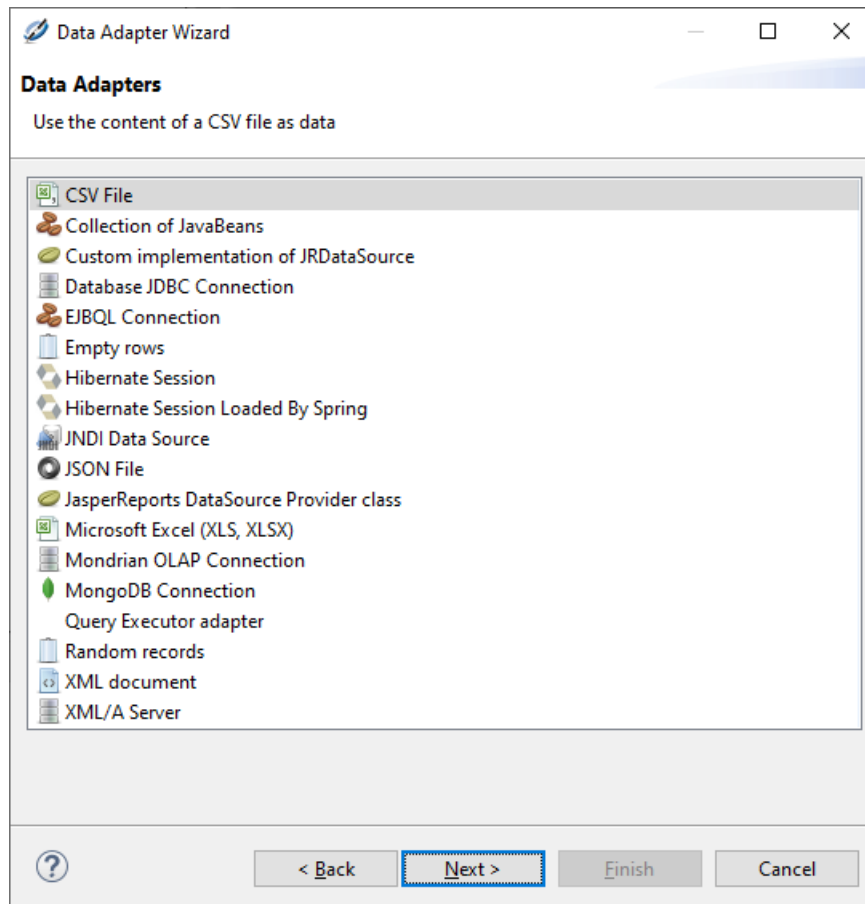
Una vez descargado a una ruta reconocible, el primer paso consiste en crear un Data Adapter que permita conectar con el fichero como fuente de datos. Para ello se acude a **File > New Data Adapter** en Jaspersoft Studio, lo que lanza un asistente que comienza solicitando el nombre del adaptador. Se le puede dar el siguiente nombre: VacasDA.



26. Asistente para creación de DataAdapters

El siguiente paso del asistente muestra los posibles orígenes de datos disponibles. Como puede verse, se pueden crear informes basados en una gran variedad de fuentes, desde un archivo

CSV como en el ejemplo a conexiones JDBC a bases de datos, sesiones Hibernate, archivos JSON, Excel, documentos XML o conexiones con bases de datos NoSQL como MongoDB.



27. Posibles fuentes de datos para informes

Obviamente, para este ejemplo se seleccionará CSV File. El siguiente paso del asistente solicita al fin el fichero sobre el que se basará, el cual puede darse localmente o desde una URL. Para este ejemplo hay que tener en cuenta que el separador no es una coma, sino un punto y coma. Por tanto, una vez seleccionado el fichero, hay que ir a la pestaña **Separators** y elegir **Semicolon**. El resto de los ítems en esta pestaña pueden quedar igual.

Volviendo al apartado principal (Columns), ahora se puede pulsar sobre **Get column names from the first row of the file** para que obtenga automáticamente los nombres de las columnas de la primera línea del fichero. Si está todo bien, se tendría una ventana similar a la de la figura, pero es fundamental que antes de pulsar **Finish**, se marque la opción **Skip the first line (the column names will be read from the first line)** para que excluya de la lista de registros la fila de encabezados. Si no se marca esta opción, es posible que ocurran errores derivados de los tipos de datos en el momento de incluir este origen de datos en el informe, ya que, por ejemplo, si se introduce un campo numérico, mostrará un error ya que el encabezado siempre será texto y, por tanto, incompatible con ese tipo.

Por último, quedaría darle un nombre al DataAdapter para ser utilizado en cualquier proyecto (en este caso, VacasDA).

Name: VacasDA

File/URL: D:\Diego\CodigoNet\VacasWPF\VacasWPF\Data\vacas.csv File

☐ Use query executor mode (the report must use the CSV query language)

Columns Separators

Column names

Get column names from the first row of the file

Column name
id
nomMunicipio
f_nacim
f_destete
alzada
peso
sexo
tipo

Add Delete Up Down

Other

Date pattern : Create

Number pattern : Create

Locale : español (España) Select...

Time zone : Europe/Madrid Select...

☒ Skip the first line (the column names will be read from the first line)

Encoding :

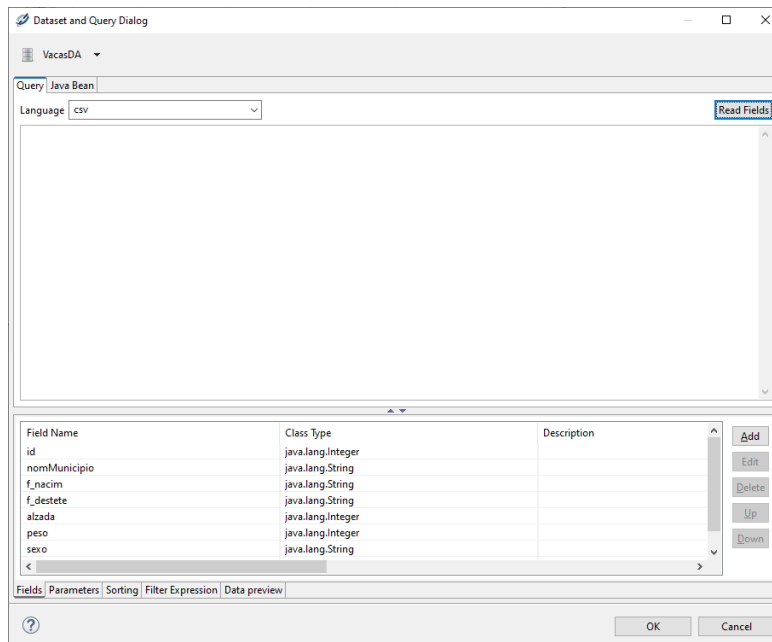
Test Connection

28. Configuración final del DataAdapter

A partir de este momento, ya se podría usar como un origen de datos más en un proyecto. La operativa es muy similar a lo visto anteriormente, aunque con algunos cambios dada la naturaleza de la fuente de datos. Conviene recordar que ya no se está tomando como fuente una base de datos, sino un archivo de texto separado por comas, por lo que, en este caso, no se pueden lanzar consultas SQL sobre él.

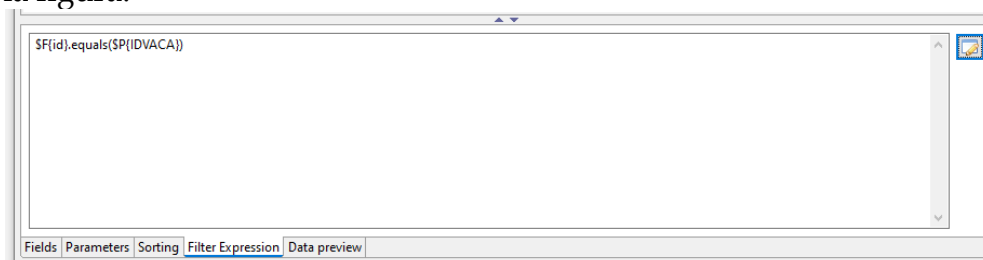
A continuación, se puede crear un nuevo informe llamado Vacas.jrxml sobre este origen de datos. Si se observa la sección Data Adapters en el Repository Explorer, en principio no aparece el recién creado; simplemente, hay que importarlo del Workspace (menú contextual sobre **Data Adapters > Import From Workspace**). A partir de ese momento, ya se puede consultar en el botón **Dataset and Query editor Dialog** el cual se usó anteriormente para obtener datos de la base de muestra.

Ahora simplemente se seleccionaría en el desplegable y como **Language** se escogería csv. Una vez que se pulsa Read Fields, el sistema mostrará todos los campos junto con sus tipos; también sería conveniente ir a Data preview y comprobar que muestra los registros. Si no se marcó la opción Skip first line... el sistema podría dar un error, tal como se indicó anteriormente.



29. Selección del DataAdapter en informe

A partir de este momento, en el informe ya pueden insertarse los campos del CSV dado que aparecen en la sección Fields. ¿Pero cómo se podría filtrar en este caso si no hay consulta SQL? La clave está en las pestañas inferiores (Fields, Parameters, Sorting, etc.) concretamente en Filter Expression. Si se añade un parámetro IDVACA de tipo Integer, se puede filtrar por él simplemente utilizando la expresión adecuada, para lo cual es de ayuda el Expression Editor que aparece en la parte inferior derecha. La expresión necesaria para realizar el filtrado se muestra en la figura.



30. Filtrado por parámetro

Como puede verse, se ha obtenido el campo a filtrar del editor y se le compara mediante el método equals con el parámetro dado. Si ahora se diseña un informe como el de la figura, se podrá obtener una vaca por su identificador.

Page Header			
ID	Municipio	Fecha Nacimiento	Fecha de destete
Column Header			
\$F{id}	\$F{nomMunicipio}	\$F{f_nacim}	\$F{f_destete}
Detail			

31. Informe de vaca por Id

ÍNDICE DE FIGURAS

1. Ejemplo de código fuente de informe JasperReports.....	6
2. Dataset and Query editor dialog.....	8
3. Estructura de bases de datos de muestra.....	8
4. Selección de plantilla de informe.....	8
5. Secciones de un informe vacío.....	9
6. Consulta de clientes.....	9
7. Campos de la base de datos en panel Outline.....	10
8. Diseño de informe básico de clientes.....	10
9. Propiedades de formato.....	11
10. Filtrado con parámetro.....	13
11. Clientes de la ciudad de Dallas.....	14
12. Creación de grupo.....	16
13. Asistente de cuadro de texto en agrupación.....	16
14. Editor de expresiones.....	17
15. Asistente de cálculos en Text Field.....	18
16. Total de clientes en informe.....	19
17. Variable de media por ciudad.....	20
18. Media por ciudad en el informe.....	20
19. Encabezado de factura.....	22
20. Subinforme con líneas de factura.....	23
21. Propiedades de subinforme.....	23
22. Asistente de creación de imágenes en JasperReports.....	25
23. Tipos de gráficos en JasperReports.....	26
24. Asistente de gráficos.....	27
25. Histograma de clientes por ciudad.....	28
26. Asistente para creación de DataAdapters.....	32
27. Posibles fuentes de datos para informes.....	33
28. Configuración final del DataAdapter.....	34
29. Selección del DataAdapter en informe.....	35
30. Filtrado por parámetro.....	35
31. Informe de vaca por Id.....	35

BIBLIOGRAFÍA - WEBGRAFÍA

Blanco, L.M. (2006). *Diseño de informes con SQL Server Reporting Services*. Editorial Netalia S.L.

González P. (2016). *Desarrollo de interfaces. Informes con IReport*.

<https://www.youtube.com/watch?v=bZJQI0Iji-Q&list=PLIfP1vJ2qaklC5L2x78oIxo9Qz1T-PG2U>

Déléchamp, F. (2018). *Desarrolle una aplicación con Java y Eclipse (2ª edición)*. Ediciones Eni.