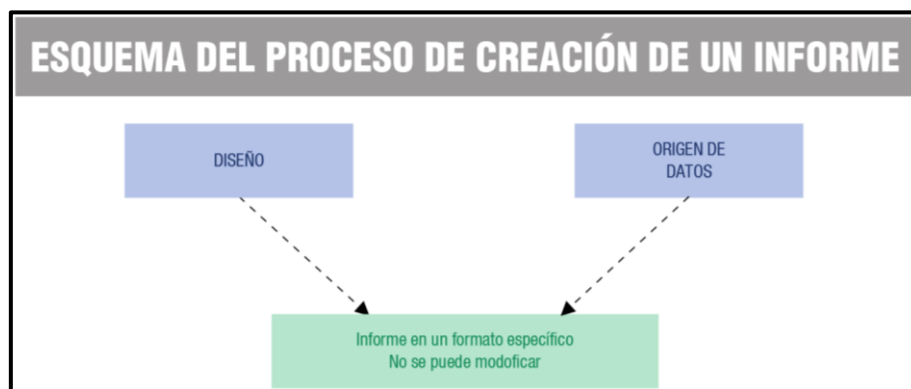


# TEMA 4

INFORMES

# Informes

Un informe es un documento que permite mostrar el contenido de un **origen de datos**<sup>1</sup> aplicando un formato que permita a los usuarios adquirir información. Son un modo eficaz de presentar la información ejerciendo un control bastante preciso sobre cómo se presenta, ya que se permite: ver, formatear y resumir datos relacionados. Por ejemplo, en un informe que muestre como datos la duración media de las estancias en el hotel organizadas por meses, la información deducible por el usuario será los meses en los que cabe esperar una mayor ocupación.



Normalmente, un informe se genera a partir de un **diseño**<sup>2</sup> en el que se determina la distribución y configuración de los elementos a mostrar y que podemos crear utilizando alguna herramienta gráfica. Luego, se combina el diseño con los datos actuales almacenados en el origen de datos volcando el resultado en documentos que faciliten su lectura e interpretación al usuario y posibilitando su impresión y almacenamiento de copias, aunque no se podrán modificar una vez generados. En caso de haber modificaciones en el origen de datos, tendremos que generar de nuevo el informe para reflejar esos cambios.

---

<sup>1</sup> Un origen de datos es una combinación de una fuente de información estructurada y la información de conexión necesaria para obtener acceso a dichos datos. Algunos ejemplos de orígenes de datos son SQL Server, Oracle RDBMS, hojas de cálculo y archivos de texto (plano o enriquecido).

<sup>2</sup> El diseño de un informe determina sus características visuales (que lo hacen amigable) y funcionales (que lo hacen usable).

¿Por qué los informes son un elemento fundamental en cualquier **modelo de negocio**<sup>3</sup>?

- Porque necesitamos distribuir información a otras personas en un formato que puedan entender.
- Porque necesitamos controlar la distribución de la información cuando la imprimamos.
- Porque necesitamos hacer cálculos y mostrarlos de forma legible.
- Porque precisamos obtener información de los datos para poder tomar decisiones.

---

<sup>3</sup> Representación abstracta de una organización, ya sea de manera textual o gráfica, de todos los conceptos relacionados, acuerdos financieros, y el portafolio central de productos o servicios que la organización ofrece y ofrecerá con base en las acciones necesarias para alcanzar las metas y objetivos estratégicos.

# Informes incrustados y no incrustados

La forma de añadir el informe a la aplicación dependerá de cómo se cree el informe. Como hemos visto, es necesario tener la definición del informe y un origen de datos para rellenarlo, sin embargo, podemos crearlo dentro de la aplicación o tenerlo en un archivo independiente e insertarlo después.

- Un **informe incrustado** es un informe que se ha importado al proyecto o que se ha creado directamente en él. Cuando se crea un informe incrustado en una aplicación, se crea una clase contenedora para el informe. Esta clase formará parte del proyecto. Cuando se importa o se crea el informe en el proyecto, se crea una **clase contenedora**<sup>4</sup>, con el mismo nombre que el informe. Esta clase contiene, o representa, el informe en el proyecto. Cuando ocurre esto, todo el código del proyecto interactúa con la clase del informe que se ha creado para representarlo, en vez de hacerlo con el propio archivo de informe original.

Al **compilar**<sup>5</sup> el proyecto, tanto el informe como su clase contenedora se incrustan en el **ensamblado**<sup>6</sup>, lo mismo que ocurriría con cualquier otro recurso del proyecto.

- Un **informe no incrustado** se ha generado con una herramienta específica aparte del proyecto y también se almacena independiente del proyecto. En este caso hay que planificar cómo se va a acceder y cargar el informe para interactuar con él. No existe una clase específica para manejar el informe. A un informe no incrustado siempre se obtiene acceso externamente y el **SDK**<sup>7</sup> puede tener acceso a él de diversas formas:
  - El informe puede estar en la unidad de disco duro en una ruta de directorio de archivos.

---

<sup>4</sup> Una **clase contenedora** representa a un objeto/elemento externo de manera que se integre en el diagrama de clases del proyecto.

<sup>5</sup> Traducir con un compilador un programa en lenguaje de alto nivel a lenguaje de la máquina.

<sup>6</sup> El ensamblado constituye una colección de uno o más archivos o ficheros agrupados juntos para formar una unidad lógica en el conjunto del programa.

<sup>7</sup> Kit de Desarrollo de *Software* (*Software Development Kit*) es un conjunto de herramientas que posibilitan el desarrollo de *software*.

- El informe puede estar expuesto a través de un servicio de informes.

Nunca se importan informes no incrustados en el proyecto y, por lo tanto, nunca se crea ninguna clase contenedora de informe, a diferencia de los informes incrustados. En su lugar, se carga el informe no incrustado en tiempo de ejecución.

# Generación de informes de forma automática: herramientas

Los motores de informes permiten, mediante una interfaz gráfica de usuario determinar la posición, aspecto final, y configuración de los elementos que aparecen en el informe, generando automáticamente los ficheros con el diseño final. Hoy día existe gran cantidad de herramientas creadas para tal fin, tanto libres como propietarias, entre las que destacamos las siguientes:

## **Crystal Reports:**

Es la solución para la generación de informes de la empresa SAP, Crystal Solutions. Esta herramienta propietaria viene integrada en Visual Studio .NET, aunque dispone de SDK específicas 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.

Los informes se almacenan en un archivo de tipo **.rpt**, que contiene información tanto del origen de datos como del diseño. Admite informes incrustados y no incrustados.

### **Para saber más**

Puedes ver más sobre Crystal Reports en su página:

[Página oficial de Crystal Reports](#)

## **JasperReport + iReport:**

Software libre perteneciente a JasperSoft. iReport genera archivos XML (con extension **jrxml**) que contienen el diseño del informe, para generar el informe se compila este archivo en otro de extensión **jasper** y se rellena con el origen de datos. Tiene su propio lenguaje para la definición de expresiones llamado Groovi, aunque es compatible con Basic y Java.

La librería JasperReport permite combinar el diseño con el origen de datos para obtener el documento final mediante código Java.

### Para saber más

Para saber más sobre JasperSoft para iReport consulta la siguiente página:

[Página oficial de iReport Designer](#)

### 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.

### Para saber más

Puedes ver más sobre BIRT en el siguiente enlace.

[Página oficial de Eclipse BIRT](#)

## Jasper Reports + iReports

**iReport** es un diseñador de informes escrito por completo en Java, visual, intuitivo, pero muy potente. Lo distribuye la comunidad JasperForge de forma gratuita bajo licencia **GPL**<sup>8</sup> versión 3. Si quieres conocer los términos para esta licencia puedes hacerlo en su web:

[Página oficial de las licencia GNU GPL](#)

**iReport** dispone de asistentes para la generación de informes, subinformes y plantillas, lo que facilita enormemente el trabajo del desarrollador, sin embargo, también permite la elaboración de informes partiendo desde cero. Sea cual sea el procedimiento elegido, el objeto de usar una herramienta de diseño es reducir el coste de escribir todo el código necesario para obtener el informe, para ello, usaremos la librería de generación de informes **JasperReports**, que es una de las más usada en Java, y constituye el núcleo de iReport. Con esta librería, podremos enviar nuestros informes a un documento de texto, PDF, imagen o impresora. Se integra fácilmente en una aplicación java, aunque iReport se puede utilizar de manera independiente.

En resumen, iReport permite diseñar informes mientras que JasperReports ejecuta y genera los informes en una aplicación Java.

---

<sup>8</sup> General Public License.

## Debes conocer

A continuación tienes un enlace al procedimiento de instalación de JasperReports con iReport en NetBeans 8 (el procedimiento debería de ser análogo para la versión 11):

### [Instalar jasperReport e iReport en NetBeans 8.0](#)

Podéis encontraros con ciertos problemas para desplegar iReport en NetBeans 11, que podréis resolver siguiendo las indicaciones de este foro:

### [Problema con iReport y NetBeans 11](#)

En el siguiente enlace podrás descargar el paquete de instalación de iReport para el sistema operativo escogido:

### [Paquete de instalación de iReport](#)

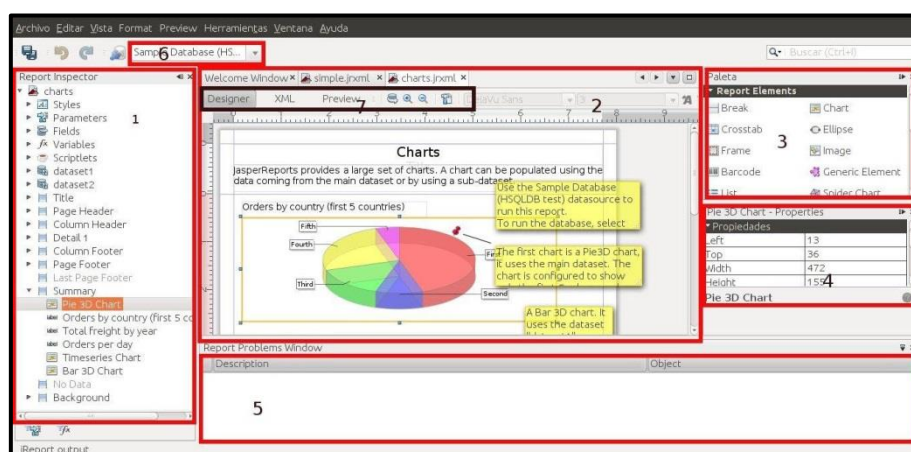
Para la instalación del plugin de iReport para NetBeans necesitaremos una serie de archivos de complemento con extensión nbm (NetBeans Module). La obtención e incorporación de estos archivos a NetBeans están disponibles en el siguiente enlace:

### [Instalación de iReport para NetBeans](#)

### [Guía de Instalación de iReport en NetBeans 11 \(RECOMENDADO\)](#)

## Interfaz de usuario de iReport

Al ejecutar iReport, vemos una interfaz parecida a esta, en el ejemplo se ha abierto un ejemplo de los que viene por defecto con la herramienta, para abrirlo, vamos a **ayuda » samples » charts**.



1. **Inspector de informes:** muestra la estructura completa del informe, que se compone de muchos objetos (tales como campos, parámetros y variables), bandas (que son las secciones del documento) y elementos (tales como campos de texto, imágenes o gráficos).

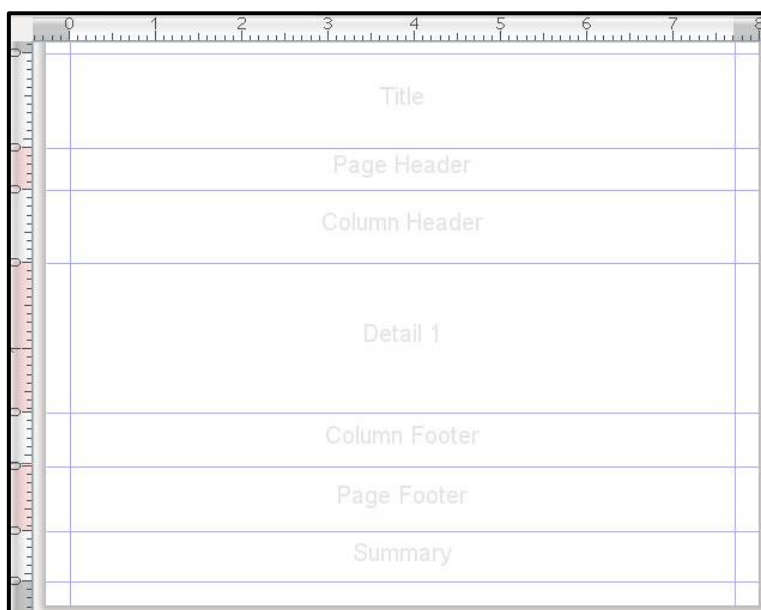


2. El **diseñador de informes** permite diseñar visualmente el informe de arrastrando, posicionando, alineando y cambiando el tamaño de los elementos del informe.
3. La **paleta de elementos** contiene los elementos de diseño que pueden ser arrastrados dentro de una banda para mostrar los datos. Para visualizarla hacemos clic en ventana » paleta.
4. La **hoja de propiedades** se utiliza para establecer las propiedades del componente seleccionado en el informe (como un campo, elemento, banda, grupo, u otros).
5. La **ventana de informe de problemas** contiene el listado de los errores encontrados al compilar el informe.
6. Sobre el diseñador de informes está el **selector del origen de datos**, que muestra la conexión activa que se utiliza para ejecutar el informe.
7. La barra de herramientas del diseñador de informes tiene tres botones para cambiar fácilmente desde el diseño a modo de vista previa. Al hacer clic en el botón de vista previa, iReport ejecuta el informe con la conexión activa y muestra el resultado utilizando un visor propio. iReport también puede ser configurado para exportar automáticamente el documento en un formato determinado y abrir el visor adecuado (como un navegador o un visor de PDF).

*La estructura de ventanas de iReport permite modificaciones, sin más que añadir o eliminar ventanas, según la necesidad de cada momento. Para eliminar una ventana, basta con pulsar la x en la esquina superior derecha. Para volver a visualizarla se selecciona desde el menú ventana.*

# Elementos estructurales de un informe

En la imagen se aprecia un informe vacío, a continuación se detallan los aspectos estructurales más destacados de cada elemento:



- **Título:** aparece sólo al inicio del informe. En esta sección se inserta el título del informe, por ejemplo "Informe de ventas del mes de marzo".
- **Encabezado de página:** aparece en la parte superior de cada página. Puede contener información como la fecha y hora, nombre de la organización, etc.
- **Encabezado de columna:** se utiliza para listar los nombres de los campos que se van a presentar (desplegar). Por ejemplo: "Producto", "Proveedor", "Precio de compra", "Precio de venta al público", "Beneficio", etc.
- **Detalle:** En esta sección se despliegan los valores correspondientes a las entradas de campos definidas en la sección anterior. Por ejemplo: "Barra de cortina metálica. 3M", "Cofrilsa distribuidores, S.A.", "2,25", "4,99", "205,30".
- **Pie de columna:** Puede presentar información resumida para cada uno de los campos. Por ejemplo: "Beneficio total del mes: 1245".
- **Pie de página:** Aparece en la parte inferior de cada página. En esta parte podemos incluir, entre otras cosas, un contador de páginas, por ejemplo: "Página 1/7".
- **Resumen:** Esta sección se usa para proporcionar información resumida de los campos presentes en la sección "detalle". Por ejemplo, para el caso del beneficio por producto se puede definir un objeto gráfico tipo "pie" para tener una mejor comparación y comprensión visual de los datos.

A cada uno de estos elementos estructurales se le denomina banda.

### Reflexiona

La distribución en bandas define y delimita cómo se mostrará la información, determinando qué elementos se repiten (la banda Detalle por ejemplo) y dónde se colocan (encabezados y pies de página).

## Iniciar el origen de datos

El ciclo de vida de un informe, en cualquier caso, pasa por una serie de pasos que se detallan a continuación, pero lo primero que necesitamos, es tener el origen de datos preparado para poder hacer la consulta, en este caso, vamos a usar una base de datos propuesta por iReport para el motor de base de datos HSQLDB, si usamos iReport de modo independiente basta con seleccionar **ayuda » samples » run sample database**, si usamos NetBeans para iniciar la base de datos seguiremos estos [sencillos pasos](#).



*Sin un origen de datos válido el informe no servirá para nada, ya que se compone de la combinación de diseño y datos, de hecho la estructura del informe depende de los datos a mostrar. Sólo en casos muy particulares se permiten informes con orígenes de datos vacíos. Por eso es necesario tener el motor de base de datos funcionando desde la fase de diseño del informe.*

### Para saber más

Para saber más acerca de este motor de base de datos para Java puedes seguir este enlace:

[Página oficial de HSQLDB](#)

## Creación de un informe sencillo

El proceso de creación de informes consta de los siguientes pasos principales:

1. Crear un **origen de datos** o una conexión al origen de datos utilizado para llenar el informe. Un origen de datos es cualquier fuente desde donde la herramienta pueda obtener el conjunto de registros con los que se rellenará el informe.
2. **Crear** el informe nuevo.
3. **Seleccionar los datos** que formarán parte del informe.
4. **Diseñar el informe**, incluyendo la disposición de sus elementos y parámetros para representar los datos.
5. **Ejecutar el informe**, a partir del archivo de origen se genera un archivo compilado y se rellena con los datos para la exportación o en pantalla.

En la siguiente presentación tienes un resumen gráfico de estos pasos para aprender a crear un informe a partir de una consulta sencilla.

### Debes conocer

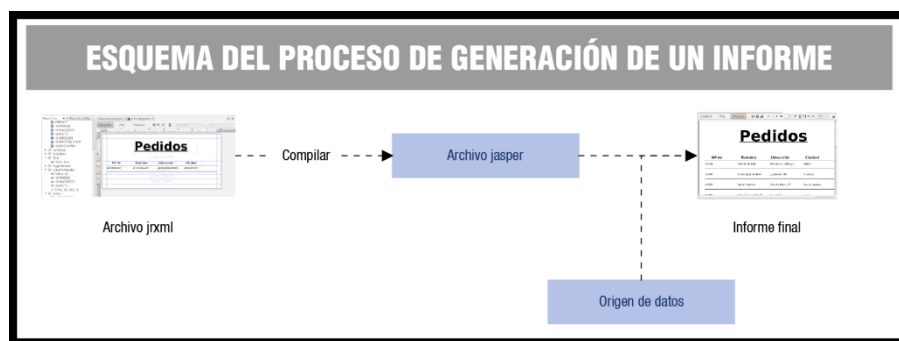
En el siguiente enlace puedes ver un resumen textual del procedimiento de elaboración de un informe sencillo:

[Elaboración de un informe](#)

Todos los ejemplos que se van a desarrollar a lo largo de la unidad se proporcionan en el siguiente archivo:

[Ejemplos de la unidad](#)

## Gestión de errores

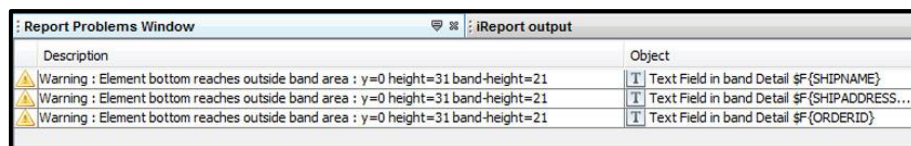


Al generar la vista previa, iReport realiza una serie de operaciones para crear el informe final.

**La primera operación consiste en compilar el archivo fuente, con extensión .jrxml en un archivo de Jasper, con extensión .jasper.** Este primer paso puede fallar si los elementos no están colocados correctamente (por ejemplo, si un elemento se coloca fuera de una banda), o si una expresión en el informe tiene errores y no puede ser compilado.

**Si la compilación se ejecuta correctamente, el archivo producido Jasper se carga y se llena con la conexión activa o fuente de datos.** Esta segunda operación, otra vez puede conducir a errores, por ejemplo, si la base de datos se hace referencia no está activo, una consulta no válida se ha proporcionado, o en un campo nulo producido un error en una expresión durante el proceso de llenado. Finalmente, si todas las operaciones se completan sin errores, el informe se muestra en el visor integrado.

Los errores se muestran en la ventana de Información sobre problemas y resultados de la operación se muestran en la salida de iReport, los cuales comparten la parte inferior de la pantalla de iReport:

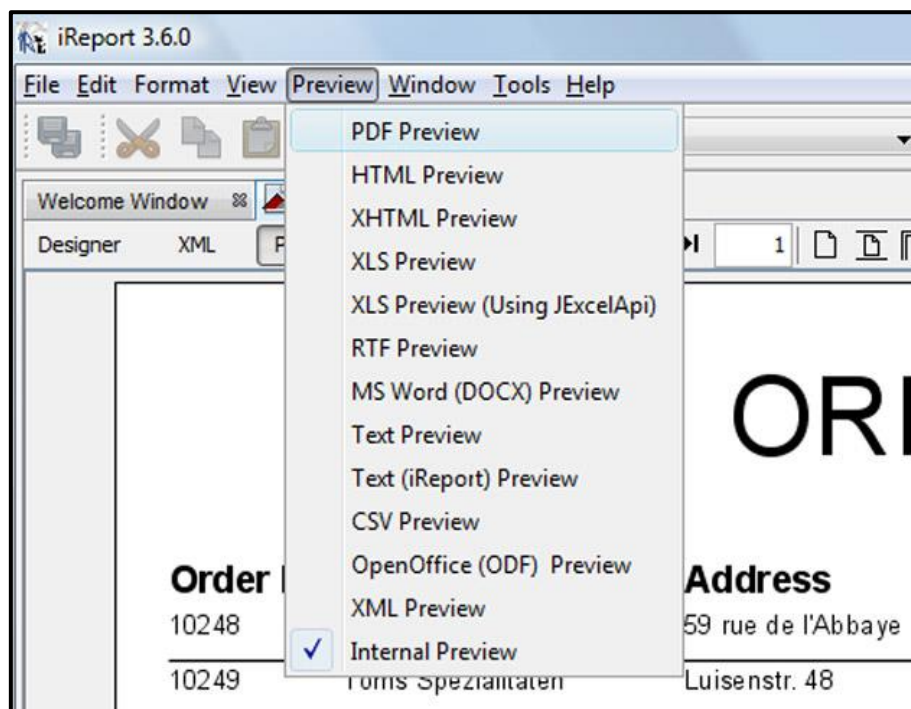


Report Problems Window		iReport output	
Description		Object	
Warning	Element bottom reaches outside band area : y=0 height=31 band-height=21	Text Field in band Detail	\$F{SHIPNAME}
Warning	Element bottom reaches outside band area : y=0 height=31 band-height=21	Text Field in band Detail	\$F{SHIPADDRESS...}
Warning	Element bottom reaches outside band area : y=0 height=31 band-height=21	Text Field in band Detail	\$F{ORDERID}

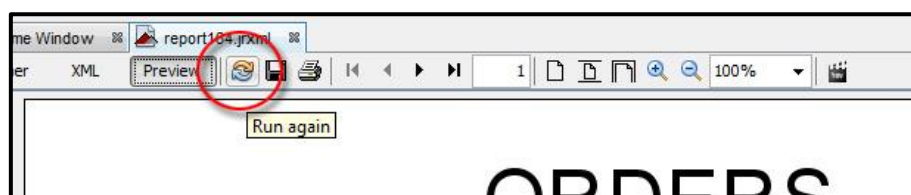
## Formatos de salida

*El objetivo de un informe es presentar información en un formato que sea accesible y fácil de distribuir, por lo que una vez generado se vuelva en archivos de texto, que se puedan almacenar e imprimir. Cualquier motor de informes debe ser capaz que exportar sus documentos a diferentes formatos de uso cotidiano, como PDF o HTML.*

En concreto iReport gestiona esto desde el menú Vista previa (**Preview**), donde podemos seleccionar el formato final del informe:



Para exportar de nuevo un informe que no se ha modificado, haga clic en **ejecutar de nuevo** en la barra de herramientas de vista previa. Una nueva ejecución de un informe es útil cuando un subinforme cambia, cuando cambia la fuente de datos, o cuando se desea ejecutar el informe con parámetros de entrada diferentes.



Cuando se establece un formato de vista previa, el informe se exporta automáticamente al formato elegido al hacer clic en vista previa, y la aplicación de visualización correspondiente se abre, por ejemplo, un visor de PDF u OpenOffice.

iReport es capaz de identificar el visor adecuado para cada formato automáticamente en función de los valores por defecto del sistema. Sin embargo, es posible ajustar manualmente la aplicación de visualización para cada formato, para hacerlo selecciona **herramientas » opciones de iReport » visualizadores (Viewers)**.

# Operaciones sobre los informes

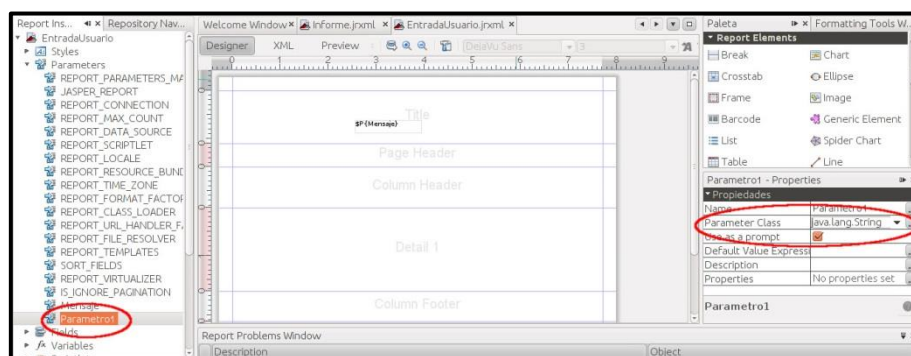
Efectivamente, Ana tiene razón. Un informe suele ser algo más que la presentación gráfica de los datos de una tabla de la base de datos. Lo habitual es realizar ciertas operaciones sobre el informe para obtener más información o mejorar el formato de presentación, entre otras cosas veremos como:

- Hacer cálculos sobre los datos para mostrar información resumida, como promedios o sumas.
- Parametrizar el informe para que se pregunte al usuario que datos desea mostrar.
- Filtrar los datos.
- Aplicar encabezados y pies de página a las hojas del informe.
- Etc.

## Uso de parámetros en un informe

La utilidad de los parámetros en un informe es permitir generar diferentes resultados a partir de un mismo archivo de diseño, en función de un dato que podemos cambiar nosotros directamente o a través de la aplicación final.

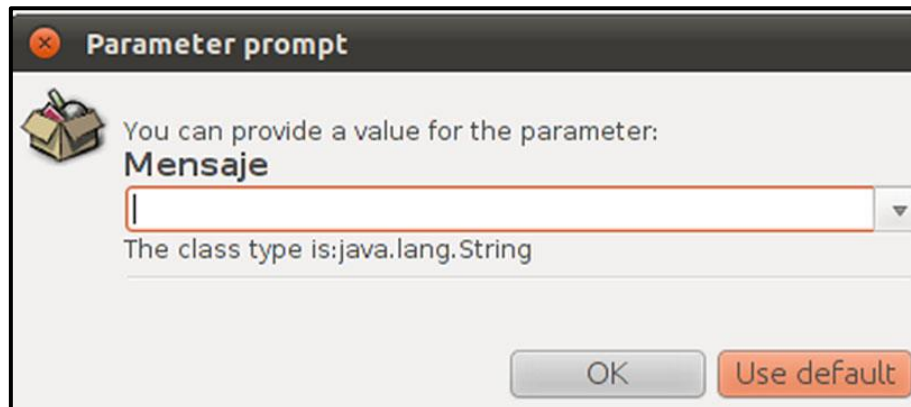
Un **parámetro** se define por un **nombre** y una **clase**, cualquier clase de Java es una clase de parámetro válido. Se emplea para pasar información al informe en tiempo de ejecución. Por ejemplo, un parámetro de tipo **java.sql.Connection** se puede utilizar para rellenar un informe integrado, mientras que un parámetro **java.lang.Boolean** se puede utilizar para mostrar u ocultar una sección del informe.



Para administrar los parámetros, utilizamos el apartado **Parameters** del inspector de informe. Desde aquí es posible agregar y quitar parámetros utilizando el menú

contextual. Podemos modificar un parámetro seleccionándolo y editando sus propiedades. Podemos modificar entre otras cosas:

- **Su nombre:** se define por medio de su propiedad **Name**.
- **Su clase:** se define en la propiedad **Class**.
- **Su valor predeterminado:** se define por medio de la propiedad "**Default Value Expression**". Esta expresión es evaluada por JasperReports sólo cuando un valor para el parámetro no ha sido proporcionada por el usuario en tiempo de ejecución.



En cuanto al tipo de parámetro podemos encontrar:

- **Parámetros integrados:** están disponibles de forma predeterminada y contienen información en tiempo de ejecución. Algunos de los más importantes son **REPORT\_CONNECTION**, que tiene la conexión JDBC para ejecutar la consulta SQL<sup>9</sup> del informe (si el informe está lleno con una conexión JDBC), el **REPORT\_DATA\_SOURCE** que contiene, en su caso, la fuente de datos utilizada para llenar el informe, o el **REPORT\_LOCALE** que contiene la configuración regional utilizada para rellenar el informe, y así sucesivamente. Los parámetros integrados no pueden ser modificados o eliminados.
- **Parámetros de usuario:** el programador determina su nombre, clase y valor. se pueden configurar para que sean insertados por el usuario cuando se ejecute el informe. Por ejemplo, vamos a crear un parámetro llamado mensaje de tipo string con la propiedad "use as a prompt" activa. Si arrastramos el parámetro desde el inspector de informe a la banda título iReport creará un campo de texto para mostrar el valor del parámetro.

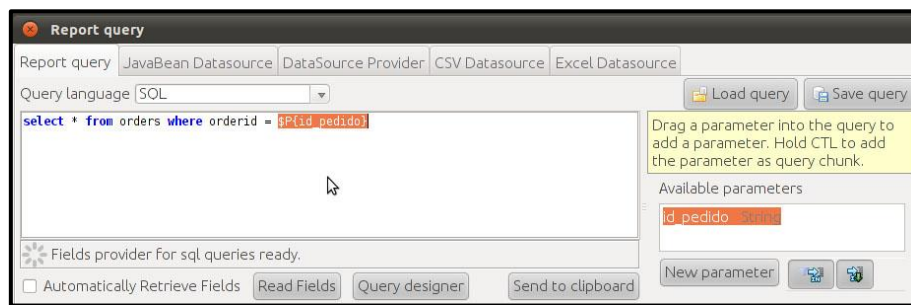
Uno de los usos más extendido de los parámetros es el filtrado de datos, que veremos a continuación.

---

<sup>9</sup> *Structured Query Language*, es un lenguaje declarativo empleado para el diseño y administración de bases de datos relacionales.



## Filtrado de datos



Se pueden utilizar parámetros en las consultas SQL para filtrar los registros en la condición where o para añadir o sustituir elementos de la consulta SQL o incluso pasar toda la cadena SQL para ejecutar. Tenemos dos posibilidades:

1. En el **primer caso** los parámetros se utilizan como parámetros estándar SQL, por ejemplo:

```
select * from orders where orderid = ${id_pedido}
```

En este ejemplo, **id\_pedido** es un parámetro de tipo **java.lang.Integer** que contiene el identificador de pedido que se debe seleccionar. Este parámetro se puede pasar al informe desde la aplicación que lo ejecuta para seleccionar sólo un pedido específico, como veremos al final de la unidad. El parámetro aquí es un verdadero parámetro SQL, lo que significa que la consulta se ejecutará mediante una sentencia como la siguiente:

```
SELECT * FROM orders where ORDER_ID =?
```

y el valor del parámetro **id\_pedido** entonces pasa a la instrucción.

2. El **segundo caso** se trata de construcciones como:

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

El parámetro será tratado como un campo de SQL. JasperReports tendrá en cuenta este parámetro como una especie de marcador de posición (ten en cuenta la sintaxis especial de **\${!{}}**), que será reemplazado por el valor de texto del parámetro (que en este caso puede ser, por ejemplo, **"OrderDate DESC"**).

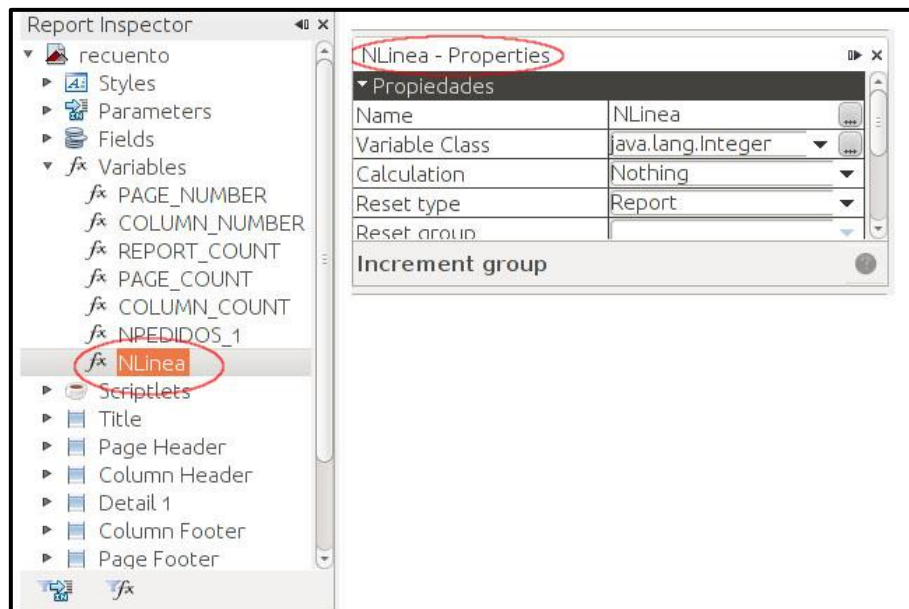
Con la misma lógica, una consulta puede pasarse íntegramente mediante un parámetro del siguiente modo:

```
$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.

Cuando se utilizan parámetros en una consulta, para que iReport pueda recuperar los campos disponibles de la consulta, se debe fijar un valor por defecto para el parámetro.

## Valores calculados



Podemos hacer cálculos usando variables. Podemos añadir y eliminar variables desde la zona variables del inspector de informe y la hoja de **propiedades** y, al igual que los parámetros, se definen por su **nombre**, **clase** y **valor**. Para editar una variable, se seleccione en el inspector de informe y modificamos la hoja de propiedades.

- 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.
- También se pueden crear **variables de usuario**, que sí pueden cambiar. Será el programador el encargado de asignar nombre (**NAME**), tipo de datos

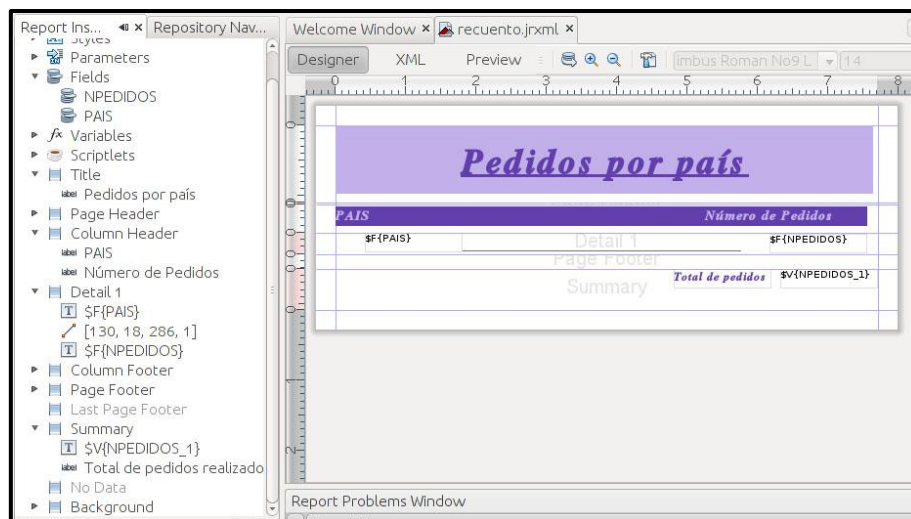
(**VARIABLE CLASS**) y valor inicial a la variable (**INIT VALUE EXPRESSION**).

A diferencia de los parámetros que toman un valor establecido por el programador como valor por defecto, o asignado desde la aplicación final en tiempo de ejecución pero no cambia, el valor de las variables cambia mientras se va creando el informe, y pueden ser evaluadas en diferentes momentos para tomar el valor más adecuado.

Veamos ahora algunos de los usos más comunes de una variable.

## Recuentos

Un recuento consiste en aplicar la función de resumen <sup>10</sup>suma a un campo concreto de una consulta. Son útiles para calcular totales y subtotales.



Realizaremos un informe sencillo que muestra un conjunto de países (**ShipCountry**) y el número de pedidos realizados en ese país (NPedidos). Para obtener estos campos trabajamos sobre la base de datos de muestra de iReport con la consulta de selección:

```
SELECT
    count(ORDERS."ORDERID") AS NPedidos,
    ORDERS."SHIPCOUNTRY" AS Pais
FROM
    "PUBLIC"."ORDERS" ORDERS
GROUP BY ORDERS."SHIPCOUNTRY"
```

<sup>10</sup> Una **función de resumen**, **función digest** o **función Hash** es una función matemática que parte de un conjunto de entrada para producir resultados en un conjunto de salida menor y con datos más simples. Se utilizan en informática para obtener índices de acceso y para comprobar la integridad de la información, entre otros fines.

Arrastramos los campos **NPedidos** y **Pais** a la banda de detalle para genera el informe, y a continuación arrastramos el campo de **NPedidos** dentro de la banda **Summary**, iReport preguntará qué valor debe mostrar. Puede ser sólo el valor de **NPedidos** (que en esta banda será sólo el último valor asumido por el campo) o el resultado de una función de agregación, como la suma.

Selecciona la **suma** y pulsa **Aceptar**.

### Reflexiona

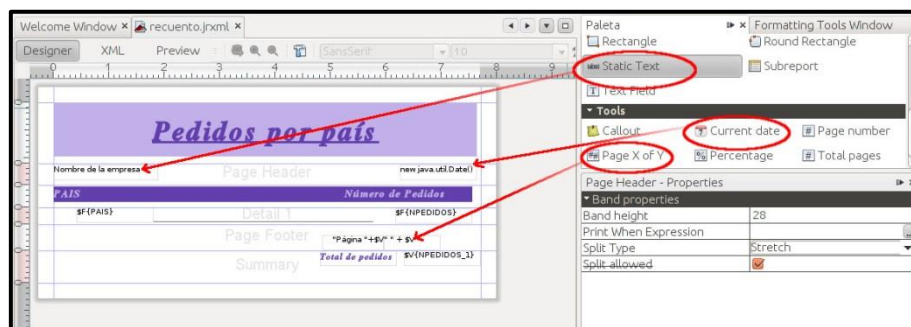
Al ejecutar el informe se observa que el valor impreso es, en efecto, la suma de todos los pedidos de todos los países. ¿Por qué se produce esto?

#### Respuesta:

Cuando le pedimos que imprima la suma, iReport crea una variable (su nombre es **NPedidos\_1**). Esta variable es de tipo entero (ya que la suma es una suma de valores enteros), la función de cálculo es la suma y la expresión utilizada para actualizar el valor de la variable (con el cálculo de la suma) es **\$F{NPedidos}** que es una expresión para obtener el valor del campo **NPedidos**. Para cada registro del informe, JasperReports obtiene el valor del campo (en realidad, se evalúa la expresión variable), y suma ese valor al conjunto de valores recogidos para realizar el cálculo.

La variable **NPedidos\_1** se muestra en un campo de texto en la banda de resumen.

## Modificar encabezados y pies de página



Para añadir la **página X de Y** en el pié de página de un informe, sólo tienes que arrastrar la herramienta página X de Y de la paleta en la banda **PAGE FOOTER**. Como se escribe Page en inglés, editamos la etiqueta y sustituimos la palabra "Page" por "Página".

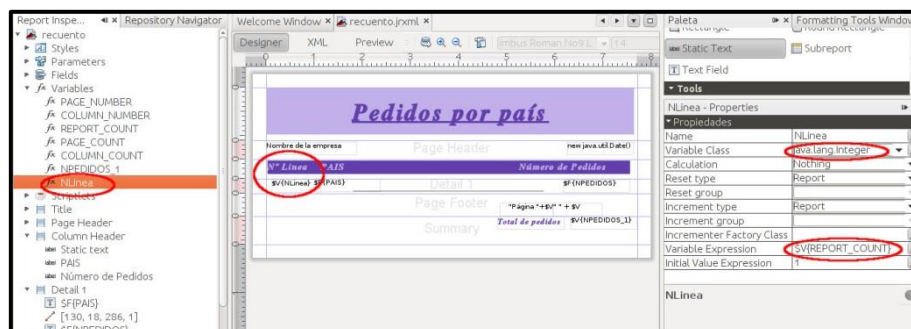
Esta herramienta crea dos campos de texto que muestran la misma variable: **PAGE\_NUMBER**. El campo de texto primero muestra la página actual, el segundo el total de páginas del informe. Esto es posible porque el tiempo de evaluación de cada campo de texto es diferente, en particular, el primer campo 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 este 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 nos permite imprimir el valor asumido por una variable en diferentes momentos. Con esta idea, podemos poner la suma total de pedidos, como se ve en el ejemplo anterior 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 iReport cuando un campo se arrastra en el título y el usuario elige para mostrar el resultado de una función de agregación).

De igual manera podemos añadir etiquetas, imágenes a la banda **PAGE HEADER** para modificar el encabezado de las páginas del informe, por ejemplo, podemos añadir una etiqueta en la que escribamos el nombre de la empresa y arrastrar la fecha (**Current date**) del apartado Tools de la paleta. Cuando colocamos el campo de fecha nos preguntará el formato en el que queremos que aparezca.

## Numeración de líneas



A veces puede ser necesario incluir el número de líneas en un informe. Para conseguirlo, usaremos una variable creada por el usuario, por ejemplo, podemos crear la variable NLineas. Para conseguir que cuente el número de línea para cada registro modificaremos las propiedades de la variable para hacer que sea de tipo entero (**java.lang.Integer**), en **Calculation** indicaremos que no haga nada (nothing), iniciaremos la variable a **1 (Init Value Expresion)** y le asignaremos la expresión **\$V{REPORT\_COUNT} (Value Expresion)**.

Para visualizar los números de línea arrastramos la variable a la banda de detalle, justo antes del país.

## Informes con agrupamientos

Agrupar los datos de un informe nos permite crear ciertas estructuras para organizar mejor los datos.

Para crear un grupo se define una **expresión** que se evalúa de tal modo que cada vez que la expresión cambia se inicia un nuevo grupo.

La expresión puede ser representada sólo por un **campo** específico (si queremos agrupar un conjunto de contactos por ciudad, o país), o puede ser más compleja (Por ejemplo, es posible agrupar un conjunto de nombres de contactos por letra inicial).

Agenda		
<b>Berne</b>		
Ott	Bill	250 - 20th Ave.
Schneider	James	277 Seventh Av.
2		
<b>Boston</b>		
Ott	Michael	339 College Av.
Heiniger	Julia	358 College Av.
2		

Es conveniente obtener los registros de la consulta de base **bien ordenados**, ya que la herramienta no va a hacer nada por ordenarlos, solo va a crear grupos cada vez que la expresión cambie.

Por ejemplo, si la expresión para el agrupamiento es la ciudad, cada vez que cambiemos de ciudad se genera un grupo nuevo, por lo que debemos obtener todos los registros ordenados por ciudad.

Cada grupo puede tener bandas de cabecera y pie de página. Los encabezados y pies de página del grupo se imprimen antes y después de la banda de detalle. Se puede definir un número arbitrario de grupos (es decir, podemos tener un grupo de primer nivel que contiene los contactos por país y un grupo anidado que contiene los contactos en cada país por la ciudad), que se anidarán según se ordenen en el inspector del informe.

A continuación, tienes dos ejemplos de grupos: uno cuya expresión es un campo de la consulta y otro con una expresión algo más compleja.

### Debes conocer

En esta presentación crearemos un informe donde se muestra una lista de personas que se agrupan por la ciudad donde viven.

#### [Resumen textual alternativo](#)

En esta presentación crearemos un informe donde se muestra la lista de personas anterior pero agrupadas por la letra inicial de su apellido.

#### [Resumen textual alternativo](#)

## Subtotales

El cálculo de subtotales implica hacer recuentos y cálculos resumidos (medias, porcentajes, mínimos o máximos) del conjunto de los datos que se representan en el informe pero aplicando una categorización que lo divide en grupos. Se realiza el cálculo para cada grupo y luego para el conjunto de datos completo también.

*El proceso de obtención del subtotal conlleva los siguientes pasos:*

- 1. Ejecutar la consulta contra la base de datos que devuelva los registros que nos interesan.*
- 2. Dividir los registros en grupos en función de un criterio, por ejemplo, por algún campo específico, como la ciudad o el tipo. Los grupos deben ser mutuamente excluyentes, es decir, cada registro pertenece siempre a un grupo, y no puede pertenecer a más de uno.*
- 3. Aplicar el cálculo o recuento a cada grupo.*
- 4. Hacer el cálculo para todo el conjunto de registros obtenidos.*

Un ejemplo sencillo de creación de subtotales lo hemos visto en el apartado anterior, basta con añadir la variable **<nombre\_grupo>\_COUNT** al pie del grupo. No obstante, podemos hacer cálculos algo más complejos aprovechando las características de las variables.

En este ejemplo vemos como crear un subtotal contra la base de ejemplo de iReport.

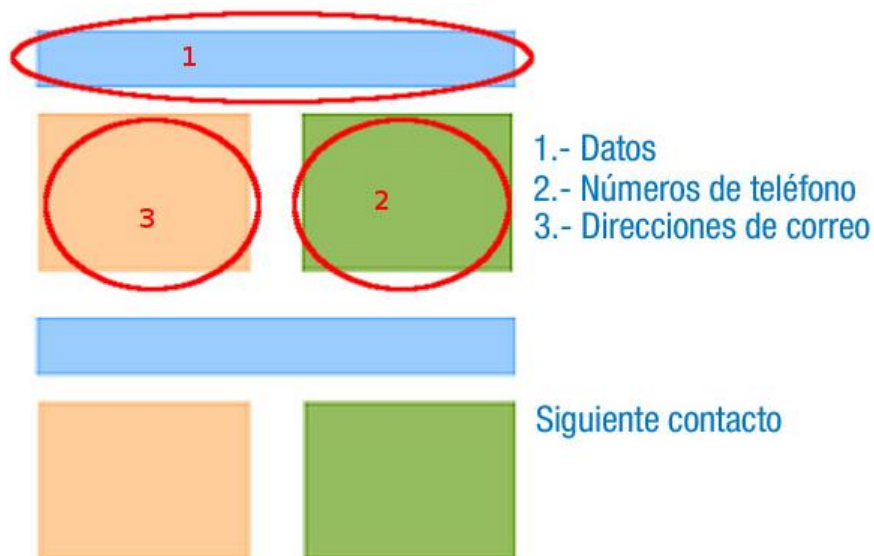
### Debes conocer

Informe donde se muestra una lista de productos indicando en qué pedidos aparece, con qué precio y cantidad, para, a continuación, calcular el total de unidades pedidas, la media de unidades y el pedido máximo y mínimo.

### Resumen textual alternativo

## Subinformes

*Un subinforme es un informe incluido dentro de otro informe. Esto permite la creación de diseños muy complejos, con diferentes partes de un documento único, que se llena con diferentes fuentes de datos e informes.*



Supongamos el caso de una agenda en la que para cada contacto tenemos varios números de teléfono y direcciones de correo electrónico almacenados en tres tablas diferentes. Queremos representar la información del siguiente modo:

- Las zonas azules corresponden a los datos del contacto.
- Las zonas naranjas serán sus números de teléfono.
- Las zonas verdes serán para las direcciones de correo.

Crearemos un informe principal para los datos de los contactos (zona azul) y después dos subinformes para los teléfonos y direcciones de correo.

El informe principal se utilizará para seleccionar a las personas de la libreta de direcciones. Crearemos un primer subinforme, para seleccionar las direcciones de correo electrónico de cada persona, en la zona de color naranja. Por último, crearemos un segundo subinforme para obtener los números de teléfono (parte verde).



Para poder realizar el ejemplo que se propone, se necesita crear una base de datos **MySQL**<sup>11</sup> que se llame **agenda**. El ejemplo se ha creado para el usuario root con contraseña **usuario**. La especificación viene dada en el siguiente archivo:

### [Base de datos agenda](#)

#### **Ejercicio resuelto**

Realiza un informe con el desarrollo de una agenda de teléfonos y direcciones de correo electrónico creado con subinformes.

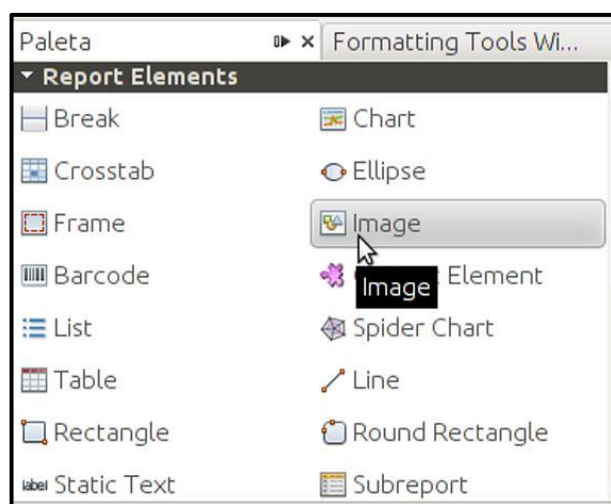
#### [Respuesta](#)

## **Añadir imágenes**

*Las imágenes son representaciones visuales de objetos que se almacenan en un fichero al que aplica un formato concreto.*

Cuando se añaden imágenes a un informe, éstas no pasan a formar parte del mismo, sino que se añade una expresión con la ruta absoluta de la imagen, por eso, cuando se arrastra un elemento de imagen en el diseñador, iReport muestra un cuadro de diálogo selector de ficheros. Esta es la forma más conveniente de especificar una imagen que se va a usar en el informe. La expresión se establece como el valor de la propiedad **Image Expression** de la imagen. He aquí una expresión de ejemplo:

**"/home/usuario/informes/flor.jpg"**

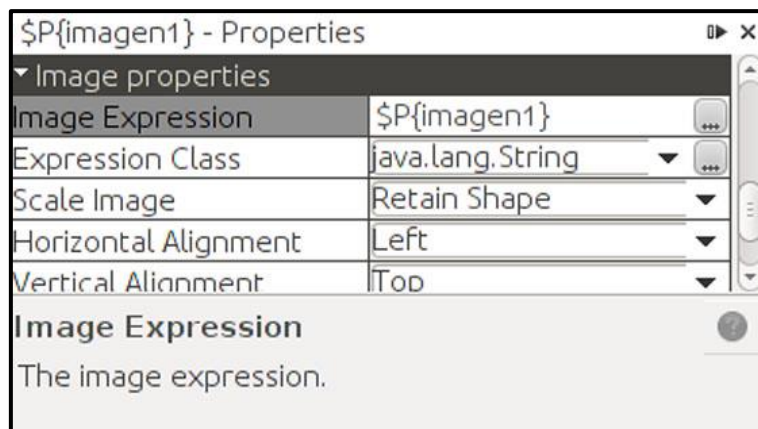


---

<sup>11</sup> Sistema gestor de bases de datos relacional desarrollado bajo licencia GPL y licencia comercial Oracle.

## Reflexiona

¿Crees que es apropiado este modo de gestionar imágenes? ¿Qué pasa cuando movemos el archivo .jrxml de sitio? ¿Cómo resolverías este problema?



### Parametrizar la Image Expression:

Este sistema tiene un gran impacto en la portabilidad informe, ya que probablemente el archivo no se encuentre en otra máquina (es decir, después de implementar el informe en un servidor web o de ejecutar el informe en un equipo diferente). Para solucionar este problema podemos parametrizar la propiedad **Image Expression** de la imagen estableciendo su valor a algo similar a esto:

En tiempo de ejecución en una aplicación hipotética, el valor del parámetro directorio\_de\_imagenes se puede ajustar mediante la aplicación en sí misma. Podemos proporcionar un valor por defecto para el parámetro. La ventaja de esta solución es que la ubicación del directorio en el que están las imágenes no se define directamente en el informe, sino que se proporciona de forma dinámica.

```
$P{DIRECTORIO_DE_IMAGENES} + "miImagen.png"
```

### Usar el Classpath:

Otra opción es utilizar el **Classpath**. Cuando una imagen se encuentra en el **Classpath**, sólo se requiere el nombre de la imagen para encontrarla. De forma predeterminada, al ejecutar un informe, iReport agrega el directorio en el que reside el informe al Classpath. Si tenemos el informe y la imagen en el mismo directorio, basta con establecer el valor de **Image Expression** al nombre del fichero con la imagen. Puesto que el directorio del informe se agrega a la ruta de clases, la imagen se encuentra de forma automática.

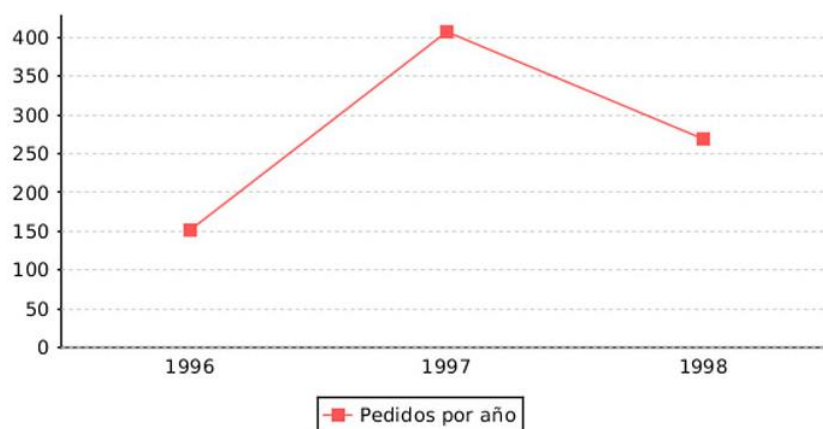
Este proceso sigue siendo válido si la imagen se encuentra en un subdirectorio de un directorio incluido en el **Classpath**. En ese caso, se deberá especificar la ruta completa

del recurso utilizando una notación de caminos al estilo Unix. Por ejemplo, si la imagen se encuentra en el directorio `/home/usuario/informes/imagenes`, el recurso se encuentra con la expresión `/imagenes/imagen.png`.

JasperReports comprobará en el directorio `/home/usuario/informes/` (que está en el **Classpath**), y, a continuación, en la ruta de recurso especificado (en este caso `/imagenes/imagen.png`).

## Gráficos

Un **gráfico** permite representar cierta información de tipo numérico (normalmente) mediante recursos gráficos (líneas, vectores, superficies o símbolos) con el objetivo de hacer más visibles los datos, poner de manifiesto su evolución temporal o espacial, o evidenciar relaciones elementos del sistema.



Existen distintos **tipos de gráficos**, entre los que destacan:

- **Gráficos lineales:** se representan los valores en dos ejes cartesianos ortogonales entre sí. Las gráficas lineales se recomiendan para representar series en el tiempo, y es donde se muestran valores máximos y mínimos; también se utilizan para varias muestras en un diagrama.
- **Gráficos de barras:** contienen barras verticales que representan valores numéricos. Normalmente representan frecuencias dentro de una categoría.
- **Gráficos circulares:** permite ver la distribución interna de los datos que representan un hecho, en forma de porcentajes sobre un total. Se suele separar el sector correspondiente al mayor o menor valor, según lo que se desee destacar.
- **Gráfico simbólico:** con imágenes que sirven para representar el comportamiento o la distribución de los datos cuantitativos de una población, utilizando símbolos de tamaño proporcional al dato representado.

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

- **Serie:** Es el conjunto de datos numéricos a representar. Podemos tener más de una serie en un gráfico, salvo en los gráficos circulares que sólo tienen una serie. 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 queremos 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.

En el ejemplo, vemos un gráfico con una serie, Pedidos por año, que toma valores en el rango de cero a cuatrocientos y cuya categoría está formada por tres elementos correspondientes a los años 1996, 1997 y 1998. representa la evolución de los pedidos por año.

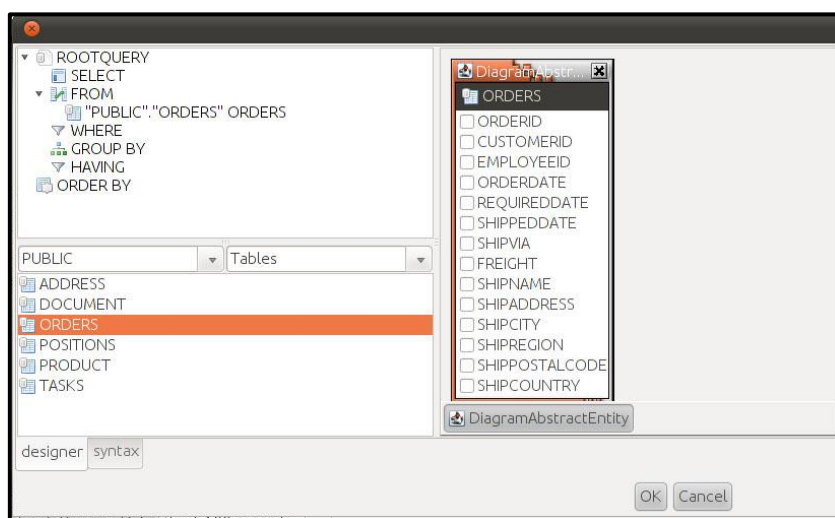
### Ejercicio resuelto

Genera un documento nuevo con iReport en el que se cree un gráfico de barras que refleje los pedidos agrupados y ordenados por mes.

### Respuesta

Te propongo generar el gráfico que ves en la imagen de arriba para practicar. Modifica las propiedades del mismo para cambiar los colores del gráfico, y añadir etiquetas a los datos de la serie.

## Informes sobre consultas complejas



A veces es preciso involucrar varias tablas en la consulta para generar un informe, incluyendo uniones, ordenación y agrupaciones. El resultado final es independiente de lo compleja que pueda ser la consulta, ya que una vez que hayamos obtenido los registros, pero para facilitar el proceso de crear la consulta podemos utilizar alguna herramienta visual. iReport, por ejemplo, proporciona un **diseñador de consultas** (query designer), accesible desde la ventana de la consulta que puede acceder a la estructura completa de la base de datos para hacer la selección de tablas y campos de manera gráfica.

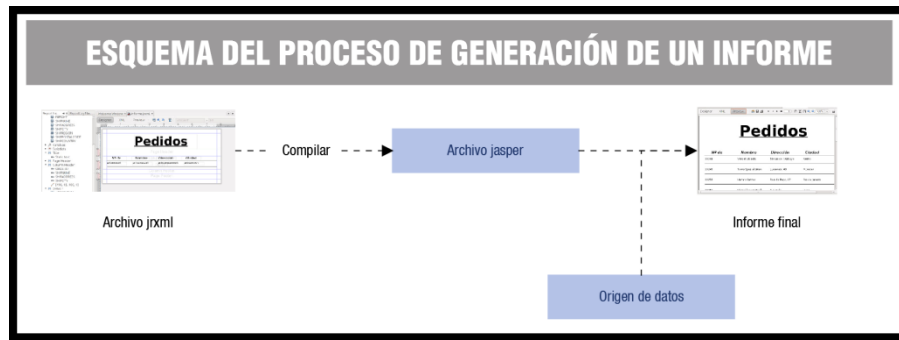
- En la zona de la izquierda abajo aparece la lista de la tablas de la base de datos seleccionando las opciones **PUBLIC** y **Tables**. Podemos desplegar una tabla en el panel de la derecha haciendo doble clic sobre su nombre. Los campos que se marquen se añadirán a la consulta.
- Haciendo clic con el botón secundario sobre la cláusula **WHERE** y seleccionando **add condition** podemos añadir condiciones.
- De igual forma se añaden cláusulas **HAVING**.
- Para añadir cláusulas **GROUP BY** u **ORDER BY** hacemos clic con el botón secundario sobre el campo que va a participar en la cláusula y seleccionamos add to **group-by** o add to **order-by**.
- Cualquier cláusula se elimina haciendo clic con el botón secundario, seleccionando la opción **remove**.

### Ejercicio resuelto

Crea un informe nuevo que muestre todos los pedidos de la base de datos organizados por documento.

[Respuesta](#)

# Análisis del código obtenido



Los informes de iReport se almacenan en archivos XML con extensión **.jrxml**. Un archivo **jrxml** se compone de un conjunto de secciones, algunas de ellas relativas a las características físicas del informe, como las dimensiones de la página, el posicionamiento de los campos, y la altura de las bandas, otras son relativas a las características lógicas, tales como la declaración de los parámetros y variables, y la definición de una consulta para la selección de datos.

Al hacer clic en el botón de vista previa en la barra de herramientas de diseño, iReport realiza una serie de operaciones para crear el informe final. La primera operación consiste en compilar el archivo fuente **jrxml** en un archivo de Jasper. Esta compilación se realiza por motivos de **rendimiento**. Si la compilación se ejecuta correctamente, el archivo producido Jasper se carga y se llena con la conexión activa o fuente de datos.

Durante la compilación del archivo **jrxml**, el archivo XML se analiza y se carga en un objeto JasperDesign, una estructura de datos que permite representar el contenido XML en la memoria. Independientemente del lenguaje que se utiliza para las expresiones dentro del archivo **jrxml**, JasperReports crea una clase especial de Java que representa la totalidad del informe, se compila, se instancia y se serializa en un archivo Jasper, que será el que se cargue en una aplicación posteriormente.

A la hora de ejecutar un informe necesitaremos este archivo de Jasper y una fuente de datos para JasperReports. Hay muchos tipos de fuentes de datos, es posible llenar un archivo de Jasper de una consulta SQL, un archivo XML, un archivo **CSV**<sup>12</sup>, un **HQL** (*Hibernate Query Language*) de la consulta, una colección de JavaBeans, etc., incluso es

---

<sup>12</sup> *Comma-Separated Values* es un tipo de documento de formato abierto que representa tablas de datos separando los campos con comas.

posible elaborar una fuente de datos personalizada. Con un archivo de Jasper y un origen de datos, JasperReports es capaz de generar el documento final en el formato que prefiera.

El archivo Jasper no contiene recursos externos, como puedan ser las imágenes utilizadas en el informe, paquetes de recursos para ejecutar el informe en diferentes idiomas, scriptlets extra u hojas de estilos externas. Todos estos recursos deben estar ubicados en tiempo de ejecución y ser proporcionados por la aplicación, como veremos a continuación.

# Repaso a la librería Jasper Report

La librería **JasperReports** permite la integración de los informes en una aplicación Java. Es posible hacerlo partiendo tanto del archivo **.jasper** como del archivo **.jrxml**. Se utiliza para compilar, rellenar, aplicar parámetros y visualizar un informe en diferentes formatos finales. La ventaja es que como podemos pasar parámetros mediante código, basta con conectar el informe a un formulario y el usuario final podrá determinar las características finales del informe o de los datos a mostrar.

Para **pasar parámetros** a un informe es necesario, en primer lugar, tener definido el parámetro en el informe con el tipo adecuado. En el código crearemos una **tabla hash**<sup>13</sup> a la que añadiremos una pareja formada por el nombre del parámetro y la variable que contiene su valor.

El despliegue del informe se hace a través de un objeto de la clase **JasperPrint** y la clase **JasperFillManager**. Para crearlo usaremos esta sentencia:

```
JasperPrint print = JasperFillManager.fillReport(ArchivoJasper,
Parámetros, Conexion);
```

Donde:

- **ArchivoJasper:** es el archivo de Jasper con el informe.
- **Parámetros:** tabla hash con los parámetros que hay que pasar al informe.
- **Conexión:** conexión al origen de datos.

**JasperFillManager.fillReport** genera el informe en memoria. Para volcar el informe a un archivo, utilizamos la clase **JasperExportManager**, que tiene varios métodos para crear archivos de salida de diferentes tipos, entre otros podremos generar un archivo PDF con el informe con el siguiente código:

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

---

<sup>13</sup> Una **tabla hash**, **tabla de dispersión** o **mapa hash** es una estructura de datos en forma de tabla que asocia unos datos simplificados que sirven de índices llamados **llaves** o **claves** con determinados **valores**.



**Para saber más**

Si quieres saber todas las posibilidades que te ofrece la librería JasperReports puede mirarlo en su página web aquí:

[Página oficial de la biblioteca de JasperReports](#)

**Ejercicio resuelto**

En el este enlace tienes un ejemplo de uso de la librería JasperReports en un proyecto Java creado con NetBeans.

[Integración de un informe iReport en un proyecto NetBeans](#)

# Anexo I- Instalación del plugin de iReport para NetBeans

**NOTA DEL PROFESOR:** se recomienda emplear la guía citada anteriormente para este fin, y que se puede encontrar en el siguiente enlace:

[Guía de Instalación de iReport en NetBeans 11 \(RECOMENDADO\)](#)

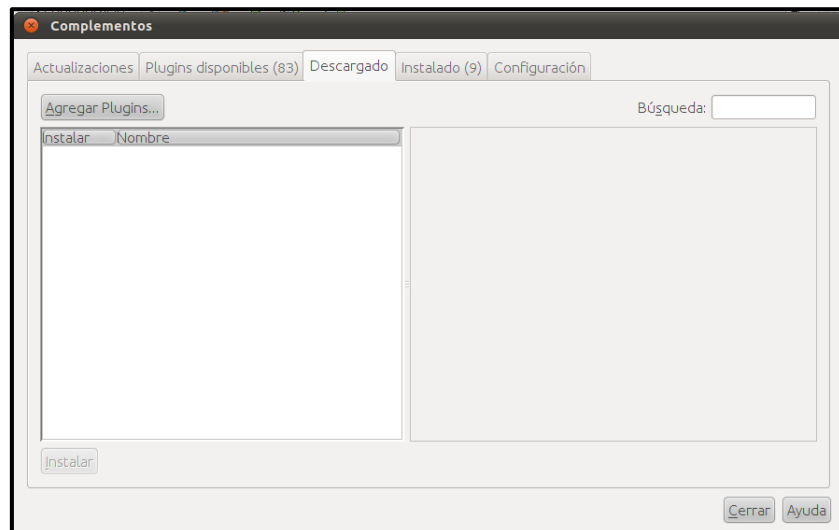
Desde la página de JasperForge para iReport podemos descargar un archivo con el plugin para NetBeans.

El archivo se llama **iReport-4.0.2.zip** y contiene varios archivos nbm con los complementos necesarios para el uso de **iReport** y las librerías **JasperReport** en NetBeans:

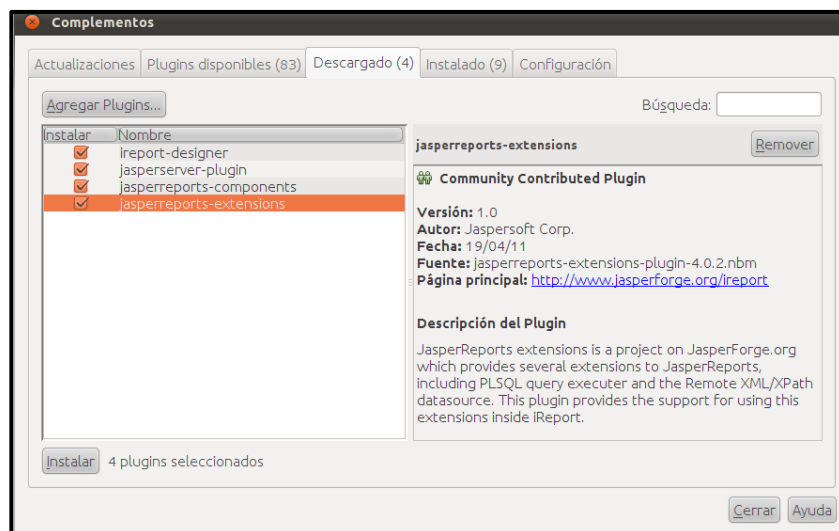
- **ireport-4.0.2.nbm.**
- **jasperreports-components-plugin-4.0.2.nbm.**
- **jasperreports-extension-plugin-4.0.2.nbm.**
- **jasperserver-plugin-4.0.2.nbm.**

Para cargar los complementos en **NetBeans** primero descomprimos el archivo en el directorio **ireport** que hemos creado antes, porque necesitamos tener los archivos disponibles en algún lugar del disco.

1. Abrimos **NetBeans** si no lo teníamos ya abierto, y vamos al menú herramientas » complementos, apartado descargado.



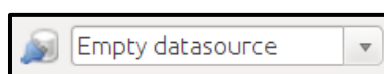
2. Pulsamos en agregar plugin y buscamos los archivos que acabamos de descomprimir.



3. Pulsamos en instalar los cuatro plugins seleccionados. Si en algún momento nos pide confirmación para realizar la instalación por que el origen no es de confianza pulsamos en aceptar, ya que en este caso, si sabemos cual es el origen del software. Una vez terminado cerramos la ventana de complementos.

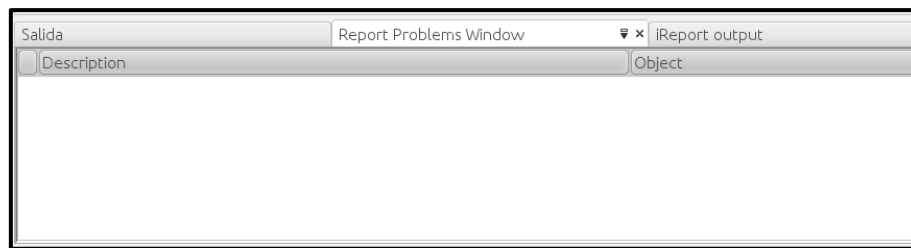
### Elementos nuevos:

La instalación del complemento genera dos elementos nuevos en la barra de herramientas de NetBeans:



El primer icono abre los orígenes de datos de los informes y la lista desplegable el origen de datos actual.

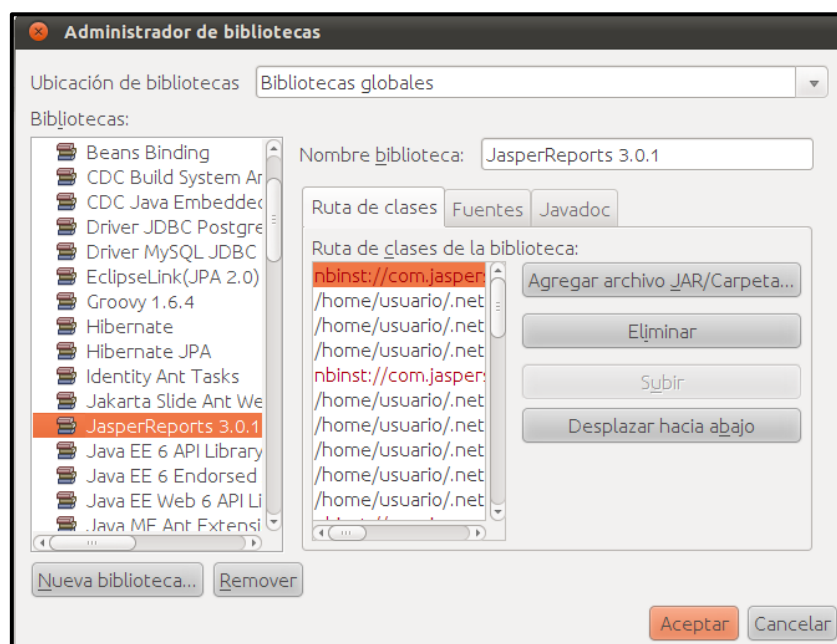
También tenemos dos paneles nuevos: la ventana de salida de iReport (**iReport output**) y una ventana para mostrar posibles problemas con los informes (**Report Problems Window**).



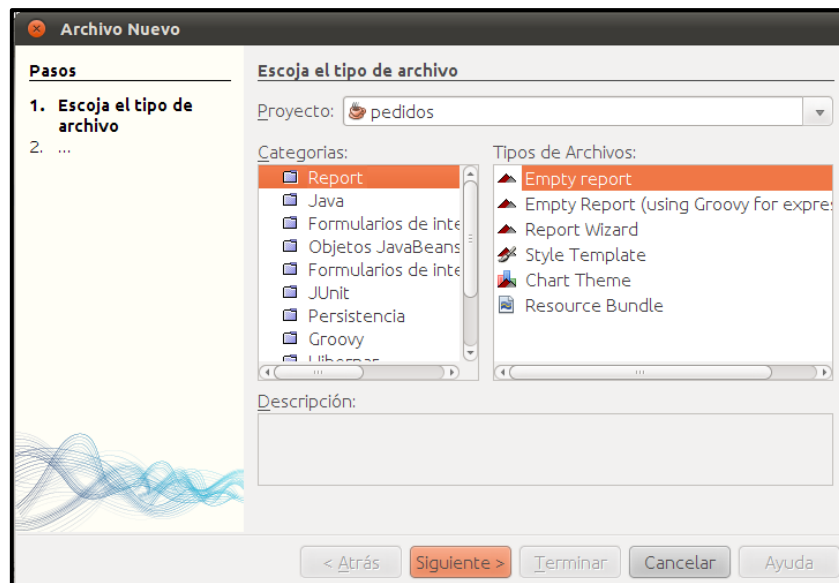
### Comprobar las bibliotecas instaladas:

La instalación de iReport añade una biblioteca a NetBeans que vamos a comprobar. Abrimos las bibliotecas cargadas en **herramientas » bibliotecas**. En la lista de bibliotecas seleccionamos **JasperReports 3.0.1**. observamos que algunas rutas aparecen en color rojo. Esto es así porque busca archivos jar que no encuentra, tendremos que indicarle su ubicación de la siguiente manera:

4. Hacemos clic en agregar archivo JAR/Carpeta...
5. Seleccionamos el directorio de instalación de NetBeans: **/home/usuario/.netbeans/6.9/modules/**.
6. Dentro de este directorio encontramos los archivos jar de jaspersoft que necesitamos.
7. Todos los archivos jar que podamos necesitar cuando programemos con la librería JasperReports los podremos encontrar en el subdirectorio (**ireport/modules/ext** del directorio de iReport).



Ahora podemos usar iReport dentro de NetBeans del mismo modo que lo hacemos con la aplicación independiente. Al agregar un archivo al un proyecto de NetBeans, podremos elegir, entre otros, un informe:



# Anexo II - Cómo utilizar la base de datos de prueba de iReport

Para realizar pruebas podemos utilizar la base de datos propuesta por iReport. Para ello tenemos que iniciar el servidor HSQLDB y crear la base de datos de prueba.

## Iniciar la base de datos en Linux-Ubuntu:

HSQLDB es un pequeño sistema gestor de bases de datos escrito totalmente en java, que se usa, entre otras en la aplicación base de openOffice. Para iniciar el servidor tenemos que localizar el archivo **hsqldb.jar**, se puede descargar de internet, de su página oficial, pero nosotros lo podemos encontrar en el directorio de iReport **ireport/modules/ext**.

En un terminal ejecutamos:

```
usuario@equipo:~/ireport/ireport/modules/ext/ java -cp hsqldb-1.8.0-10.jar org.hsqldb.Server -database.0 file:../../database/test
```

**-database.0 file:../../database/test**. Este parámetro indica cual es la base de datos que se va a crear físicamente. Como pueden crearse varias bases de datos, debemos diferenciar cada una usando un número después del parámetro **-database**. Es decir, en este caso, nuestra base de datos tiene el número 0 (**-database.0**). Luego con el parámetro file: se especifica dónde se va a ubicar físicamente el archivo de la base de datos. En este caso se creará el archivo test existe en el directorio ireport/database y contiene los comandos necesarios para crear la base de datos. Puede ser una ruta relativa o absoluta de disco. (**file:data/demo\_db**).

## Iniciar la base de datos en Windows:

En el caso de Windows para disponer de la base de datos debes descargarla de su origen en:

[Base de datos](#)

En la que encontrarás un enlace a la versión más actual de la base de datos.

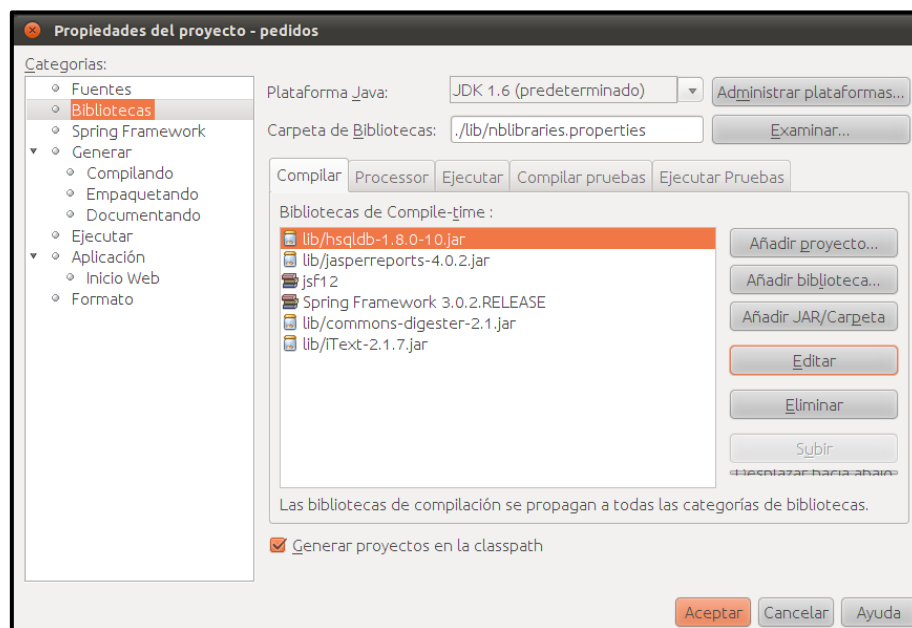
Cuando la tengas descomprímela y ejecuta la siguiente orden para inicializarla en el directorio donde esté:

```
java -cp lib/hsqldb.jar org.hsqldb.Server -database.0  
file:data/demo_db -dbname.0 xdb
```

### Utilización de la base de datos de prueba en NetBeans:

Ahora podemos crear una conexión a la base de datos de prueba desde NetBeans de igual forma que lo hemos hecho en iReport. Si bien será necesario añadir el driver para bases de datos Hsqldb que se encuentra disponible en el archivo **hsqldb-1.8.0-10.jar** del directorio **/ireport/ireport/modules/ext/** en Linux, en Windows puedes usar el archivo **.jar** que has descargado.

Simplemente añadiendo la librería al proyecto estará disponible:



# Anexo III - Despliegue del informe en una aplicación Java

Antes de comenzar es preciso que instales y configures NetBeans para poder usar el complemento de iReport como veíamos al principio de la unidad y que inicies el servidor HSQLDB con la base de datos de prueba.

En el proyecto que crees debes añadir las librerías: **hsqldb**, **iText**, **jasperreports**, **common-digester**, **common-collections** y **spring**. Se pueden encontrar en el subdirectorio de iReport **/ireport/modules/ext**, si estás usando Linux, si usas Windows y nos las tienes ya instaladas puedes encontrarlas fácilmente en internet.

Cuando hemos finalizado el proceso de configuración, añadimos un archivo nuevo a nuestro proyecto, dentro de la categoría report, de tipo empty iReport. Crearemos un informe parametrizado, con las siguientes características:

1. Creamos un parámetro llamado año, de tipo **java.lang.integer** con valor por defecto 1998.
2. La consulta sobre la base de datos es:

## Código:

```
select * from orders
where year(orderdate)=$P{año}
order by shipcountry
```



3. Generamos un informe parecido a este, que tiene un agrupamiento llamado país sobre el campo **shipcountry**:

<h1>Pedidos por pais</h1>			
Año: \$P{ año}		Page Header	
		new java.util.Date()	
<h2>\$F{SHIPCOUNTRY}</h2>			
Group Header 1			
ORDERDATE	SHIPNAME	SHIPCITY	FREIGHT
\$F{ORDERDATE}	\$F{SHIPNAME}	\$F{SHIPCITY}	\$F{FREIGHT}
Detail			
Pais Group Footer			
Total embarcado para el pais:			\$V{FREIGHT_1}
Column Footer			
Total embarcado en esta página:			\$V{FREIGHT_2}
Page Footer			
Total embarcado:			\$V{FREIGHT_3}
Summary			

4. Añadimos un archivo java al proyecto con este código.

```
import java.awt.Desktop;
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.swing.*.*;
import java.util.HashMap;
import java.util.Map;
import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.view.save.JRPdfSaveContributor.*;
import net.sf.jasperreports.view.JRViewer.*;
import
net.sf.jasperreports.view.save.JRMultipleSheetsXlsSaveContributor.*;

public class pedidosAño {
    public static Connection conexion = null;
    String baseDatos = "jdbc:hsqldb:hsqldb://localhost";
    String usuario = "sa";
    String clave = "";
    //Constructor que conecta a la base de datos de prueba
    public pedidosAño() {
        try {
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
            conexion =
DriverManager.getConnection(baseDatos,usuario,clave);
        } catch (ClassNotFoundException cnfe){
            System.err.println("Fallo al cargar JDBC");
            System.exit(1);
        } catch (SQLException sqle){
```

```

        System.err.println("No se pudo conectar a BD");
        System.exit(1);
    } catch (java.lang.InstantiationException sqlex){
        System.err.println("Imposible Conectar");
        System.exit(1);
    } catch (Exception ex){
        System.err.println("Imposible Conectar");
        System.exit(1);
    }
}
//El método ejecutar recibe el parametro del informe
public void ejecutar(int año) {
    //Ruta del informe respecto del proyecto NetBeans
    String archivojasper="./informes/pedidos.jasper";
    try {
        //Cargamos los parametros del informe en una tabla Hash
        Map parametros = new HashMap();
        parametros.put("año",año);

        //Generamos el informe en memoria
        JasperPrint print =
        JasperFillManager.fillReport(archivojasper, parametros, conexion);
        // Exporta el informe a PDF
        JasperExportManager.exportReportToPdfFile(print,
        "informe.pdf");
        //Abre el archivo PDF generado
        File path = new File ("informe.pdf");
        Desktop.getDesktop().open(path);
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null, e.toString(), "Error",
        JOptionPane.WARNING_MESSAGE);
    }
}
}
}

```

Completamos el método **main** con este código:

```

pedidosAño informe = new pedidosAño();
int año=1998;
informe.ejecutar(año);

```

Si creamos un formulario para seleccionar el año mediante un campo de texto o lista desplegable y ejecutamos el formulario desde un botón tendremos una aplicación gráfica para generar nuestro informe parametrizado.

# ÍNDICE

---

<b>INFORMES .....</b>	<b>1</b>
<b>INFORMES INCRUSTADOS Y NO INCRUSTADOS .....</b>	<b>3</b>
<b>GENERACIÓN DE INFORMES DE FORMA AUTOMÁTICA: HERRAMIENTAS .....</b>	<b>5</b>
JASPER REPORTS + IREPORTS.....	6
INTERFAZ DE USUARIO DE IREPORT .....	7
ELEMENTOS ESTRUCTURALES DE UN INFORME .....	9
INICIAR EL ORIGEN DE DATOS .....	10
CREACIÓN DE UN INFORME SENCILLO .....	11
GESTIÓN DE ERRORES.....	11
FORMATOS DE SALIDA .....	12
<b>OPERACIONES SOBRE LOS INFORMES.....</b>	<b>14</b>
USO DE PARÁMETROS EN UN INFORME .....	14
VALORES CALCULADOS .....	17
INFORMES CON AGRUPAMIENTOS.....	21
SUBTOTALES.....	22
SUBINFORMES .....	23
AÑADIR IMÁGENES .....	24
GRÁFICOS.....	26
INFORMES SOBRE CONSULTAS COMPLEJAS.....	27
<b>ANÁLISIS DEL CÓDIGO OBTENIDO .....</b>	<b>29</b>
<b>REPASO A LA LIBRERÍA JASPER REPORT.....</b>	<b>31</b>
<b>ANEXO I- INSTALACIÓN DEL PLUGIN DE IREPORT PARA NETBEANS .....</b>	<b>33</b>
<b>ANEXO II - CÓMO UTILIZAR LA BASE DE DATOS DE PRUEBA DE IREPORT.....</b>	<b>37</b>
<b>ANEXO III - DESPLIEGUE DEL INFORME EN UNA APLICACIÓN JAVA.....</b>	<b>39</b>