

1. Tras visualizar los diferentes [Vídeos para realizar la tarea](#) incluidos como recursos en la unidad, genera un informe con iReport llamado "facturas". Para ello:

- Utiliza la base de datos "Sample Database (HSQLDB test)" que viene por defecto en iReport.
- El **Jaspersoft iReport Designer 5.6.0 funciona en Windows 10 con JDK 7**. Ambos están disponibles [aquí](#).
- Para completar la instalación, recuerda cambiar el valor de la variable **jdkhome** en el fichero *ireport.conf* (ruta por defecto C:\Program Files (x86)\Jaspersoft\iReport-5.6.0\etc) y borrar el carácter '#'.

```
# ${HOME} will be replaced by user home directory according
default_userdir="${HOME}/.${APPNAME}/5.6.0"
default_mac_userdir="${HOME}/Library/Application Support/${

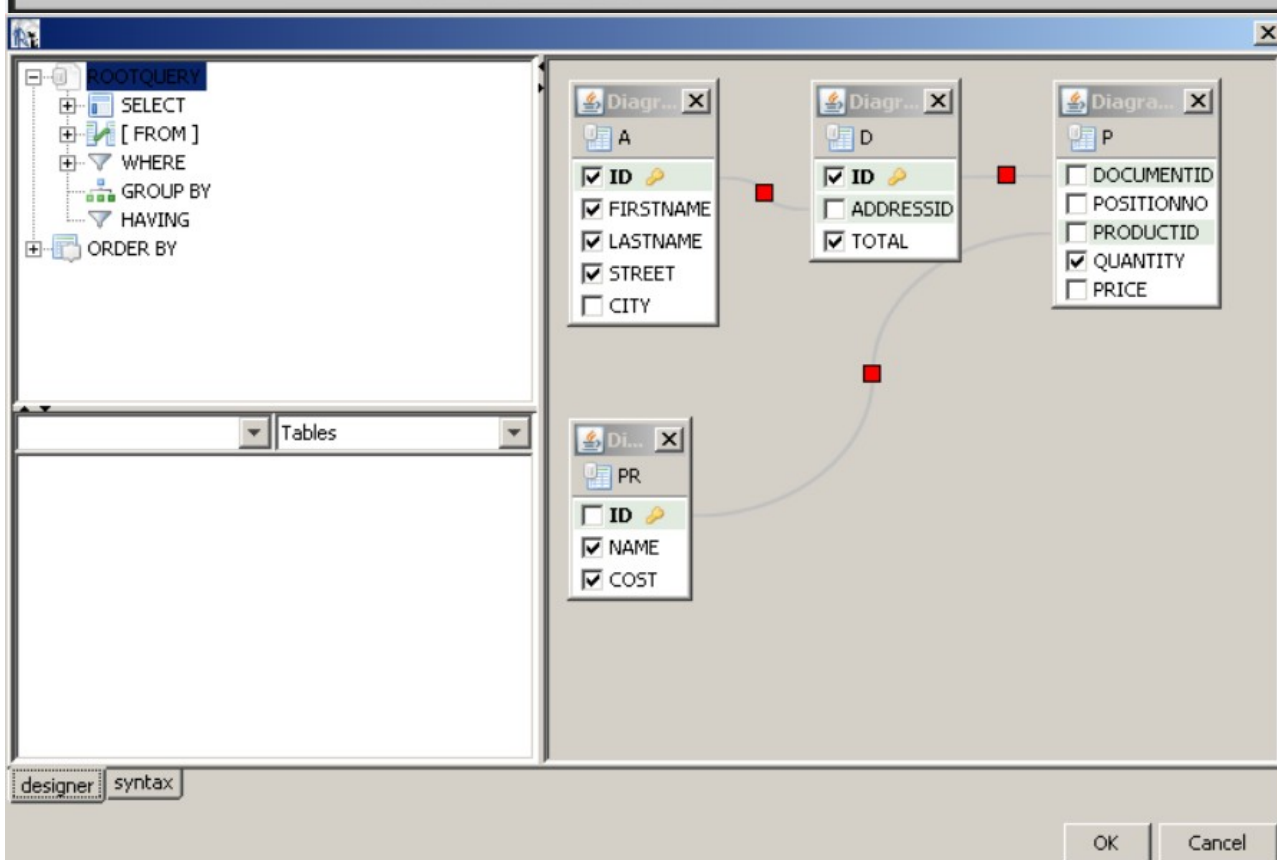
# options used by the launcher by default, can be overridde
# command line switches
default_options="-J-Xms256m -J-Xmx512m -J-Dorg.netbeans.Pro
Dapple.laf.useScreenMenuBar=true -J-Dapple.awt.graphics.Use
# for development purposes you may wish to append: -J-Dnetb

# default location of JDK/JRE, can be overridden by using -
#jdkhome="/path/to/jdk"
jdkhome="C:\Program Files\Java\jdk1.7.0_80"
```

- De la base de datos debes utilizar 4 de sus tablas (DOCUMENT, POSITIONS, PRODUCT y ADDRESS).

En el informe deben aparecer los datos personales (id, nombre, apellidos y dirección) de los clientes de la empresa (tabla ADDRESS), el número de la factura (tabla DOCUMENT), los productos (tabla PRODUCT) con su cantidad, precio y su total, es decir, cantidad * precio (tabla POSITIONS). Por último, añadir un campo con el **importe total a pagar para cada factura** y también el **importe total a pagar por cada cliente**.

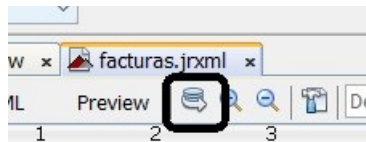
ADDRESS	DOCUMENT	ORDERS	POSITIONS	PRODUCT	TASKS
ID	ID	ORDERID	DOCUMENTID	ID	SERIES
FIRSTNAME	ADDRESSID	CUSTOMERID	POSITIONNO	NAME	TASK
LASTNAME	TOTAL	EMPLOYEEID	PRODUCTID	COST	SUBTASK
STREET	50 rows	ORDERDATE	QUANTITY	50 rows	STARTTIMESTAMP
QTY		REQUIREDDATE	PRICE		ENDTIMESTAMP
50 rows		SHIPPEDDATE	650 rows		PERCENT
		SHIPVIA			16 rows
		FREIGHT			
		SHIPNAME			
		SHIPADDRESS			
		SHIPCITY			
		SHIPREGION			
		SHIPPOSTALCODE			
		SHIPCOUNTRY			
		830 rows			



Abrimos iReport y en la parte superior izq (en el desplegable) seleccionamos **Sample Database**.

Creamos un **nuevo archivo** Archivo>new y seguimos los pasos, se crea el informe.

Hacemos la consulta clicando en el botón para ello.



Desactivamos el **Automatically Retrieve Fields**

Report query

Report query | JavaBean Datasource | DataSource Provider | CSV Datasource | Excel Datasource

Query language: SQL

Load query | Save query

SELECT

```
AD."FIRSTNAME" AS NOMBRE,  
AD."LASTNAME" AS APELLIDO,  
AD."STREET" AS DIRECCION,  
DO."ID" AS NUMFACTURA,  
PR."NAME" AS PRODUCTOS,  
PR."COST" AS PRECIO,  
PO."QUANTITY" AS CANTIDAD,  
PO."QUANTITY"*PR.COST AS TOTAL
```

FROM

```
"DOCUMENT" DO INNER JOIN "ADDRESS" AD ON DO."ADDRESSID" = AD."ID"  
INNER JOIN "POSITIONS" PO ON DO."ID" = PO."DOCUMENTID"  
INNER JOIN "PRODUCT" PR ON PO."PRODUCTID" = PR."ID"
```

ORDER BY

```
NUMFACTURA
```

Drag a parameter into the query to add a parameter. Hold CTL to add the parameter as query chunk.

Available parameters

Fields provider for sql queries ready.

☐ Automatically Retrieve Fields | Read Fields | Query desig... | Send to clipbo...

New parameter

Field name	Field type	Description
ID	java.lang.Integer	
NOMBRE	java.lang.String	
APELLIDO	java.lang.String	
DIRECCION	java.lang.String	
NUMFACTURA	java.lang.Integer	
PRODUCTOS	java.lang.String	
PRECIO	java.math.BigDecimal	
CANTIDAD	java.lang.Integer	
TOTAL	java.math.BigDecimal	

Filter expression... | Sort options... | Preview data ▼ | OK | Cancel

Con la consulta creada, lo siguiente es meter los campos creados (se encuentran en Fieds) en el informe.

Para agrupar las facturas de cada cliente y saber el importe total a pagar por cada cliente, lo hacemos creando un grupo, donde se agruparán todas las facturas relacionadas con un cliente en concreto.

Para hacer el grupo, hacemos clic con el botón derecho, encima del nombre del informe y entramos en Add Report Group, le ponemos nombre al grupo y elegimos el campo por el que queremos agrupar, en nuestro caso, por el numero de factura.

The screenshot shows the 'New group wizard' dialog box with the 'Group criteria' tab selected. On the left, a 'Pasos' (Steps) pane shows '1. Group criteria' as the current step and '2. Details' as the next step. The main area contains the following fields and options:

- Group name:** A text box containing 'NUMEROFACTURA'.
- Group by the following report object:** A radio button that is selected, followed by a dropdown menu showing 'NUMFACTURA Field Integer'.
- Group by the following expression:** An unselected radio button followed by a large empty text box and a small icon of a notepad and pencil.

At the bottom of the dialog are five buttons: '< Atras', 'Siguiete >' (highlighted with a blue border), 'Terminar', 'Cancelar', and 'Ayuda'.

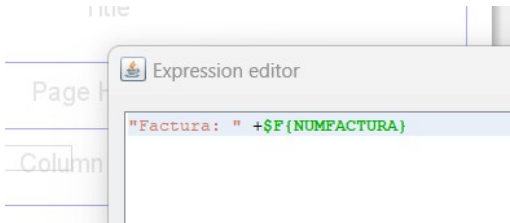
The screenshot shows the 'New group wizard' dialog box with the 'Details' tab selected. The 'Pasos' (Steps) pane on the left shows '1. Group criteria' and '2. Details' as the current step. The main area contains two checked checkboxes:

- ☒ Add the group header
- ☒ Add the group footer

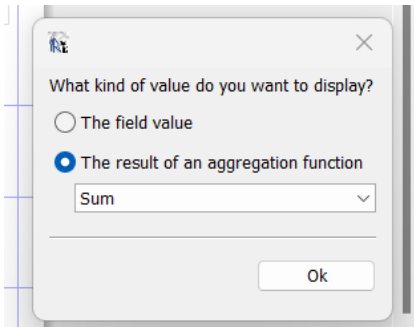
Esto añadirá una cabecera y un
pié de página.

Lo añadimos al grupo que se ha creado en nuestro informe, con los datos de los clientes.

\$F{NUMFACTURA}		CLIENTE: \$F{NOMBRE} \$F{APELLIDO}
		DIRECCIÓN: \$F{DIRECCION}



Le cambiamos la estructura al texto para que nos muestre un título informativo, para ello, con el botón derecho sobre el campo introducido (\$F{NUMFACTURA}) y clicamos en **Edit Expression.**



Por último, para mostrar el importe total a pagar por cada cliente, arrastramos nuestro campo **Total** al **group Footer** con una función de agregación (cuadro que sale cuando arrastras el elemento al informe) de suma.

APARTADO 1			
Page Header			
PRODUCTOS	PRECIO	CANTIDAD	TOTAL
"Factura: " + \$F{NUMFACTURA}		CLIENTE: \$F{NOMBRE} \$F{APELLIDO}	
		DIRECCIÓN: \$F{DIRECCION}	
\$F{PRODUCTOS}	\$F{PRECIO}	\$F{CANTIDAD}	\$F{TOTAL}
IMPORTE TOTAL:			\$V{TOTAL_2}
Column Footer			
Page Footer			
Summary			

IMPORTE TOTAL: 1738.4

Factura: 1

CLIENTE: Mary King

DIRECCIÓN: 491 College Av.

Shoe Shoe	16.2	9	145.8
Shoe Shoe	16.2	19	307.8
Clock Ice Tea	21.6	8	172.8
Ice Tea Shoe	19.4	16	310.4
Iron Iron	5.4	9	48.6
Ice Tea Ice Tea	11.0	6	66.0
Shoe Clock	2.8	8	22.4

IMPORTE TOTAL: 1073.8

- Creamos un proyecto en java, dentro del proyecto creamos una carpeta (Informes) y metemos los informes creados con el iReport, el .jrxml (creado con el iReport) y el .jasper (el informe compilado) al que tenemos que acceder desde el proyecto netbeans.
- En el proyecto, he creado:

-**Una clase**, donde he introducido el código para:

La conexión

Dos métodos para generar informes (uno con parámetros y otro sin parámetros)

Un método para que nos devuelva el resultado de la consulta realizada a la BBDD.

```

    */
    private static void crearConexion() throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {
        Class.forName(className: "org.hsqldb.jdbcDriver").newInstance();
        connection = DriverManager.getConnection(url: jdbcdir, user: Usuario, password: pass);
    }

    /**
     * Genera un informe que no necesita parametros, solo indicar en donde se encuentra el fichero *.jasper y donde guardaremos el fichero *.pdf
     * @param dirJasper ruta del ficher *.jasper
     * @param dirPdf ruta del archivo que crearemos *.pdf
     */
    public static void generarInforme(String dirJasper, String dirPdf) {

        try {
            crearConexion();

            Map parametros = new HashMap();

            JasperPrint print = JasperFillManager.fillReport(sourceFileName: dirJasper, params: parametros, connection);
            JasperExportManager.exportReportToPdfFile(jasperPrint: print, destFileName: dirPdf);

        } catch (Exception e) {

            System.out.println(x: e.getCause());

        }

    }
}

```

```

    * Genera un informe que necesita parametros, solo indicar en donde se encuentra el fichero *.jasper y donde guardaremos el fichero *.pdf
    * @param dirJasper ruta del ficher *.jasper
    * @param dirPdf ruta del archivo que crearemos *.pdf
    * @param parametros fichero tipo Map que tiene guardado de donde recoge los parametros.
    */
    public static void generarInforme(String dirJasper, String dirPdf, Map parametros) {

        try {
            Class.forName(className: "org.hsqldb.jdbcDriver").newInstance();
            Connection connection = DriverManager.getConnection(url: jdbcdir, user: Usuario, password: pass);

            JasperPrint print = JasperFillManager.fillReport(sourceFileName: dirJasper, params: parametros, connection);
            JasperExportManager.exportReportToPdfFile(jasperPrint: print, destFileName: dirPdf);

        } catch (Exception e) {

            System.out.println(x: e.getCause());

        }

    }

    /**
     * devuelve el resultado (ResultSet) de una consulta a la base de datos.
     * @param sql
     * @return
     */
    public static ResultSet consulta(String sql) {
        ResultSet rs = null;
        try {
            crearConexion();

            Statement stm = connection.createStatement();
            rs = stm.executeQuery(sql);
        } catch (Exception e) {

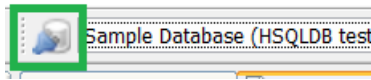
        }

        return rs;
    }
}

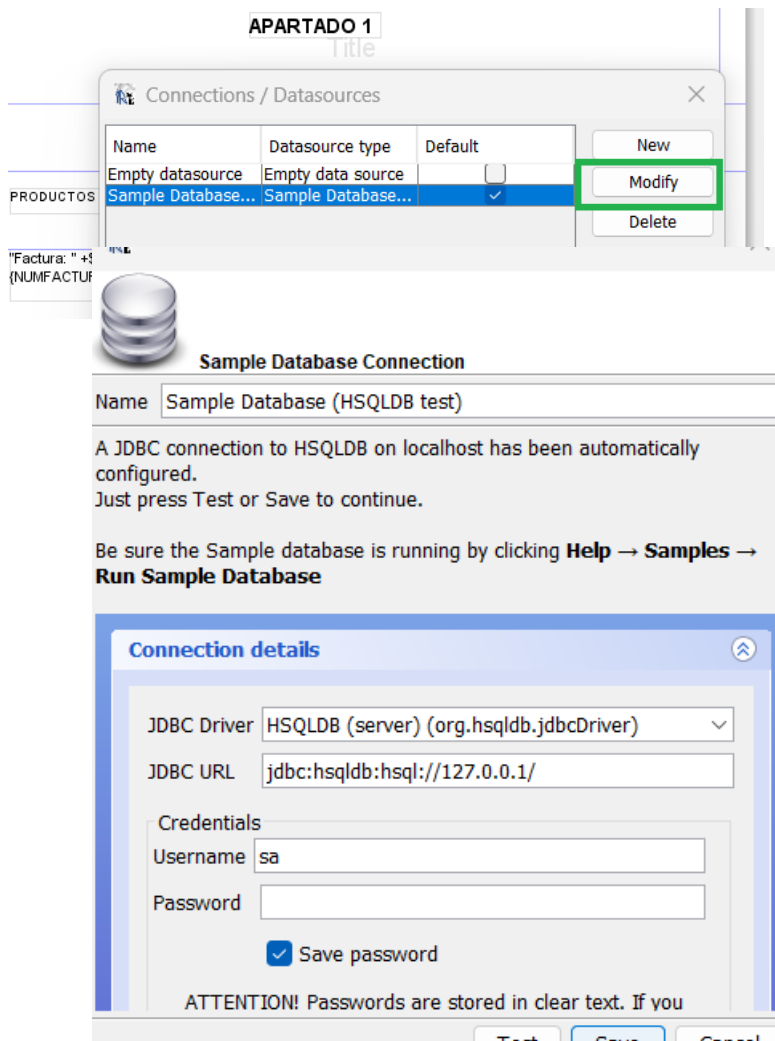
```

- **-Un JFrame**, donde he añadidos los elementos necesarios para crear los informes de los punto 1, 2, 3, y 4, en nuestro caso, 1 comboBox y 4 Button.

Para saber los datos de la conexión, lo hacemos desde iReport.



En la ventana que aparece, marcamos Sample Database y clicamos en el botón Modify



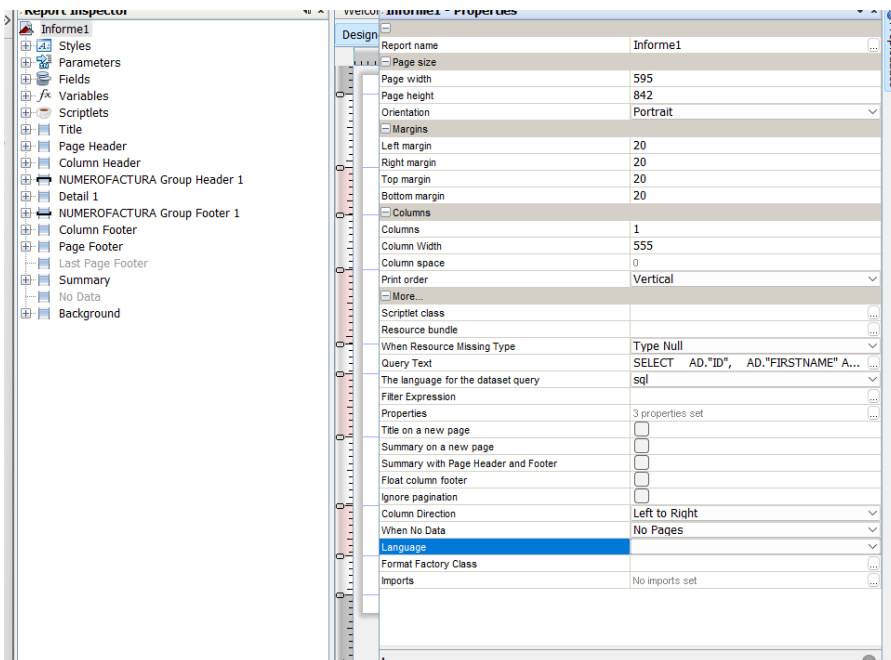
- En cada botón, llamamos a los métodos creados en la clase de nuestro proyecto.
- Importamos las librerías de jasper al proyecto, para poder crear un informe (en nuestro caso, será un informe PDF)

Librerías>Add JAR/Folder, las buscamos en la carpeta de iReport, ireport>modules>ext.

- Libraries
 - > commons-beanutils-1.8.2.jar
 - > commons-collections-3.2.1.jar
 - > commons-digester-2.1.jar
 - > commons-logging-1.1.jar
 - > hsqldb-1.8.0-10.jar
 - > iText-2.1.7.js2.jar
 - > jasperreports-5.6.0.jar
 - > servlet-api-2.4.jar
 - > groovy-4.0.20-grooid.jar
 - > jcommon-1.0.15.jar
 - > jfreechart-1.0.12.jar
 - > JDK 1.8 (Default)

- Cambiamos, en iReport, el lenguaje para que al ejecutar no nos salte error de groovy

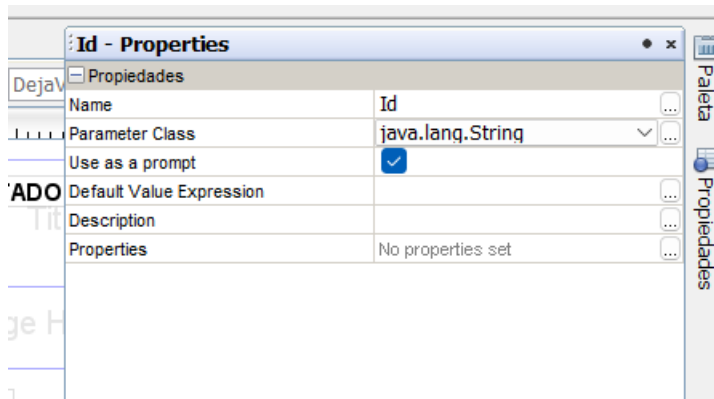
Propiedades>Lenguaje, seleccionamos java



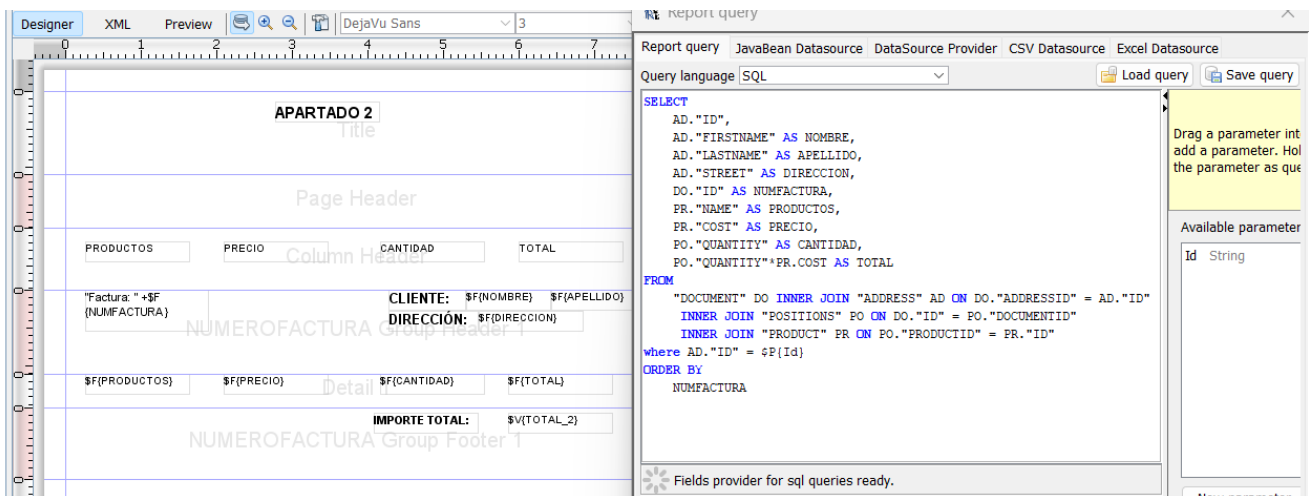
2. Incluir el informe para generar facturas en una aplicación Java que lo muestre en formato PDF, informando al usuario del resultado de la operación, y teniendo en cuenta que el **código del cliente (id de la tabla ADDRESS)** se debe pasar al informe como **parámetro**. Además, es importante saber que para que la aplicación Java funcione, la base de datos de iReport debe estar ejecutándose (Ayuda -> Samples -> Run Sample Database).

- Lo primero, es crear un parámetro con los id, en iReport.

Para ello, encima de parameters (panel izq), con el *botón derecho>agregar parámetro y* desde propiedades (panel derecho) cambiamos sus propiedades, nombre y tipo



Creamos el informe añadiéndole el parámetro creado.



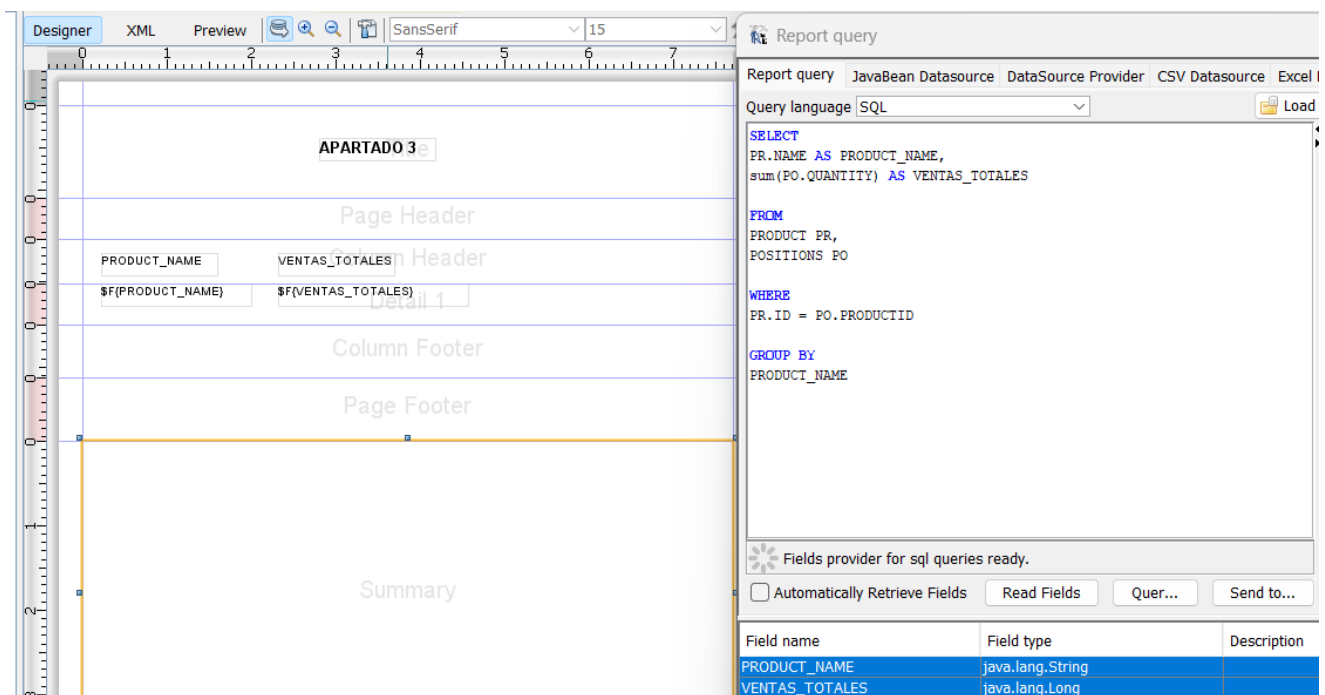
3. Crear un informe en el que aparezca un listado con el nombre de cada producto y sus ventas totales, es decir, la suma de las cantidades vendidas -sum(quantity) AS VENTAS_TOTALES agrupando por PRODUCT_NAME-. A continuación, añadir un gráfico al informe que permita visualizar estos totales. Se debe seleccionar el tipo de gráfico que mejor se adapte al problema.

Creamos el SQL con lo que nos pide en el punto.

“el nombre de cada producto” de la tabla product el name, renombramos a
PRODUCT_NAME

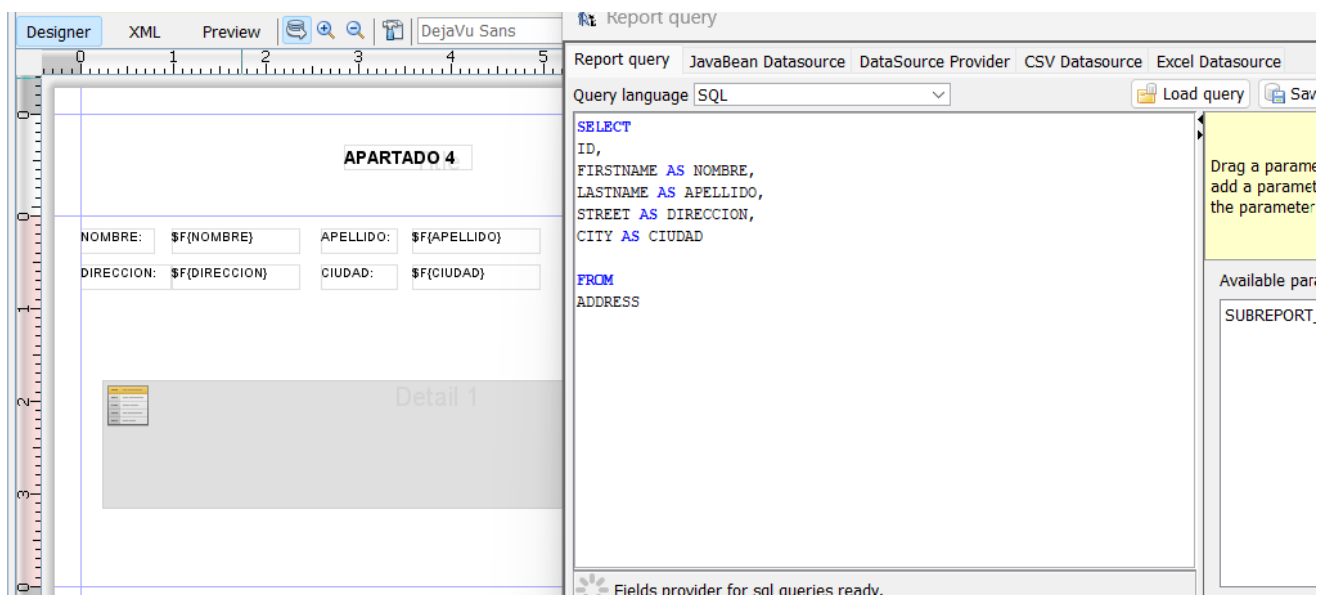
“suma de las cantidades vendidas” de la tabla positions, la suma de Quantity,
renombramos a VENTAS_TOTALES

Creamos el informe con los datos que nos piden y añadimos el gráfico (en la paleta el
chart) en el bloque *Summary*.



4.Repite la primera parte del apartado 1 utilizando subinformes, es decir, parte de todos los datos de la tabla ADDRESS y obtén un subinforme empleando el resto de tablas y manejando el *addressid* como parámetro. Deben aparecer los datos personales (id, nombre, apellidos y dirección) de los clientes de la empresa (tabla ADDRESS), el número de la factura (tabla DOCUMENT), así como los productos (tabla PRODUCT) con su cantidad y precio (tabla POSITIONS). No es necesario incluir los totales.

Generamos el informe y creamos el sql con todos los datos de la tabla ADDRESS y con el subinforme (en la paleta, subreport) y lo generamos. Lo colocamos en **detail**.



En el subinforme, introducimos el resto de datos de las demás tablas y creamos el parámetro ID.

The screenshot shows the Jaspersoft Studio Designer interface. On the left, the report layout is visible with a title 'Title', a 'Page Header' section, a table with columns 'NUMERO FACTURA', 'NOM_PRODUCTO', 'PRECIO', 'CANTIDAD_PROD', and 'TOTAL', a 'Detail 1' section, a 'Column Footer' section, a 'Page Footer' section, and a 'Summary' section. On the right, the 'Report query' window is open, showing the following SQL query:

```
SELECT
DO.ID AS NUM_FACTURA,
PR.NAME AS NOM_PRODUCTO,
PR.COST AS PRECIO,
PO.QUANTITY AS CANTIDAD_PROD,
PO.QUANTITY * PR.COST AS TOTAL
FROM
PRODUCT PR,
POSITIONS PO,
DOCUMENT DO
WHERE
PR.ID = PO.PRODUCTID AND PO.DOCUMENTID = DO.ID AND PO.DOCUMENTID =
${P{ID}}
ORDER BY
NUM_FACTURA
```

The 'Query language' is set to 'SQL'. The 'Fields provider for sql queries ready.' message is visible at the bottom of the query window. Below the query window, there is a table with the following data:

Field name	Field type	Description
NUM_FACTURA	java.lang.Integer	
NOM_PRODUCTO	java.lang.String	
PRECIO	java.math.BigDecimal	
CANTIDAD_PROD	java.lang.Integer	
TOTAL	java.math.BigDecimal	

```

private void btn_GConsulta2ActionPerformed(java.awt.event.ActionEvent evt) {

    //Creamos el objeto Map
    Map parametro = new HashMap();

    parametro.put("Id", jComboBox.getSelectedItem());

    iReport.generarInforme("Informes/Apartado4/factura.jasper", "Informes/Apartado4/factura.pdf", parametro);

    System.out.println("Se ha creado el archivo pdf con éxito");
}

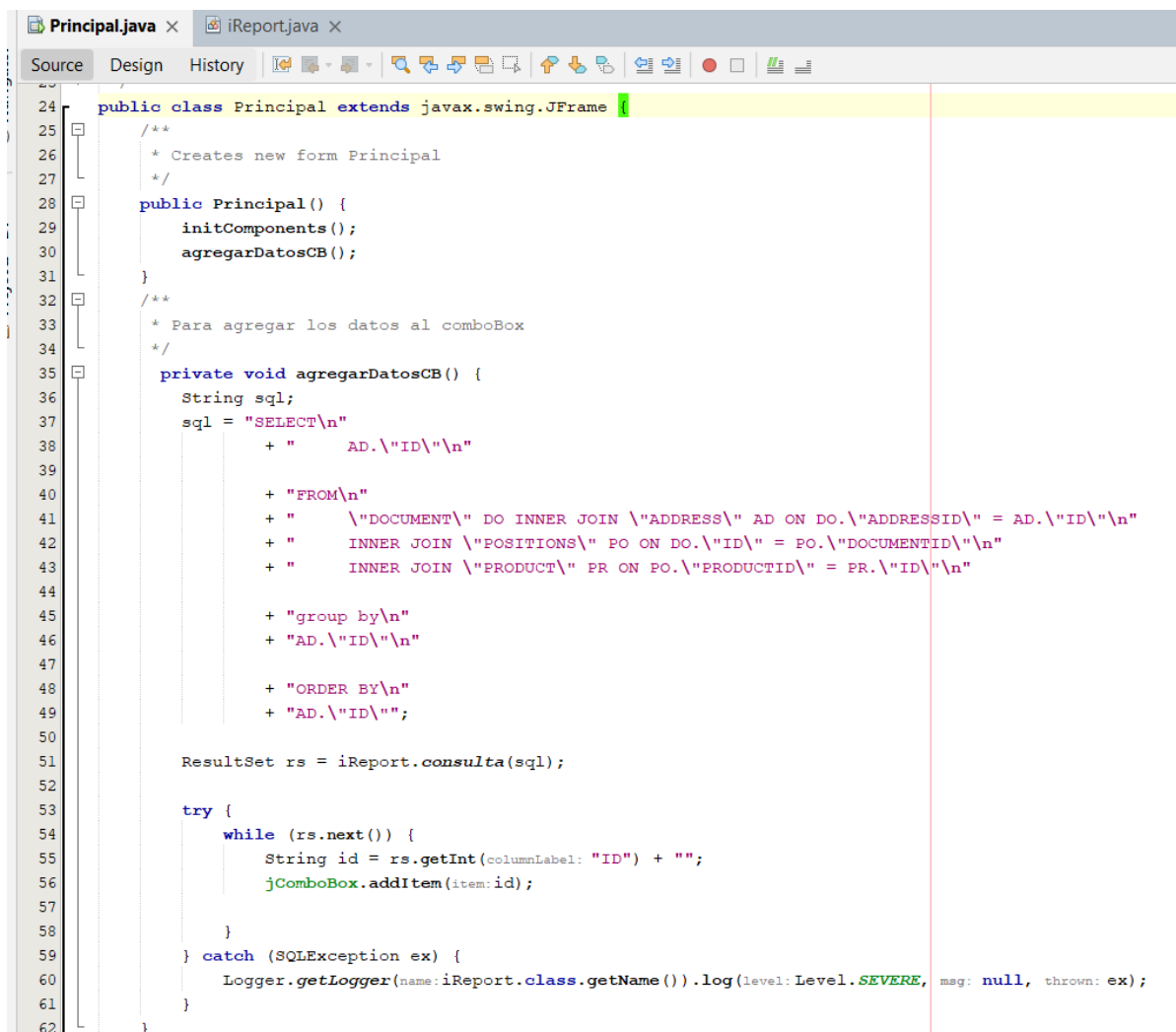
private void btn_GConsulta3ActionPerformed(java.awt.event.ActionEvent evt) {
    iReport.generarInforme("Informes/Apartado3/factura.jasper", "Informes/Apartado3/factura.pdf");
    System.out.println("Se ha creado el archivo pdf con éxito");
}

private void btn_GConsulta1ActionPerformed(java.awt.event.ActionEvent evt) {
    iReport.generarInforme("Informes/Apartado1/facturas.jasper", "Informes/Apartado1/factura.pdf");
    System.out.println("Se ha creado el archivo pdf con éxito");
}

private void btn_GConsulta4ActionPerformed(java.awt.event.ActionEvent evt) {
    iReport.generarInforme("Informes/Apartado2/facturas.jasper", "Informes/Apartado2/factura.pdf");
    System.out.println("Se ha creado el archivo pdf con éxito");
}

```

Método para agregar datos al comboBox



```

Principal.java x iReport.java x
Source Design History
24 public class Principal extends javax.swing.JFrame {
25     /**
26      * Creates new form Principal
27      */
28     public Principal() {
29         initComponents();
30         agregarDatosCB();
31     }
32     /**
33      * Para agregar los datos al comboBox
34      */
35     private void agregarDatosCB() {
36         String sql;
37         sql = "SELECT\n"
38             + "    AD.\nID\n\n"
39             + "FROM\n"
40             + "    \nDOCUMENT\n DO INNER JOIN \nADDRESS\n AD ON DO.\nADDRESSID\n = AD.\nID\n\n"
41             + "    INNER JOIN \nPOSITIONS\n PO ON DO.\nID\n = PO.\nDOCUMENTID\n\n"
42             + "    INNER JOIN \nPRODUCT\n PR ON PO.\nPRODUCTID\n = PR.\nID\n\n"
43             + "group by\n"
44             + "AD.\nID\n\n"
45             + "ORDER BY\n"
46             + "AD.\nID\n";
47
48         ResultSet rs = iReport.consulta(sql);
49
50         try {
51             while (rs.next()) {
52                 String id = rs.getInt(columnLabel: "ID") + "";
53                 jComboBox.addItem(item:id);
54             }
55         } catch (SQLException ex) {
56             Logger.getLogger(name:iReport.class.getName()).log(level:Level.SEVERE, msg: null, thrown: ex);
57         }
58     }
59 }
60
61
62

```

