UD5. Confección de informes

RA5. Crear informes, para lo que se utiliza y evalua herramientas gráficas.

– CA5.1. Establecer la estructura del informe.

– CA5.2. Se diseñan informes básicos a partir de una fuente de datos mediante asistentes.

– CA5.3. Se establecieron filtros sobre los datos en los que se basan los informes.

– CA5.4. Se incluyen valores calculados, recuentos e totales.

– CA5.5. Se incluyen gráficos generados a partir de los datos.

– CA5.6. Se utilizan herramientas para generar el código correspondiente a los informes de una aplicación.

– CA5.7. Se modifica el código correspondiente a los informes.

– CA5.8. Se desarrolla una aplicación que contiene informes incrustados.

BC4. Confección de informes

- Informes incrustados y no incrustados en la aplicación.
- Herramientas gráficas integradas en el IDE y externas a él.
- Estructura general: secciones. Uso de agrupamientos. Filtración de datos.
- Numeración de liñas, recuentos e totales. Valores calculados.
- Librerías para la generación de informes: clases, métodos e atributos.
- Informes con parámetros.
- Conexión coas fuentes de datos y ejecución de consultas.

1. Informes incrustados y no incrustados en la aplicación

Al planear la creación de una aplicación una de las consideraciones más importantes es la posibilidad de la inclusión de informes.

Los informes de una aplicación se pueden dividir en dos grandes grupos desde el punto de vista de su diseño e inclusión en la misma: **informes incrustados** e **informes no incrustados.**

Un **informe incrustado** es un informe que se ha importado a un proyecto, o que se ha creado en él. Cuando se incrusta un informe en el proyecto, automáticamente **se genera una clase contenedora para el informe.**

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.

En el caso de que sea una aplicación compilada, tanto el informe como su clase contenedora se incrustan en el ensamblado, lo mismo que ocurriría con cualquier otro recurso, módulo, paquete o clase del proyecto.

En Microsft Visual Studio tienen la clase contenedora denominada **ReportDocument** y en Java hay varias opciones pero quizás la mas utilizada es **JasperReports.** Una de las herramientas más utilizadas en el campo de la ofimáica, el **ACCESS y su homólogo en Openoffice BASE,** generan también sus propios informes incrustados

Un **informe no incrustado** es un informe **externo al proyecto.** Existen muchas formas de tener acceso al informe y cargarlo en el proyecto, para habilitar la interacción con el informe mediante programación, pero el informe siempre continuará siendo externo.

A un informe no incrustado siempre se obtiene acceso externamente y se 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 como servicio Web de informes y obtenerse en formato HTML o XML.
- El informe puede formar parte de un grupo de informes expuestos a través de herramientas externas. **Crystal Reports**, el generador de informes de mayor influencia, puede trabajar y generar informes no incrustados.

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 uno de los modelos de objetos en tiempo de ejecución**.**

¿Cuándo elegir informes incrustados o no incrustados?

Si desea simplificar la implementación del proyecto es preferible **informes incrustados**. El problema de estos es que no todos los lenguajes y sus correspondientes entornos de desarrollo soportan la creación de informes o tienen la herramienta o asistente para llevar a cabo su diseño.

Los **informes incrustados** tendrán menos archivos con los que trabajar y no se tendrá que preocupar de si los informes están mal colocados en la ruta de directorio de archivos equivocada. Además, esta solución es **aconsejable** siempre y cuando los informes no se exponen a modificaciones tras el compilado del proyecto.

Los informes **no incrustados son más sencillos y seguros,** pero requieren **más trabajo**. Podemos modificar su diseño sin necesidad de volver a compilar el proyecto.

Si los informes **se deben modificar regularmente**, utilice informes no incrustados para facilitar su acceso y modificación, sin preocuparse por la necesidad de volver a compilar los ensamblados cada vez. Además, existen **límites en el tamaño que puede tener un informe incrustado**. Un informe muy grande se compila como un recurso incrustado. Además, los informes no incrustados tienen **ventajas de escalabilidad.**

```
private ReportDocument ChooseReport(int i)

{
        switch(i)
        {
                case 1:
                Chart chartReport = new Chart();
                return (ReportDocument)chartReport;
                case 2:
                Hierarchical_Grouping hierarchicalGroupingReport = new
                Hierarchical_Grouping();
                return (ReportDocument)hierarchicalGroupingReport;
                default:
                World_Sales_Report worldSalesReport = new
                World_Sales_Report ();
                return (ReportDocument)worldSalesReport;

        };
}
```

Ejemplo de Código Generador de un Informe Incrustado en C#

```
string reportPath="C:\\Files Program\\VisualStudio" + "CrystalReports\\Report\\Business\\" +
                                        "SalesReport.rpt";
 ReportDocument reportDocument = new ReportDocument();
```

Ejemplo de enlace con un Generador de un Informe no Incrustado

2. Herramientas gráficas integradas en el IDE y externas a él

- **Crystal Reports:** es una aplicación utilizada para diseñar y generar informes desde una amplia gama de fuentes de datos.

  Varias aplicaciones, como Microsoft Visual Studio, incluyen una versión OEM de *Crystal Reports* como una herramienta de propósito general para informes/reportes. **Crystal Reports** se convirtió en el generador de informes estándar cuando Microsoft lo liberó con Visual Basic. Permite incluir datos de varias tablas, la inclusión de sentencias SQL y de estructuras condicionales e iterativas. El formato de salida puede ser .pdf, .rpt, .xls, .txt y .CVS[1] entre otros. CrystalReports es quizás la herramientas más utilizada en programación para la obtención de reportes.

- **JasperReports** es una herramienta de creación de informes que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML. Está escrito completamente en Java y puede ser usado en gran variedad de aplicaciones de Java, incluyendo J2EE o aplicaciones web, para generar contenido dinámico. Su propósito principal es ayudar a crear documentos de tipo páginas, preparados para imprimir en una forma simple y flexible. JasperReports se usa comúnmente con **iReport,** un front-end gráfico de código abierto para la edición de informes. Se encuentra bajo licencia libre GNU, por lo que es **Software libre**. Forma parte de la iniciativa open source Lisog.

- **ReportLab** es una librería para la obtención de reportes o informes en Python. Es software libre. Es uno de los más utilizados por los desarrolladores en Python para la elaborar los informes de sus proyectos.

- **Datareport** es una herramienta eficaz y es fácil crear informes complejos arrastrando y soltando campos fuera de la ventana de entorno de datos. Es de muy fácil uso, sin apenas necesidad de implementar código y se encuentra en el entorno de desarrollo Visual Net de Microsoft. Se utiliza para aplicaciones no demasiado complejos pero no tiene la potencialidad de CrystalReports.

- **R&OS o pdf class** es una clase para PHP la cual provee métodos muy potentes y simplificados para la creación de archivos PDF. Otra clase útil es PHP es **PHPReport Generator.**

1Los ficheros **CSV** (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: España, Francia, Italia...) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles, por ejemplo, → luis, 98789, vigo

- **Geraldo**: es un motor de informes para aplicaciones de Python o Django. Utiliza **ReportLab** para generar informes con encabezado y pie de página.. Informe comenzar y bandas de resumen, agreggation y elementos gráficos, etc.. También es sencilla de utilizar.
- **pyfpdf:** el que utilizamos en las prácticas, destaca por su sencillez. Utiliza una librería para PHP (FPDF) adaptada a python. La librería **xhtml2pdf** permite obtener, utilizando pyfpdf, informes en html.
- **Docutils** es un sistema de texto de código abierto para el procesamiento de documentos de procesamiento de texto plano en formatos útiles, tales como HTML, LaTeX o XML. Incluye **reStructuredText**, es fácil de leer, fácil de usar, estilio ***lo-que-ve-es-lo-que-hay*** (WYSWYG).

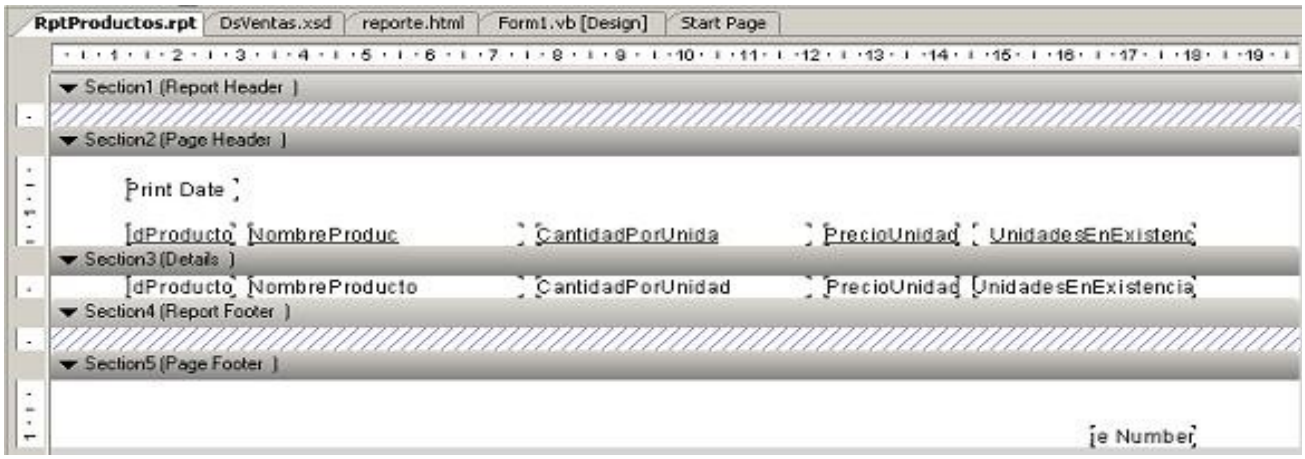
Las diferentes herramientas que están contenidas en las suites ofimáticas Openoffice.org, LibreOffice y Office de Microsoft también permiten la obtención de informes o reportes en diferentes formatos, principalmente PDF en el primer caso y XPS en el segundo.

Existen algunas herramientas más dedicadas a la generación de informes en diferentes formatos pero las más utilizadas son las mencionadas.

3. Estructura general: secciones. Uso de agrupamentos. Filtrado de datos

Básicamente un informe, de forma general, contiene las siguientes secciones:

- **Seccion 1: Report Header**. ***Cabecera del informe***, donde se imprime una sola vez al inicio del reporte. Puede contener desde el *logo de la empresa, fechas, nombre del informe*...
- **Seccion 2: Page Header. *Cabecera de pagina***, donde se imprime al inicio de cada pagina impresa. Puede contener *anotaciones generales*, por ejemplo si es una factura, el nombre y dirección del cliente, número de factura.
- **Seccion 3: Details**. ***Detalle del reporte***, donde las filas o registros que conforman el reporte. Es alli donde se alojan los *campos del origen de datos*.
- **Seccion 4: Report Footer. *Pie del reporte***, donde se imprime una sola vez al finalizar el reporte. Se utiliza esta seccion para imprimir los *totales generales, promedios*...
- **Seccion 5: Page Footer. *Pie de pagina,*** donde se imprime al final de cada pagina. Se utiliza esta seccion para imprimir la *paginacion, los totales por pagina.*



Aunque no obligatoriamente debe contener todas las secciones si es aconsejable que se mantenga una estructura por secciones por su sencillez y posibilidad a la hora de utilizar dicha estructura en otros informes.

Los datos que aparecen en el informe finalizado dependen de las **opciones de organización**. En particular, los datos del informe varían según las secciones en las que desee insertar objetos de informe concretos. Por ejemplo, si inserta un objeto de gráfico en la sección *Encabezado de informe*, el gráfico sólo aparecerá una vez al principio del informe y resumirá los datos que contiene el informe. Además, si un objeto de gráfico se añade a la sección *Encabezado de grupo,* aparecerá un gráfico individual al principio de cada grupo de datos y sólo se resumirán los datos relacionados con dicho grupo.

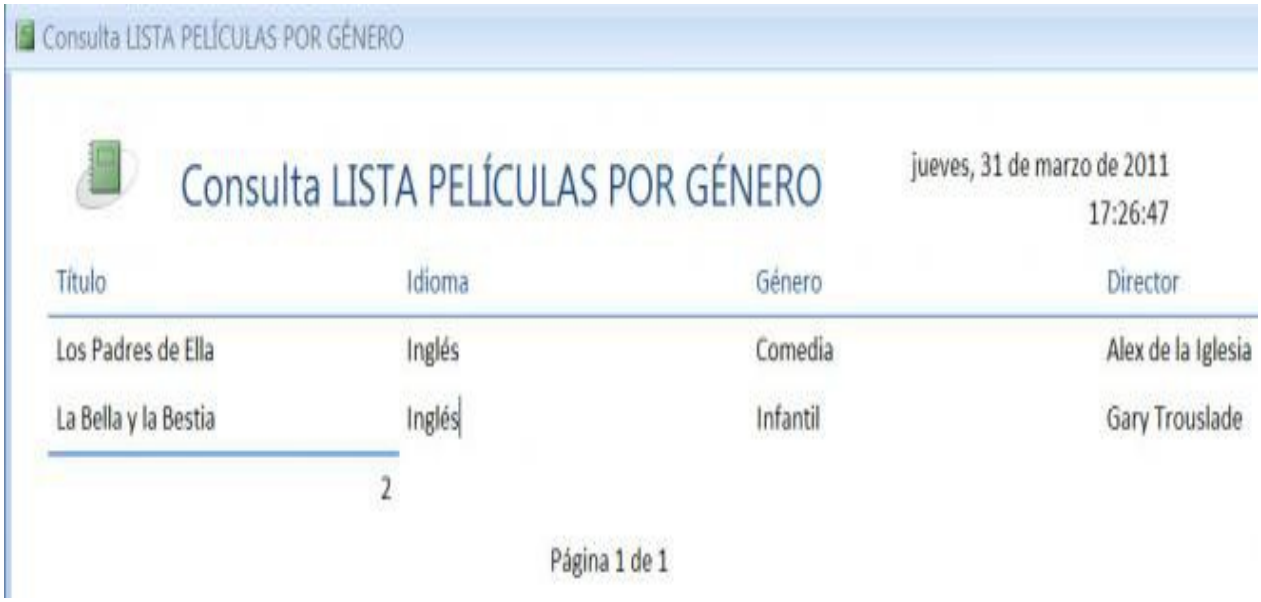
Los informes necesitará separar los datos en grupos para **facilitar la lectura y el análisis.** Se puede agrupar los datos de un informe **para que muestren relaciones jerárquicas**. Al agrupar datos jerárquicamente, se ordena la información según la relación entre dos campos. En los datos que se utilicen para el informe debe ser inherente una relación jerárquica.

- Para que el programa reconozca una relación entre los campos principal y secundario, ambos deben pertenecer al mismo tipo de datos.
- Los **datos del campo principal deben ser un subconjunto de los datos del campo secundario.**
- Para que el nivel superior de la jerarquía aparezca en el informe, el valor debe aparecer en los datos secundarios y la fila correspondiente de los datos principales debe estar vacía.
- **No puede existir lógica circular en los datos** (es decir, A no puede estar relacionado con B si B está relacionado con C y C está a su vez relacionado con A).

Por ejemplo, si se desea mostrar la estructura jerárquica de un departamento, se puede agrupar datos por **Id. de empleado y especificar la jerarquía usando el campo de datos que muestra a quién informa el empleado.** Otro agrupamiento sería listado de empleados por departamento

siendo el campo principal el **id del departamento** y el secundario los empleados que a él pertenecen.

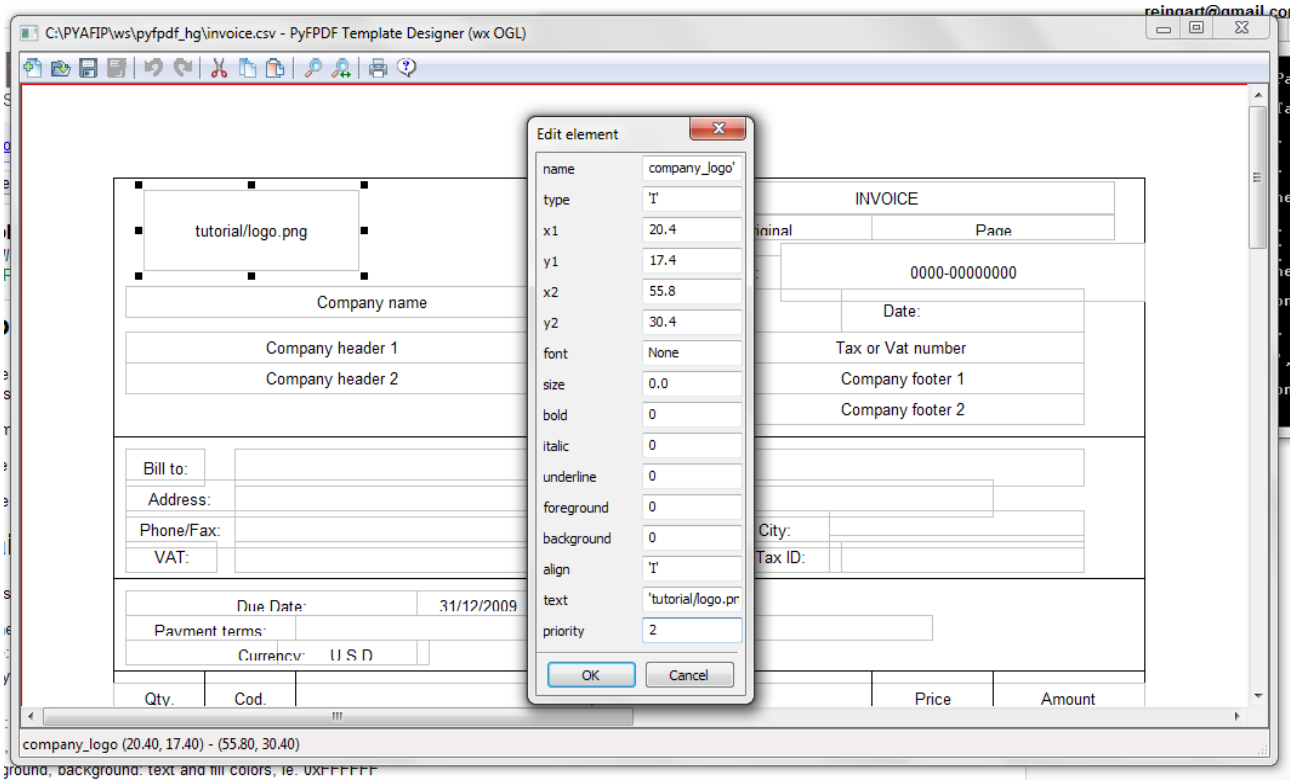
Se puede definir filtros en una región de datos para **seleccionar o excluir partes del conjunto de datos que utiliza la región de datos.** Los filtros limitan los datos que se muestran al usuario tras la recuperación de todos los datos. Dado que se recupera el conjunto completo de datos y luego se filtra cuando se procesa el informe, es posible que el informe no sea tan bueno como cuando el informe obtiene datos filtrados de otra manera (específicamente, si escribe un código que filtra datos antes de que pasen al informe). De esta forma en los agrupamientos pueden ser interesantes filtrados de datos, como por ejemplo listado de empleados por departamentos que obtuvieron un mínimo de X ventas o que pertenezcan a tal o cual ciudad.



4. Numeración de liñas, recontos e totais. Valores calculados.

Cuando calculamos un **resumen y/o totalizamos** los datos, hay que **dividir en grupos** y realizar la operación especificada sobre los valores de cada grupo. Todo lo que hay que hacer es especificar lo siguiente:

- el **campo** que deseamos resumir o totalizar,
- el tipo de operación que va a realizar con ese campo: **media, unidades x precio-unidad, subtotal, iva x subtotal, iva + subtotal....**
- el campo que va a desencadenar un nuevo grupo siempre que cambie su valor, por ejemplo, **modificaciones del iva, descuentos pronto-pago...**
- el orden de clasificación.



Plantilla de Web2Py para realizar informes con PyfPDF

Para insertar un subtotal (sólo en campos numéricos), o totales determinadas herramientas como **CrystalReports** tienen asistentes que facilitan dichos cálculos. En otros casos, como en Python, debemos generar mediante código la obtención de dichos resultados para incluirlos en el informe final. De hecho el lenguaje SQL permite que determinadas operaciones sean obtenidas en las sentencia de consulta, por ejemplo, el count, la media...

5. Librerías para la generación de informes: clases, métodos e atributos

Generalmente la mayoría de los API para creación de informes se han utilizando herramientas integradas en el caso de IDE integrados. Se suele aconsejar reducir el uso de **sql + programación** para la generación de informes por alto coste de trabajo aunque es cierto que tiene la ventaja de reducir código redundante.

Existen en el mercado muchas herramientas que integran las librerías necesarias para generar informes:

- Microsoft SQL Server Reporting Services
- Crystal Reports
- BEA Portal Reporting Solutions
- MicroStrategy Report Services
- Actuate BIRT (Business Intelligence and Reporting Tools)

Y otras que son Open Source:

- Pentaho Report Designer
- OpenRPT para PostgreSQL
- Eclipse BIRT
- JasperReports

De algunas de ellas ya hablamos en apartados anteriores.

La mayoría de las librerías que realizan la generación de informes establecen la siguiente arquitectura en su elaboración:

- **modelo de datos**
- **acceso a los datos**
- **presentación de los datos**

Para estudiar la estructura de clases de una herramientas de elaboración de informes utilizaremos **JasperReports.** Dicha herramienta elabora reportes predefinidos en XML. La estructura de clases es la siguiente:

- net.sr.jasperreports.engine.*jasperexportManager*
- net.sr.jasperreports.engine.*jasperprintManager*
- net.sr.jasperreports.engine.*jasperfillManager*
- net.sr.jasperreports.engine.*jaspercompileManager*
- *net.sr.jasperreports.engine.jasperdesignViewer*
- *net.sr.jasperreports.engine.jasperViewer*

Las cuatro primeras permiten manejar el motor de generación del reporte, y las dos últimas se encargan de la presentación o visualización del mismo.

En primer lugar llevamos a cabo el diseño del informe que no se más que una “plantilla” que utilizará el motor del generación del informe. Como es un documento XML este **lleva un DTD** asociado que se encuentra en las librerías de JasperReport. Su estructura es similar a la de cualquier documento de texto:

title	Título del reporte
pageHeader	Encabezado del documento
columnHeader	Encabezado de las columnas
detail	Detalle del documento. Cuerpo
columnFooter	Pie de la columna
pageFooter	Pie del documento
sumary	Cierre del documento

Una vez el diseño está listo llega el momento de la compilación que se encarga el método *compileReport* de la clase *compileManager* para posteriormente completar su contenido con el método *fiilReport* de la clase *jasperFillManager* obteniendo los datos de la base de datos o con cálculos intermedios.

La clase *jasperManager* se encarga de llamar a la API adecuado sea ODBC, JDBC para obtener los datos. Finalmente *jasperPrint* se encarga de su prepración para imprimir, *jasperViewer* para visualizarlo en pantalla o *exportManager* para exportalo a otro formato.

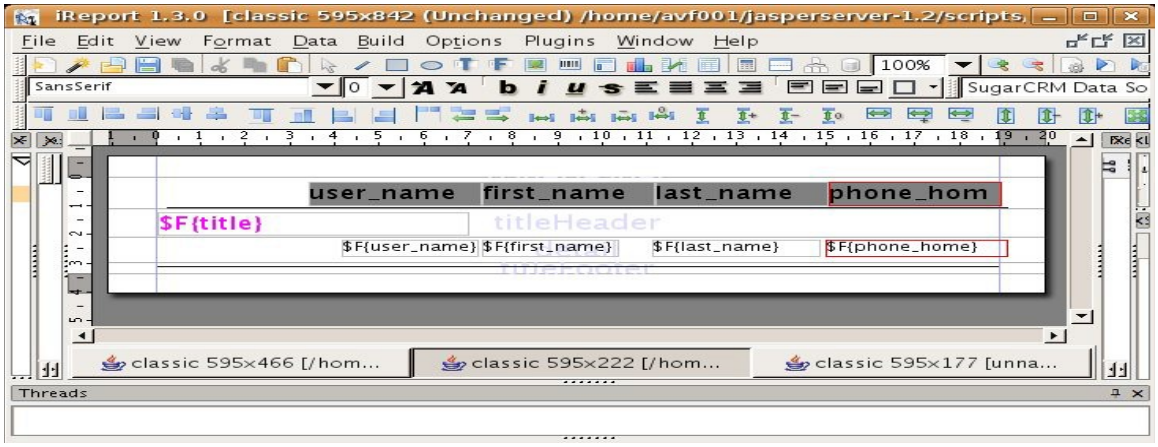
Un ejemplo sería:

```
Map parametrosReporte;          // Mapa de parámetros del reporte
JasperDesign disenioReporte;    // diseño del reporte
JRDesignQuery queryReporte;     // consulta a la base para
                                   armar el reporte

// Armamos el MAP de parámetros adicionales a pasar al reporte
  (los nombres están determinados por el XML diseñado
parametrosReporte.put("Identificacion", "121");
parametrosReporte.put("NumeroSecuencia", "01");

// Cargamos el diseño desde el archivo xml, la extensión por
  defecto: .jrxml
disenioReporte = JRXmlLoader.load("DisenoXML.jrxml");
```

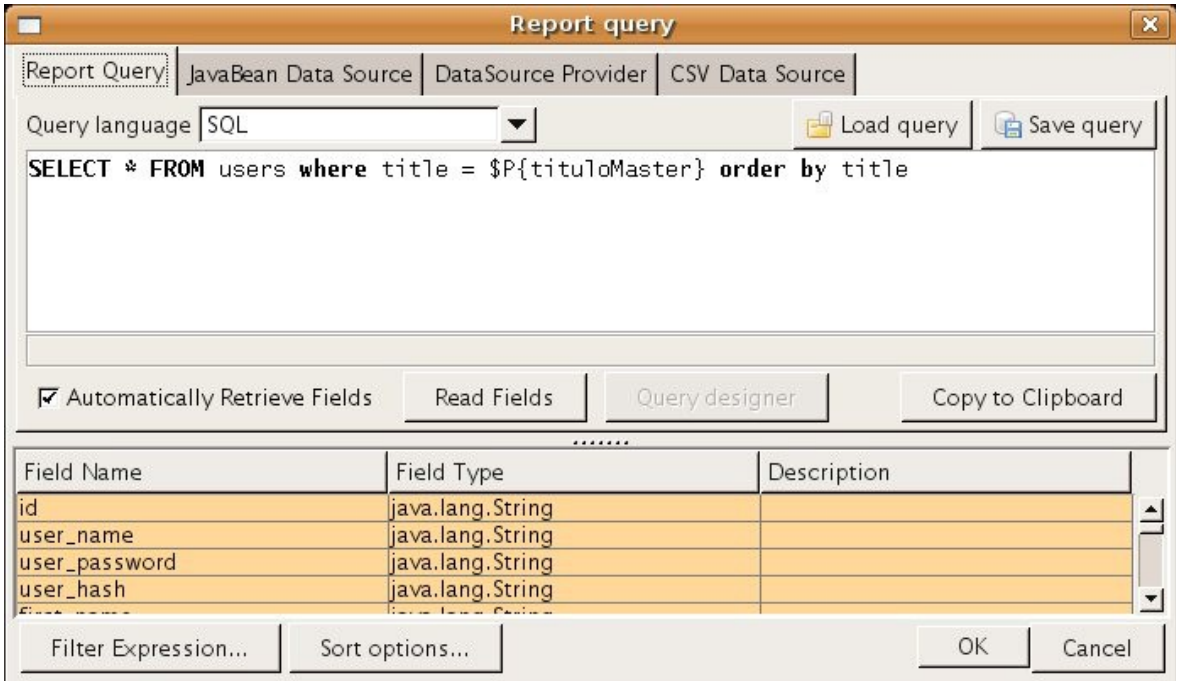

```
Map parametrosReporte;          // Mapa de parámetros del reporte
JasperDesign disenioReporte;    // diseño del reporte
JRDesignQuery queryReporte;     // consulta a la base para
                                   armar el reporte

// Armamos el MAP de parámetros adicionales a pasar al reporte
  (los nombres están determinados por el XML diseñado
parametrosReporte.put("Identificacion", "121");
parametrosReporte.put("NumeroSecuencia", "01");

// Cargamos el diseño desde el archivo xml, la extensión por
  defecto: .jrxml
disenioReporte = JRXmlLoader.load("DisenoXML.jrxml");
```


6. Informes con parámetros

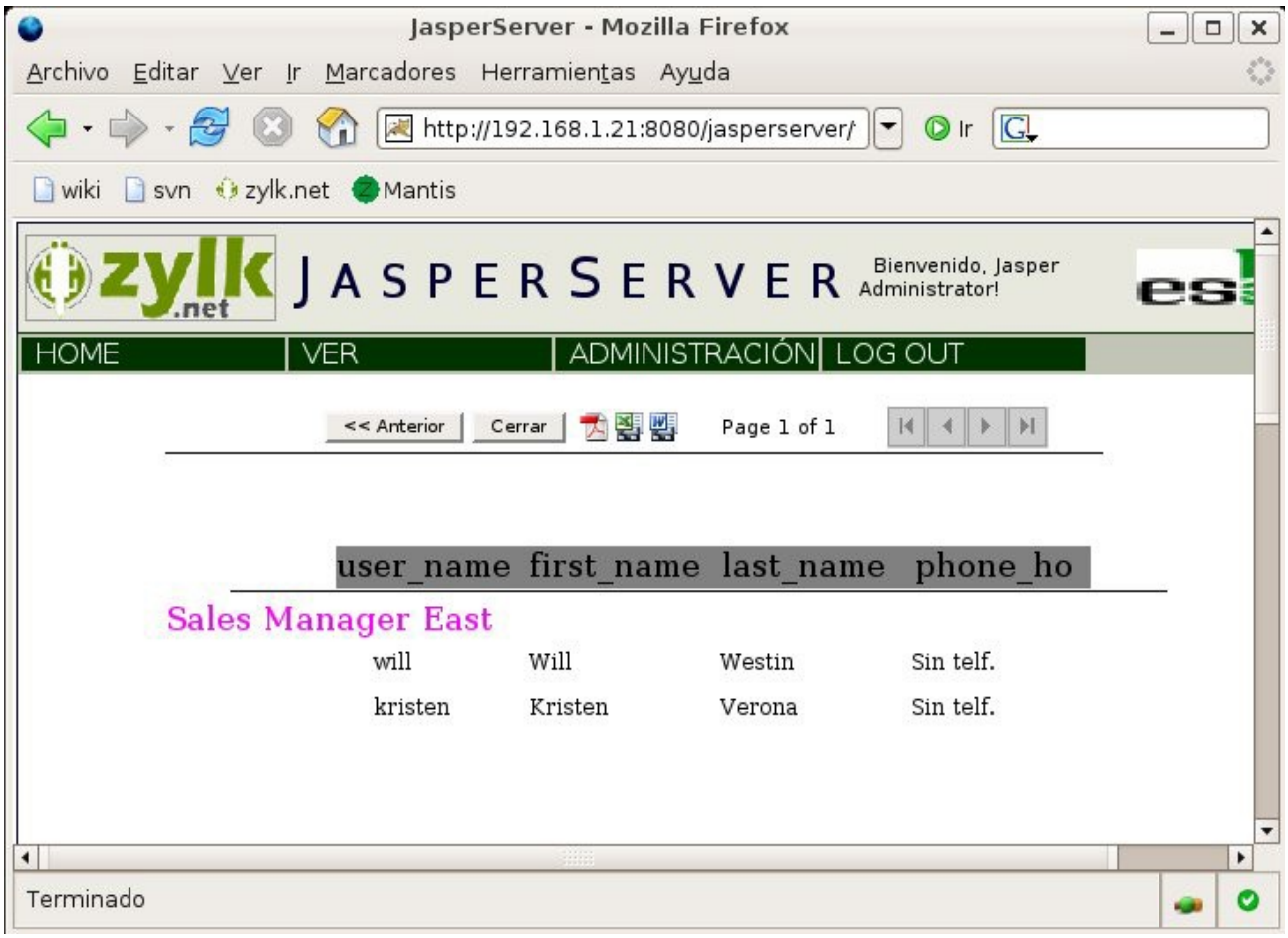
Las consultas que dan lugar a informes son útiles para poder trabajar sólo con los campos de una tabla que corresponden a una tarea determinada. Cuando se desea limitar aún más los datos con los que se va a trabajar, basándose en el valor de un campo, se pueden usar **criterios en la consulta.** Los criterios son reglas que se incluyen en el diseño de una consulta. Estas reglas especifican valores o modelos con los que los campos deben coincidir o que los campos deben contener para que la consulta los devuelva.



Ejemplo de consulta basada en el parámetro título



Ejemplo de informe basada en el parámetro título de la consulta anterior



Informe resultante

Cuando se desea que una consulta pida un valor o un modelo cada vez que se ejecuta, se puede crear una consulta con parámetros. La manera más sencilla de crear un informe que acepte parámetros **es usar una consulta de parámetros como origen de registros del informe**. Por ejemplo, puede crear un informe de ingresos mensuales basado en una consulta de parámetros que solicite un valor para el mes.

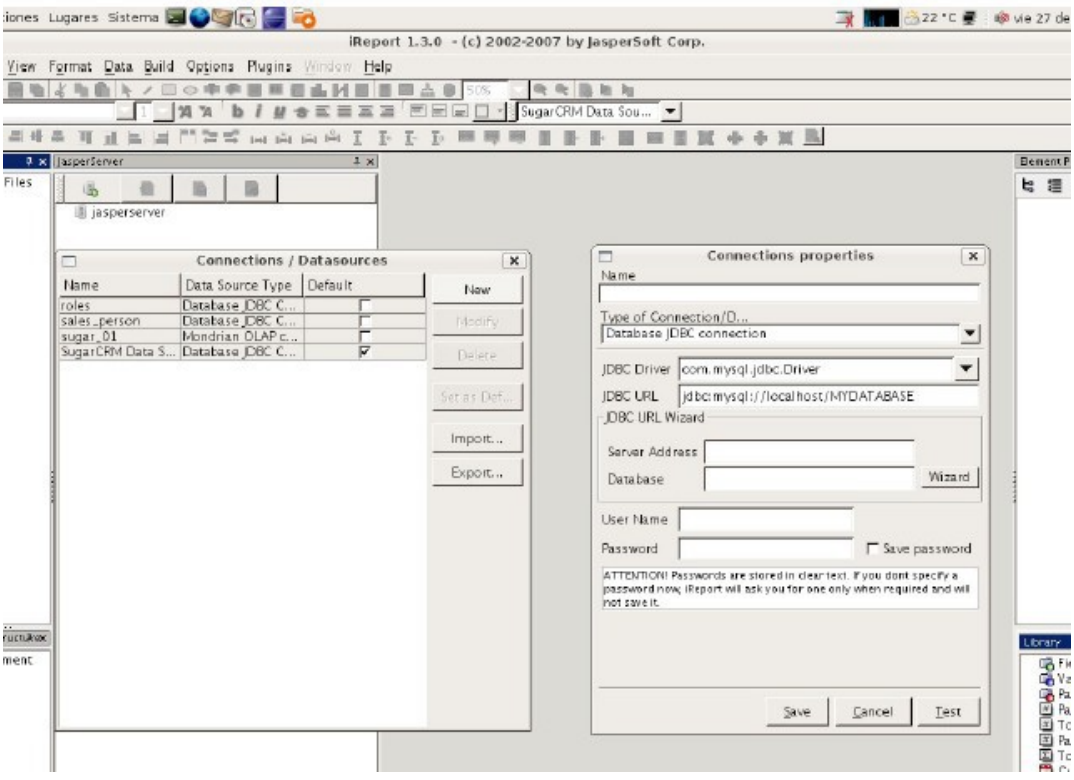
7. Conexión con las fuentes de datos y ejecución de consultas.

Las conexiones con las fuentes de datos son las mismas con las que se realizan las consultas. La diferencia de las diferentes herramientas de generación de informes radican en la automatización de dichas tareas.

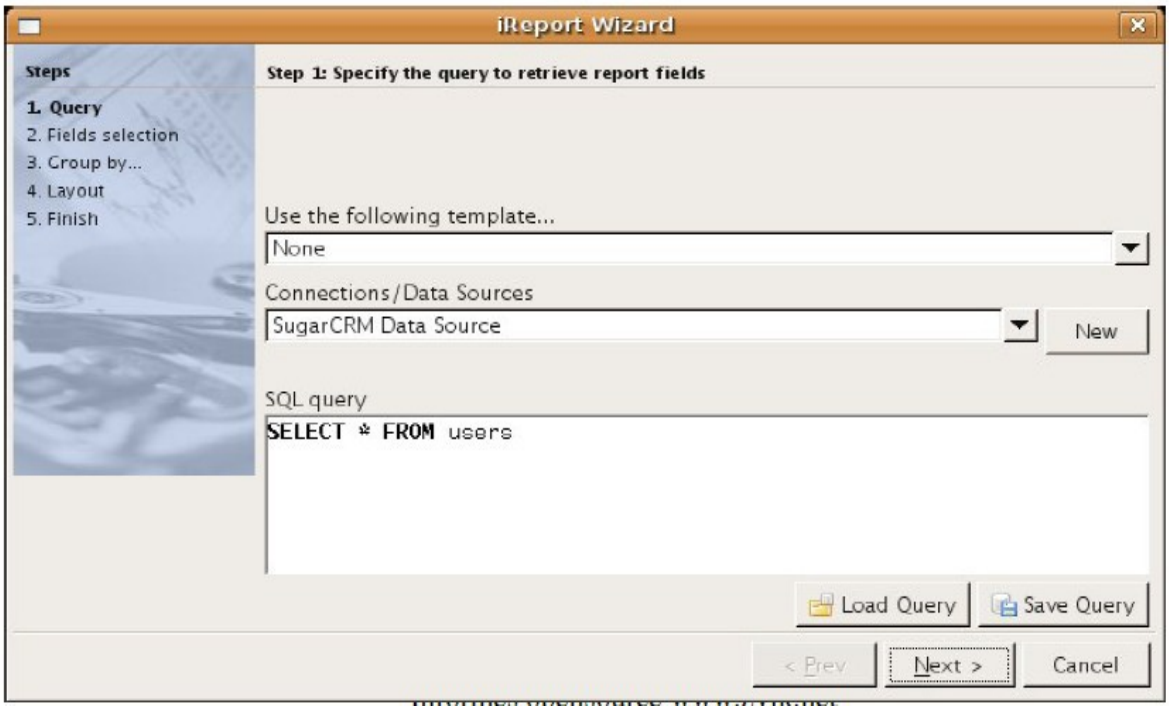
Mientras en VB.NET un asistente puede guiar al programador en la conexión de datos, en Python, la conexión más habitual se la manual. Eso no implica que en el primer caso la podamos hacer manual, y en el segundo existan herramientas que lo permitan hacer de forma automática. En resumen, todos los lenguajes de programación, o casi todos, permiten ambas formas de conectar la aplicación la base o fuente de datos para la elaboración de consultas e informes detallados respectivamente.

En la siguiente figura podemos observar como JaserServer realiza una conexión a una base de datos de forma automatizada monstrándonos la herramienta utilizada para ello:

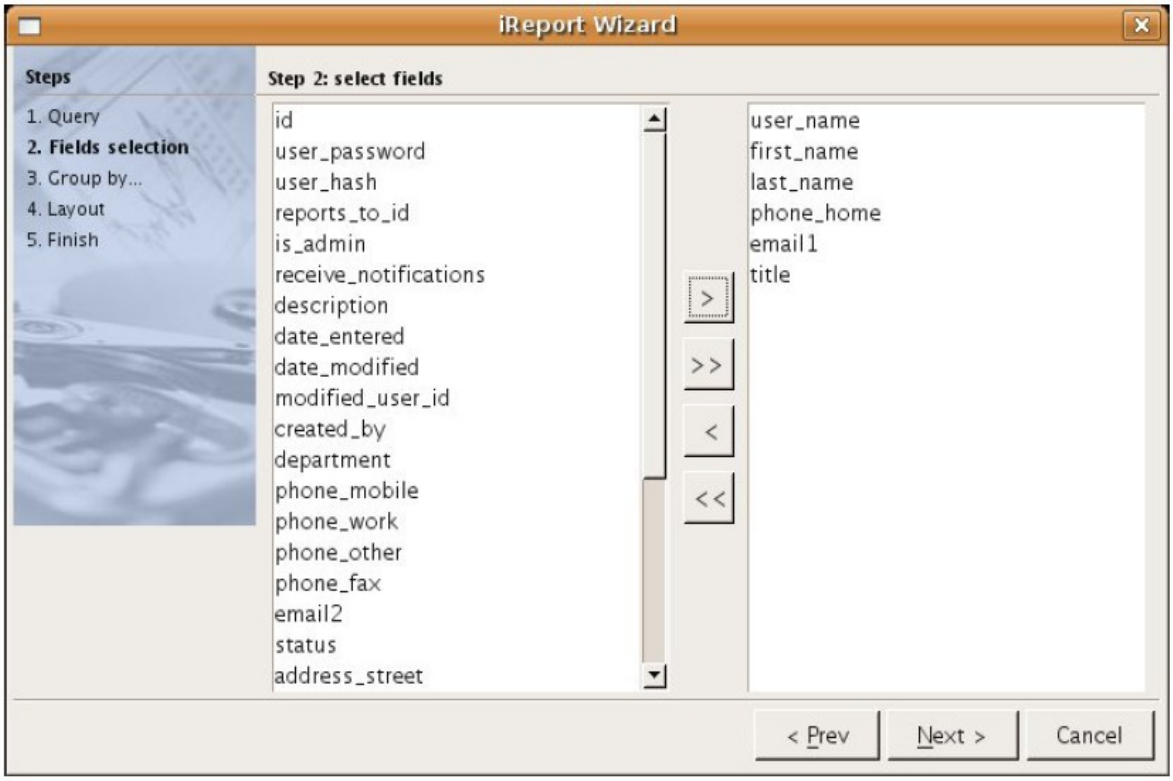
1. **Conexión con la base de datos**



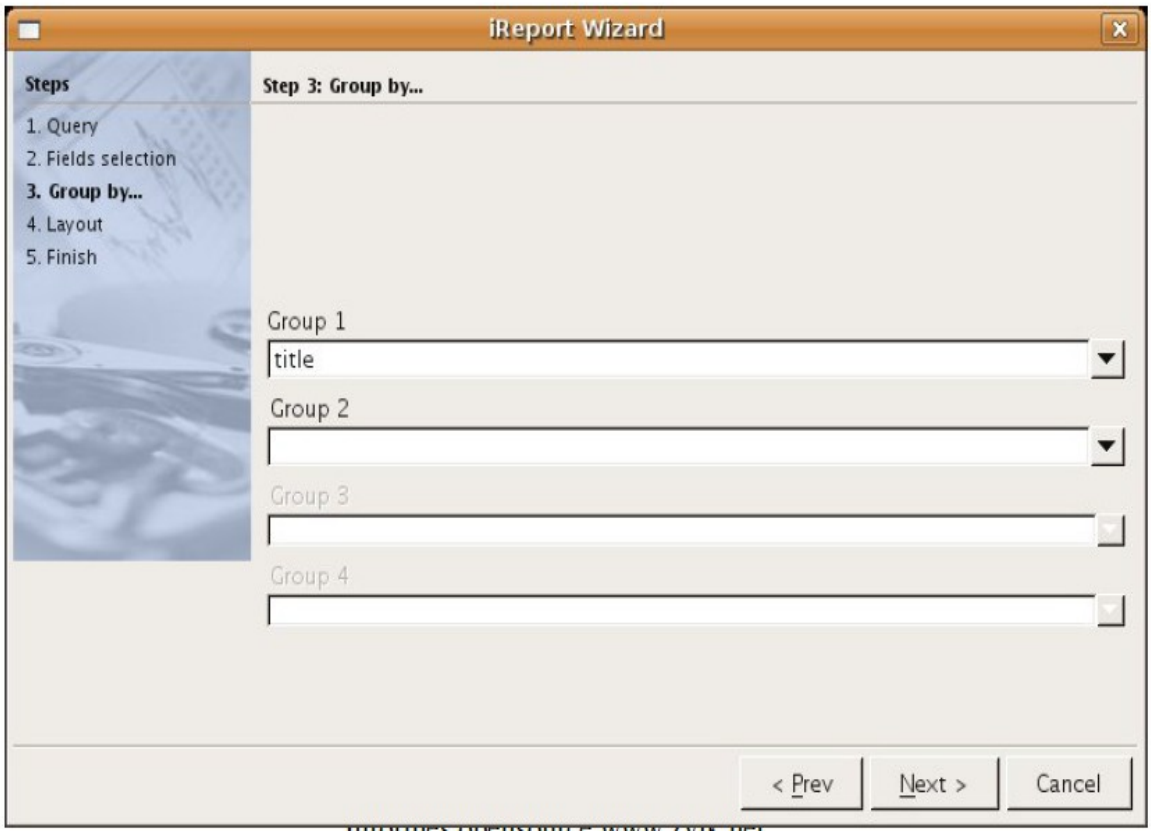
2. Inicio del asistente



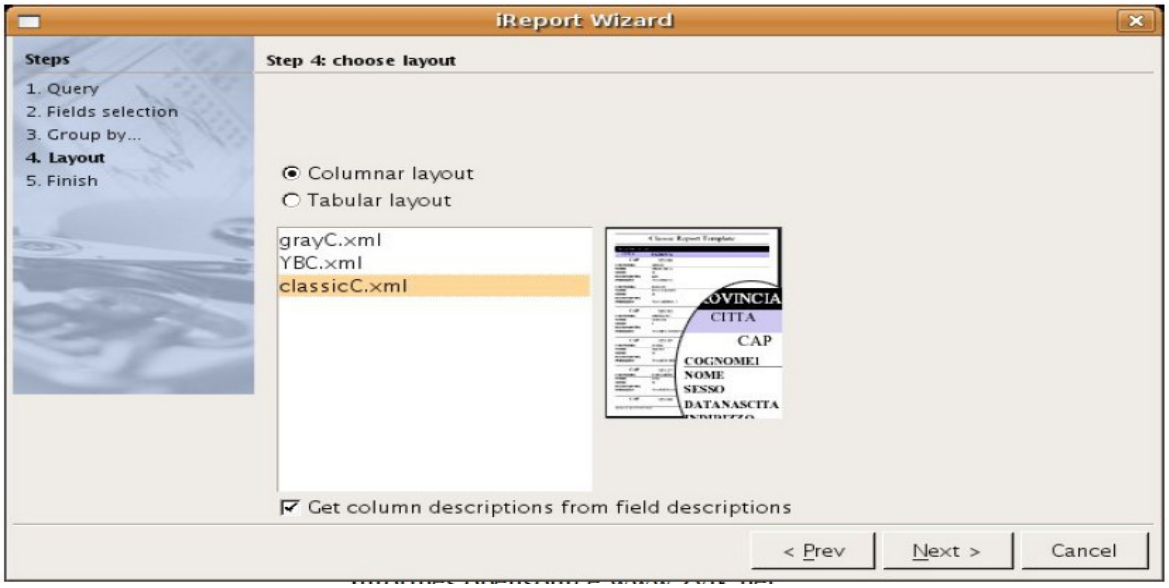
3. Seleccionamos los campos



4. Seleccionamos algún criterio o parámetro



5. Elegimos la presentación



La mayoría de los generadores de informes automatizados funcionan de una manera similar a esta.

Por otro lado puede que la conexión a la base de datos se tenga que hacer de forma manual. Un ejemplo podría ser este basándonos en PHP:

```
<?php
#Conectamos con MySQL
$conexion = mysql_connect("Host","User","Pass") //Host=ip o nombre del servidor
or die ("Fallo en el establecimiento de la conexión");

#Seleccionamos la base de datos a utilizar
mysql_select_db("NombreBaseDatos")
or die("Error en la selección de la base de datos");

#Efectuamos la consulta SQL
$result = mysql_query ("select * from personal" )
or die("Error en la consulta SQL");

#Mostramos los resultados obtenidos o enviamos a informe
while( $row = mysql_fetch_array ( $result )) {
    echo $row [ "id" ];
    echo $row [ "nombre" ];
}
mysql_close();    //cerramos la conexión
?>
```

En el caso de los informes el código anterior se añadiría en el módulo o librería que genera el informe tal como vimos con pyfpdf en Python.

Bibliografía