Caso práctico

Ministerio de Educación y Formación Profesional
(Elaboración propia)

Ana sigue inmersa en el proceso de desarrollo de una aplicación de gestión sobre una base de datos con un entorno gráfico muy potente. Han hecho un buen trabajo, cuentan con una interfaz de usuario depurada y muy ergonómica, que ha costado no pocas horas de trabajo y que funciona sobre una base de datos con un diseño impecable. Por si fuera poco, la aplicación puede ejecutarse sin problema en diferentes plataformas. Llegados a este punto, el siguiente paso es dotar a la aplicación de un mecanismo que permita generar documentación con información relevante para la empresa, como: listados, recuentos, o resúmenes, incluso gráficos.

Al comentar el caso con el equipo, **Juan** le recuerda la importancia que tiene para cualquier empresa disponer de una información precisa y actualizada sobre su estado en el proceso de toma de decisiones, y no solo eso, sino que la información se debe presentar en un formato accesible para personas no relacionadas con el mundo de la informática. Por ejemplo, para poder decidir sobre la política de aprovisionamiento, en cualquier empresa se debería hacer un estudio previo de la evolución de las ventas de su producto.

Después de escuchar las aportaciones de **Juan**, **Ada** decide que el mejor modo de resolver este problema es instalar e integrar en el entorno de programación un motor de informes, que le permita generar no sólo los listados sencillos que necesitan, sino otros más complejos que impliquen la realización de cálculos o de agrupamientos, o la generación de algún gráfico.

Ministerio de Educación y Formación Profesional
(Elaboración propia)

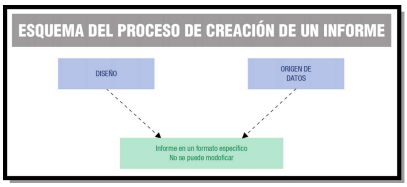
1.- ¿Qué es un informe?

Caso práctico

Al salir de la reunión, **Ana** tiene claro que debe instalar un motor que le permita generar informes, pero aún no comprende muy bien qué es lo que tiene que hacer, ¿qué es exactamente un informe?, ¿qué necesita para obtenerlo?, ¿por qué es tan necesario que su completa aplicación genere listas con información que puede obtener de la base de datos?

Ministerio de Educación y Formación Profesional
(Elaboración propia)

Un informe es un documento que permite mostrar el contenido de un origen de datos 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.



Ministerio de Educación y Formación Profesional
(Elaboración propia)

Normalmente, ***un informe se genera a partir de un*** **diseño 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.

¿Por qué los informes son un elemento fundamental en cualquier modelo de negocio?

- 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.

Autoevaluación

¿Quién determina el aspecto final del informe?

- ○ El origen de datos.
- ○ La plantilla de diseño.
- ○ El formato del archivo final.
- ○ La herramienta seleccionada.

No es correcta, el origen de datos determina el contenido del informe.

Efectivamente la plantilla de diseño contiene la distribución y diseño de los elementos que forman el informe y que determinan su aspecto final.

No es la respuesta correcta, independientemente del formato que escojamos tendremos un aspecto más o menos similar al final.

No es así, el diseño es independiente de la herramienta.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

2.- Informes incrustados y no incrustados.

Caso práctico

De acuerdo, necesitamos los informes para presentar información con una estructura y un formato que cualquier persona que no sepa manejar una base de datos pueda entender. Y ahora, ¿cómo añadimos un informe a una aplicación que hayamos implementado?

Ministerio de Educación y Formación Profesional
(Elaboración propia)

Paul Sherman (Dominio público)

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, 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 el proyecto, tanto el informe como su clase contenedora se incrustan en el ensamblado, 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 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.
  - El informe puede estar expuesto a través de un servicio web 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.

Autoevaluación

¿En qué caso no necesitamos un archivo con la definición del informe para integrarlo en una aplicación?:

- ○ En el caso de los informes incrustados.
- ○ En el caso de los informes no incrustados.
- ○ En ambos casos.
- ○ En ninguno de los dos.

No es cierto, aunque exista una clase contenedora en el proyecto necesitamos un archivo con la definición del informe para importarlo.

No es cierto, el informe se crea de modo independiente al proyecto y se necesita de un archivo con la definición.

No es correcto. El archivo con la definición se va a necesitar siempre.

Efectivamente en cualquier caso necesitaremos un archivo con la definición del informe, tanto si lo cargamos en una clase propia como si lo hacemos por otros medios.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

3.- Generación de informes de forma automática: herramientas.

Caso práctico

Ana ya tiene un poco más claro qué es un informe, para qué sirve y cómo lo puede integrar en su aplicación. Ahora es el momento de buscar entre los distintos motores para la generación de informes aquel que se adecúa más a sus necesidades, para empezar a probarlo. BK siempre ha apostado por el software libre, y, aunque están abiertos a cualquier formato, suelen desarrollar sus aplicaciones en Java.

Ministerio de Educación y Formación Profesional
(Elaboración propia)

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'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.

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.

Enlace a la página de Crystal Reports.

JasperReport + iReport:

María José Navascués González. (Elaboración propia)

Software libre perteneciente a JasperSoft. iReport genera archivos XML (con extensión `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 puedes ver más en la siguiente página.

Enlace a la página de JasperSoft para iReport.

Eclipse Birt:

BIRT son las siglas de Bussiness Inteligence 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.

Enlace a la página de BIRT.

3.1.- JasperReports + 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 AGPL. Si quieres conocer los términos para esta licencia puedes hacerlo en su web:

Licencia AGPL.

María José Navascués González. (Elaboración propia)

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.

Debes conocer

Instalación de JasperReports en NetBeans.

Para poder utilizar JasperReports a través de NetBeans es necesario instalar los plugins y librerías de JasperReports e iReports. Los plugins y las librerías necesarias las puedes descargar desde el siguiente enlace: Enlace para descargar librerías y plugins. (No olvides descomprimir el fichero para utilizarlo).

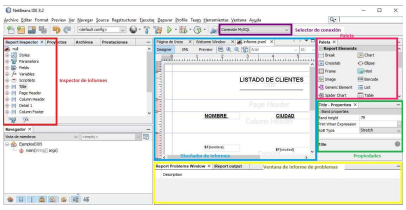
Comenzaremos por instalar los plugins de NetBeans, para ello, dentro de NetBeans seleccionamos la opción Herramientas -> Plugins --> Descargados --> Agregar Plugins. Seleccionamos los ficheros que nos hemos descargados.

A continuación, procederemos a instalar las librerías necesarias. Para ello, pulsamos Herramientas --> Librerías --> Nueva Librería. Asignamos un nombre, por ejemplo, JasperReport 6.11. Pulsaremos el botón Agregar jar / carpeta y seleccionaremos todas las librerías. Para finalizar pulsaremos Aceptar.

Una vez que tenemos configurado NetBeans, crearemos un nuevo proyecto. Por ejemplo, le llamaremos EjemplosDI05 y será de tipo Aplicación Java.

3.2.- Interfaz de usuario de iReport.

Al ejecutar iReport, vemos una interfaz parecida a la que aparece a continuación:



Montaña Martín Vergel (Elaboración propia)

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).
El **diseñador de informes** permite diseñar visualmente el informe de arrastrando, posicionando, alineando y cambiando el tamaño de los elementos del informe.
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.
La **hoja de propiedades** se utiliza para establecer las propiedades del componente seleccionado en el informe (como un campo, elemento, banda, grupo, u otros).
La **ventana de informe de problemas** contiene el listado de los errores encontrados al compilar el informe.
Sobre el diseñador de informes está el **selector del origen de datos (selector de conexión)**, que muestra la conexión activa que se utiliza para ejecutar el informe.

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.

Si queremos restablecer la configuración de las ventanas, debemos de seleccionar la opción Ventas --> Restablecer ventanas.

3.3.- 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:

María José Navascués González. (Elaboración propia)

Título (Title): 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 (Page Header): aparece en la parte superior de cada página. Puede contener información como la fecha y hora, nombre de la organización, etc.

Encabezado de columna (Column Header): 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 (Detail): 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 (Column Footer): Puede presentar información resumida para cada uno de los campos. Por ejemplo: "Beneficio total del mes: 1245".

Pie de página (Page Footer): 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 (Summary): 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 delimita cómo se mostrará la información, determina qué elementos se repiten (la banda Detalle, por ejemplo) y dónde

se colocan (encabezados y pies de página).

Autoevaluación

¿En qué banda aparecerían los registros obtenidos de una tabla mediante una consulta `select`?

- ○ En el Encabezado de columna.
- ○ En la banda de Detalle.
- ○ En el pie de columna.
- ○ En la banda de resumen.

No es cierto, en esta banda aparecen los nombres de los campos.

Así, en esta banda aparecen los registros seleccionados de la base de datos.

No es correcto. Esta banda se usa para mostrar datos resumidos, como totales o promedios.

No es correcto. En esta banda se suelen poner gráficos, o datos resumidos.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

3.4.- 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 acceder a los datos que vamos a utilizar para construir nuestros informes.

Para poder realizar los ejemplos de esta unidad, necesitarás tener instalado una base de datos y que se encuentre accesible desde NetBeans. Vamos a utilizar una base de datos creada con MySQL.

> 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.

A lo largo de esta unidad, vamos a utilizar la base de datos Fabrica perteneciente a la compañía SumiMetalicos S.A. Se trata de una empresa que se dedica a la fabricación de artículos para ventas en ferreterías. La empresa recibe pedidos de otras empresas ubicadas en diferentes ciudades. La base de datos tiene las siguientes tablas:

1-. Clientes: almacena información sobre los clientes a los cuales suministra artículos la fábrica.

2-. Artículos: almacena información sobre los artículos que vende la fábrica.

3-. Pedidos: Almacena la información sobre los pedidos que ha recibido la fábrica.

4-. Detalle_Pedidos: almacena los artículos que se han realizado en cada pedido.

5-. Emails: almacena los emails de contacto de las empresas clientes.

6-. Teléfonos: almacena los números de teléfonos de los contactos de las empresas clientes.

La estructura de la base de datos junto con los datos, que se emplearán en los ejemplos de esta unidad, la puedes descargar desde el siguiente enlace: Estructura y datos de la base datos SumiMetalicos S.A

Para poder realizar los ejemplos de esta unidad, utilizaremos una base de datos MySQL. Por lo que es imprescindible que tengas instado MySQL. Te recomendamos que instales XAMPP, es una forma fácil de tener instalado MySQL junto con la herramienta phpMyadmin que nos permitirá gestionar la base de datos desde un navegador.

Te recomendamos que importes en MySQL el fichero descargado. De esta forma, tendrás los datos y la estructura de la base de datos creada correctamente.

Una vez que tenemos creada la base de datos y nos aseguramos que está corriendo MySQL, procederemos a configurar la conexión de nuestro proyecto con la base de datos. Seleccionamos la opción *Report DataSources* y procedemos a configurar una nueva

conexión a base de datos. Para ello, pulsamos *New* (Nuevo) --> *DatabaseJdbcConnection* y completamos los datos necesarios para configurar la conexión:

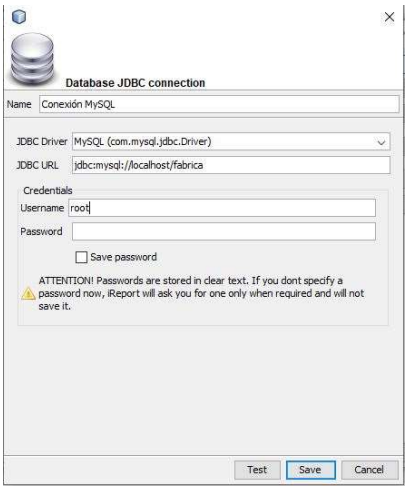
Nombre (**Name**): Nombre simbólico que le vamos a asignar a la conexión. Por ejemplo, "Conexión MySQL".

JDBC Driver: seleccionamos Mysql (com.mysql.jdbc.Driver)

JDBC URL: configuramos la ruta de acceso a la base de datos. Si nuestra base de datos se llama fábrica, debemos de indicar: jdbc:mysql://localhost/fabrica e introduciremos las credenciales necesarias para acceder. Recuerda, que el usuario que utilizamos por defecto a través de PhpMyAdmin es el usuario root y no tiene contraseña asignada. Por lo tanto, lo dejamos en blanco.

Username: nombre de usuario con el cual accederemos a la base de datos.

Password: contraseña del usuario con el cual accederemos a la base de datos.



Montaña Martín Vergel (Elaboración propia)

Para poder continuar, necesitamos que al seleccionar el botón *Test*, la comunicación sea correcta.

Debes conocer

Para poder seguir los contenidos de este punto es imprescindible que tengas conocimientos básicos de MySQL, puesto que es necesario que lo tengas instalado y que crees una base de datos para hacer el ejemplo. En estos enlaces tienes toda la información acerca de MySQL y de la herramienta XAMPP, que además de MySQL instala el servidor web Apache, PHP y la herramienta phpmyadmin para gestionar MySQL en modo gráfico.

Página oficial de MySQL.

Página oficial de XAMPP.

3.5.- Creación de un informe sencillo.

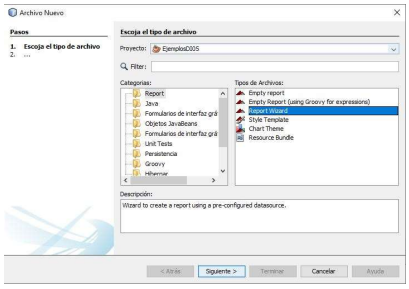
El proceso de creación de informes consta de los siguientes pasos principales:

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.

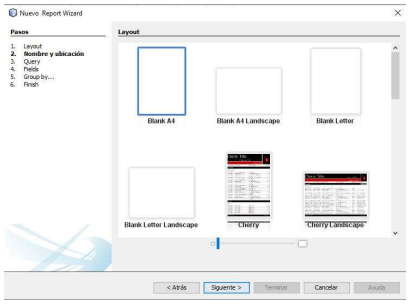
Utilizando el proyecto que hemos configurado en el apartado anterior, para el cual ya tenemos configurado el acceso a la base de datos, vamos a ver cómo construir un informe sencillo.

Agregaremos un nuevo fichero a nuestro proyecto que utilizaremos para crear el informe. Para ello, en NetBeans, seleccionamos *Archivo* --> *Archivo Nuevo* --> *Categorías*, seleccionamos *Report* --> *ReportWizard.*



Montaña Martín Vergel (Elaboración propia)

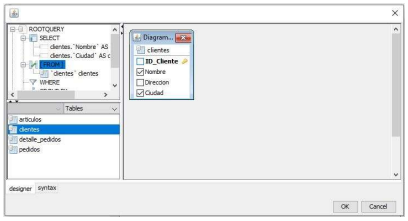
Podemos ver que existen diferentes formatos predefinidos que podemos utilizar, En nuestro caso, vamos a proceder a crear un informe vacío. Para ello, seleccionamos Blank A4 y pulsamos Siguiente.



Montaña Martín Vergel (Elaboración propia)

A continuación, introducimos el nombre que va a tener nuestro fichero dentro del proyecto. Por ejemplo, podemos asignarle el nombre "Informe.jrxml". Podemos indicar en que carpeta

podemos almacenar el fichero, el nombre del proyecto en donde queremos crearlo y la ruta absoluta del fichero que se va a crear.



Montaña Martín Vergel (Elaboración propia)

A continuación, seleccionamos la conexión que queremos utilizar, que en nuestro caso se denomina "Conexión MySQL" e introduciremos la orden SQL que utilizaremos para recuperar los datos que se van a presentar en el informe. En nuestro caso vamos a realizar una consulta a la base de datos para recuperar el nombre de nuestro cliente junto con la ciudad en la que se encuentran. Para ello, escribiríamos la orden SQL: *Select Nombre, Ciudad from clientes*

También podemos utilizar un asistente para crear o diseñar la consulta que queramos realizar contra la base de datos. Para ello, pulsaremos la opción *Design Query*. Al seleccionarlo, deberemos de introducir la contraseña del usuario que definimos en la configuración de la conexión en el caso de que no marcáramos la opción de guardar contraseña.

A continuación, podemos indicar que campos queremos que aparezcan en el informe. A pulsar Siguiente, podemos agrupar la información que queremos mostrar. De esta manera podemos clasificar los resultados obtenidos a la hora de presentarlos en el informe. Para finalizar el diseño del informe, pulsaremos Terminar.

Llegado este momento, ya tenemos creado el informe vacío y configurado para que pueda recuperar la información que vamos a presentar en el informe. Para finalizar, la creación del informe, deberemos de indicar que información queremos que se nos muestre en el informe. Para ello, podemos utilizar el panel *Report Inspector* para diseñar el informe. A través de este panel, podemos seleccionar los elementos que queramos que aparezcan en el informe. Para ello, solo hay que seleccionarlo y arrastrarlo hacía la parte del informe en donde queramos que se muestren.

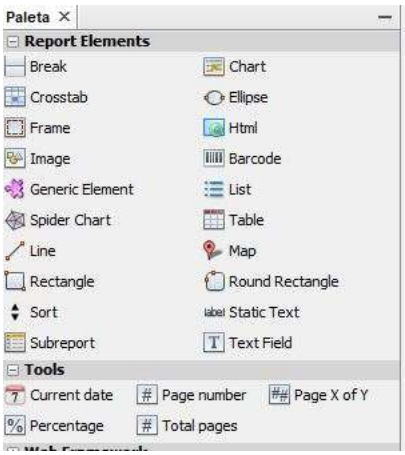
Dentro del panel Report Inspector, podemos acceder al apartado ***Fields***, en donde se nos mostrará los campos que definimos en la configuración de la consulta a la base de datos. Estos campos deberemos de incluirlos dentro de cada banda del informe en donde queramos que se muestre.

Montaña Martín Vergel (Elaboración propia)

Cuando el objeto de campo se arrastra dentro de la banda de detalle, se crea un elemento de campo de texto y establece el campo de texto la expresión de ese elemento a

$F{Nombre_Campo}. Esta es una expresión simple para imprimir el valor del campo Nombre_Campo.

Para añadir otros elementos (tales como líneas o etiquetas), arrástralos desde la Paleta **Reports Elements** en la vista del diseñador para, a continuación, cambiar el tamaño y organizarlos como desee.



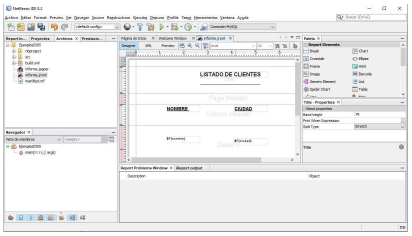
Montaña Martín Vergel (Elaboración propia)

Desde la ventana de **Propiedades** se podrán modificar en su aspecto (color, tipo de fuente, tamaño, etc).

En este caso, añadimos una etiqueta en la banda de título para el título del informe, se añade etiquetas de las columnas con elementos de la etiqueta colocada en la banda de cabecera de la columna, y colocamos una fina línea justo debajo de los campos de texto en la banda de detalle.

Por último, cambiamos el tamaño de algunas bandas y eliminamos los demás mediante el establecimiento de una su altura a 0. Cambiaremos la altura de una banda arrastrando su borde inferior. Una forma directa para reducir la altura de la banda es hacer doble clic en su borde inferior, esto se establece en el borde inferior de su elemento más bajo.

Para ver el resultado del informe que hemos creado solo hay que pulsar el botón **Preview.** Al pulsar este botón, se compila primero el fichero fuente jrxml. Si la compilación es correcta, el archivo producido se carga y se llena con la conexión activa.



Montaña Martín Vergel (Elaboración propia)

Ejercicio resuelto

Crea un informe de tipo listado por cada una de las tablas de la base de datos que hemos utilizado en el ejemplo anterior. Añade diferentes elementos de la paleta y modifica las propiedades de campos y objetos para mejorar el diseño. Ejecuta los informes y exporta su contenido a archivos PDF y a hojas de cálculo.

Mostrar retroalimentación

Para ayudarte con esta tarea puedes recurrir a los siguientes recursos (tutorial en inglés de la página oficial de iReport):

Tutorial de iReport.

3.6.- Gestión de errores.

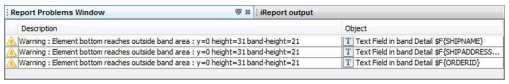
Al generar la vista previa, iReport realiza una serie de operaciones para crear el informe final.

Ministerio de Educación y Formación Profesional (Elaboración propia)

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:



Montaña Martín Vergel (Elaboración propia)

Autoevaluación

¿Desde cuándo necesitamos que el origen de datos esté activo, en el proceso de elaboración de un informe?

- ○ Desde el primer momento cuando vamos a comenzar a diseñar el informe.
- ○ Cuando vayamos a generar el informe.
- ○ Cuando vayamos a cargar el informe en la aplicación de usuario.

Así es, es imprescindible tener activo el origen de datos porque lo necesitamos para obtener los registros y los nombres de los campos.

No es correcto, puesto que necesitaremos antes los nombres de los campos para diseñar el informe.

No es correcto. A veces, ni siquiera cargaremos el informe desde una aplicación de usuario.

Solución

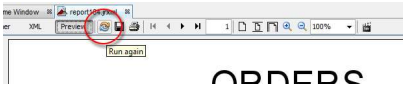
1. Opción correcta
2. Incorrecto
3. Incorrecto

3.7.- 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 vuelca 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.

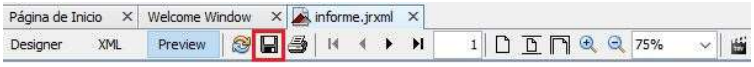
En concreto iReport gestiona esto desde el menú Vista previa (**Preview**), donde podemos seleccionar el formato final del informe.

Para generar 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.



Montaña Martín Vergel (Elaboración propia)

Para almacenar el resultado en un formato diferente al que aparece por defecto, por ejemplo, PDF debemos de pulsar el botón Guardar (Save) y elegir el formato de salida que queramos utilizar.



Montaña Martín Vergel (Elaboración propia)

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--> iReport --> visualizadores (Viewers)**.

4.- Operaciones sobre los informes.

Caso práctico

Ana continúa con la investigación. Ya ha creado un informe con un conjunto de datos sencillo obtenido de una única tabla de una base de datos, pero sabe que para su trabajo no bastará con eso.

Además, va a necesitar hacer informes basados en consultas de varias tablas, y, en otros casos, informes que resuman la información haciendo cálculos como sumas o medias de datos.

Ministerio de Educación y Formación Profesional
(Elaboración propia)

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 mejor el formato de presentación, entre otras cosas veremos cómo:

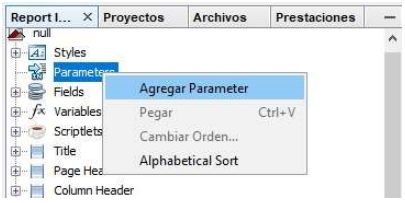
- Hacer cálculos sobre los datos para mostrar información resumida, como promedios o sumas.
- Parametrizar el informe para que se pregunte al usuario que datos desea mostrar.
- Filtrar los datos.
- Aplicar encabezados y pies de página a las hojas del informe.
- Etc.

4.1.- 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.

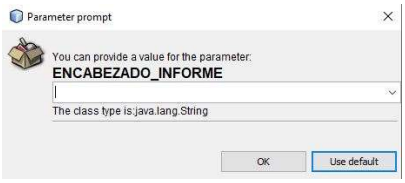
Para administrar los parámetros, utilizamos el apartado `Parameters` del inspector de informe. Desde aquí es posible agregar y quitar parámetros utilizando el menú contextual. Podemos modificar un parámetro seleccionándolo y editando sus propiedades. Podemos modificar entre otras cosas:



Montaña Martín Vergel (Elaboración propia)

- **Name**: nos permite asignar un nombre a un parámetro.
- **Parameter class:** lugar en donde ubicaremos la clase asociada al parámetro.
- **Default Value Expression:** valor predeterminado asociado al parámetro. Esta expresión es evaluada por JasperReports sólo cuando un valor para el parámetro no ha sido proporcionado por el usuario en tiempo de ejecución. Si el parámetro es de tipo String, el valor que introduzcamos debe de ir encerrado entre comillas.
- **Use as prompt:** activaremos esta opción cuando el valor del parámetro sea asignado durante el instante de tiempo en el cual se está generando el informe. En caso de no activarlo, el parámetro tendrá asignado el valor Default Value Expression.

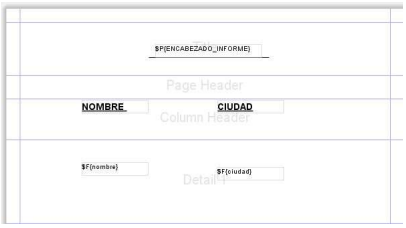
Los parámetros se pueden clasificar en:



Montaña Martín Vergel (Elaboración propia)

- **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 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. En cuanto al tipo de parámetro podemos encontrar:
- **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 **ENCABEZADO_INFORME** de tipo **String** con la propiedad "**Use as 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. Por lo tanto, al generar el informe nos solicitará un valor para este parámetro que se visualizará como título del informe.

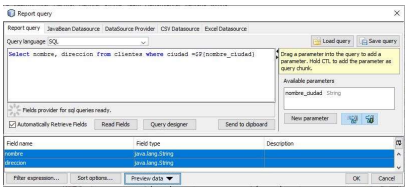


Montaña Martín Vergel (Elaboración propia)

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

4.1.1.- 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:



Montaña Martín Vergel (Elaboración propia)

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

```
select nombre, direccion from clientes where Ciudad = $P{nombre
```

En este ejemplo, **nombre_ciudad** es un parámetro de tipo **`java.lang.String (Text)`** que contiene el nombre de la ciudad debe seleccionar. Este parámetro se puede pasar al informe desde la aplicación que lo ejecuta para seleccionar sólo una ciudad específica.

El parámetro aquí es un verdadero parámetro SQL, lo que significa que la consulta se ejecutará mediante una sentencia como la siguiente:

```
select nombre, direccion from clientes where Ciudad = ?
```

y el valor del parámetro **nombre_ciudad** entonces pasa a la instrucción.

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

```
SELECT * FROM clientes ORDER BY $P!{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 **`$P!{}`**), que será reemplazado por el valor de texto del parámetro (que en este caso puede ser, por ejemplo, "**`Fecha_pedido 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 arbitrario. 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.

4.2.- 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, ***COLUMN_NUMBER*** que contiene el número de columnas, ***PAGE_COUNT*** que contiene el número de registros que han sido procesados por página. 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.

Autoevaluación

¿Cuál es la principal diferencia entre un parámetro y una variable?

- ○ No hay diferencia entre ambas.
- ○ Se colocan en bandas diferentes.
- ○ El valor de una variable puede cambiar a lo largo de la ejecución de un informe mientras que los parámetros no cambian.

No es correcto, son elementos diferentes que funcionan de forma diferente.

No es correcto, tanto un parámetro como una variable se pueden colocar en cualquier zona del informe.

Cierto. Cuando se crea una variable hay que asignarle un tipo, un nombre y un valor inicial. Este valor puede ir cambiando según cuando se evalúe la variable, que dependerá de la configuración de la misma y de la zona donde se coloque.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta

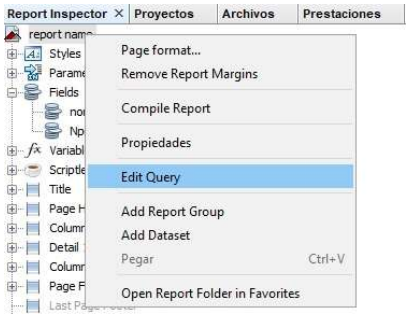
4.2.1.- Recuentos.

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

Realizaremos un informe sencillo que muestra un listado en donde se muestre el número de pedidos realizado por cada cliente. La sentencia SQL que utilizaremos es la que aparece a continuación:

```
Select nombre, count(id_pedido) as Npedidos from pedidos, clientes where pedidos.ID_Cliente =
```

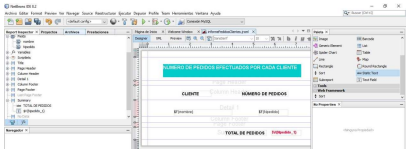
Comenzaremos creando un nuevo informe vacío. Modificaremos la consulta asociada al informe para introducir la consulta anterior. Para ello, podemos acceder al menú contextual sobre la opción report name que aparece en el panel Report Inspector.



Montaña Martín Vergel (Elaboración propia)

Arrastraremos los campos **Nombre** y **Npedidos** a la banda de detalle, en la banda Column Header modificaremos las etiquetas de cada columna para indicar los textos Cliente y Número de pedidos, 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** (Sum) y pulsa Aceptar.



Montaña Martín Vergel (Elaboración propia)

En la banda del título (Title) introduciremos una etiqueta (static text) con el texto NÚMERO DE PEDIDOS EFECTUADOS POR CADA CLIENTE. Podemos modificar el color del texto de la etiqueta y el fondo utilizando las propiedades ForeColor y Bakcolor junto con la propiedad Opaque.

Si pulsamos el botón Preview para visualizar el informe, nos aparecerá:

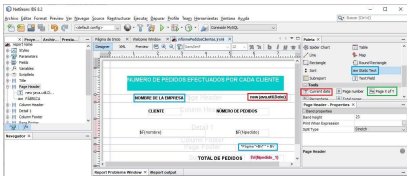

NÚMERO DE PEDIDOS EFECTUADOS POR CADA CLIENTE

CLIENTE	NÚMERO DE PEDIDOS
Ferretería	2
Ferretería La Madrila	2
Ferretería Lavapies	1
Ferretería Ramírez	1
	TOTAL DE PEDIDOS: 6

Montaña Martín Vergel (Elaboración propia)

4.2.2.- Modificar encabezados y pies de página.

Para añadir la **página X de Y** en el pie 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".



Montaña Martín Vergel (Elaboración propia)

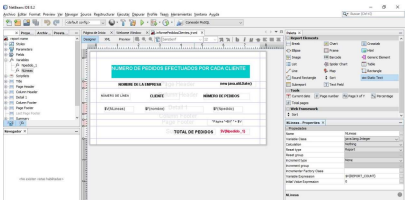
Esta herramienta crea dos campos de texto que muestran la misma variable: **`PAGE_NUMBER`**. El campo de texto primero muestra la página actual, el segundo el total de páginas del informe. Esto es posible porque el tiempo de evaluación de cada campo de texto es diferente, en particular, el primer campo 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, 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 a una banda 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.

4.2.3.- 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 0 (`Init Value Expresion`) y le asignaremos la expresión `$V{REPORT_COUNT}` (`Value Expresion`).



Montaña Martín Vergel (Elaboración propia)

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

Autoevaluación

Relaciona cada expresión con el tipo de elemento del inspector de informe a la que hace referencia, escribiendo el número asociado a la característica en el hueco correspondiente.

Ejercicio de relacionar

Expresión	Relación	Información del inspector de informe
$F.	☐	1. Campo de la consulta.
$P.	☐	2. Variable.
$V.	☐	3. Parámetro.

Enviar

Cada uno de estos elementos se identifican con un elemento del inspector de informe, los campos obtenidos al ejecutar la consulta se identifican con la expresión `$F` (de Field), las variables con `$V` y los parámetros con `$P`.

4.3.- 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 pedidos por ciudad), o puede ser más compleja (Por ejemplo, es posible agrupar un conjunto de clientes por letra inicial).

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 pedidos entregados por ciudad y un grupo anidado que contiene los artículos de cada pedido), que se anidarán según se ordenen en el inspector del informe.

A continuación, veremos dos ejemplos de agrupamiento: uno cuya expresión es un campo de la consulta y otro cuyo agrupamiento es el resultado de una expresión.

4.3.1.- Ejemplo: Informe con agrupamientos simple.

A continuación, vamos a ver un ejemplo de una agrupación simple. Vamos a ver cómo obtener un informe de los pedidos que se han efectuado desde cada ciudad a la fábrica. Por lo tanto, el informe mostrará un listado de pedidos agrupados por ciudades.

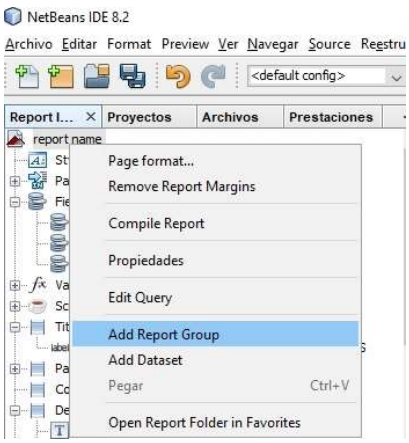
Comenzaremos creando un informe vacío a través de JasperReport. Le asignaremos como nombre PedidosCiudad. De cada pedido mostraremos el número que identifica al pedido (guardado en la base de datos como ID_Pedido) y la fecha del pedido. Al realizar la consulta a la base de datos ordenaremos los datos por el campo ciudad, utilizando la cláusula Order by.

La sentencia SQL que utilizaremos que vamos a utilizar para obtener la información de nuestro informe es:

```
SELECT ciudad, id_pedido, fecha_pedido FROM
clientes, pedidos where clientes.ID_Cliente = pedidos.ID_Cliente
order by ciudad
```

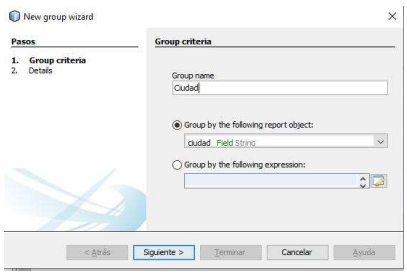
Desde el panel Report Inspector, agregamos los campos Id_pedido y fecha de nuestra consulta en la parte de detalle del informe. Eliminaremos las etiquetas que nos aparecerá en la banda Column Header y pondremos las bandas que no utilicemos a una altura de cero.

Para agrupar los registros por ciudad, añadiremos un nuevo grupo ***Add Report Group.*** Esta opción la podemos encontrar en panel *Report Inspector* si accedemos al menú contextual sobre el nodo raíz, como puede verse en la siguiente imagen:



Montaña Martín Vergel (Elaboración propia)

Crearemos un nuevo grupo, denominado Ciudad, para agrupar por el campo Ciudad. Pulsaremos Siguiente y activaremos las opciones *Add the group header* y *Add the group footer* para añadir las bandas de encabezado y pie de grupo.



Montaña Martín Vergel (Elaboración propia)

Las dos nuevas bandas (encabezado y pie de grupo) aparecerán en la ventana diseño, y los nodos correspondientes se añadirán a la estructura del informe en la vista de esquema.

Montaña Martín Vergel (Elaboración propia)

Arrastramos el campo Ciudad, a la banda encabezado de grupo, podemos modificar su tamaño, transparencia y color de fondo (modificando los valores de las propiedades width, height, opaque y backcolor). Con esto conseguimos que aparezca el nombre de cada ciudad al comienzo de cada grupo.

Arrastramos el campo Ciudad al pie de grupo indicando que queremos utilizar la función de contar (COUNT). Podemos conseguir que los valores aparezcan de color azul, lo seleccionamos y modificamos la propiedad ForeColor.

Montaña Martín Vergel (Elaboración propia)

El diseño final tendrá el siguiente aspecto:

Montaña Martín Vergel (Elaboración propia)

4.3.2.- Ejemplo: Informe con agrupamiento complejo.

En nuestro segundo ejemplo, vamos a obtener un informe en donde se muestren los datos agrupados por el resultado del cálculo de una expresión. En este caso vamos a mostrar en el informe el nombre de cada cliente, la dirección y ciudad agrupados por la primera inicial del nombre de los clientes.

Montaña Martín Vergel (Elaboración propia)

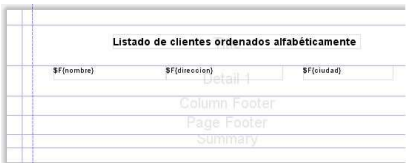
Comenzaremos creando un informe nuevo y vacío con el nombre Ejemplo_agrupamiento_complejo. Configuraremos la sentencia SQL que se va a asociar a dicho informe. Para este ejemplo, utilizaremos la siguiente consulta SQL:

```
SELECT nombre, direccion, ciudad FROM clientes order by nombre
```

Los registros seleccionados ordenados por el nombre de los clientes.

Al informe, le asignaremos como título "*Listado de clientes ordenados alfabéticamente*". Para ello, introduciremos una etiqueta en la banda Title con dicho texto.

Por cada cliente, se mostrará el nombre del cliente (cliente), la dirección del cliente (direccion) y la ciudad del cliente (Ciudad). Por lo tanto, añadiremos los campos nombre, direccion y ciudad a la banda Detail. Eliminaremos las etiquetas que nos aparecerán de forma automática en la banda Column Header. Reduciremos la altura de las bandas que no utilicemos a cero. El diseño del informe quedará de la siguiente forma:

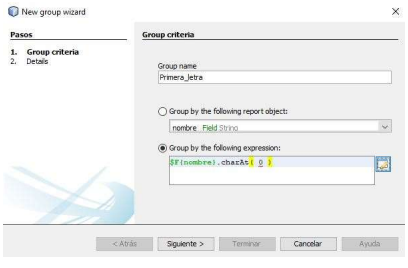


Montaña Martín Vergel (Elaboración propia)

En este ejemplo, añadiremos en el encabezado del grupo, la letra por la que comienza el nombre del cliente de cada persona.

En el pie del grupo mostraremos el número de registros que hay en cada grupo.

Comenzaremos añadiendo un nuevo agrupamiento. Para ello, pulsaremos la opción *Add report group*, de la misma forma que lo hicimos en el ejemplo anterior. Le asignaremos el nombre Primera_letra. En esta ocasión, vamos a configurar el agrupamiento para que sea por el resultado del cálculo de una expresión.



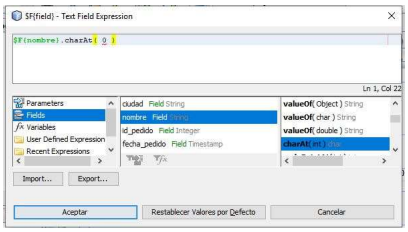
Montaña Martín Vergel (Elaboración propia)

Para ello, activaremos la opción *Group by the following expression*, e introduciremos la sentencia para obtener la primera inicial del nombre del cliente:

```
$F{nombre}.charAt(0)
```

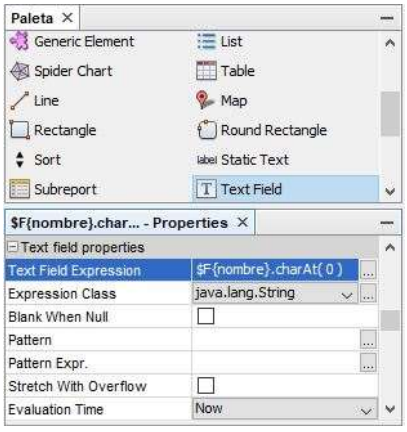
Activaremos las opciones *Add the group header* y *Add the group footer* para mostrar el encabezado de grupo y pie de grupo.

Arrastraremos al encabezado del grupo, desde la paleta, un campo de texto (Text Field) y en su propiedad Text Field Expression introduciremos el valor: `$F{nombre}.charAt(0)`



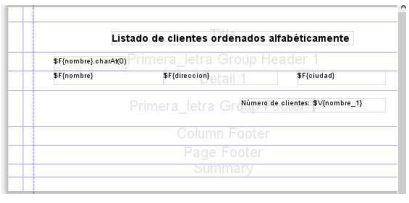
Montaña Martín Vergel (Elaboración propia)

En el pie de grupo introduciremos una etiqueta con el texto "Número de clientes" y el campo nombre, pero indicando que se aplique la fórmula de contar (Count). De esta forma conseguiremos que se muestre el número de clientes por cada agrupamiento.



Montaña Martín Vergel (Elaboración propia)

El diseño del informe quedará de la siguiente forma:



Montaña Martín Vergel (Elaboración propia)

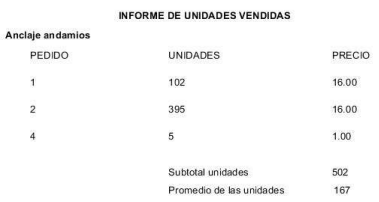
4.4.- Subtotales.

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 al pie del grupo. No obstante, podemos hacer cálculos algo más complejos aprovechando las características de las variables.



INFORME DE UNIDADES VENDIDAS

Anclaje andamios

PEDIDO	UNIDADES	PRECIO
1	102	16.00
2	395	16.00
4	5	1.00
	Subtotal unidades	502
	Promedio de las unidades	167

Montaña Martín Vergel (Elaboración propia)

En el siguiente ejemplo, vamos a añadir a nuestro proyecto, un nuevo fichero de tipo report con el nombre *Informe_Subtotales*. En este informe, vamos a proceder a consultar los datos de los artículos vendidos en la fábrica para mostrar por cada artículo vendido, el número de pedido en el cual se vendió, las unidades que se vendieron en cada pedido y el precio del artículo cuando se efectuó la venta. Por cada artículo que se muestre queremos, además, mostrar el nombre o descripción del artículo, el número de unidades vendidas y el promedio de las unidades vendidas por cada artículo. Al final del informe, mostraremos el número total de unidades que ha vendido la fábrica y el promedio de unidades totales. (Estos dos valores aparecerán al final del informe).

La instrucción SQL que utilizaremos para recuperar la información que queremos mostrar será la siguiente:

```
select descripcion, detalle_pedidos.ID_Articulo, ID_pedido, unidades, detalle_pedidos.precio
order by ID_ARTICULO
```

Por lo tanto, deberemos de configurar dicho report para que utilice la sentencia SQL de la misma forma que lo hemos hecho en los ejemplos anteriores.

A continuación, procedemos a diseñar el informe. Para ello, agregaremos una etiqueta (*Static Text*) a la banda *Title* con el texto "INFORME DE UNIDADES VENDIDAS".

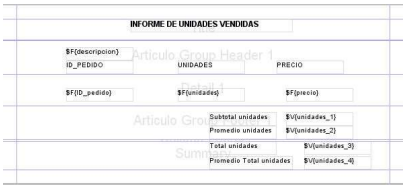
En la banda *Detail*, procedemos a arrastrar los campos ID_pedido, unidades y precio. Creamos un nuevo agrupamiento, pulsando la opción *Add Report Group* del panel *Report Inspector*, con el nombre Articulo y mostraremos tanto el encabezado del agrupamiento como el pie.

En la banda *Group Header*, arrastraremos el campo descripción, para conseguir que se muestre el nombre de los artículos e introduciremos las etiquetas ID_PEDIDO, UNIDADES y PRECIO.

En la banda *Group Footer* introduciremos dos etiquetas con los textos "Subtotal unidades" y "Promedio de las unidades". Acompañando a estas etiquetas, arrastraremos la variable unidades dos veces para indicar en la primera que nos muestre la aplicación de la fórmula suma (*SUM*) y la segunda para indicarle que nos muestre el resultado de aplicar la función promedio (*Average*).

Para conseguir que al final del informe, nos aparezca la suma y el promedio de las unidades totales vendidas, arrastraremos a la banda Summary, de nuevo el campo unidades, para que se aplique la función suma (*SUM*) y la función promedio (*Average*). Ambos valores deberán de ir precedido de una etiqueta (*Static Text*) donde se indique la descripción de los valores.

El diseño del informe final será el que aparece a continuación:



Montaña Martín Vergel (Elaboración propia)

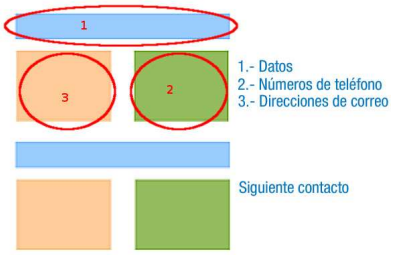
Con la propiedad del agrupamiento ***Start on a new Page*** activada conseguimos que cada agrupamiento comience en una nueva página.

4.5.- 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.

Cuando utilizamos subinformes, tendremos que construir al menos un informe principal y tantos informes secundarios como necesitemos.

Supongamos, con respecto a la base de datos fabrica, que cada empresa que realiza pedidos a la fábrica puede tener varios teléfonos y emails de contactos. Por lo tanto, la base de datos fábrica almacena, para cada cliente, varios teléfonos y emails de contacto. Esta información la guarda en la tabla Teléfonos y Emails.



María José Navascués González (Elaboración propia)

Queremos realizar un informe representando 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 las empresas clientes de la fábrica (zona azul) y después dos subinformes secundarios para los teléfonos y direcciones de correo de dichas empresas.

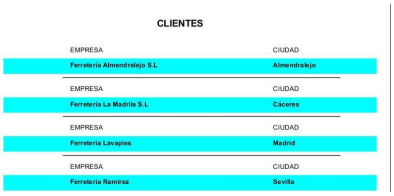
El informe principal se utilizará para seleccionar a los clientes. Crearemos un primer subinforme, para seleccionar las direcciones de correo electrónico asociadas a cada cliente, en la zona de color naranja. Por último, crearemos un segundo subinforme, para obtener los números de teléfono de cada empresa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

Montaña Martín Vergel (Elaboración propia)

En los siguientes apartados de esta unidad vamos a ver cómo crear el informe principal y los dos informes secundarios.

4.5.1.- Ejemplo de informe principal: Datos clientes.

Comenzaremos añadiendo a nuestro proyecto un nuevo fichero report, denominado ***Informe_Principal,jrxml***. Crearemos un informe vacío con este nombre que tendrá conexión con la base de datos Fábrica y nos aseguraremos de tener conexión con dicha base de datos. Recuerda que, para poder tener conexión con la base de datos, el servidor MySQL debe de estar corriendo. Te recomendamos que utilices phpMyAdmin para administrar la base de datos.



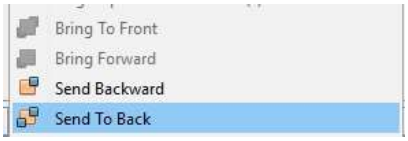
Montaña Martín Vergel (Elaboración propia)

La sentencia SQL que utilizaremos para generar el informe es: `select nombre, ciudad from clientes`

Vamos a insertar en la banda *Title* una etiqueta de texto (Static Text) para introducir un título a nuestro informe. A esta etiqueta le asignaremos el texto Clientes.

En la banda Detail (detalle) vamos a hacer que aparezca el nombre de la empresa junto con la ciudad en donde se encuentra. Por lo tanto, esta banda mostrará todas las empresas clientes de la fábrica junto con su localización.

Para que el informe aparezca más vistoso, vamos a insertar un *Frame* (marco) de color azul. Este marco lo enviaremos al fondo de la



Montaña Martín Vergel (Elaboración propia)

banda para que los valores de los campos nombres y ciudad se puedan visualizar encima del marco. Para enviar el marco al fondo de la banda, seleccionamos el marco, accedemos al menú contextual y seleccionamos la opción *Send To back* (Enviar al fondo). Para modificar el color del marco modificaremos la propiedad *BackColor* y activaremos la opción *Opaque*.

A continuación, arrastramos los campos nombres y ciudad a la banda detail, colocándolo encima del marco azul introducido anteriormente. Por cada empresa que aparezca en el informe, queremos que aparezca los textos Empresas y Ciudad justo antes de los datos de cada empresa. Por esta razón, no incluiremos ninguna etiqueta en la banda *Column Header* y las introduciremos en la banda *Detail* (Detalle).

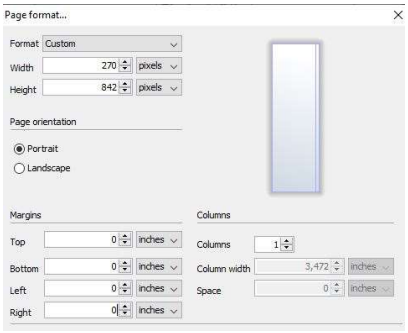
Para diferenciar los datos de cada empresa, vamos a insertar una línea (Paleta, objeto *Line*) después del marco que tendrá de ancho todo el espacio que ocupe el nombre de la empresa junto con la ciudad en donde se ubican. A la línea, le modificaremos la propiedad *Position Type* para asignarle el valor *Float*. Con esto conseguimos que, sea cual sea la altura de los informes, la línea pueda desplazarse y adaptarse en tiempo de ejecución.

Montaña Martín Vergel (Elaboración propia)

4.5.2.- Ejemplo de subinforme: emails de los clientes.

Vamos a crear el primer informe secundario o subinforme.

Para ello, agregaremos a nuestro proyecto, un nuevo fichero de tipo report vacío. Le asignaremos el nombre: Informe_Secundario_Emails.jrxml.

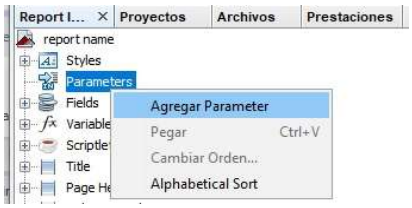


Montaña Martín Vergel (Elaboración propia)

Agregaremos a la banda *Title*, una etiqueta con el texto (*Static Label*) Direcciones de correo.

En los subinfome no son útiles los márgenes y reduciremos el formato de la página a 270 pixeles. Para ello, accedemos a Formato (*Format*)---> Formato de página (*Page Format*)

La altura del informe no es interesante porque será gestionado por el informe principal una vez que este informe sea utilizado como subinforme.



Montaña Martín Vergel (Elaboración propia)

Este informe nos mostrará la lista de correos electrónicos asociados a los datos de la empresa que se muestre en el informe principal. Por lo tanto, tendremos que filtrar las direcciones de correo. Para esto utilizaremos un parámetro. Le asignaremos el nombre ID_Contacto. Para ello, seleccionamos en nodo *Parameters* (Parámetros) y mediante el menú contextual seleccionamos la opción *Agregar Parameter* (*Add parameter*).

Una vez creado el parámetro, en la hoja de propiedades, le asignaremos los valores:

- **Name**: Id_contacto.
- **Clase**: java.lang.Integer.
- **Default Value Expression**: le asignaremos el valor 1.

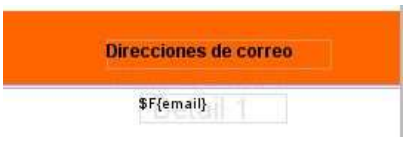
Para generar este informe vamos a utilizar la sentencia SQL:

```
select email from emails where id_cliente = $P{Id_contacto}
```

La sintaxis **`$P{Id_contacto}`** permite el uso de un parámetro dentro de una consulta, en este caso para filtrar el resultado utilizando una condición where.

Arrastraremos a la banda Detail, el campo correo para que nos muestre los valores asociados a dicho campo.

El diseño del informe queda como el que sigue, teniendo en cuenta que hemos eliminado, poniendo su altura a cero, todas las bandas que no usamos:



Montaña Martín Vergel (Elaboración propia)

4.5.3.- Ejemplo de subinforme: teléfonos de los clientes.

Vamos a crear el segundo informe secundario o subinforme. Para ello, agregaremos a nuestro proyecto, un nuevo fichero de tipo report vacío. Le asignaremos el nombre: Informe_Secundario_Telefonos.jrxml.

Agregaremos a la banda *Title*, una etiqueta con el texto (*Static Label*) *Teléfonos*

Reduciremos el formato de la página a 270 pixeles. Para ello, accedemos a Formato (*Format*)---> Formato de página (*Page Format*) y eliminaremos todos los márgenes poniendo a cero el valor correspondiente. De la misma forma, que hicimos en el primer subinforme.

De la misma forma que hicimos en el informe anterior, crearemos un parámetro con el nombre Id_contacto. Le configuramos de la misma forma:

- **Name**: Id_contacto.
- **Clase**: java.lang.Integer.
- **Default Value Expression**: le asignaremos el valor 1.

La sentencia SQL que utilizaremos para obtener los números de teléfono de contacto de cada empresa es:

```
select telefono from telefonos where id_cliente =$P{Id_Contacto}
```

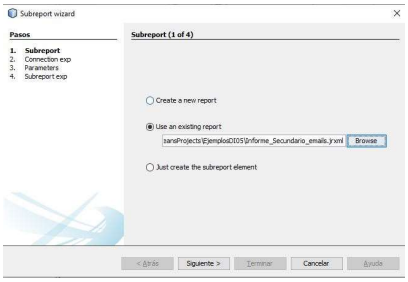
Pondremos a cero la altura de las bandas que no utilicemos y arrastramos a la banda Detail el campo teléfono.

Montaña Martín Vergel (Elaboración propia)

4.5.4.- Unificación de informes y subinformes.

Vamos a proceder a conectar el informe principal con los subinformes. Para ello, accedemos al informe principal cuyo nombre es Informe_principal.jrxml.

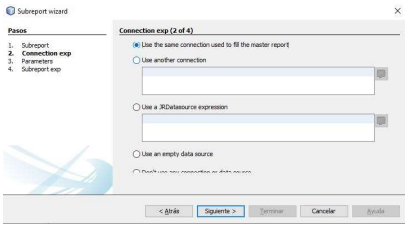
Desde la paleta, arrastramos al diseñador el elemento subinforme (*subreport*) a la banda detalle, veremos cómo se lanza un asistente.



Montaña Martín Vergel (Elaboración propia)

Seleccionaremos la opción Utilizar un report existente y pulsando el botón de Browse (Examinar), localizaremos el primer subinforme que definimos con el nombre Informe_secundario_emails.jrxml.

A continuación, se nos presentará diferentes opciones para indicar cómo se van a recuperar los datos que se van a utilizar. Vamos a seleccionar la primera opción, utilizaremos la misma conexión que el informe principal (*Use the same connection used to fill the master report*). Pulsaremos Next (Siguiente).



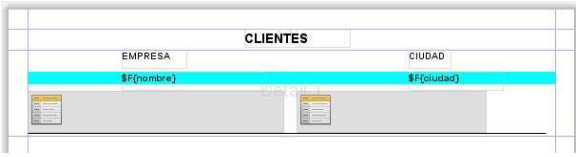
Montaña Martín Vergel (Elaboración propia)

En la siguiente venta, se nos mostrará los parámetros definidos, que en nuestro caso es ID_Contacto y se le tiene que asignar la expresión $F{id_cliente}. Utilizando el desplegable asociado al parámetro ID_Contacto seleccionamos *id_cliente Field Integer.*

Montaña Martín Vergel (Elaboración propia)

Finalizaremos la definición del subinforme, seleccionando la opción *Store the directory name in a parameter* para almacenar el nombre de la carpeta junto con el parámetro.

Para añadir el segundo subinforme, repetimos el proceso y lo colocaremos al lado del primero. El diseño del informe principal tiene que quedar de la siguiente forma:



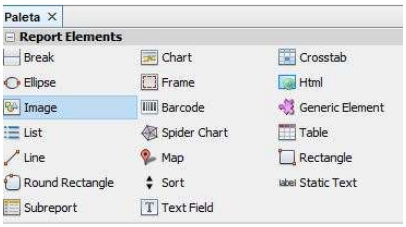
Montaña Martín Vergel (Elaboración propia)

4.6.- 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:

```
"c:\usuarios\alumno\Documentos\Imagenes\informes\flor.jpg".
```



Montaña Martín Vergel (Elaboración propia)

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 Imagen 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:

María José Navascués González (Elaboración propia)

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

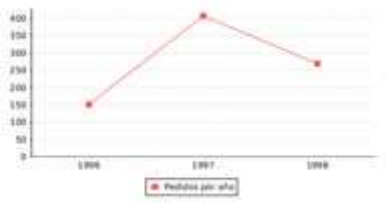
Otra opción es utilizar el **Classpath** Cuando una imagen se encuentra en el **Classpath**, sólo se requiere el nombre de la imagen para encontrarla. De forma predeterminada, al ejecutar un informe, iReport agrega el directorio en el que reside el informe al **Classpath**. Si tenemos el informe y la 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.

4.7.- 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:



María José Navascués González (Elaboración propia)

- **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.

Los gráficos se pueden insertar en cualquier banda de nuestro informe, pero lo más lógico y habitual es insertarlo en la banda de resumen (Summary)

En la imagen anterior puede verse un 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.

En el siguiente apartado, vamos a ver un ejemplo de cómo construir un gráfico con JasperReport.

4.7.1.- Ejemplo de creación de gráficos.

Vamos a ver cómo crear un gráfico de barras que refleje los pedidos agrupados y ordenados por año. Es decir, crearemos un gráfico en donde pueda apreciarse el número de pedidos que se hayan efectuado cada año.

Montaña Martín Vergel (Elaboración propia)

JasperReports soporta un gran número de los gráficos incorporados creados con la popular colección de código abierto JfreeChart. Se incluyen los tipos de gráficos circulares, barras, barras apiladas, línea, área, burbuja, de Gantt, etc.

Vamos a agregar un gráfico de múltiples series en el informe para representar el número de pedidos recibidos cada año. Para empezar, vamos a crear un informe en blanco, denominado *Ejemplo_grafico.jrxml*, basado en la siguiente consulta SQL:

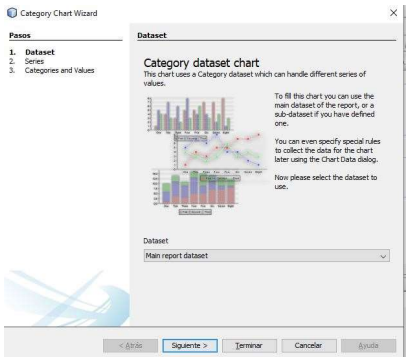
```
select count(id_pedido) as Total, year(fecha_pedido) as Año from pedidos group by year(fecha_
```

La consulta devuelve el número de pedidos realizados cada año. Agrupamos por año de modo que cada grupo va a componer una serie.

Desde la paleta, arrastramos el elemento de gráfico (Char) dentro de la banda de título y seleccione como tipo de gráfico Bar 3D chart (Tres dimensiones).

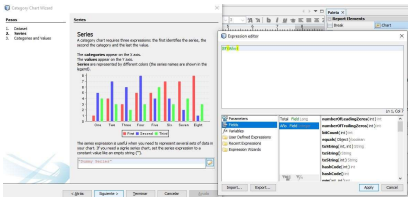
Para generar el informe tenemos que rellenar seguir **tres pasos:**

1.- Seleccionar el **Dataset** que obtendremos del informe principal. Seleccionaremos la opción *Main report dataset.* Esto nos permite recoger los datos de la consulta definida en la configuración del informe.



Montaña Martín Vergel (Elaboración propia)

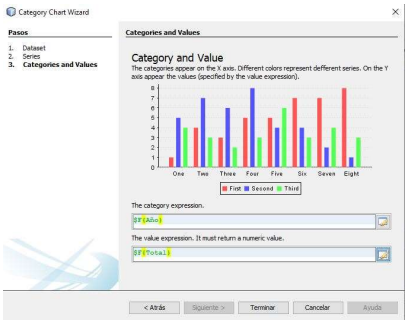
2-. Seleccionar las series, en nuestro caso la serie viene definida por el campo **$F{AÑO}**. Para ello, accedemos al editor y seleccionamos el campo año. Pulsaremos Aplicar (Apply) y el botón Siguiente (Next).



Montaña Martín Vergel (Elaboración propia)

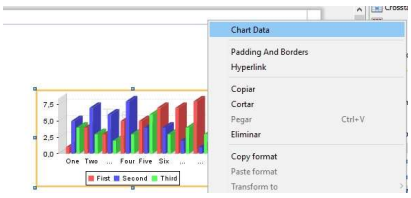
3.- Seleccionar Categorías y valores. En el apartado **Category expression** indicaremos los valores que queremos representar en el eje de abscisas. En nuestro ejemplo, queremos representar los diferentes años para los cuales la fábrica ha recibido pedidos.

El apartado **Value expression**, mostrará los valores para cada uno de los valores que se muestran en el eje de las abscisas. En nuestro ejemplo, queremos mostrar el número total de pedidos recibidos en cada año. Por lo tanto, introduciremos el valor **$F{total}.**



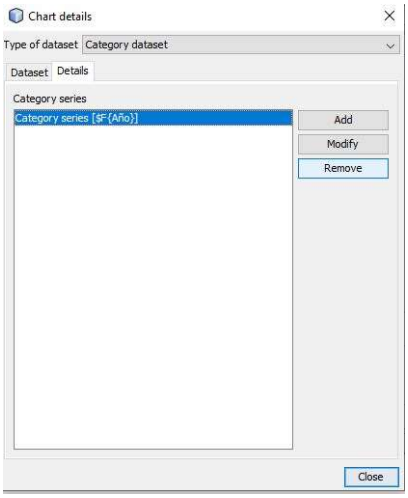
Montaña Martín Vergel (Elaboración propia)

Para finalizar, pulsaremos el botón Terminar. Deberemos de tener en cuenta que, el gráfico se mostrará, cuando visualicemos el informe. Es decir, cuando pulsemos el botón Preview.



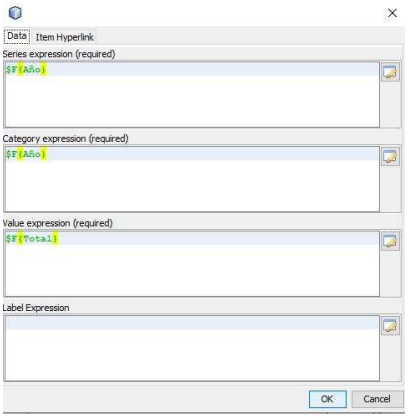
Montaña Martín Vergel (Elaboración propia)

Una vez generado el gráfico que queremos representar en el informe, podemos modificar su configuración si lo seleccionamos y accedemos a la opción ***Chart Data*** que se encuentra en el menú contextual.



Montaña Martín Vergel (Elaboración propia)

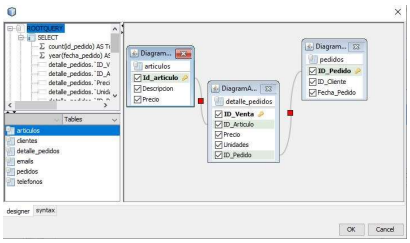
Al acceder a la ventana *Chart Details*, deberemos de pulsar la pestaña ***Detalle*** (Details) y se nos mostrarán las diferentes series que estamos representando. A través de esta ventana podemos añadir nuevas series si pulsamos el botón ***Add***, eliminar series, si se seleccionan, y se pulsa el botón ***Remove*** o modificar las series existentes si pulsamos el botón ***Modify***



Montaña Martín Vergel (Elaboración propia)

4.8.- 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.



Montaña Martín Vergel (Elaboración propia)

- En la zona de la izquierda abajo aparece la lista de las 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.***

6.- Repaso a la librería Jasper Report.

Caso práctico

De acuerdo, ya sabemos con exactitud cuáles son los archivos involucrados y cómo funcionan en la creación del informe, pero ¿cómo se pueden añadir a una aplicación Java generada con NetBeans? **Ana** ha realizado la instalación de iReport como herramienta independiente y también como complemento de NetBeans, ha creado un proyecto Java, ha generado el informe y lo ha compilado. Dispone de un archivo `.jrxml` y un archivo de `jasper`, pero ¿cómo lo pone todo junto?

Ministerio de Educación y Formación Profesional
(Elaboración propia)

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*** a la que añadiremos una pareja formada por el nombre del parámetro y la variable que contiene su valor. Una Hashtable Java es una estructura de datos que utiliza una función hash para identificar datos mediante una llave o clave

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");
```

Para saber más

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

Librería JasperReports.

6.1.- Creación de informe desde una aplicación

Antes de comenzar es preciso que instales y configures NetBeans para poder usar el complemento de JasperReport, si no lo has hecho ya, como veíamos al principio de la unidad y que inicies el servidor MySQL con la base de datos *Fabrica* que hemos venido utilizando en los ejemplos anteriores.

En el proyecto que crees debes añadir las librerías:

Driver MySql JDBC: para incluirla accede al panel proyectos, pulsas el botón derecho del ratón (menú contextual) y seleccionas la opción Agregar Biblioteca. Lo buscamos en la lista que nos aparece y pulsamos Añadir librería (Add Library).

Spring Framework 4.0.1: repetimos el proceso anterior y seleccionamos dicho valor en la lista.

JasperReport 6.11: repetimos el proceso de la misma forma que hicimos con el Driver de MySQL y Spring Framework.

Incluiremos los siguientes ficheros .jar a nuestra Biblioteca. Accedemos, de nuevo, al menú contextual sobre Bibliotecas y seleccionamos Añadir ficheros jar /carpeta. Debemos de incluir los siguientes ficheros jar: **commons-beanutils-1.8.2.jar, commons-collections4-4.4.jar, commons-digester-2.1.jar, commons-logging-1.1.2.jar, itext-2.1.7.jar, joda-time-2.4.jar, jollyday-0.4.9.jar**.

Todos estos ficheros los puedes descargar desde el siguiente enlace: Conjunto de ficheros jars necesarios para el ejemplo.

Lo que pretendemos con este ejemplo, es desde una aplicación Java, construir un informe en formato PDF. Es decir, al ejecutar la aplicación vamos a crear un nuevo fichero, en formato PDF, que contendrá el informe.

Vamos a utilizar el informe PedidosCiudad.jrxml, que creamos de ejemplo durante el desarrollo de esta unidad. Vamos a duplicarlo y le vamos a cambiar el nombre para que se llame *PedidosCiudadParametro.jrxml*. Este informe muestra los pedidos que se han recibido de cada ciudad. Lo vamos a modificar para que reciba un parámetro, que será el año del cual queremos consultar los pedidos. Por lo tanto, obtendremos un informe de los pedidos que se han recibido agrupados por ciudad durante el año indicado.

Utilizando este fichero, accederemos al panel *Report Inspector* para definir el parámetro año. Las propiedades asociadas a este parámetro son:

Name: año

Parameter Class: java.lang.Integer

Default Value Expression:0. (Lo inicializamos con el valor cero.)

A continuación, modificaremos la sentencia SQL asociada a este informe para que reciba por parámetro el año del cual queremos mostrar los pedidos. La sentencia SQL que vamos a utilizar es la siguiente:

```
SELECT ciudad, id_pedido, fecha_pedido FROM
                    clientes, pedidos where clientes.ID_Cliente = pedidos.ID_Cliente
                    and year(fecha_pedido)=$P{año}
                    order by ciudad
```

A continuación, procedemos a modificar el diseño del informe para introducir las etiquetas (static text) para agregar los títulos Número de pedido, Fecha de pedido y Total pedidos. También agregaremos, en la banda Title, el texto AÑO, utilizando una etiqueta (static label) y el valor del parámetro año.

El diseño del informe tiene que quedar de la siguiente forma:

Montaña Martín Vergel (Elaboración propia)

Una vez creado correctamente el informe, procederemos a utilizarlo desde una aplicación en Java. Para ello, vamos agregar a nuestro proyecto un nuevo fichero java, denominado PedidosAnio.java, que contendrá las instrucciones necesarias para establecer una conexión con la base de datos, obtener los datos para crear el informe y crear el fichero PDF en donde se guardará el informe.

```
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.*;
            public class PedidosAnio {
            public static Connection conexion = null;
            String baseDatos = "jdbc:mysql://localhost/fabrica";
            String usuario = "root";
            String clave = "";
            public PedidosAnio(){
            try{
            Class.forName("com.mysql.jdbc.Driver").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);
            }
```

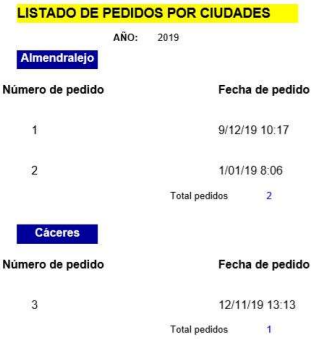
```
catch (java.lang.InstantiationException sqlex){
System.err.println("Imposible Conectar");
System.exit(1);
}
catch (Exception ex){
System.err.println("Imposible Conectar");
System.exit(1);
}
}
//El método ejecutar recibe el parametro del informe
public void ejecutar(int año)
{
//Ruta del informe respecto del proyecto NetBeans
String archivojasper="PedidosCiudadParametro.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, conex
// 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_MESSA
}
}
}
```

Con este código estamos utilizando el informe `PedidosCiudadParametro.jasper` para generar el informe. El fichero resultante de la ejecución del código se almacenará en un fichero llamado informe.pdf

A continuación, vamos a crear un nuevo fichero, llamado EjemploDI05.java, donde tendremos definido el método main con este código:

```
public static void main(String[] args) {
PedidosAnio informe = new PedidosAnio();
int anio=2019;//Valor que se va a pasar para que lo recoja el parámetro
informe.ejecutar(anio);
}
```

Si ejecutamos la aplicación, se nos generará el fichero informe.pdf, cuyo contenido es:



LISTADO DE PEDIDOS POR CIUDADES

AÑO: 2019

Almendralejo

Número de pedido	Fecha de pedido
1	9/12/19 10:17
2	1/01/19 8:06
Total pedidos	2

Cáceres

Número de pedido	Fecha de pedido
3	12/11/19 13:13
Total pedidos	1

Montaña Martín Vergel (Elaboración propia)

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.

5.- Análisis del código obtenido.

Caso práctico

Ana ya tiene una idea bastante clara de qué es un informe, cómo hacer un diseño complejo, bien porque utilice estructuras elaboradas como agrupamientos o porque utilice una consulta que involucre varias tablas. También ha aprendido a añadir elementos que lo hagan más completo y atractivo, como gráficos e imágenes, y todo esto sacando el mejor partido a las posibilidades que le ofrece la herramienta que ha escogido.

Ministerio de Educación y Formación Profesional (Elaboración propia)

Ahora su intención es unir los informes que necesita a la aplicación de usuario que está elaborando. **Juan** le explica que el diseño de los informes con los que trabaja se almacenan en archivos `xml`, pero que antes de poder utilizarlos se deben compilar para optimizar el proceso de mezcla con los datos de la base de datos y la carga en la aplicación. Antes de avanzar, **Ana** comprende que debe analizar todo esto con un poco más de detalle.

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.

Montaña Martín Vergel (Elaboración propia)

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, una colección de

JavaBeans, etc., incluso es 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.