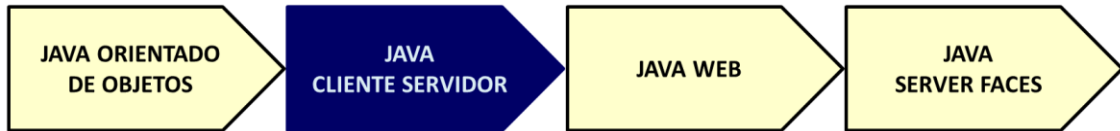


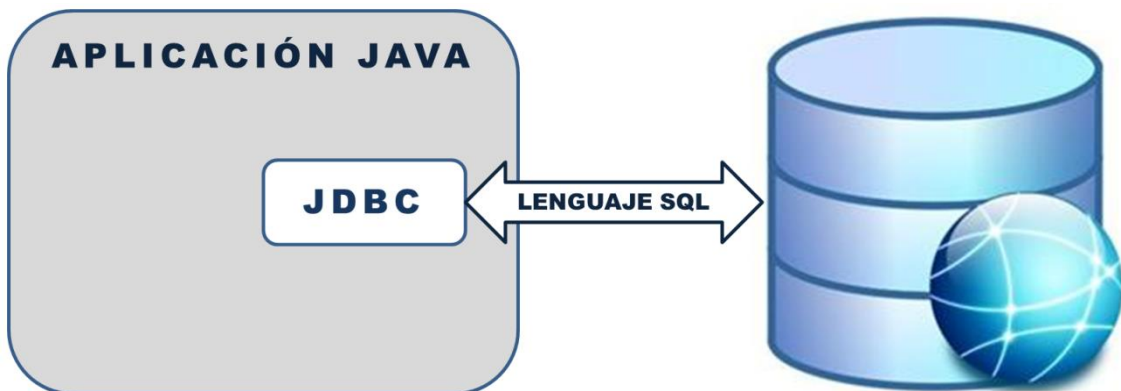


**CEPSUNI**  
Centro de Extensión y Proyección Social  
Universidad Nacional de Ingeniería



# **JAVA**

## **CLIENTE - SERVIDOR**



CEP - UNI  
Febrero - 2016

Programa: JAVA PROFESSIONAL DEVELOPER

Curso: Java Cliente - Servidor

Edición: Febrero – 2016

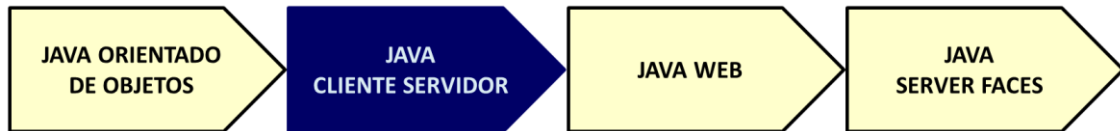
**Eric Gustavo Coronel Castillo**

Instructor Java EE

[www.desarrollasoftware.com](http://www.desarrollasoftware.com)

[gcoronelc@gmail.com](mailto:gcoronelc@gmail.com)

## PRESENTACION



Las aplicaciones de escritorio son quizás las más utilizadas por las empresas, también se les llama Aplicaciones Cliente-Servidor.

Este tipo de aplicaciones se instalan en la computadora del usuario que la va a utilizar y se conecta a una base de datos utilizando JDBC a través de la red de la empresa.

También se pueden conectar con servidores de bases de datos remotas utilizando Internet, una técnica muy utilizada es utilizar redes VPN.

La tecnología JDBC permite ejecutar sentencias SQL, procedimientos almacenados y manejar transacciones de manera muy eficiente y segura.

JDBC proporciona los siguientes objetos para programar los componentes de persistencia:

- Connection: Para establecer la conexión con la fuente de datos y manejar las transacciones.
- Statement: Para ejecutar sentencias sin parámetros.
- PreparedStatement: Para ejecutar sentencias con parámetros.
- CallableStatement: Para ejecutar procedimientos almacenados.
- ResultSet: para procesar conjunto de resultados.
- SQLException: Para la gestión de errores.

Para la creación de reporte se utiliza iReport, esta aplicación permite el diseño de reportes muy versátiles y para su ejecución se utiliza la librería Jasper Report.

Eric Gustavo Coronel Castillo  
Instructor Java EE  
CEPS - UNI



# INDICE

<b>CAPÍTULO 1 INTRODUCCIÓN A GIT Y GITHUB .....</b>	<b>6</b>
OBTENER EL SOFTWARE .....	6
PROCESO DE INSTALACIÓN .....	6
COMANDOS INICIALES .....	11
<i>Verificar la instalación de Git .....</i>	<i>11</i>
<i>Consultar la versión de Git instalada .....</i>	<i>11</i>
<i>Registra tu nombre .....</i>	<i>12</i>
<i>Registra tu correo electrónico .....</i>	<i>12</i>
CREAR UN REPOSITORIO EN GITHUB .....	13
<i>Crear una cuenta en GitHub .....</i>	<i>13</i>
<i>Procede a crear un repositorio .....</i>	<i>14</i>
CLONAR REPOSITORIO DEMO .....	15
EDITAR REPOSITORIO .....	17
<i>Modificar el archivo .gitignore .....</i>	<i>17</i>
<i>Crea un archivo .....</i>	<i>17</i>
ACTUALIZAR EL REPOSITORIO EN GITHUB .....	18
<i>Verificar el estado del repositorio .....</i>	<i>18</i>
<i>Confirmar los cambios en el repositorio local .....</i>	<i>18</i>
<i>Subir los cambios a GitHub .....</i>	<i>19</i>
ACTUALIZAR EL REPOSITORIO LOCAL .....	21
<i>Modifica el archivo into.txt en GitHub .....</i>	<i>21</i>
<i>Actualiza tu repositorio local .....</i>	<i>22</i>
 <b>CAPÍTULO 2 ARQUITECTURA DE UNA BASE DE DATOS ORACLE .....</b>	<b>23</b>
ESQUEMA GENERAL .....	23
ESQUEMAS DE BASE DE DATOS .....	23
USUARIOS ADMINISTRADORES .....	24
SYS .....	24
SYSTEM .....	24
PRIVILEGIOS ESPECIALES DE ADMINISTRACIÓN .....	24
SYSDBA .....	24
SYSOPER .....	24
DESBLOQUEAR LA CUENTA SCOTT .....	25
CREAR EL ESQUEMA SCOTT .....	25
VERIFICAR EL SERVICIO DE LA BASE DE DATOS .....	26
 <b>CAPÍTULO 3 CREACION DE UN NUEVO ESQUEMA .....</b>	<b>27</b>
INICIAR SESIÓN COMO SUPER USUARIO .....	27
CREAR EL USUARIO .....	27

ASIGNAR RECURSOS AL USUARIO DEMO .....	28
INICIAR SESIÓN CON USUARIO DEMO.....	28
CREACIÓN DE UNA TABLA .....	28
INGRESO DE DATOS.....	29
USO DEL EDITOR DE TEXTO.....	29
 <b>CAPÍTULO 4 ORACLE SQL DEVELOPER .....</b>	<b>31</b>
EJECUTAR SQL DEVELOPER.....	31
CONFIGURAR LA CONEXIÓN CON SCOTT .....	32
SQL DEVELOPER EN ACCIÓN .....	35
CONEXIÓN REMOTA CON SQL*PLUS .....	35
 <b>CAPÍTULO 5 CREACIÓN DEL ESQUEMA EUREKA .....</b>	<b>36</b>
DIAGRAMA E-R .....	36
OBTENER EL SCRIPT .....	37
EJECUCIÓN DE LOS SCRIPTS .....	38
CONSULTANDO EL CATALOGO .....	39
CONEXIÓN CON SQL DEVELOPER.....	40
CONEXIÓN DESDE NETBEANS .....	41
<i>Database</i> .....	41
<i>Registrar Driver</i> .....	42
<i>Registrar la Conexión</i> .....	43
<i>Ejecución de Sentencias SQL</i> .....	45
 <b>CAPÍTULO 6 APLICACIONES CLIENTE – SERVIDOR.....</b>	<b>46</b>
ACLARANDO CONCEPTOS .....	46
<i>Cliente-Servidor</i> .....	46
<i>Aplicación Servidor</i> .....	46
<i>Aplicación Cliente</i> .....	46
ARQUITECTURA CLIENTE-SERVIDOR.....	47
<i>Modelo Clásico</i> .....	47
<i>Modelo en Internet</i> .....	47
 <b>CAPÍTULO 7 API JDBC .....</b>	<b>48</b>
JDBC .....	48
TIPOS DE DRIVER.....	49
<i>Tipo 1: JDBC-ODBC bridge driver</i> .....	49
<i>Tipo 2: Native API/Partly Java Driver</i> .....	50
<i>Tipo 3: Pure Java Driver</i> .....	50
<i>Tipo 4: Native Protocol Java Driver</i> .....	50
COMPONENTES PRINCIPALES.....	51
<i>Objeto: Connection</i> .....	51

Objeto: Statement .....	51
Objeto: PreparedStatement .....	51
Objeto: CallableStatement .....	51
Objeto ResultSet .....	51
Objeto: SQLException .....	51
CONSULTAR DATOS .....	52
Acceso a la base de datos .....	52
Mapeo de Tablas .....	53
Consultar un Registro .....	55
Consultar un Conjunto de Registros .....	56
ESTRATEGIAS PARA PROGRAMAR TRANSACCIONES .....	58
Controladas Desde el Cliente .....	58
Controladas en la Base de Datos .....	58
PROGRAMANDO TRANSACCIONES .....	59
Transacción Controlada Desde el Cliente .....	59
Transacción de Base de Datos .....	61
<b>CAPÍTULO 8 APACHE POI .....</b>	<b>64</b>
INTRODUCCIÓN .....	64
LEER UN ARCHIVO EXCEL .....	65
CREAR UN ARCHIVO EXCEL .....	67
<b>CAPÍTULO 9 IREPORT &amp; JASPERREPORT .....</b>	<b>73</b>
INTRODUCCIÓN .....	73
Requerimientos de JasperReports .....	73
FUNCIONAMIENTO DE JASPERREPORTS .....	73
Compilación, exportación de reportes de JasperReports .....	75
IREPORT .....	76
Funcionamiento de iReport .....	76
Configuración de la conexión a una base de datos .....	77
Creación del Reporte .....	78
Secciones de un Reporte en iReport .....	79
Diseño del Reporte .....	81
Compilación y Ejecución del Reporte .....	84
CREACIÓN DE GRÁFICOS EN IREPORT .....	88
<b>CAPÍTULO 10 PRACTICAS DE LABORATORIO .....</b>	<b>96</b>
PRACTICA 01 .....	96
Objetivo .....	96
Actividades Previas .....	96
Creación del Esquema SCOTT .....	96
Creación del Esquema EUREKA .....	96
PRACTICA 02 .....	97



<i>Objetivo</i> .....	97
<i>Proyecto 1</i> .....	97
<i>Proyecto 2</i> .....	97
PRACTICA 03 .....	98
<i>Objetivo</i> .....	98
<i>Proyecto</i> .....	98
PROYECTO 04.....	99
<i>Objetivo</i> .....	99
<i>Proyecto</i> .....	99
PRACTICA 05 .....	101
<i>Objetivo</i> .....	101
<i>Proyecto</i> .....	101
PROYECTO 06.....	102
<i>Objetivo</i> .....	102
<i>Proyecto</i> .....	102
PROYECTO 07.....	103
<i>Objetivo</i> .....	103
<i>Proyecto</i> .....	103

# Capítulo 1

## Introducción a Git y GitHub

### OBTENER EL SOFTWARE

Te recomiendo que instales GIT FOR WINDOWS, la ruta es la siguiente:

<https://git-for-windows.github.io/>

El archivo que debes descargar para Windows de 64 Bits es el siguiente:

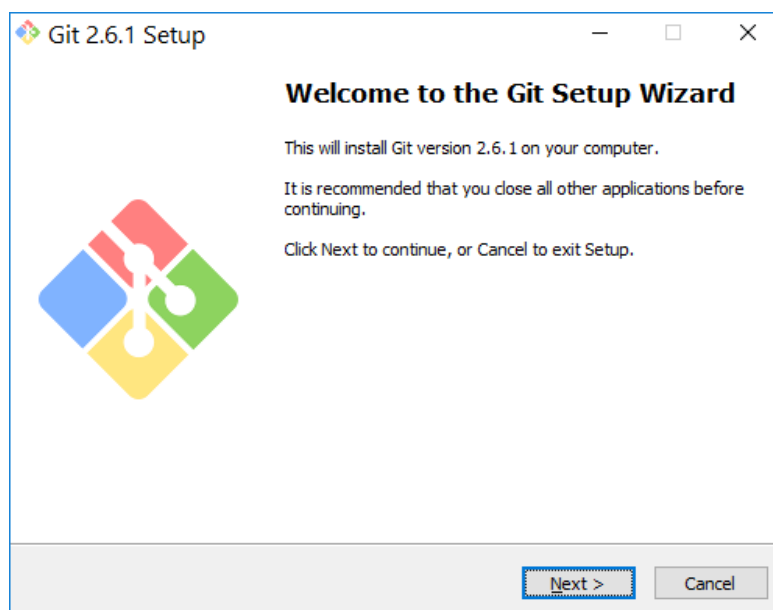
Git-2.6.1-64-bit.exe

Es posible que en este momento que estás leyendo este documento ya exista una nueva versión.

### PROCESO DE INSTALACIÓN

Durante el proceso de instalación debes integrarlo con la consola de Windows.

1. Ventana de bienvenida, solo debes hacer click en el botón **Next**.

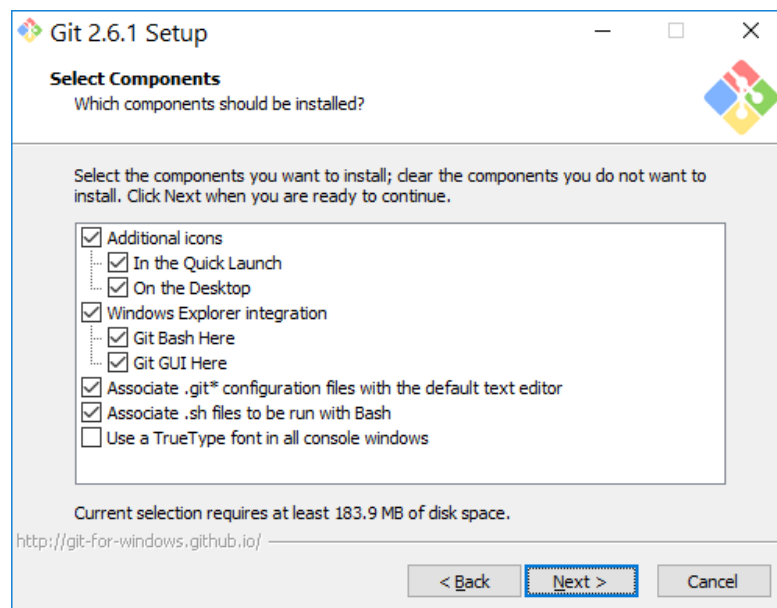




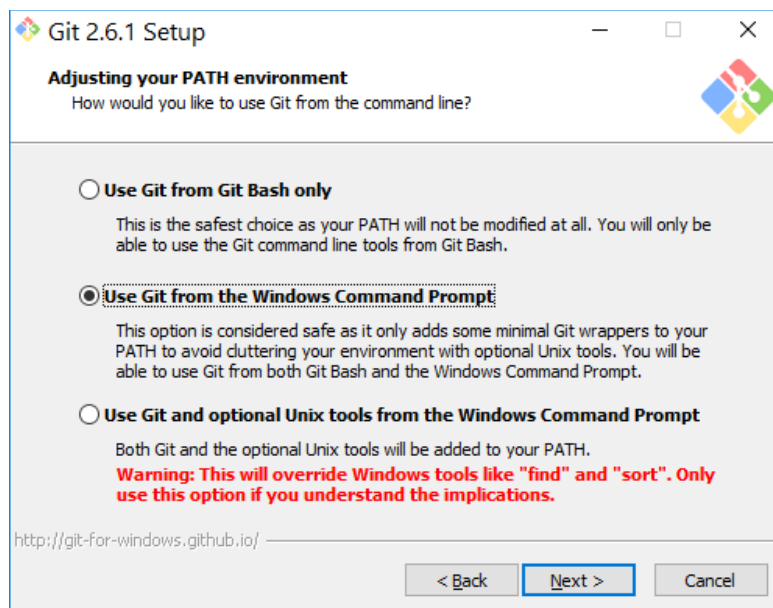
2. Ventana de licencia del software, solo debes hacer click en el botón **Next**.



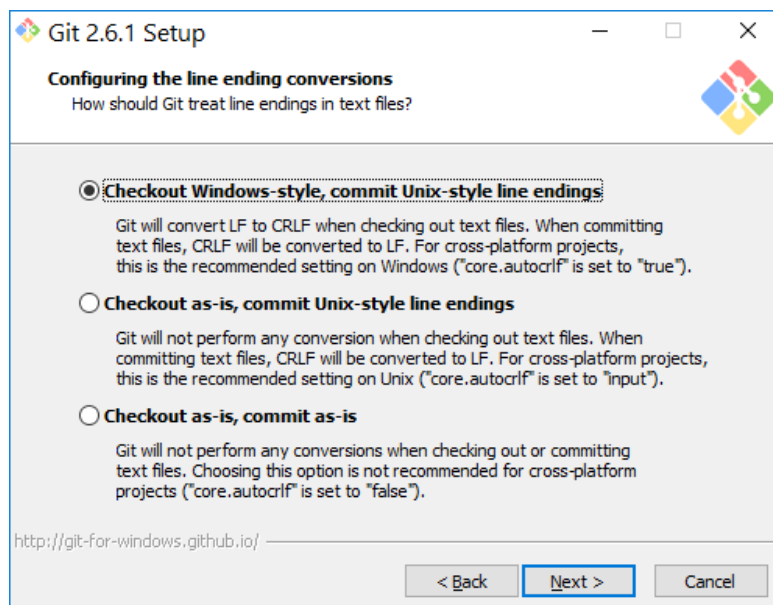
3. Ventana de componentes a instalar, seleccione los componentes tal como se ilustra en la siguiente imagen y haga click en el botón **Next**.



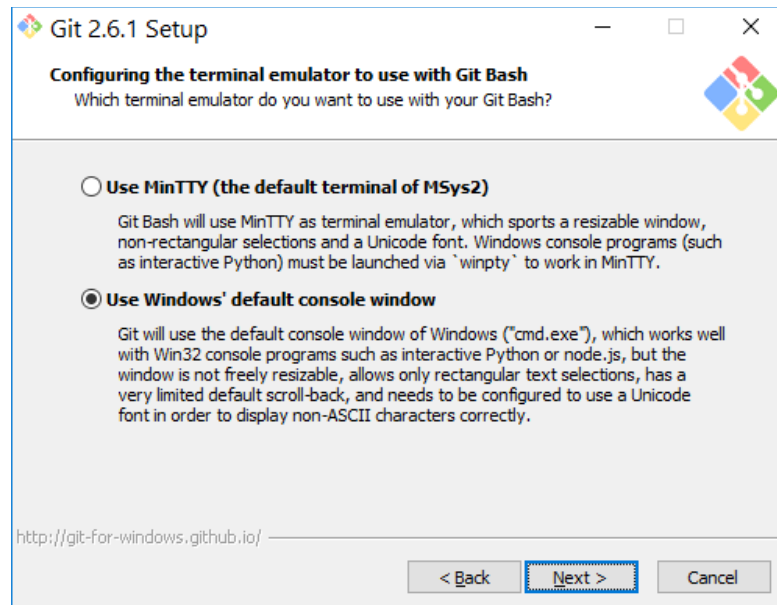
4. Ventana de configuración de la variable PATH de entorno de Windows. Seleccione la opción tal como aparece en la siguiente imagen y haga click en el botón **Next**.



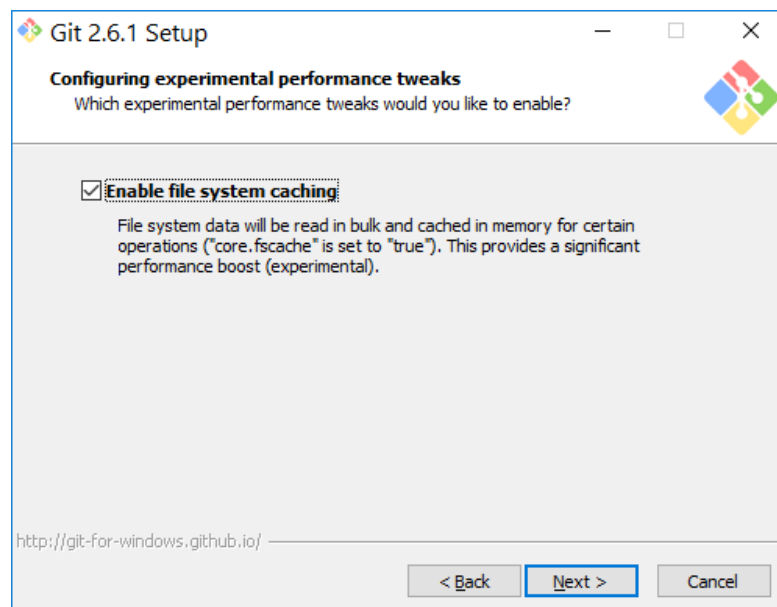
5. Ventana de configuración de conversiones de fin de línea. Deje la opción por defecto y haga click en el botón **Next**.



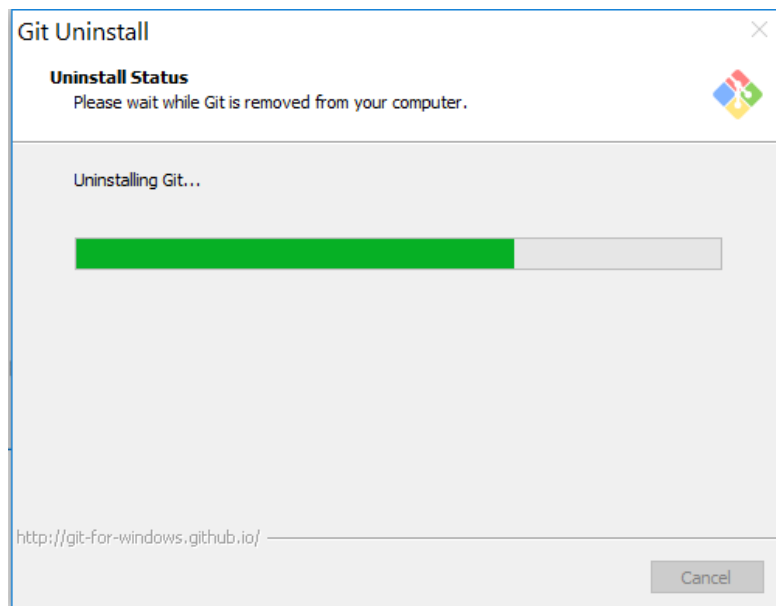
6. Ventana de configuración de terminal. Seleccione la opción que aparece en la siguiente imagen y haga click en el botón **Next**.



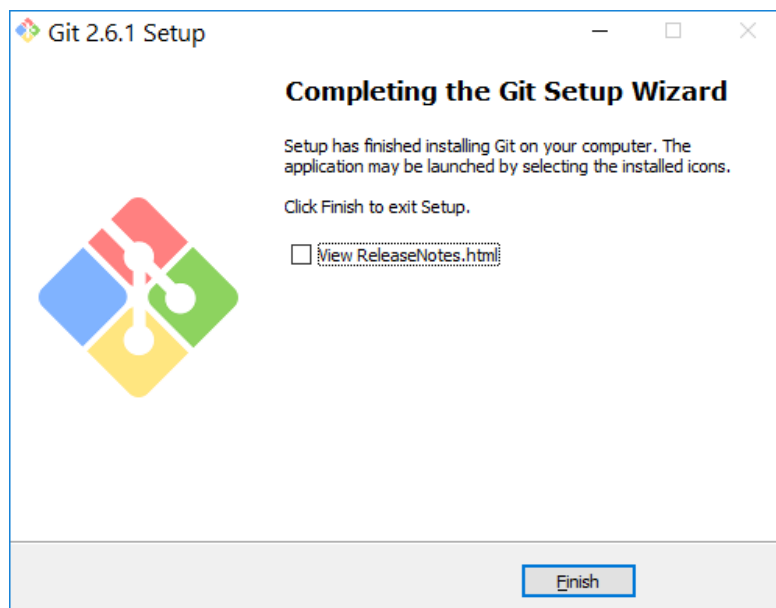
7. En la siguiente ventana configure el uso de sistema de cache y haga click en el botón **Next**.



8. Luego se inicia el proceso de instalación.

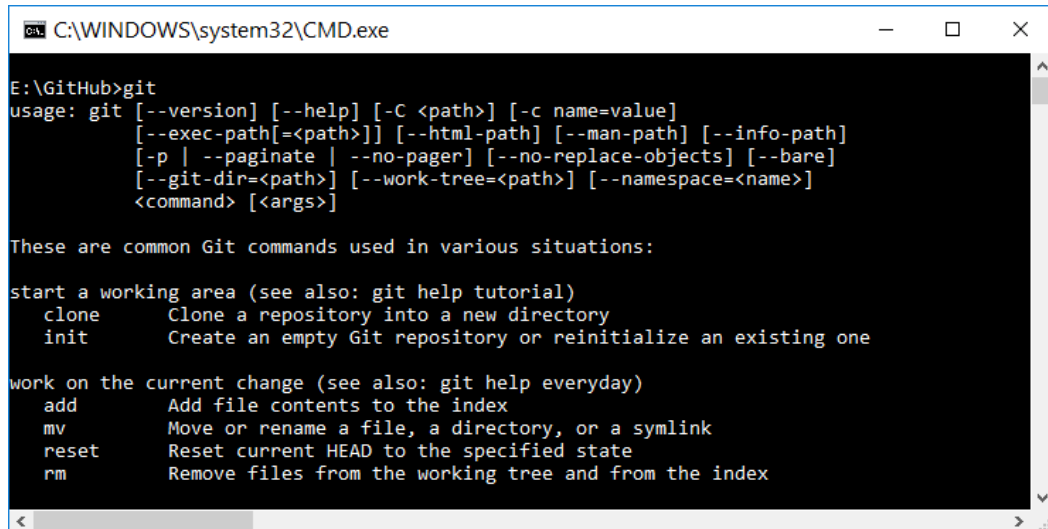


9. Finalmente, el proceso de instalación finaliza, haga click en el botón **Finish**.



## COMANDOS INICIALES

Todos los comandos se deben ejecutar en la consola de Windows. Personalmente, tengo una carpeta GitHub donde tengo todos mis repositorios que mantengo en GitHub.



```
C:\WINDOWS\system32\CMD.exe

E:\GitHub>git
usage: git [--version] [--help] [-C <path>] [-c name=value]
         [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
         [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
         [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
         <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
```

### Verificar la instalación de Git

Cuando ejecutas el comando **git** muestra su sintaxis una ayuda sobre cada uno de los comandos que puedes ejecutar para gestionar un repositorio git.

```
E:\GitHub>git
usage: git [--version] [--help] [-C <path>] [-c name=value]
         [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
         [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
         [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
         <command> [<args>]
```

### Consultar la versión de Git instalada

```
E:\GitHub>git --version
git version 2.6.1.windows.1
```



## Registra tu nombre

Es muy importante que registres tu nombre para saber quién o quienes estan registrando cambios en el repositorio.

La sintaxis es la siguiente:

```
git config --global user.name "Aquí escribe tu nombre y apellido"
```

Por ejemplo, para mi caso sería así:

```
E:\GitHub>git config --global user.name "Eric Gustavo Coronel Castillo"
```

## Registra tu correo electrónico

Es muy importante que registres tu correo electrónico para saber quién o quienes están registrando cambios en el repositorio.

La sintaxis es la siguiente:

```
git config --global user.email "Aquí escribe tu correo electrónico"
```

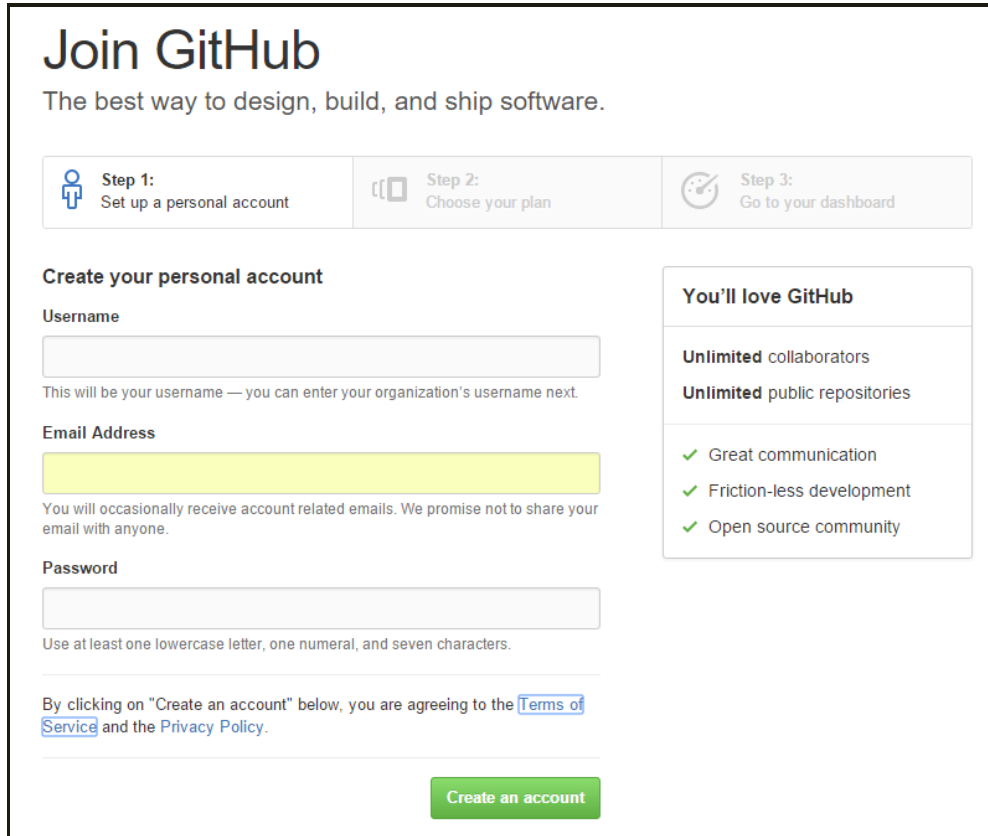
Por ejemplo, para mi caso sería así:

```
E:\GitHub>git config --global user.email "gcoronelc@gmail.com"
```

## CREAR UN REPOSITORIO EN GITHUB

### Crear una cuenta en GitHub

En GitHub (<https://github.com>) procede a crear tu cuenta de usuario.



**Join GitHub**  
The best way to design, build, and ship software.

**Step 1:** Set up a personal account  
**Step 2:** Choose your plan  
**Step 3:** Go to your dashboard

**Create your personal account**

**Username**  
  
 This will be your username — you can enter your organization's username next.

**Email Address**  
  
 You will occasionally receive account related emails. We promise not to share your email with anyone.

**Password**  
  
 Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

**Create an account**

**You'll love GitHub**

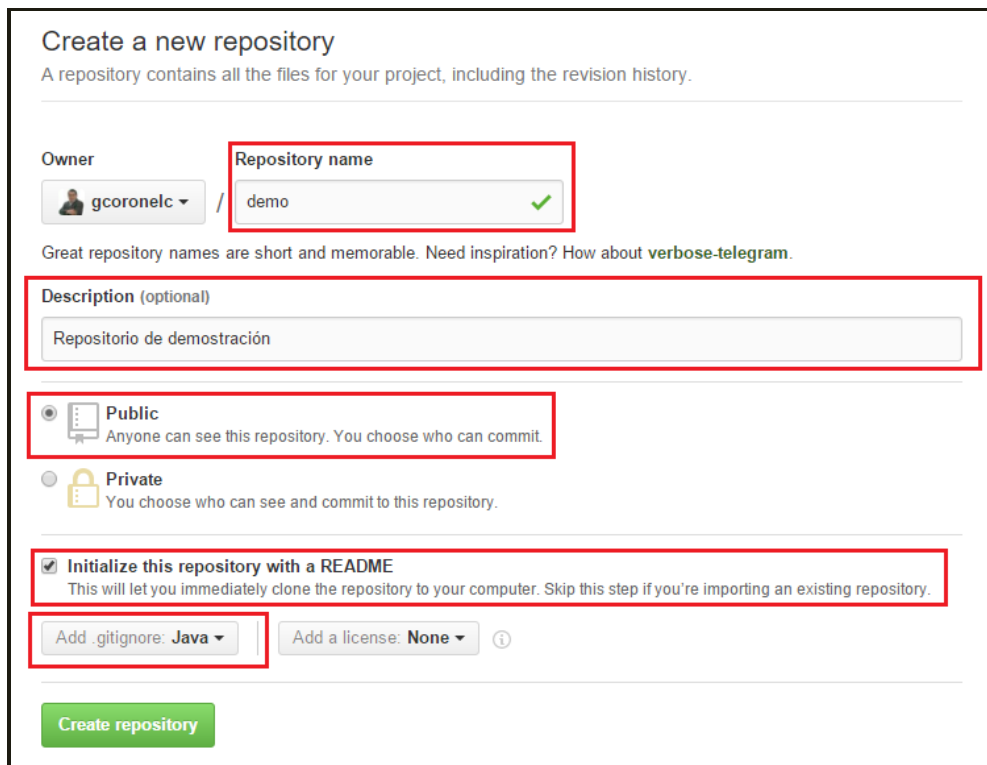
- Unlimited collaborators
- Unlimited public repositories
- ✓ Great communication
- ✓ Friction-less development
- ✓ Open source community

GitHub te enviará un correo electrónico para que confirmes la creación de la cuenta.

## Procede a crear un repositorio

Procede a crear un repositorio de demostración de nombre **demo**.

La siguiente imagen se ilustra la creación de repositorio **demo** con mi cuenta.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: gcoronelc / Repository name: demo

Great repository names are short and memorable. Need inspiration? How about [verbose-telegram](#).

Description (optional): Repositorio de demostración

Public: Anyone can see this repository. You choose who can commit.

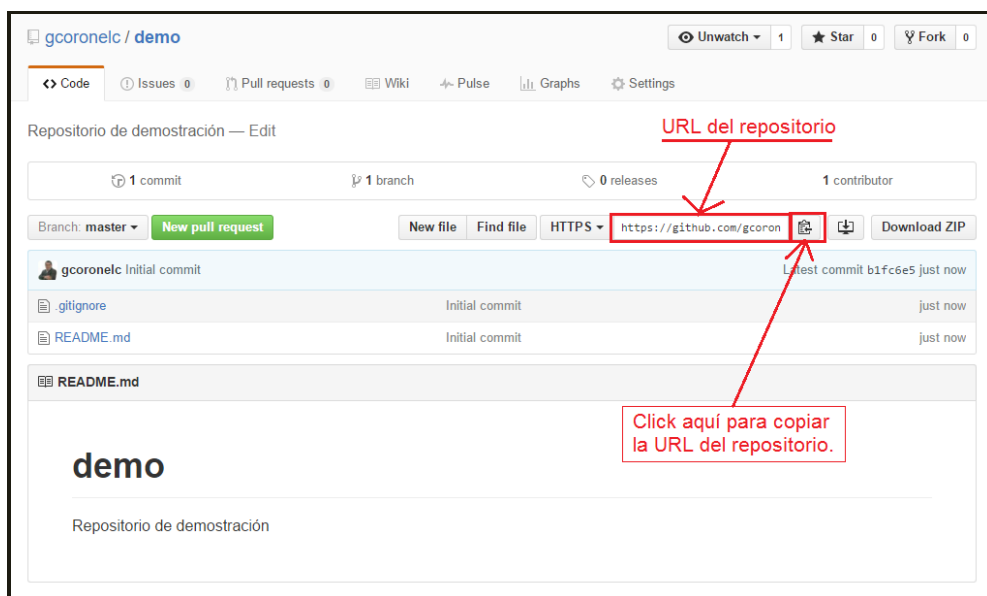
Private: You choose who can see and commit to this repository.

☒ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Java Add a license: None

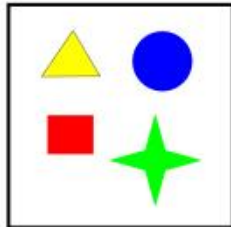
Create repository

A continuación tienes la imagen que confirma la creación del repositorio **demo**.





## CLONAR REPOSITORIO DEMO



Ahora vas a clonar el repositorio **demo** creado en el ítem anterior.

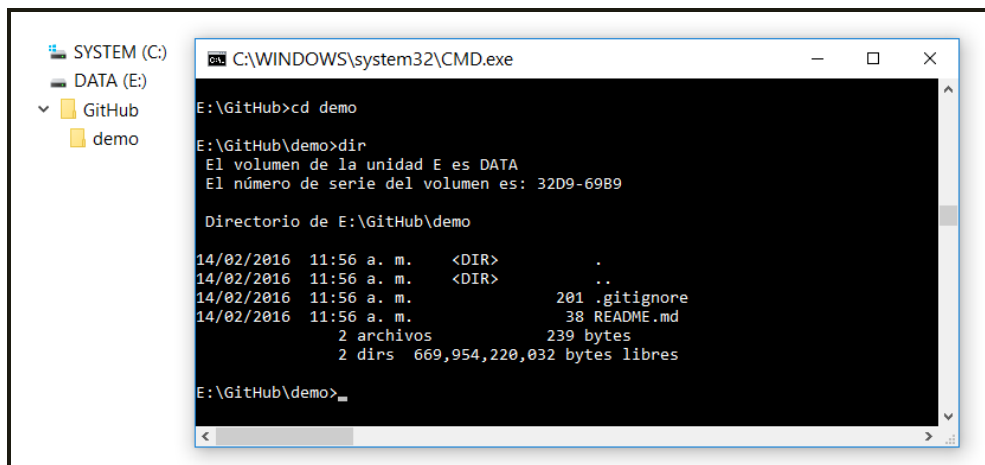
Para clonar un repositorio la sintaxis es:

```
git clone <URL del repositorio>
```

Para el caso del repositorio **demo**, sería así:

```
E:\GitHub>git clone https://github.com/gcoronelc/demo.git
Cloning into 'demo'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
```

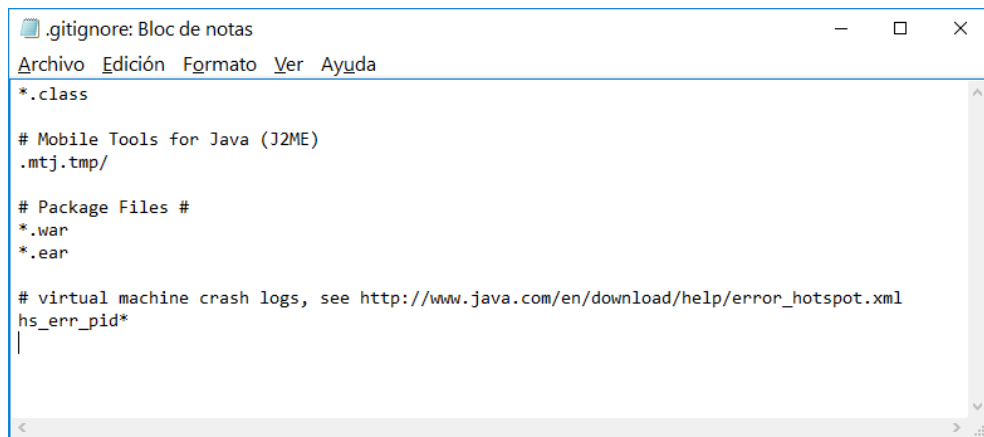
El repositorio ha sido clonado correctamente, no presenta ningún mensaje de error. En la unidad E: puedes verificar que se ha creado la carpeta demo correspondiente a repositorio, tal como se ilustra en la siguiente imagen:



## EDITAR REPOSITORIO

### Modificar el archivo .gitignore

En el editor de texto carga el archivo `.gitignore` y elimina la línea que tenga `"*.jar"`, debe quedar como se muestra en la siguiente imagen:



```
.gitignore: Bloc de notas
Archivo Edición Formato Ver Ayuda

*.class

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.war
*.ear

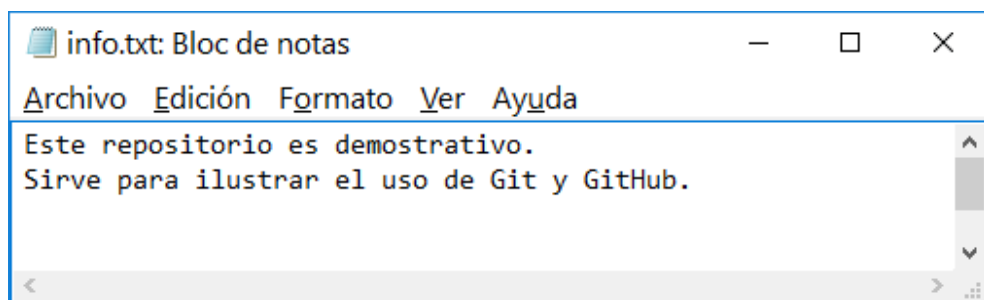
# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
```

Graba y cierra el archivo.

### Crea un archivo

En la carpeta demo del repositorio procede a crear un archivo nuevo de nombre `info.txt`, y registra información sobre información que registrarás en tu repositorio.

La siguiente imagen muestra lo que podría ser el archivo `info.txt`.



```
info.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

Este repositorio es demostrativo.
Sirve para ilustrar el uso de Git y GitHub.
```

## ACTUALIZAR EL REPOSITORIO EN GITHUB

### Verificar el estado del repositorio

El comando a ejecutar es:

```
git status
```

Para mi caso sería así:

```
E:\GitHub\demo>git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        info.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

El resultado nos indica que existen un archivo modificado y uno nuevo.

### Confirmar los cambios en el repositorio local

Para confirmar los cambios, primero debes preparar la lista de archivos a confirmar con el comando **git add**.

Para mi caso, si quiero agregar todos los archivos a la lista:

```
E:\GitHub\demo>git add .
```

Para confirmar la lista de archivos preparada con **git add**, se debe utilizar el comando **git commit**.

Para mi caso, sería así:

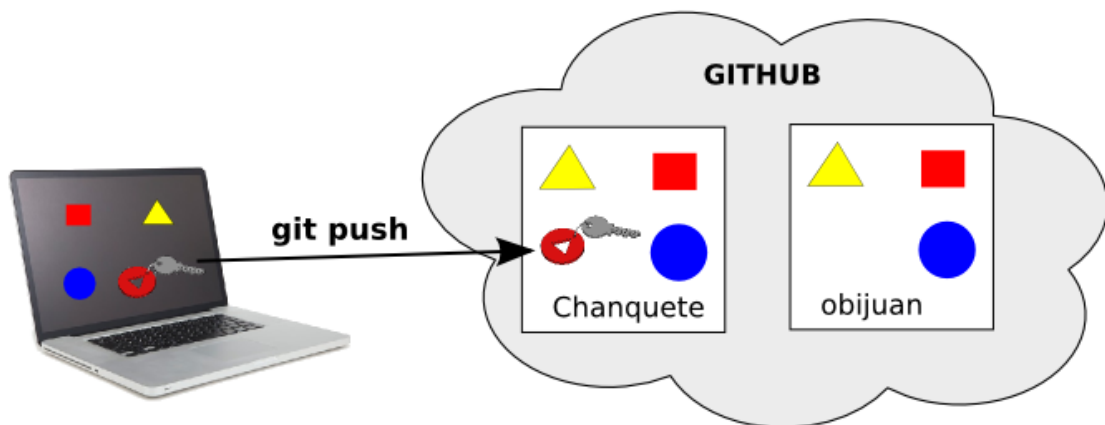
```
E:\GitHub\demo>git commit -m "Probando el repositorio."
[master 5f781ff] Probando el repositorio.
 2 files changed, 1 deletion(-)
 create mode 100644 info.txt
```

Puedes utilizar nuevamente el comando `git status` para verificar el estado de tu repositorio.

```
E:\GitHub\demo>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
```

En este caso indica que el repositorio esta adelantado 1 commit con respecto a repositorio origen, y se debe utilizar `git push` para subir los cambios.

## Subir los cambios a GitHub



Para subir los cambios a GitHub se utiliza el comando `git push`.

```
git push origin master
```

Cuando ejecutas este comando te solicita tu cuenta de usuario y clave, salvo que ya se encuentre configurado, como es mi caso.

A continuación se tiene el resultado para mi caso:

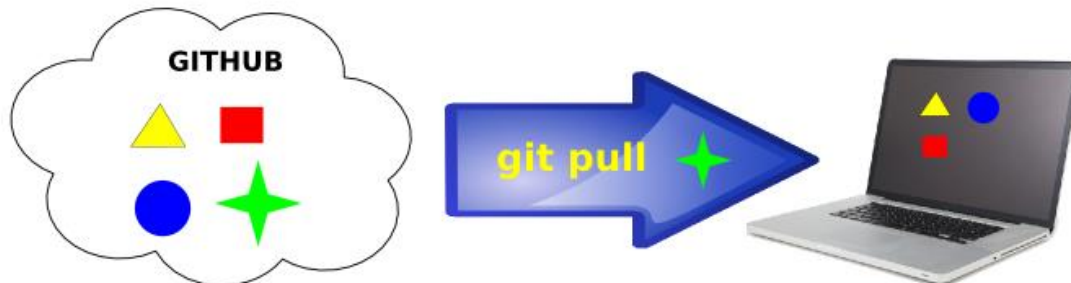
```
E:\GitHub\demo>git push origin master
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 371 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
To https://github.com/gcoronelc/demo.git
    b1fc6e5..5f781ff  master -> master
```

Ahora puedes nuevamente verificar el estado de tu repositorio.

A continuación tienes el resultado para mi repositorio.

```
E:\GitHub\demo>git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

## ACTUALIZAR EL REPOSITORIO LOCAL



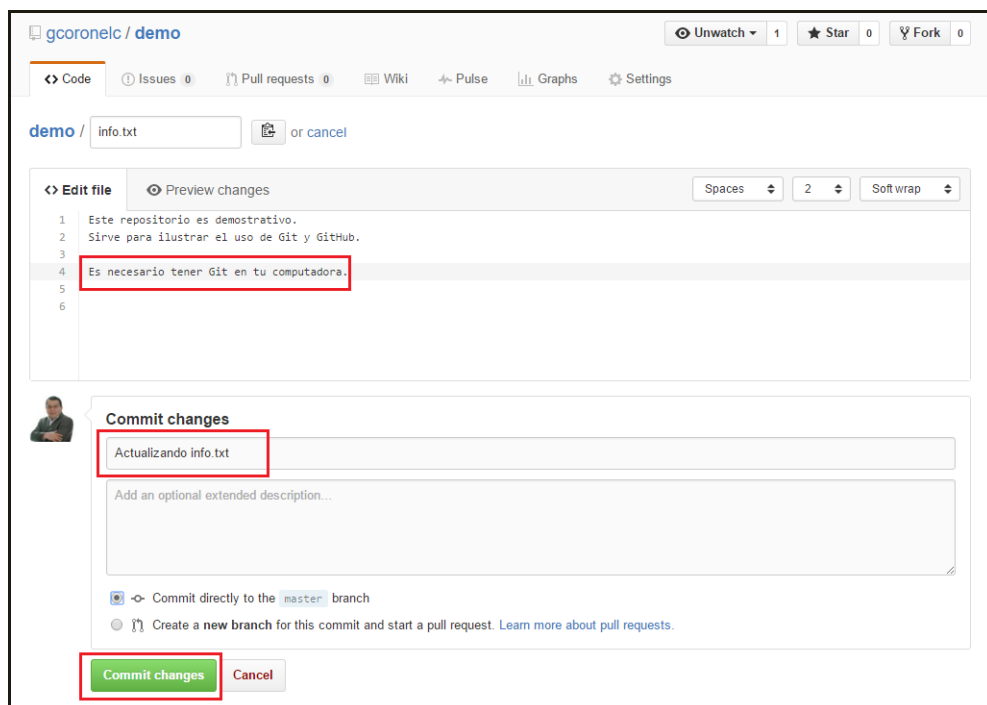
Puede suceder que el repositorio lo clones en la universidad y has realizado varias actualizaciones, luego quieres actualizar el repositorio de tú computadora en tu casa.

En estas situaciones debes usar el comando `git pull` para actualizar tu repositorio local.

### Modifica el archivo `into.txt` en GitHub

Procede a editar el archivo `into.txt` en GitHub y agrégale una línea, y luego hazle `commit`.

A continuación tienes una imagen de como podrías realizarlo:



## Actualiza tu repositorio local

Antes de que realices cualquier cambio en tu repositorio local, se recomienda que lo actualices con los últimos cambios, para eso debe utilizar el comando **git pull**.

Aquí tienes su sintaxis:

```
git pull origin master
```

Aquí tienes un ejemplo con mi repositorio:

```
E:\GitHub\demo>git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/gcoronelc/demo
 * branch                master      -> FETCH_HEAD
    2a5a48e..2317f84      master      -> origin/master
Updating 2a5a48e..2317f84
Fast-forward
 info.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

El proceso se ha ejecutado correctamente, un archivo ha cambiado.

El archivo que ha cambiado es **info.txt**, he indica que una fila se ha insertado, pero, también una fila se ha eliminado, a continuación tienes el script para consultar el contenido del archivo **info.txt**:

```
E:\GitHub\demo>type info.txt
Este repositorio es demostrativo.
Sirve para ilustrar el uso de Git y GitHub.

Es necesario tener Git en tu computadora.
```

Esto ha sido una introducción a **Git** y **GitHub**, suficiente para utilizarlo como repositorio en los cursos de programación, pero si lo que necesitas es usarlo para control de versiones te recomiendo que consultes el material oficial de Git.

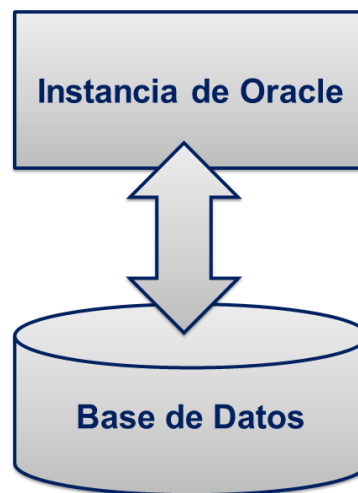


## Capítulo 2

# ARQUITECTURA DE UNA BASE DE DATOS ORACLE

### ESQUEMA GENERAL

La instancia representa la base de datos, pero en memoria.



### ESQUEMAS DE BASE DE DATOS

La base de datos se organiza en esquemas.

Para que exista un esquema, primero debe existir el usuario.

El usuario que tiene privilegios y recursos para crear objetos, será también un esquema.



El esquema **SYS** pertenece al usuario **SYS**, y es donde se guarda el diccionario de datos.

## USUARIOS ADMINISTRADORES

Se deben utilizar para hacer tareas administrativas.

### SYS

Es el propietario del esquema **SYS** y del diccionario de datos.

### SYSTEM

Tiene el rol **DBA** y se recomienda su uso para hacer tareas administrativas.

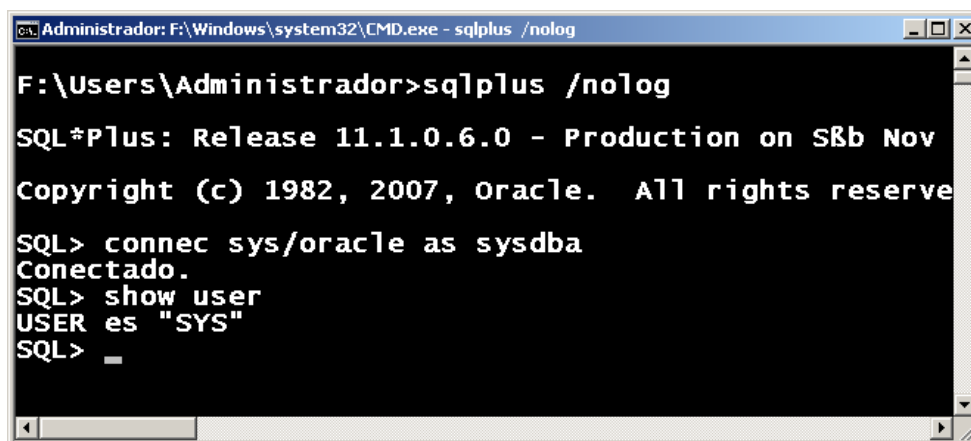
## PRIVILEGIOS ESPECIALES DE ADMINISTRACIÓN

### SYSDBA

Se utiliza para realizar la conexión como usuario **SYS**.

Sintaxis usando SQL\*Plus:

```
CONNECT sys/<clave> AS SYSDBA
```



```
Administrador: F:\Windows\system32\CMD.exe - sqlplus /nolog

F:\Users\Administrador>sqlplus /nolog

SQL*Plus: Release 11.1.0.6.0 - Production on SBb Nov

Copyright (c) 1982, 2007, Oracle. All rights reserve

SQL> connec sys/oracle as sysdba
Conectado.
SQL> show user
USER es "SYS"
SQL>
```

### SYSOPER

Se utiliza para realizar la conexión como usuario **PUBLIC**.

Tiene menos privilegios que **SYSDBA**.

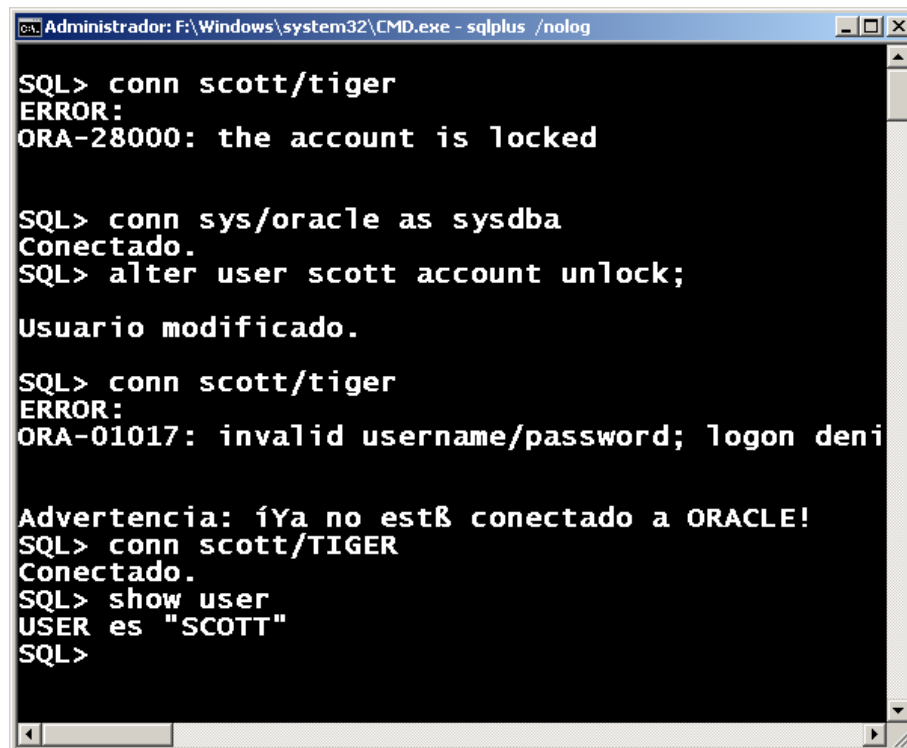
Sintaxis usando SQL\*Plus:

```
CONNECT <usuario>/<clave> AS SYSOPER
```

## DESBLOQUEAR LA CUENTA SCOTT

Si tú instalación de Oracle ya trae el esquema **SCOTT** es posible que la cuenta se encuentre bloqueada, lo primero que se debe hacer es desbloquearla.

Se utiliza el comando **ALTER USER** como se ilustra a continuación:



```
Administrador: F:\Windows\system32\CMD.exe - sqlplus /nolog

SQL> conn scott/tiger
ERROR:
ORA-28000: the account is locked

SQL> conn sys/oracle as sysdba
Conectado.
SQL> alter user scott account unlock;
Usuario modificado.

SQL> conn scott/tiger
ERROR:
ORA-01017: invalid username/password; logon denied

Advertencia: ¡Ya no estás conectado a ORACLE!
SQL> conn scott/TIGER
Conectado.
SQL> show user
USER es "SCOTT"
SQL>
```

De los resultados observados, se puede constatar que la clave está mayúsculas.

## CREAR EL ESQUEMA SCOTT

En caso de que tú instalación de Oracle no tenga el esquema **SCOTT**, se debe crear el esquema.

El script se encuentra en la carpeta **RDBMS\ADMIN**.

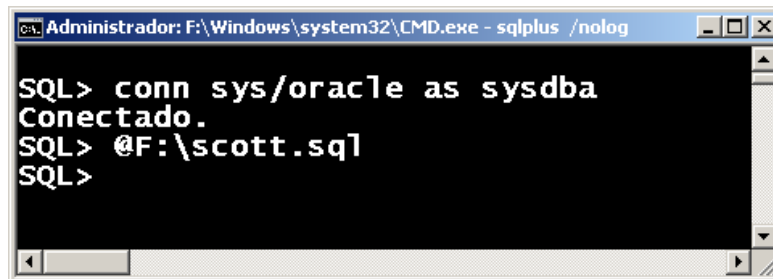
Si tienes instalado el Oracle 11, la ruta es la siguiente:

```
F:\app\Administrador\product\11.1.0\db_1\RDBMS\ADMIN
```

Si tu Oracle está instalado en la partición C, debes ubicar la ruta en la partición C.

Aquí encontraras el archivo llamado **scott.sql**, debe copiarlo en la raíz de la unidad de trabajo, en este caso F:.

Para ejecutar un archivo .sql se debe utilizar el comando **RUN** o **@**, a continuación se ilustra la ejecución del archivo **scott.sql** que se encarga de crear el esquema **SCOTT**.



```
Administrador: F:\Windows\system32\CMD.exe - sqlplus /nolog

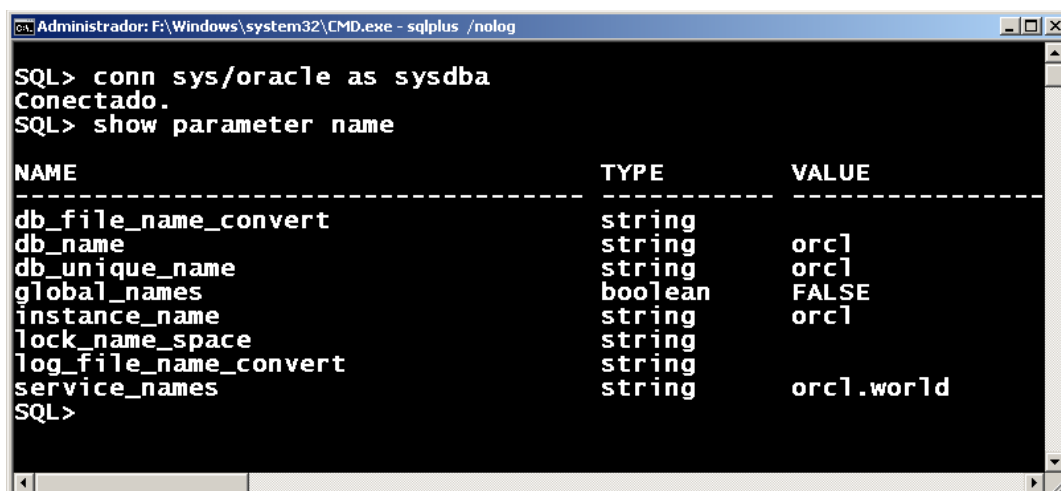
SQL> conn sys/oracle as sysdba
Conectado.
SQL> @F:\scott.sql
SQL>
```

## VERIFICAR EL SERVICIO DE LA BASE DE DATOS

Como usuario **SYS** debe ejecutar el siguiente comando:

```
SHOW PARAMETER NAME
```

Se muestran todos los parámetros que tienen en su nombre la palabra **name**.



```
Administrador: F:\Windows\system32\CMD.exe - sqlplus /nolog

SQL> conn sys/oracle as sysdba
Conectado.
SQL> show parameter name

NAME                                TYPE                                VALUE
-----                                -                                -
db_file_name_convert                string                              orcl
db_name                             string                              orcl
db_unique_name                      string                              orcl
global_names                        boolean                             FALSE
instance_name                      string                              orcl
lock_name_space                    string                              orcl
log_file_name_convert               string                              orcl
service_names                      string                              orcl.world
SQL>
```

El parámetro **instance\_name** muestra el valor del **SID**, en este caso es **orcl**.

El parámetro **service\_names** muestra el nombre del servicio, en este caso **orcl.world**.

## Capítulo 3

# CREACION DE UN NUEVO ESQUEMA

### INICIAR SESIÓN COMO SUPER USUARIO

Se puede utilizar el usuario **SYS** con privilegio **SYSDBA** o el usuario **SYSTEM**.

En este caso se está utilizando el usuario **SYS** con autenticación del **SO**:

```
C:> sqlplus / as sysdba  
  
sql>
```

En caso de no lograr conectarte debes utilizar la contraseña del usuario **SYS**, por lo general es **oracle**:

```
C:> sqlplus sys/oracle as sysdba  
  
sql>
```

Si utilizas el usuario **SYSTEM**, usualmente la contraseña es **oracle**:

```
C:> sqlplus system/oracle  
  
sql>
```

### CREAR EL USUARIO

Crearemos el usuario **demo** con contraseña **admin**:

```
sql> CREATE USER demo IDENTIFIED BY admn;
```

## ASIGNAR RECURSOS AL USUARIO DEMO

Para que este usuario **demo** pueda tener privilegio de poder crear objetos y ser considerado como esquema se le asignará los roles **connect**, y **resource**:

```
sql> GRANT connect, resource TO demo;
```

## INICIAR SESIÓN CON USUARIO DEMO

Probaremos si la creación del usuario **demo** ha sido correcta:

```
SQL> CONNECT demo/admn  
Conectado.
```

```
SQL> SHOW USER  
USER es "DEMO"
```

## CREACIÓN DE UNA TABLA

Con el usuario **demo** procederemos a crear la tabla **mensaje**:

```
SQL> SHOW USER  
USER es "DEMO"
```

```
SQL> CREATE TABLE mensaje(  
2 id number(5,0) not null primary key,  
3 de varchar2(20) not null,  
4 para varchar2(20) not null,  
5 texto varchar2(144) not null  
6 );
```

```
Tabla creada.
```

También se creará una secuencia para los datos de la columna **id**, este objeto proporcionará valores únicos iniciando en el valor 1, luego 2, 3, 4 y así sucesivamente:

```
SQL> CREATE SEQUENCE sq_mensaje;
```

Secuencia creada.

## INGRESO DE DATOS

Probaremos ingresar datos a la tabla **mensaje**:

```
SQL> INSERT INTO mensaje
  2  VALUES(sq_mensaje.nextval,'profesor','estudiantes','Hola a todos');
```

1 fila creada.

## USO DEL EDITOR DE TEXTO

Por defecto, SQL\*PLUS guarda en memoria la última sentencia SQL ejecutada, y la puedes cargar en el editor de texto con el comando **edit**.

Al ejecutar el comando **edit** se cargara el editor de texto con la última sentencia **SQL**, proceda a editarla, luego cierre el editor, el resultado será el siguiente:

```
SQL> edit
Escrito file afiedt.buf

  1  INSERT INTO mensaje
  2* VALUES(sq_mensaje.nextval,'profesor','estudiantes','Que les parece Oracle?')
SQL> /

1 fila creada.
```

Ahora procederemos a consultar la tabla, primero formateamos las columnas:

```
SQL> set linesize 100
SQL> column texto format a30
SQL> column id format 999
SQL> column de format a15
SQL> column para format a15
```

Ejecutamos la sentencia **select**:

```
SQL> select * from mensaje;
```

ID DE	PARA	TEXTO
1 profesor	estudiantes	Hola a todos
2 profesor	estudiantes	Que les parece Oracle?

Para confirmar los **insert** se debe ejecutar la sentencia **commit**.

```
SQL> commit;
```

```
Confirmación terminada.
```

Listo, ahora está en condiciones de crear cualquier otro esquema.



## Capítulo 4

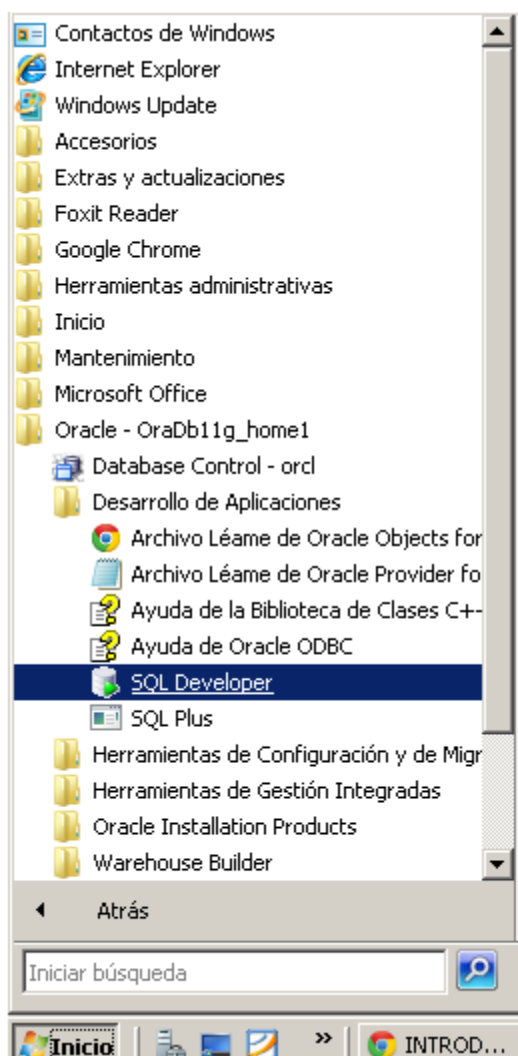
# ORACLE SQL DEVELOPER

### EJECUTAR SQL DEVELOPER

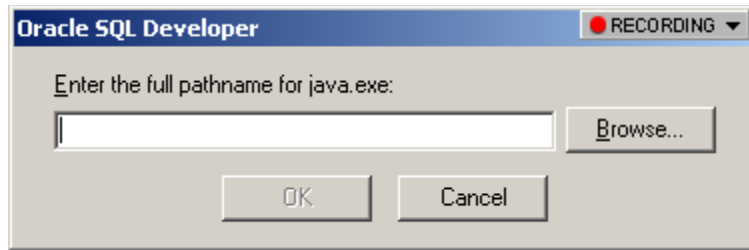
Esta herramienta se puede obtener como un producto independiente desde el portal de Oracle.

A partir de Oracle 11g ya viene integrada con el producto y la puedes encontrar desde el menú **Inicio** de Windows.

Es una herramienta gráfica para trabajar con bases de datos Oracle.



Cuando intentamos ejecutar SQL Developer, solo la primera vez, solicita la ruta del archivo `java.exe`:

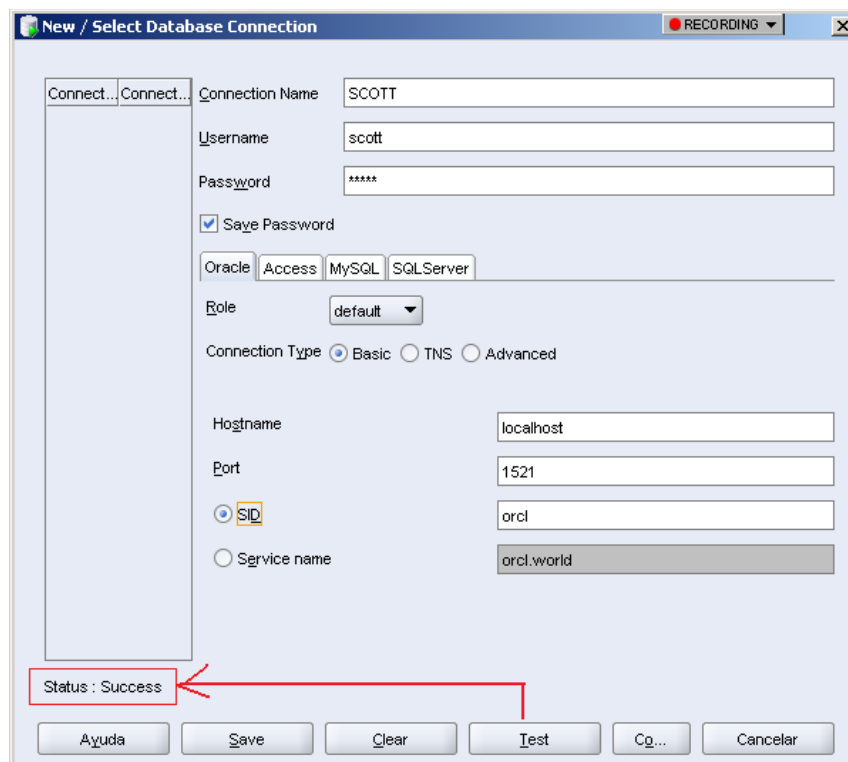


Este archivo se encuentra en la siguiente ruta:

`F:\app\Administrador\product\11.1.0\db_1\jdk\bin`

Haciendo click en el botón **Browse** se debe llegar a la carpeta **bin** y seleccionar el archivo `java.exe`.

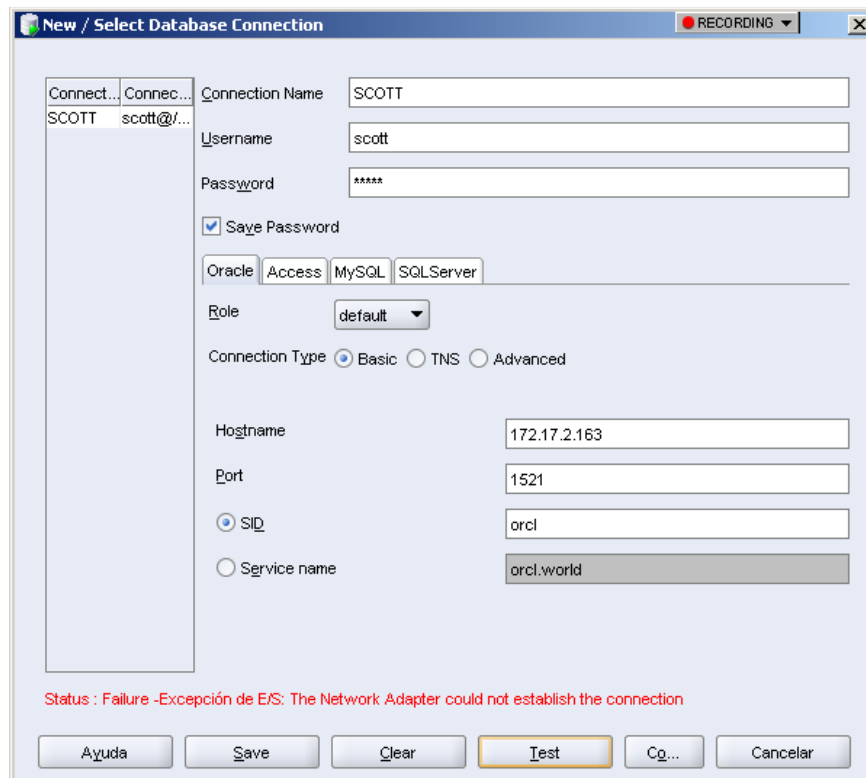
## CONFIGURAR LA CONEXIÓN CON SCOTT



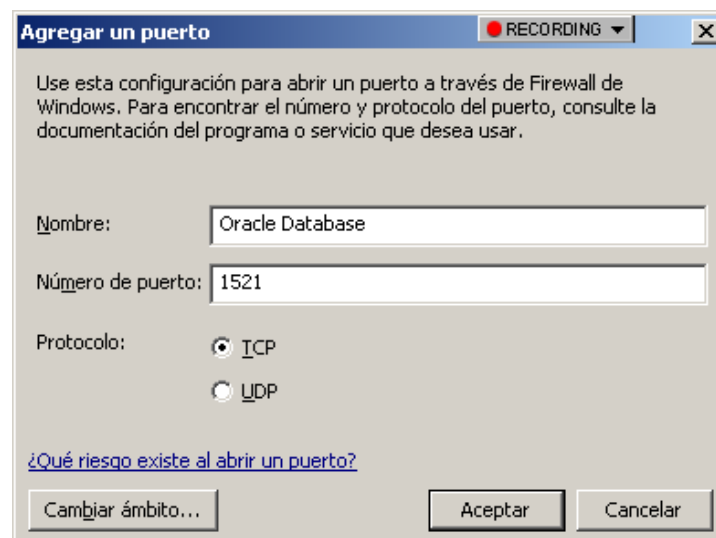
Se puede usar el **SID** o el nombre del servicio.

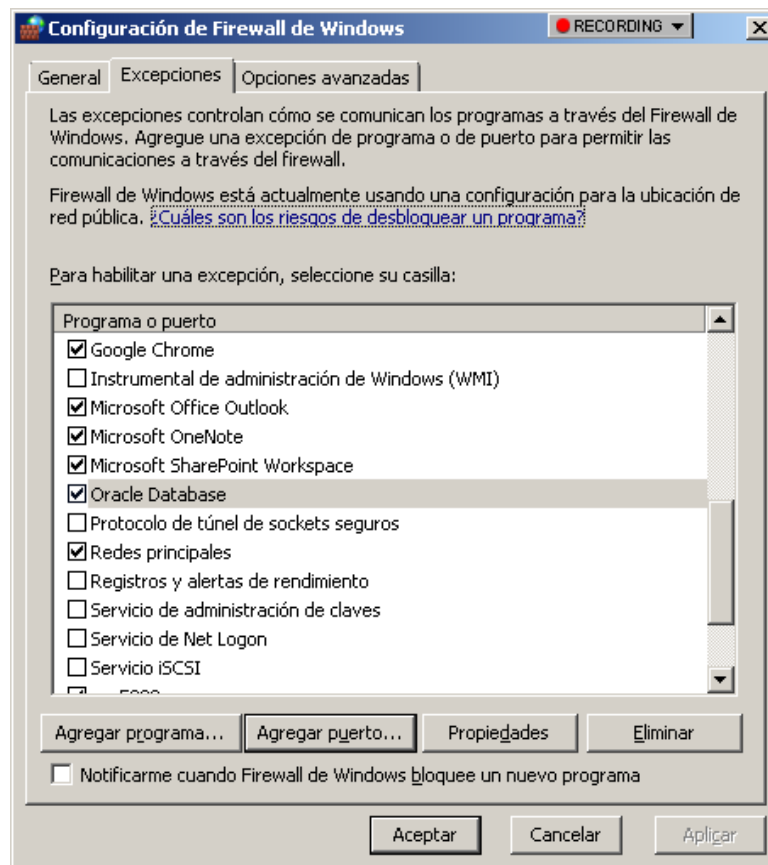
En lugar de `localhost` se puede utilizar el IP del computador, de esta manera es posible realizar conexiones a otros servidores de la red.

Si intentas la conexión con otro servidor, por ejemplo el **172.17.2.163**, es posible que obtenga el siguiente mensaje de error:

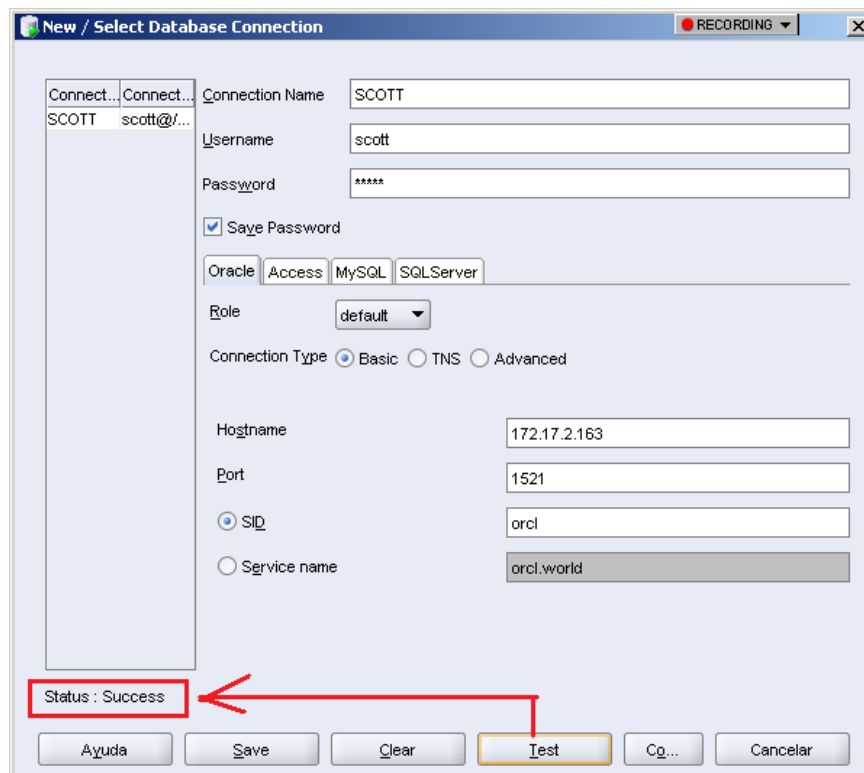


El problema es el Firewall, el puerto de Oracle Database es por defecto el **1521**, por lo tanto se debe abrir en el servidor 172.17.2.163:



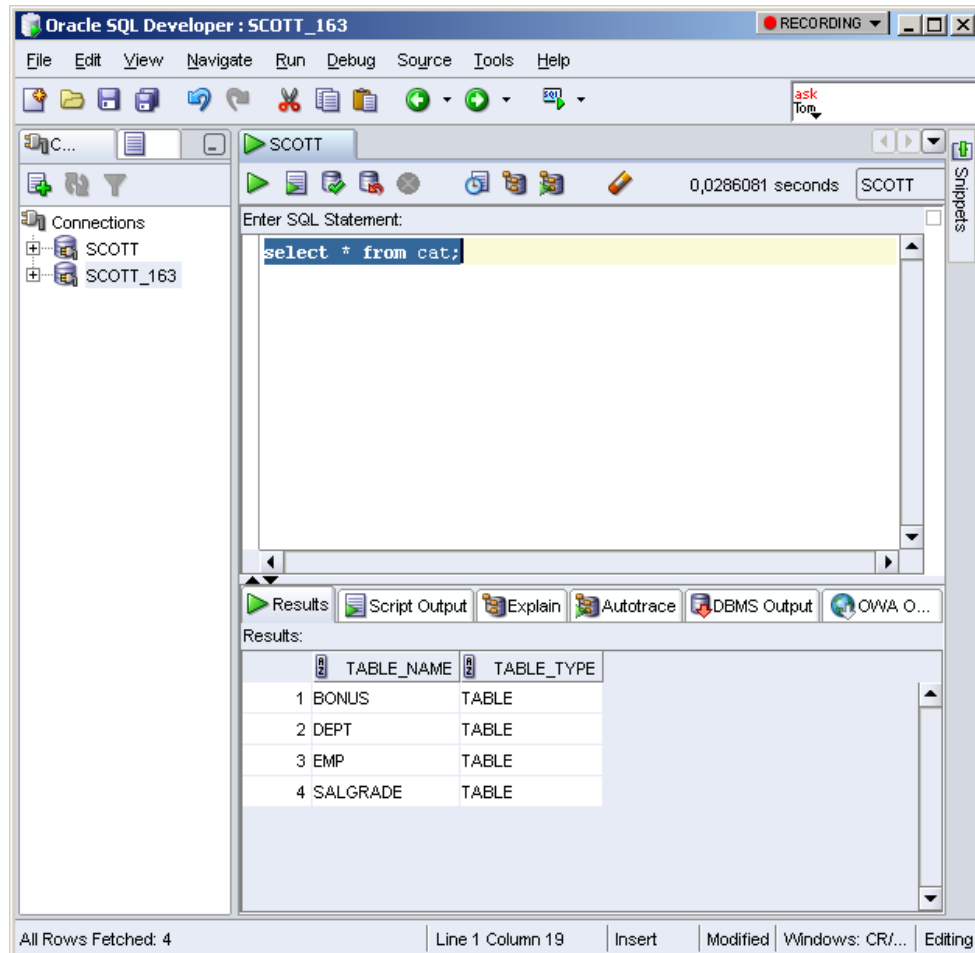


Luego realice nuevamente la prueba:



## SQL DEVELOPER EN ACCIÓN

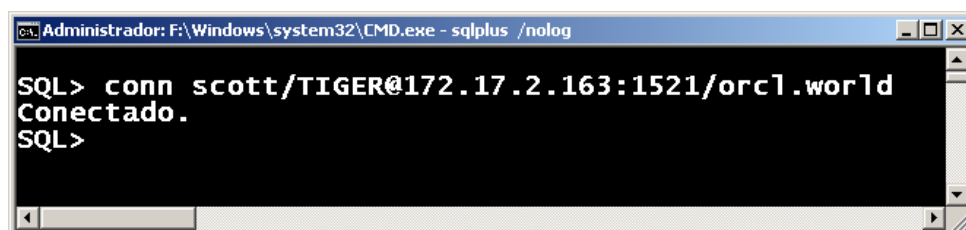
Consultando el catalogo:



## CONEXIÓN REMOTA CON SQL\*PLUS

Sintaxis:

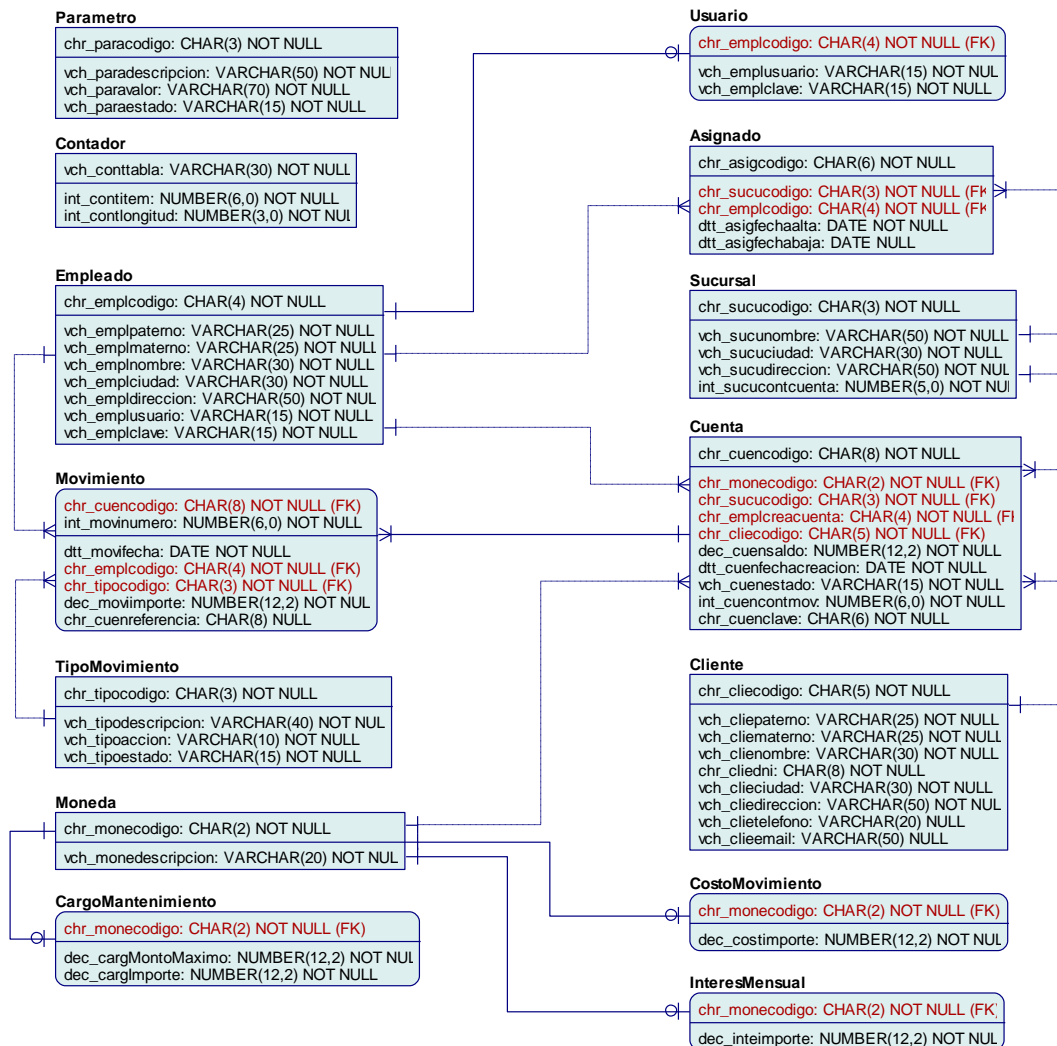
```
CONNECT <usuario>/<clave>@<equipo>:1521/servicio
```



# Capítulo 5

## CREACIÓN DEL ESQUEMA EUREKA

### DIAGRAMA E-R



## OBTENER EL SCRIPT

El script lo puedes tener del siguiente repositorio GIT:

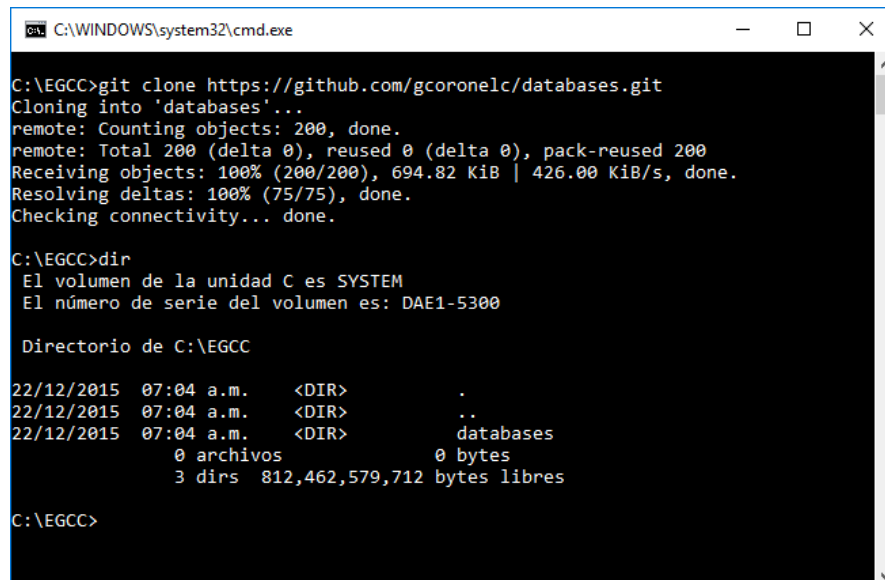
```
https://github.com/gcoronelc/databases
```

Lo puedes clonar o descargar en ZIP.

Para clonarlo el comando es el siguiente:

```
git clone https://github.com/gcoronelc/databases.git
```

Tal como se ilustra a continuación:



```
C:\WINDOWS\system32\cmd.exe

C:\EGCC>git clone https://github.com/gcoronelc/databases.git
Cloning into 'databases'...
remote: Counting objects: 200, done.
remote: Total 200 (delta 0), reused 0 (delta 0), pack-reused 200
Receiving objects: 100% (200/200), 694.82 KiB | 426.00 KiB/s, done.
Resolving deltas: 100% (75/75), done.
Checking connectivity... done.

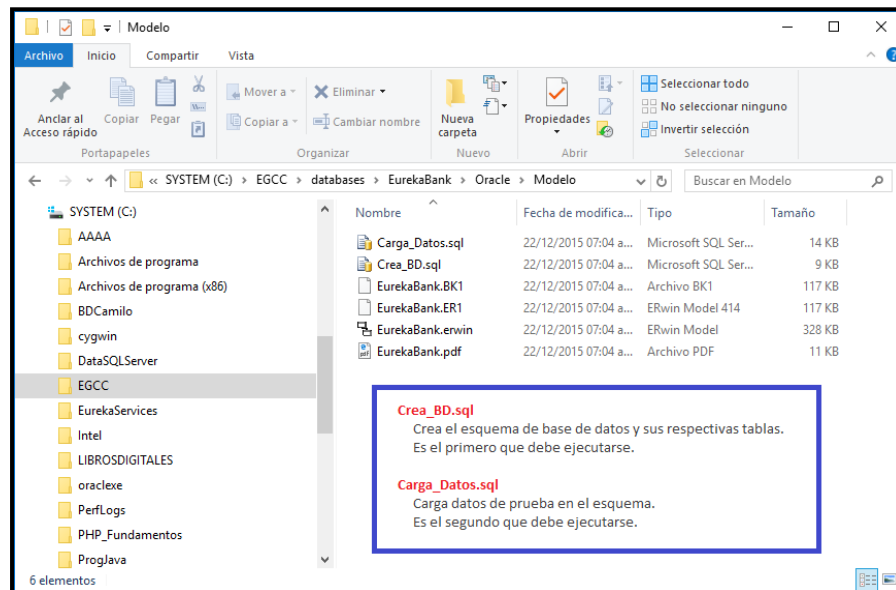
C:\EGCC>dir
El volumen de la unidad C es SYSTEM
El número de serie del volumen es: DAE1-5300

Directorio de C:\EGCC

22/12/2015  07:04 a.m.    <DIR>          .
22/12/2015  07:04 a.m.    <DIR>          ..
22/12/2015  07:04 a.m.    <DIR>          databases
                   0 archivos             0 bytes
                   3 dirs  812,462,579,712 bytes libres

C:\EGCC>
```

Los script a ejecutar se ilustran en la siguiente imagen:



## EJECUCIÓN DE LOS SCRIPTS

Para ejecutar los scripts debes usar el comando **RUN** o **@**, tal como se ilustra a continuación:

```
SQL> show user
USER es "SYS"

SQL> @C:\EGCC\databases\EurekaBank\Oracle\Modelo\Crea_BD.sql

SQL> @C:\EGCC\databases\EurekaBank\Oracle\Modelo\Carga_Datos.sql

SQL> conn eureka/admin
Conectado.
```

Puedes ver el siguiente video, donde se explica la creación del esquema EUREKA:

<https://www.youtube.com/watch?v=q0W90RXepUs>



## CONSULTANDO EL CATALOGO

A continuación tenemos la consulta del catálogo:

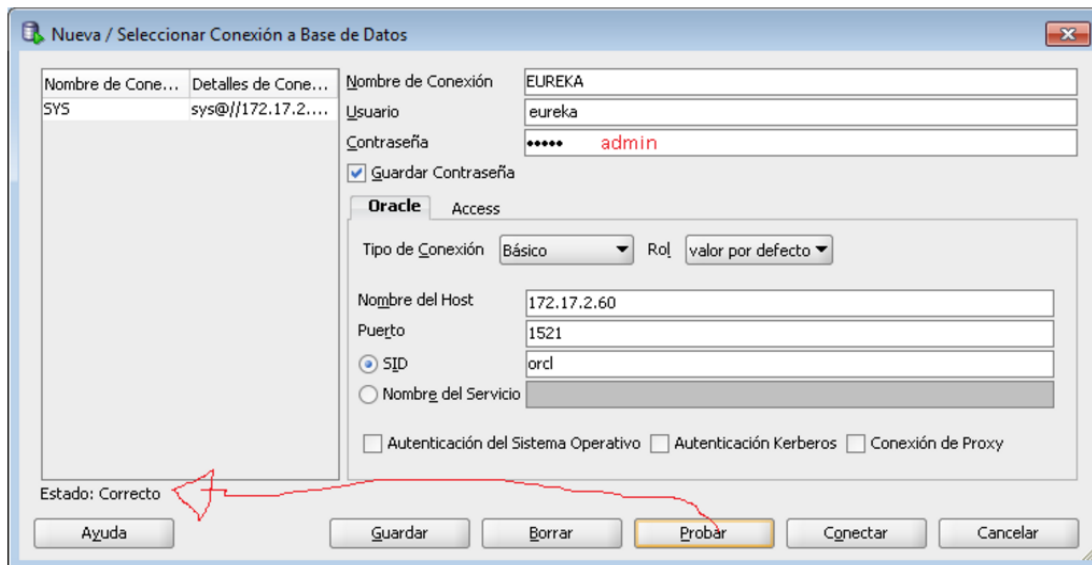
```
SQL> set pagesize 5000
SQL> set linesize 80
SQL> column table_name format a30
SQL> select * from cat;
```

TABLE_NAME	TABLE_TYPE
-----	-----
CONTADOR	TABLE
CARGOMANTENIMIENTO	TABLE
COSTOMOVIMIENTO	TABLE
INTERESMENSUAL	TABLE
PARAMETRO	TABLE
MOVIMIENTO	TABLE
CUENTA	TABLE
MONEDA	TABLE
CLIENTE	TABLE
TIPOMOVIMIENTO	TABLE
ASIGNADO	TABLE
SUCURSAL	TABLE
EMPLEADO	TABLE

13 filas seleccionadas.

## CONEXIÓN CON SQL DEVELOPER

Puede crear la conexión con SQL Developer, la siguiente imagen muestra la configuración respectiva:



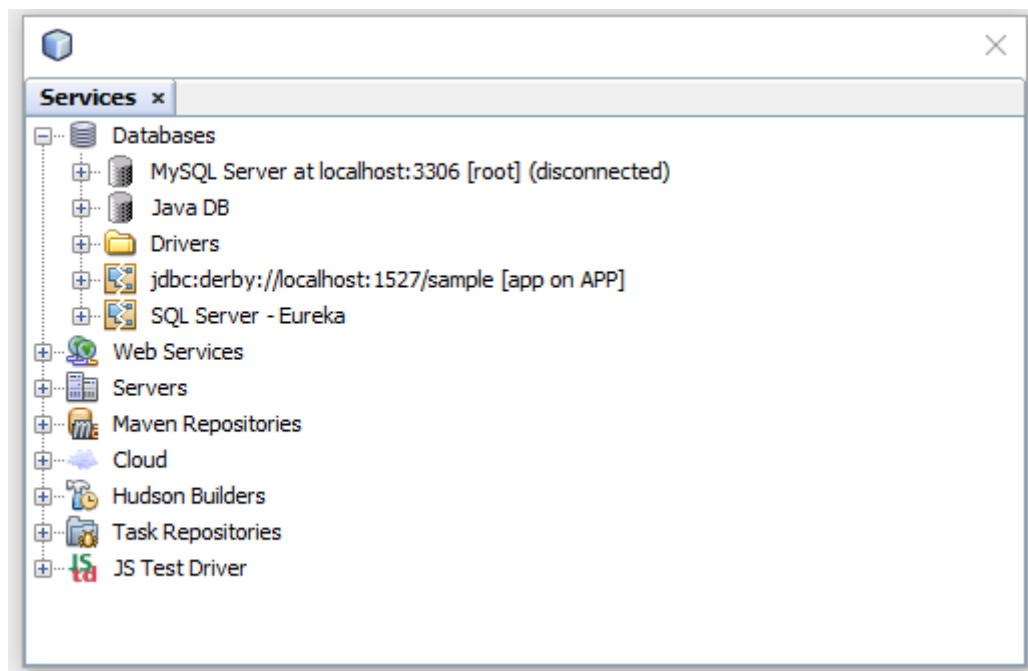
Luego de configurar la conexión, ya puede utilizarla para ejecutar sentencias SQL contra el esquema EUREKA.

## CONEXIÓN DESDE NETBEANS

### Database

Database es una herramienta que provee NetBeans para trabajar con bases de datos desde el mismo IDE, sin tener la necesidad de utilizar otras herramientas externas al IDE.

Esta herramienta se encuentra en la ventana **Service**, tal como se muestra en la siguiente figura:



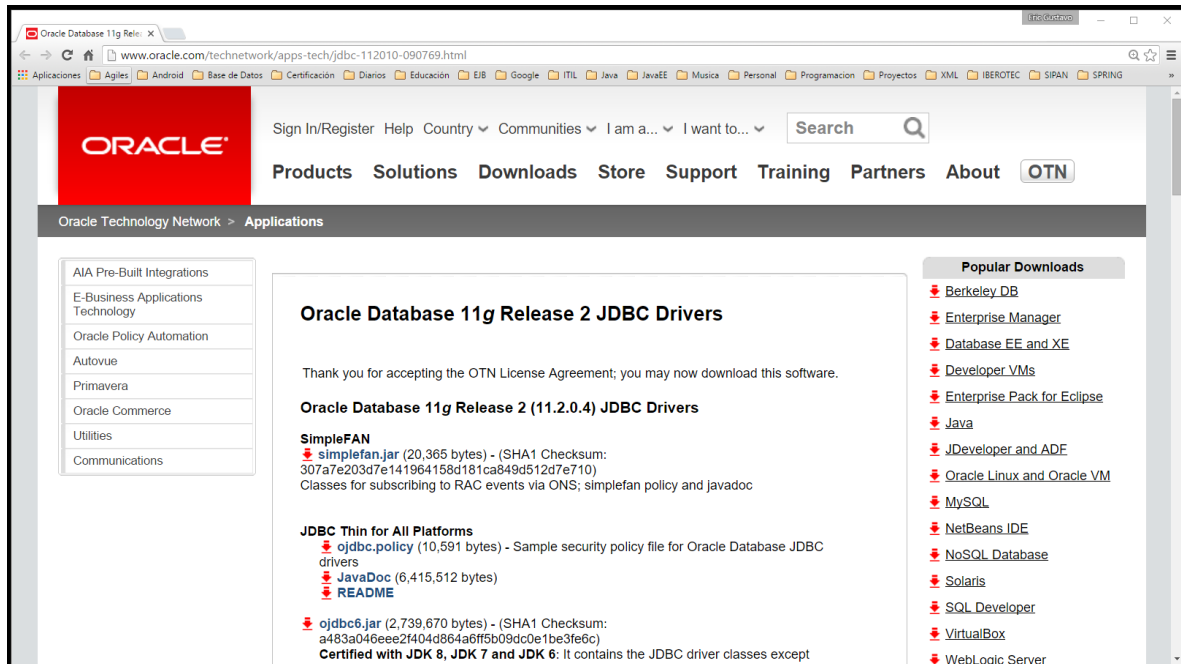
Esta herramienta ya trae algunos drivers JDBC, específicamente para Oracle no trae ningún driver. Así que el proceso para trabajar con Oracle es:

1. Registrar el driver.
2. Registrar la conexión. Se pueden registrar varias conexiones.
3. Utilizar la conexión para ejecutar sentencias SQL.

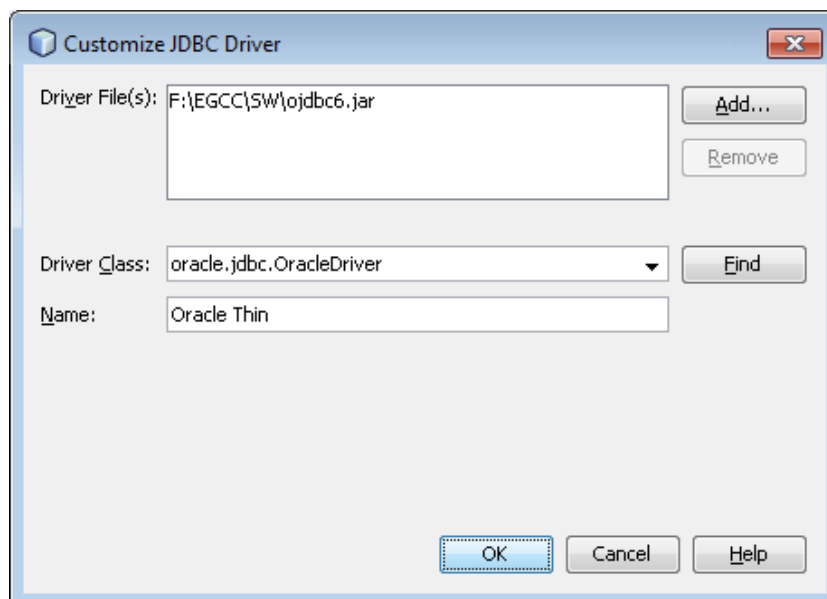
## Registrar Driver

Puede descargar el driver de la siguiente dirección web:

<http://www.oracle.com/technetwork/apps-tech/jdbc-112010-090769.html>

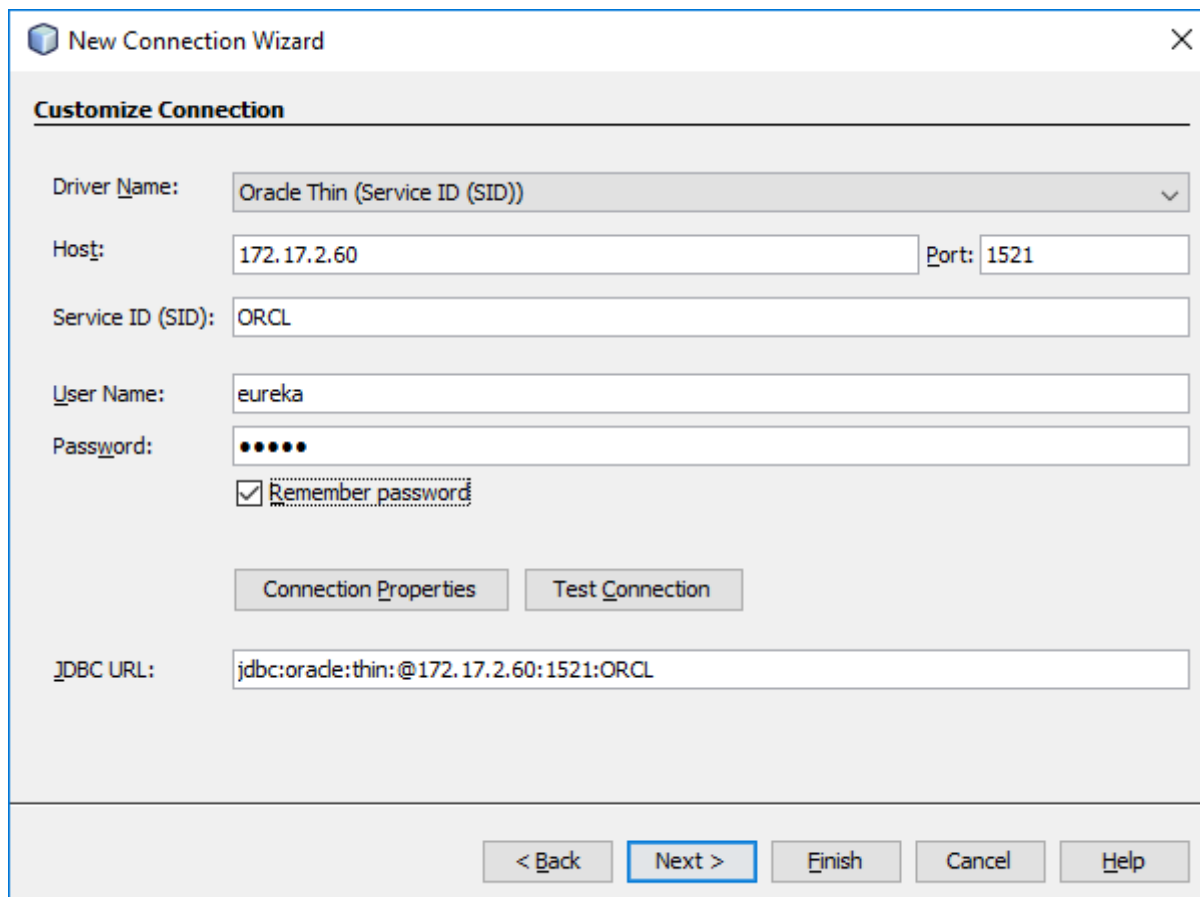


Después proceda a registrarlo en la herramienta **Database**, tal como se ilustra en la siguiente imagen:



## Registrar la Conexión

Después de haber registrado correctamente el driver, debe proceder a registrar la conexión, tal como se muestra en la siguiente imagen:



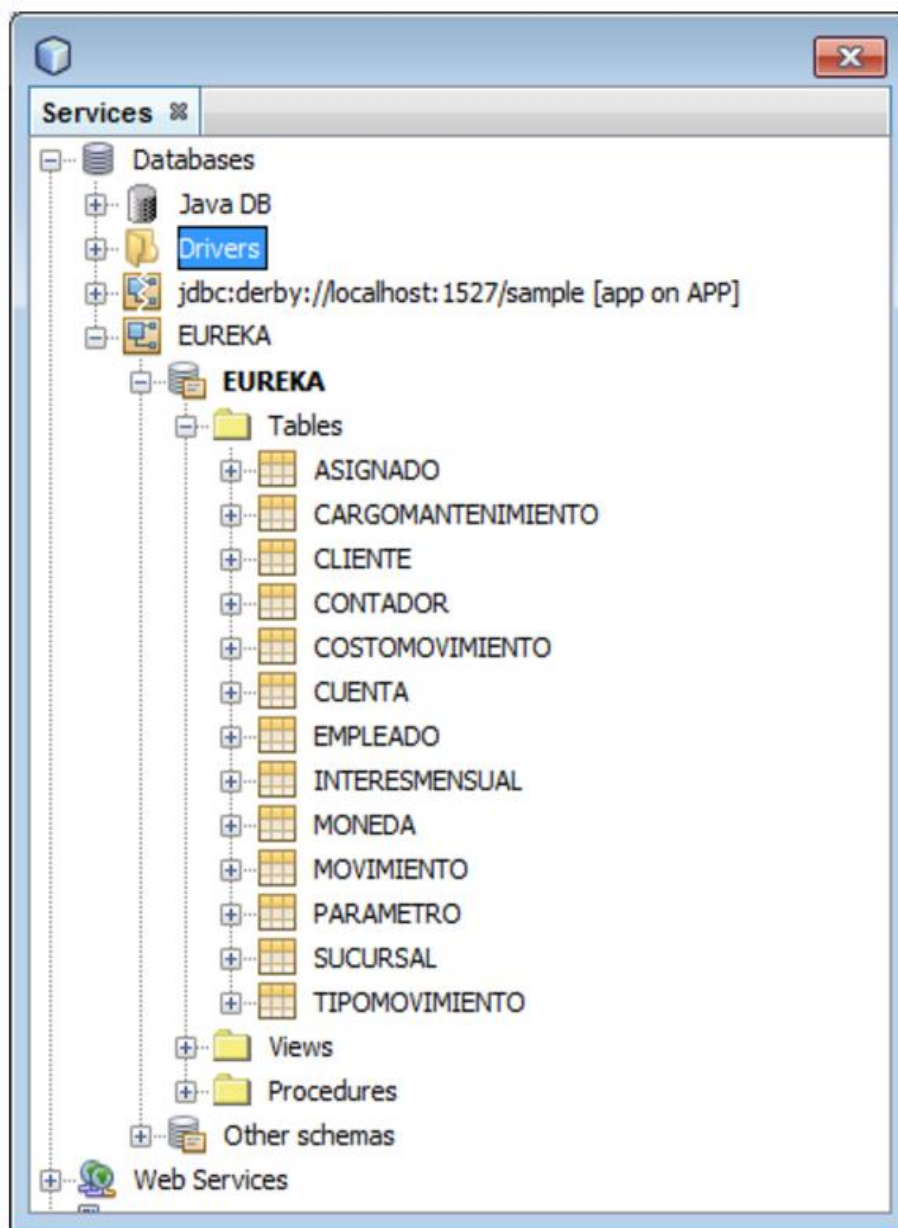
The screenshot shows the 'New Connection Wizard' dialog box, specifically the 'Customize Connection' step. The fields are as follows:

- Driver Name:** Oracle Thin (Service ID (SID))
- Host:** 172.17.2.60
- Port:** 1521
- Service ID (SID):** ORCL
- User Name:** eureka
- Password:** (masked with dots)
- ☒ **Remember password**
- Buttons:** Connection Properties, Test Connection
- JDBC URL:** jdbc:oracle:thin:@172.17.2.60:1521:ORCL
- Navigation Buttons:** < Back, Next > (highlighted), Finish, Cancel, Help

Se recomienda marcar el check box **Remember password** para que no solicite la contraseña cada vez que realice la conexión con la base de datos.

Puede registrar varias conexiones con diferentes esquemas de la base de datos local o bases de datos remotas.

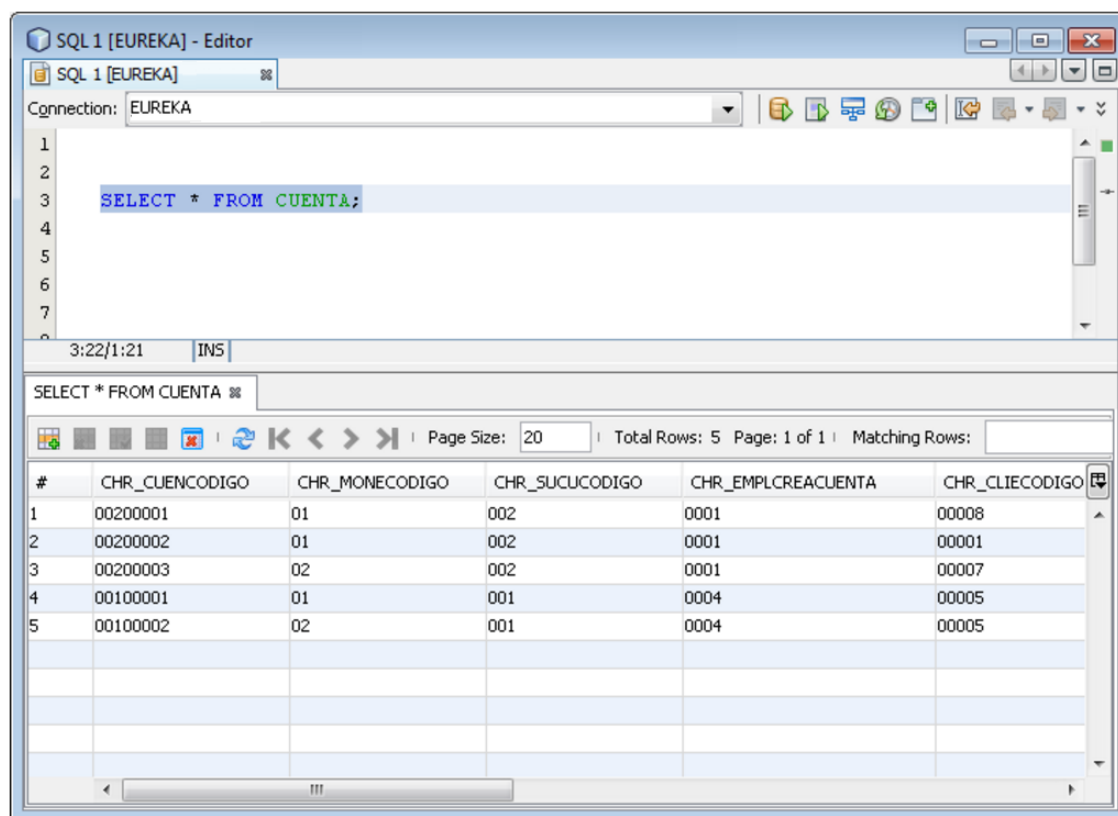
Después de registrar la conexión puede explorar los objetos del esquema:



Incluso, puede revisar objetos de otros objetos en la carpeta **Other schemas**.

## Ejecución de Sentencias SQL

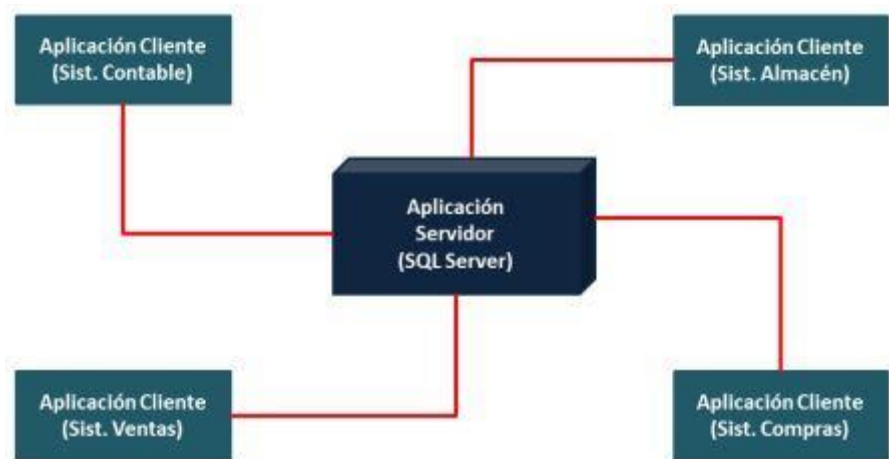
Esta herramienta Database también permite ejecutar sentencias SQL, por ejemplo puede ejecutar consultas como se muestra en la siguiente imagen:



## Capítulo 6

# APLICACIONES CLIENTE - SERVIDOR

### ACLARANDO CONCEPTOS



### Cliente-Servidor

Es un modelo basado en la cooperación e interacción de dos partes conocidas como servidor o back-end y cliente o front-end.

### Aplicación Servidor

Es la aplicación que provee servicios, por ejemplo, los servidores de base de datos ofrecen servicios de persistencia de datos, podríamos mencionar a SQL Server, Oracle, MySQL, etc,

### Aplicación Cliente

Es la aplicación que hace uso o consume los servicios de la Aplicación Servidor; por ejemplo, las aplicaciones comerciales como los sistemas de ventas y compras necesitan que sus datos persistan en el tiempo, para lo cual se recurren a los servidores de base de datos que ofrecen estos servicios.



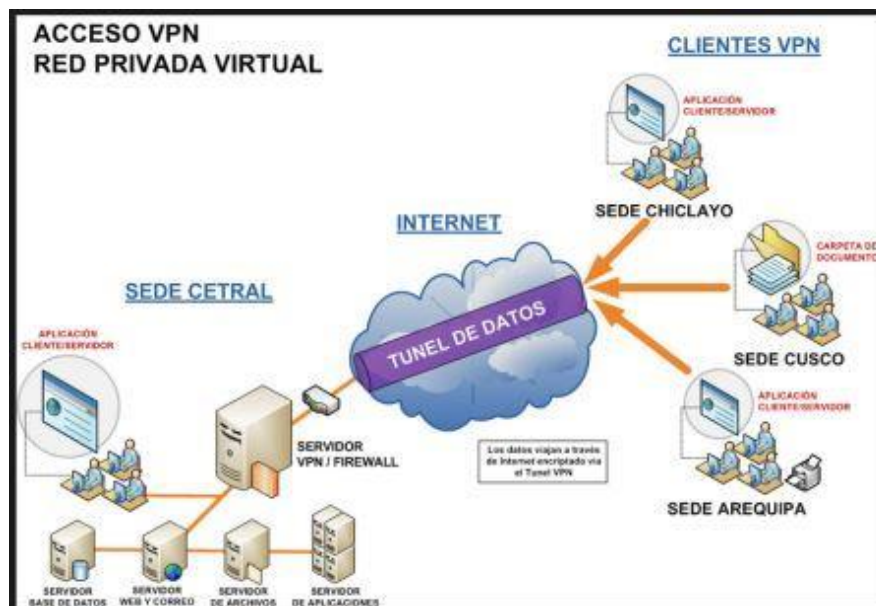
## ARQUITECTURA CLIENTE-SERVIDOR

### Modelo Clásico



El cliente se encuentra del lado del usuario y se define como un proceso consumidor de servicios, mientras que el servidor provee los servicios requeridos y se encuentra de manera remota al usuario y es transparente al cliente.

### Modelo en Internet



Los clientes deben estar en la red de la empresa, estas pueden ser redes locales, pero también es posible tener clientes remotos a través por ejemplo de una red VPN.

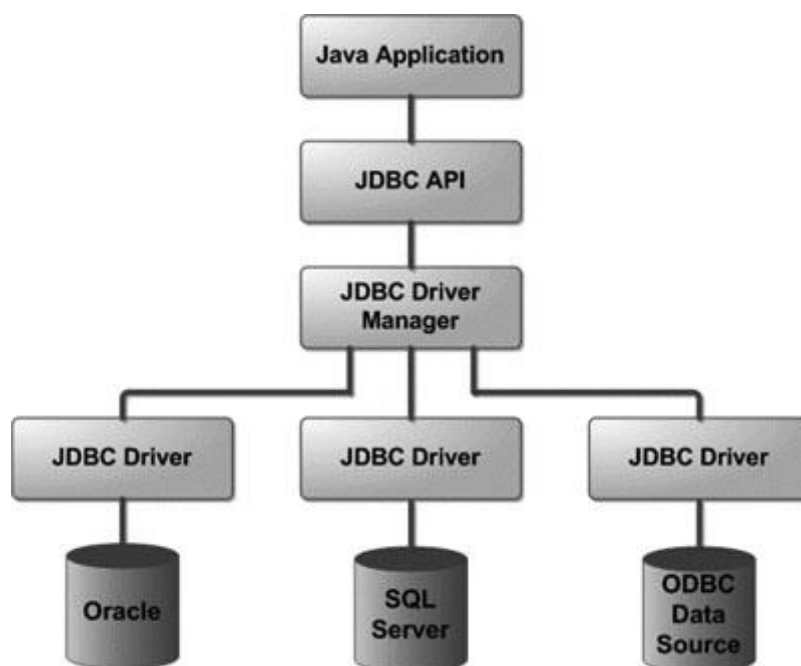
## Capítulo 7

### API JDBC

#### JDBC

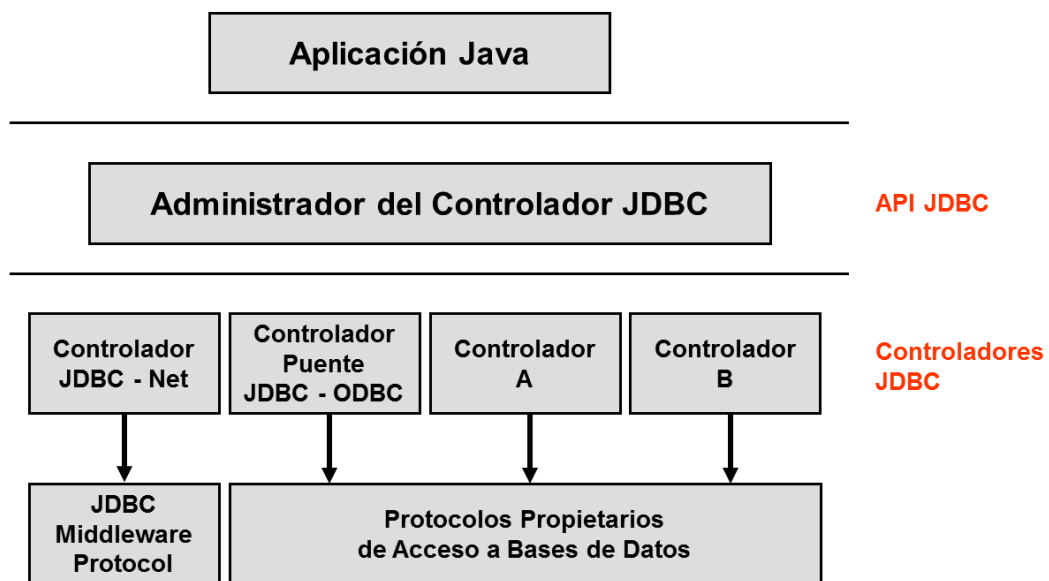
Recomiendo que visiten el siguiente enlace:

<http://gcoronelc.blogspot.pe/2013/06/resumen-seminario-java-cliente-servidor.html>



De JDBC podemos afirmar:

- Es la sigla de Java Database Connectivity.
- Es un API conformada por un conjunto de interfaces y clases Java que nos permiten acceder de una forma genérica a las bases de datos independiente del proveedor.
- Cada proveedor dispondrá de una implementación para comunicarse con su motor de base de datos.
- Se encuentra en el paquete `java.sql`.



Básicamente una aplicación que usa JDBC realiza los siguientes pasos:

1. Establece una conexión con la base de datos.
2. Crea y envía una sentencia SQL a la base de datos.
3. Procesa el resultado.
4. Cierra la conexión.

## TIPOS DE DRIVER

Tenemos cuatro tipos de driver:

- Tipo 1: JDBC-ODBC bridge driver
- Tipo 2: Native API/Partly Java Driver
- Tipo 3: Pure Java Driver
- Tipo 4: Native Protocol Java Driver

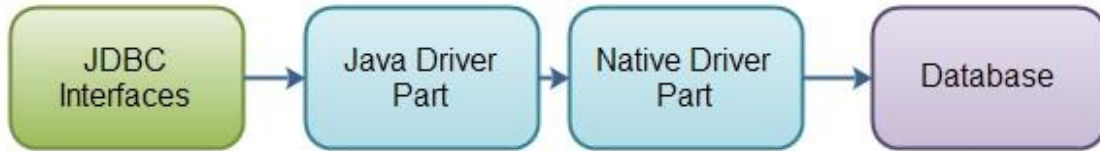
### Tipo 1: JDBC-ODBC bridge driver

Permite utilizar drivers ODBC, su desventaja es que se debe instalar el driver ODBC en cada cliente.



## Tipo 2: Native API/Partly Java Driver

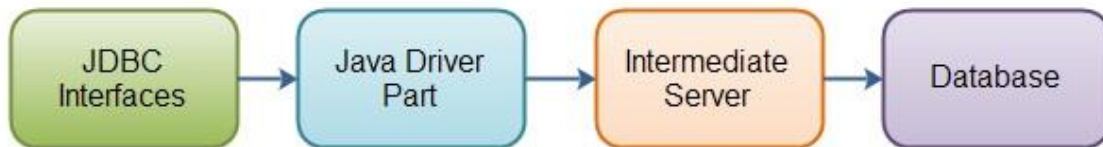
Requiere que se instalen las librerías nativas en cada cliente.



## Tipo 3: Pure Java Driver

Este tipo de driver traduce llamadas JDBC en un protocolo de red independiente del DBMS y luego, a un protocolo de DBMS.

Un Servidor JDBC, realiza la tarea de middleware, siendo capaz de conectar un cliente Java a diferentes bases de datos.

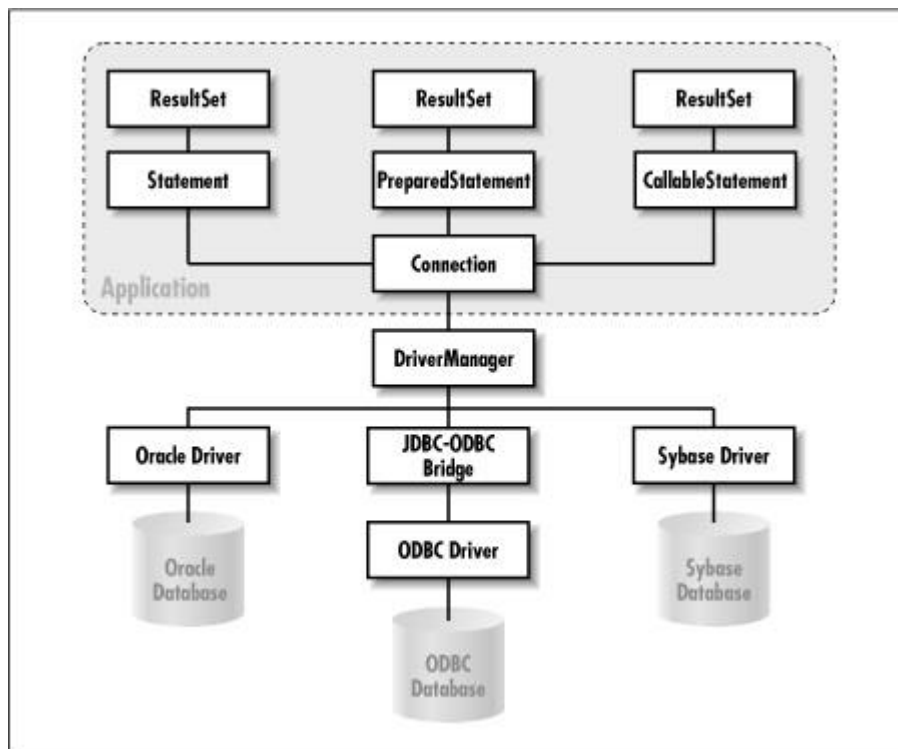


## Tipo 4: Native Protocol Java Driver

El driver tipo 4 convierte llamadas JDBC directamente en el protocolo de red usado por el DBMS. Esto permite llamadas directas desde la máquina del cliente al servidor del DBMS, es una solución muy práctica y la más utilizada.



## COMPONENTES PRINCIPALES



### Objeto: Connection

Establece la conexión con la base de datos y maneja las transacciones.

### Objeto: Statement

Se utiliza para ejecutar sentencias sin parámetros.

### Objeto: PreparedStatement

Se utiliza para ejecutar sentencias con parámetros.

### Objeto: CallableStatement

Se utiliza para ejecutar procedimientos almacenados.

### Objeto ResultSet

Se utiliza para procesar resultados.

### Objeto: SQLException

Se utiliza para el manejo de error de base de datos.

## CONSULTAR DATOS

### Acceso a la base de datos

El acceso a la base de datos debe estar programado en una clase de manera centralizada, para este caso se propone la clase **AccesoDB.java**, el script se muestra a continuación:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
public final class AccesoDB {

    // Para que no se pueda crear instancias de esta clase
    private AccesoDB() {
    }

    // Retorna un objeto de tipo Connection
    public static Connection getConnection() throws SQLException {
        Connection cn = null;
        try {
            // Paso 1: Cargar el driver a memoria
            Class.forName("oracle.jdbc.OracleDriver").newInstance();
            // Paso 2: Obtener el objeto Connection
            String url = "jdbc:oracle:thin:@localhost:1521:ORCL";
            cn = DriverManager.getConnection(url, "eureka", "admin");
        } catch (SQLException e) {
            throw e;
        } catch (ClassNotFoundException e) {
            throw new SQLException("No se encontró el driver de la base de datos.");
        } catch (Exception e) {
            throw new SQLException("No se puede establecer la conexión con la BD.");
        }
        return cn;
    }
}
```

## Mapeo de Tablas

Cuando se trata de trabajar con datos de una tabla, se recomienda mapear la tabla en una clase, a continuación tenemos la clase **ClienteBean** para trabajar con la tabla de clientes:

```
/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
public class ClienteBean {

    private String codigo;
    private String paterno;
    private String materno;
    private String nombre;
    private String dni;
    private String ciudad;
    private String direccion;
    private String telefono;
    private String email;

    public ClienteBean() {
    }

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public String getPaterno() {
        return paterno;
    }

    public void setPaterno(String paterno) {
        this.paterno = paterno;
    }

    public String getMaterno() {
        return materno;
    }
}
```



```
public void setMaterno(String materno) {
    this.materno = materno;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getDni() {
    return dni;
}

public void setDni(String dni) {
    this.dni = dni;
}

public String getCiudad() {
    return ciudad;
}

public void setCiudad(String ciudad) {
    this.ciudad = ciudad;
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

public String getEmail() {
    return email;
}
```



```
public void setEmail(String email) {  
    this.email = email;  
}  
}
```

## Consultar un Registro

El siguiente script, permite consultar los datos de un registro, en este caso los datos de un cliente, lo que retorna el método es un objeto de tipo **ClienteBean** o **NULL**.

```
/**  
 *  
 * @author Eric Gustavo Coronel Castillo  
 * @blog www.desarrollasoftware.com  
 * @email gcoronelc@gmail.com  
 */  
public ClienteBean consultarPorCodigo(String codigo) {  
    ClienteBean clienteBean = null;  
    Connection cn = null;  
    try {  
        cn = AccesoDB.getConnection();  
        String sql = "select chr_cliecodigo, vch_cliepaterno, vch_cliematerno, "  
            + "vch_clienombre, chr_cliedni, vch_clieciudad, "  
            + "vch_cliedireccion, vch_clietelefono, vch_clieemail "  
            + "from cliente "  
            + "where chr_cliecodigo = ?";  
        PreparedStatement pstm = cn.prepareStatement(sql);  
        pstm.setString(1, codigo);  
        ResultSet rs = pstm.executeQuery();  
        if (rs.next()) {  
            clienteBean = new ClienteBean();  
            clienteBean.setCodigo(rs.getString("chr_cliecodigo"));  
            clienteBean.setPaterno(rs.getString("vch_cliepaterno"));  
            clienteBean.setMaterno(rs.getString("vch_cliematerno"));  
            clienteBean.setNombre(rs.getString("vch_clienombre"));  
            clienteBean.setDni(rs.getString("chr_cliedni"));  
            clienteBean.setCiudad(rs.getString("vch_clieciudad"));  
            clienteBean.setDireccion(rs.getString("vch_cliedireccion"));  
            clienteBean.setTelefono(rs.getString("vch_clietelefono"));  
            clienteBean.setEmail(rs.getString("vch_clieemail"));  
        }  
        rs.close();  
        pstm.close();  
    } catch (SQLException e) {  
        throw new RuntimeException(e.getMessage());  
    } catch (Exception e) {  

```

```
        throw new RuntimeException("Error de acceso a la tabla Cliente.");
    } finally {
        try {
            cn.close();
        } catch (Exception e) {
        }
    }
    return clienteBean;
}
```

## Consultar un Conjunto de Registros

En este caso, el método debe retornar una lista de objetos, para el ejemplo se está retornando una lista de objetos ClienteBean. Si no encuentra ninguna coincidencia retorna una lista vacía.

```
/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
public List<ClienteBean> consultarPorNombre(String nombre) {
    List<ClienteBean> lista = new ArrayList<ClienteBean>();
    Connection cn = null;
    try {
        cn = AccesoDB.getConnection();
        String sql = "select chr_cliecodigo, vch_cliepaterno, vch_cliematerno, "
            + "vch_clienombre, chr_cliedni, vch_clieciudad, "
            + "vch_cliedireccion, vch_clietelefono, vch_clieemail "
            + "from cliente "
            + "where lower(vch_cliepaterno) like ? "
            + "or lower(vch_cliematerno) like ? "
            + "or lower(vch_clienombre) like ?";
        PreparedStatement pstm = cn.prepareStatement(sql);
        nombre = nombre.toLowerCase() + "%";
        pstm.setString(1, nombre);
        pstm.setString(2, nombre);
        pstm.setString(3, nombre);
        ResultSet rs = pstm.executeQuery();
        while (rs.next()) {
            clienteBean = new ClienteBean();
            clienteBean.setCodigo(rs.getString("chr_cliecodigo"));
            clienteBean.setPaterno(rs.getString("vch_cliepaterno"));
            clienteBean.setMaterno(rs.getString("vch_cliematerno"));
            clienteBean.setNombre(rs.getString("vch_clienombre"));
        }
    } catch (Exception e) {
    }
    return lista;
}
```

```

        clienteBean.setDni(rs.getString("chr_cliedni"));
        clienteBean.setCiudad(rs.getString("vch_clieciudad"));
        clienteBean.setDireccion(rs.getString("vch_cliedireccion"));
        clienteBean.setTelefono(rs.getString("vch_clietelefono"));
        clienteBean.setEmail(rs.getString("vch_clieemail"));
        lista.add(bean);
    }
    rs.close();
    pstmt.close();
} catch (SQLException e) {
    throw new RuntimeException(e.getMessage());
} catch (Exception e) {
    throw new RuntimeException("Error de acceso a la tabla Cliente.");
} finally {
    try {
        cn.close();
    } catch (Exception e) {
    }
}
}
return lista;
}

```

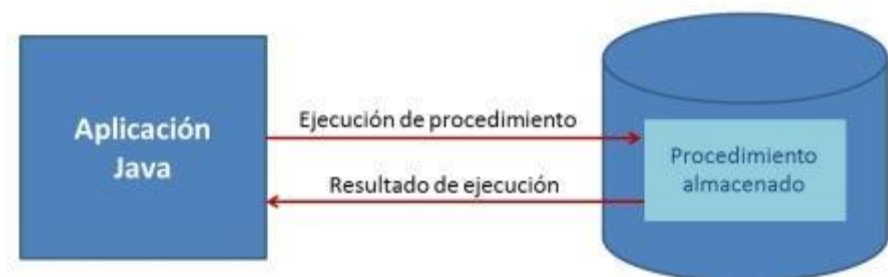
## ESTRATEGIAS PARA PROGRAMAR TRANSACCIONES

### Controladas Desde el Cliente



En este caso, desde la aplicación Java programamos el inicio y fin de la transacción.

### Controladas en la Base de Datos



En este caso programamos la transacción en un procedimiento almacenado.

## PROGRAMANDO TRANSACCIONES

### Transacción Controlada Desde el Cliente

En el siguiente ejemplo, tienes programado el proceso para registrar un nuevo cliente, como parte del proceso se genera el código haciendo uso de la tabla PARAMETRO.

Toda la transacción está controlada por el código Java.

```
/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
public void insertar(ClienteBean clienteBean) {
    Connection cn = null;
    try {
        // Variables
        String sql, codigo;
        ResultSet rs;
        PreparedStatement pstmt;
        Statement stm;
        int cont, longitud;
        // Inicio de tx
        cn = AccesoDB.getConnection();
        cn.setAutoCommit(false); // Inicia la Tx
        // Obtener contador de sucursal
        sql = "select int_contitem, int_contlongitud from contador "
            + "where vch_conttabla = 'Cliente' for update";
        stm = cn.createStatement();
        rs = stm.executeQuery(sql);
        rs.next();
        cont = rs.getInt("int_contitem") + 1;
        longitud = rs.getInt("int_contlongitud");
        rs.close();
        // Crear codigo
        String formato = "";
        for (int i = 1; i <= longitud; i++) {
            formato += "0";
        }
        DecimalFormat df = new DecimalFormat(formato);
        codigo = df.format(cont);
        // Insertar cliente
        sql = "insert into cliente(chr_cliecodigo,vch_cliepaterno,vch_cliematerno,"
            + "vch_clienombre,chr_cliedni,vch_clieciudad,vch_cliedireccion,"
```

```
        + "vch_clitelefono,vch_clieemail) values(?,?,?,?,?,?,?,?,?)";
    pstmt = cn.prepareStatement(sql);
    pstmt.setString(1, codigo);
    pstmt.setString(2, clienteBean.getPaterno());
    pstmt.setString(3, clienteBean.getMaterno());
    pstmt.setString(4, clienteBean.getNombre());
    pstmt.setString(5, clienteBean.getDni());
    pstmt.setString(6, clienteBean.getCiudad());
    pstmt.setString(7, clienteBean.getDireccion());
    pstmt.setString(8, clienteBean.getTelefono());
    pstmt.setString(9, clienteBean.getEmail());
    pstmt.executeUpdate();
    pstmt.close();
    // Actualizar contador
    sql = "update contador set int_contitem = int_contitem + 1 "
        + "where vch_conttabla='Cliente'";
    stm.executeUpdate(sql);
    stm.close();
    cn.commit(); // Confirma Tx
    clienteBean.setCodigo(codigo);
} catch (SQLException e) {
    try {
        cn.rollback();
    } catch (Exception e1) {
    }
    throw new RuntimeException(e.getMessage());
} catch (Exception e) {
    try {
        cn.rollback();
    } catch (Exception e1) {
    }
    throw new RuntimeException("Error en proceso de creación de cuenta.");
} finally {
    try {
        cn.close();
    } catch (Exception e) {
    }
}
}
```

## Transacción de Base de Datos

En este caso se debe desarrollar un procedimiento almacenado en la base de datos, luego desde Java solo se debe invocar con los parámetros respectivos.

A continuación se tiene un procedimiento para registrar un retiro:

```
/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
create or replace procedure usp_egcc_retiro
( p_cuenta varchar2, p_importe number,
  p_empleado varchar2, p_clave varchar2 )
as
  v_msg varchar2(1000);
  v_saldo number(12,2);
  v_moneda char(2);
  v_cont number(5,0);
  v_estado varchar2(15);
  v_costoMov number(12,2);
  v_clave varchar2(10);
  v_excep1 Exception;
begin
  select
    dec_cuensaldo, chr_monecodigo, int_cuencontmov,
    vch_cuenestado, chr_cuenclave
  into v_saldo, v_moneda, v_cont, v_estado, v_clave
  from cuenta
  where chr_cuencodigo = p_cuenta;
  if v_estado != 'ACTIVO' then
    raise_application_error(-20001,'Cuenta no está activa.');
```

```
update cuenta
  set dec_cuensaldo = v_saldo,
  int_cuencontmov = int_cuencontmov + 2
  where chr_cuencodigo = p_cuenta;
-- Movimiento de retiro
v_cont := v_cont + 1;
insert into movimiento(chr_cuencodigo,int_movinúmero,dtm_movifecha,
  chr_emplcodigo,chr_tipocodigo,dec_moviimporte,chr_cuenreferencia)
  values(p_cuenta,v_cont,sysdate,p_empleado,'004',p_importe,null);
-- Movimiento Costo
v_cont := v_cont + 1;
insert into movimiento(chr_cuencodigo,int_movinúmero,dtm_movifecha,
  chr_emplcodigo,chr_tipocodigo,dec_moviimporte,chr_cuenreferencia)
  values(p_cuenta,v_cont,sysdate,p_empleado,'010',v_costoMov,null);
-- Confirmar la Tx
commit;
exception
  when v_excep1 then
    rollback; -- cancelar transacción
    raise_application_error(-20001,'Clave incorrecta.');
```

```
when others then
  v_msg := sqlerrm; -- capturar mensaje de error
  rollback; -- cancelar transacción
  raise_application_error(-20001,v_msg);
end;
/
```

A continuación tienes la programación del procedimiento almacenado, tener en cuenta que la transacción ya no la debes controlar desde Java.

```
/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */

public void registrarRetiroConSP(String cuenta, String clave,
  double importe, String empleado) {
  Connection cn = null;
  try {
    // Obtener el objeto connection
    cn = AccesoDB.getConnection();
    // Cancelar el control de transacciones
    cn.setAutoCommit(true);
    // Proceso
```



```
String sql = "{call egcc_retiro(?,?,?,?)}";
CallableStatement cstm = cn.prepareCall(sql);
cstm.setString(1, cuenta);
cstm.setDouble(2, importe);
cstm.setString(3, empleado);
cstm.setString(4, clave);
cstm.executeUpdate();
cstm.close();
System.err.println("PROCESO: Con procedimiento.");
} catch (SQLException e) {
    // Propagar excepcion
    throw new RuntimeException(e.getMessage());
} catch (Exception e) {
    // Propagar excepcion
    throw new RuntimeException("ERROR: no se tiene acceso a la BD.");
} finally {
    try {
        if (cn != null) {
            cn.close();
        }
    } catch (Exception e) {
    }
}
}
```

## Capítulo 8

# Apache POI

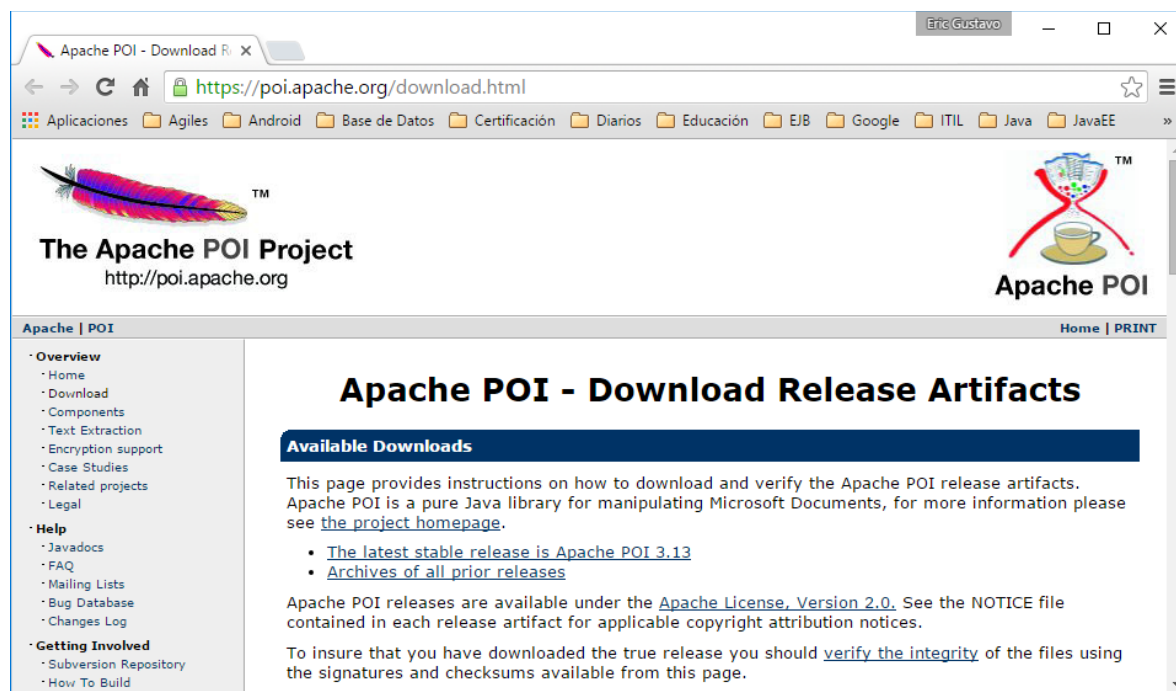
### INTRODUCCIÓN

Apache POI es una librería Java que se utiliza para poder interactuar con hojas de cálculos de Microsoft, en sus distintos formatos.

Es importante tener en cuenta que cada archivo de Excel representa un LIBRO, dentro de cada libro tenemos HOJAS, dentro de cada HOJA tenemos FILAS, y, finalmente, en cada FILA tenemos CELDAS. Entender esto nos ayudará a ver cómo se organiza la información en el archivo.

Es necesario que descargues la librería y la agregues a tu proyecto. La ruta de descarga es la siguiente:

```
https://poi.apache.org/download.html
```



## LEER UN ARCHIVO EXCEL

Vamos a ver el código que se necesita para poder leer los datos contenidos en un archivo Excel.

Vamos a ver las clases que utilizamos en el ejemplo para leer los datos de un archivo Excel.

- **POIFSFileSystem:** Esta clase se utiliza para gestionar el ciclo de vida completo del sistema de archivos.
- **HSSFWorkbook:** Este es el primer objeto construido para escribir o leer un archivo de Excel.
- **HSSFSheet:** Esta clase se utiliza para crear hojas de cálculo.
- **HSSFRow:** Esta clase representa la fila de una hoja de cálculo.
- **HSSFCell:** Esta clase representa la celda en una fila de la hoja de cálculo.

El código que utilizamos es el siguiente:

```
import java.io.FileInputStream;
import java.util.Iterator;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;

/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
public class LeerExcel {

    /**
     * Este método se utiliza para leer la data un archivo Excel.
     *
     * @param fileName - Ruta y nombre del archivo Excel.
     */
    private void leerExcel(String fileName) {

        try {

            /**
             * Crea una instancia de la clase FileInputStream.
             */
        }
    }
}
```

```
    */
    FileInputStream fileInputStream = new FileInputStream(fileName);

    /**
     * Crea una instancia de la clase POIFSFileSystem.
     */
    POIFSFileSystem fsFileSystem = new POIFSFileSystem(fileInputStream);

    /*
     * Crea una instancia de la clase HSSFWorkbook.
     */
    HSSFWorkbook workbook = new HSSFWorkbook(fsFileSystem);

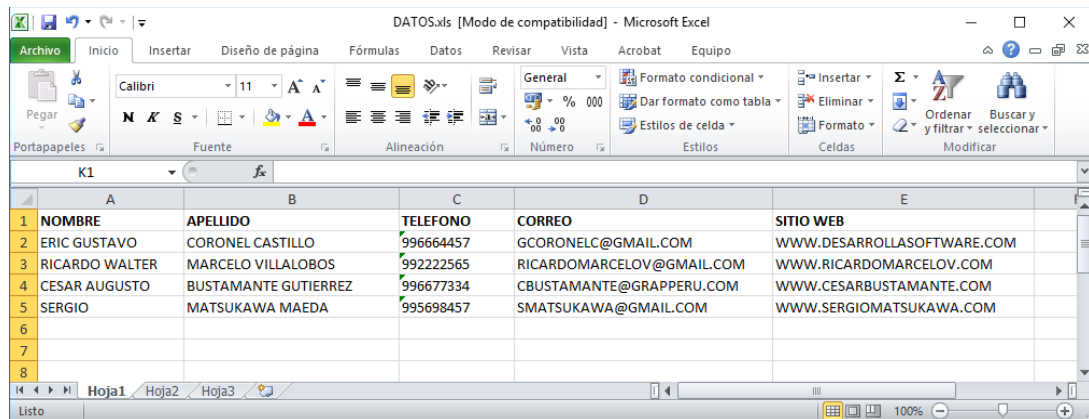
    /*
     * Se accede a la prima hoja del archivo Excel.
     */
    HSSFSheet hssfSheet = workbook.getSheetAt(0);

    /**
     * Se leen todos los datos, fila x fila, celda x celda.
     */
    Iterator rowIterator = hssfSheet.rowIterator();
    while (rowIterator.hasNext()) {
        HSSFRow hssfRow = (HSSFRow) rowIterator.next();
        Iterator iterator = hssfRow.cellIterator();
        while (iterator.hasNext()) {
            HSSFCell hssfCell = (HSSFCell) iterator.next();
            String stringCellValue = hssfCell.toString();
            stringCellValue = String.format("%1$-30s", stringCellValue);
            System.out.print(stringCellValue);
        }
        System.out.println("\n");
    }
} catch (Exception e) {
    e.printStackTrace();
}

}

public static void main(String[] args) {
    String fileName = "E:\\DATOS.XLS";
    LeerExcel leerExcel = new LeerExcel();
    leerExcel.leerExcel(fileName);
}
}
```

El archivo de prueba que se está utilizando es DATOS.xls y se encuentra en la unidad E, tal como se ilustra en la siguiente imagen:



	A	B	C	D	E
	NOMBRE	APELLIDO	TELEFONO	CORREO	SITIO WEB
1	ERIC GUSTAVO	CORONEL CASTILLO	996664457	GCORONELC@GMAIL.COM	WWW.DESARROLLASOFTWARE.COM
2	RICARDO WALTER	MARCELO VILLALOBOS	992222565	RICARDOMARCELOV@GMAIL.COM	WWW.RICARDOMARCELOV.COM
3	CESAR AUGUSTO	BUSTAMANTE GUTIERREZ	996677334	CBUSTAMANTE@GRAPPERU.COM	WWW.CESARBUSTAMANTE.COM
4	SERGIO	MATSUKAWA MAEDA	995698457	SMATSUKAWA@GMAIL.COM	WWW.SERGIOMATSUKAWA.COM

El resultado de su ejecución es el siguiente:

```
run:
NOMBRE      APELLIDO      TELEFONO      CORREO      SITIO WEB
ERIC GUSTAVO  CORONEL CASTILLO  996664457  GCORONELC@GMAIL.COM  WWW.DESARROLLASOFTWARE.COM
RICARDO WALTER  MARCELO VILLALOBOS  992222565  RICARDOMARCELOV@GMAIL.COM  WWW.RICARDOMARCELOV.COM
CESAR AUGUSTO  BUSTAMANTE GUTIERREZ  996677334  CBUSTAMANTE@GRAPPERU.COM  WWW.CESARBUSTAMANTE.COM
SERGIO        MATSUKAWA MAEDA  995698457  SMATSUKAWA@GMAIL.COM  WWW.SERGIOMATSUKAWA.COM
```

## CREAR UN ARCHIVO EXCEL

Primero, necesitamos crear un LIBRO haciendo uso de la clase [HSSFWorkbook](#):

```
// Procesar información y generar el xls.
HSSFWorkbook objWB = new HSSFWorkbook();
```

A continuación, creamos la hoja con la clase [HSSFSheet](#):

```
// Crear la hoja
HSSFSheet hoja1 = objWB.createSheet("hoja 1");
```

Luego, creamos la fila con [HSSFRow](#):

```
// Crear la fila.
HSSFRow fila = hoja1.createRow((short)1);
```

Note que el valor que se envía al método encargado de crear las filas es de tipo **short**, el mismo que indica el número correspondiente a la fila que debemos trabajar. El índice de las filas empieza en "0", aunque ello no nos impide trabajar directamente con otras filas.

Una vez creada la fila, empezamos a trabajar con las celdas.

```
// Aunque no es necesario podemos establecer estilos a las celdas.
// Primero, establecemos el tipo de fuente
HSSFFont fuente = objLibro.createFont();
fuente.setFontHeightInPoints((short)11);
fuente.setFontName(fuente.FONT_ARIAL);
fuente.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);

// Luego creamos el objeto que se encargará de aplicar el estilo a la celda
HSSFCellStyle estiloCelda = objLibro.createCellStyle();
estiloCelda.setWrapText(true);
estiloCelda.setAlignment(HSSFCellStyle.ALIGN_JUSTIFY);
estiloCelda.setVerticalAlignment(HSSFCellStyle.VERTICAL_TOP);
estiloCelda.setFont(fuente);

// También, podemos establecer bordes...
estiloCelda.setBorderBottom(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setBottomBorderColor((short)8);
estiloCelda.setBorderLeft(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setLeftBorderColor((short)8);
estiloCelda.setBorderRight(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setRightBorderColor((short)8);
estiloCelda.setBorderTop(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setTopBorderColor((short)8);

// Establecemos el tipo de sombreado de nuestra celda
estiloCelda.setFillForegroundColor((short)22);
estiloCelda.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);

// Creamos la celda, aplicamos el estilo y definimos
// el tipo de dato que contendrá la celda
HSSFCell celda = objFila.createCell((short)0);
celda.setCellStyle(estiloCelda);
celda.setCellType(HSSFCell.CELL_TYPE_STRING);

// Finalmente, establecemos el valor
celda.setCellValue("Un valor");
```

Como podemos apreciar en el código tenemos la posibilidad de establecer estilos mediante las clases **HSSFFont** y **HSSFCellStyle**.

La primera, nos permite establecer el tipo de fuente que se empleará para la celda que hemos de utilizar. Para ello, contamos con los métodos `setPointHeightInPoints` que recibe un valor de tipo `short` que representa el tamaño de la fuente; el método `setFontName` el mismo que recibe una constante de la misma clase que nos permite establecer la fuente que se ha de emplear, y, otros métodos como: `setBoldweight` y `setUnderline`, entre otros, que nos permitirán aplicarle otros estilos y efectos al valor que ocupe nuestra celda.

La segunda, es la clase que, finalmente, nos ayudará a aplicar el estilo a la celda. Podemos acomodar y alinear el texto mediante los métodos `setWrapText`, `setAlignment` y `setVerticalAlignment`; aplicar la fuente trabajada, con el método `setFont`; configurar los bordes mediante los métodos: `setBorderBottom`, `setBorderLeft`, `setBorderRight`, `setBorderTop`, para el tipo; y, `setBottomBorderColor`, `setLeftBorderColor`, `setRightBorderColor`, `setTopBorderColor` para establecer el color de los bordes; y, establecer el sombreado de las celdas mediante los métodos `setFillForegroundColor`, `setFillBackgroundColor` y `setFillPattern`.

Aunque, es un poco engorroso trabajar estos estilos, celda por celda, de esta forma, lo mejor es encapsular todo este proceso en métodos que nos permitan ahorrar líneas de código, preestableciendo, los estilos que se emplearán.

Según la versión de la librería que se esté empleando, podremos contar o no, con algunas constantes para la configuración del color y el establecimiento de los sombreados.

Finalmente, volcamos nuestro libro a un archivo de la siguiente forma:

```
// Volcamos la información a un archivo.
String strNombreArchivo = "E:\\LIBRO.xls";
File objFile = new File(strNombreArchivo);
FileOutputStream archivoSalida = new FileOutputStream(objFile);
objWB.write(archivoSalida);
archivoSalida.close();
```

A continuación les presento el código completo:

```
import java.io.File;
import java.io.FileOutputStream;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFFont;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
```

```
/**
 *
 * @author Eric Gustavo Coronel Castillo
 * @blog www.desarrollasoftware.com
 * @email gcoronelc@gmail.com
 */
public class CrearExcel {

    public static void main(String[] args) {

        try {
            // Procesar la información y generar el xls
            HSSFWorkbook objLibro = new HSSFWorkbook();

            // Crea la hoja
            HSSFSheet hoja1 = objLibro.createSheet("hoja 1");

            // Procesar la información y genero el xls.
            HSSFRow objFila = hoja1.createRow((short) 0);

            // Aunque no es necesario podemos establecer estilos a las celdas.
            // Primero, establecemos el tipo de fuente
            HSSFFont fuente = objLibro.createFont();
            fuente.setFontHeightInPoints((short) 11);
            fuente.setFontName(HSSFFont.FONT_ARIAL);
            fuente.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);

            // Luego se crea el objeto que se encargará de aplicar el estilo a la celda
            HSSFCellStyle estiloCelda = objLibro.createCellStyle();
            estiloCelda.setWrapText(true);
            estiloCelda.setAlignment(HSSFCellStyle.ALIGN_JUSTIFY);
            estiloCelda.setVerticalAlignment(HSSFCellStyle.VERTICAL_TOP);
            estiloCelda.setFont(fuente);

            // También se puede establecer bordes
            estiloCelda.setBorderBottom(HSSFCellStyle.BORDER_MEDIUM);
            estiloCelda.setBottomBorderColor((short) 8);
            estiloCelda.setBorderLeft(HSSFCellStyle.BORDER_MEDIUM);
            estiloCelda.setLeftBorderColor((short) 8);
            estiloCelda.setBorderRight(HSSFCellStyle.BORDER_MEDIUM);
            estiloCelda.setRightBorderColor((short) 8);
            estiloCelda.setBorderTop(HSSFCellStyle.BORDER_MEDIUM);
            estiloCelda.setTopBorderColor((short) 8);

            // Establecer el tipo de sombreado de la celda
            estiloCelda.setFillForegroundColor((short) 22);
            estiloCelda.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);
```



```

// Crear la celda, se aplica el estilo y se define
// el tipo de dato que contendrá la celda
HSSFCell celda = objFila.createCell((short) 0);
celda.setCellStyle(estiloCelda);
celda.setCellType(HSSFCell.CELL_TYPE_STRING);

// Finalmente se establece el valor
celda.setCellValue("GUSTAVO CORONEL");

// Ajusta el ancho de la primera columna
hoja1.autoSizeColumn(0);

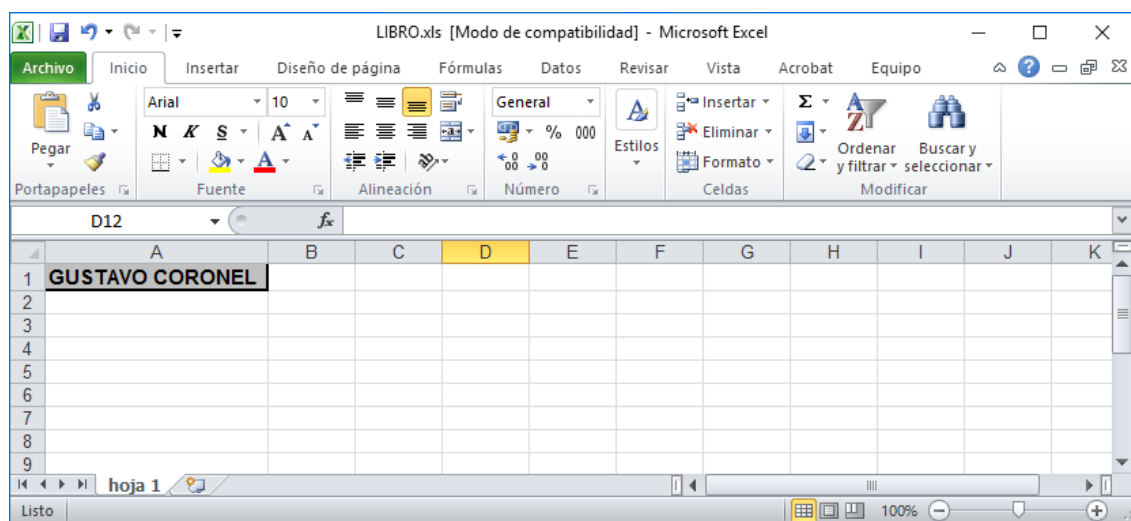
// Se vuelca la información a un archivo.
String strNombreArchivo = "E:\\LIBRO.xls";
File objFile = new File(strNombreArchivo);
FileOutputStream archivoSalida = new FileOutputStream(objFile);
objLibro.write(archivoSalida);
archivoSalida.close();

System.out.println("Todo ok.");
} catch (Exception e) {
    e.printStackTrace();
}

}

}
    
```

El resultado lo puedes ver a continuación:





Aunque el ejemplo es sencillo, todo lo presentado aquí generalmente se combina con arreglos de beans y bucles, los cuales te ayudarán a presentar más datos en tus hojas de Excel.

El objetivo es presentar un pequeño caso práctico sobre el uso de HSSF (POI) para la generación de archivos Excel.

## Capítulo 9

# I REPORT & JASPERREPORT

### INTRODUCCIÓN

JasperReports es la mejor herramienta de código libre en Java para generar reportes. Puede entregar ricas presentaciones o diseños en la pantalla, para la impresora o para archivos en formato PDF, HTML, RTF, XLS, CSV y XML.

Está completamente escrita en Java y se puede utilizar en una gran variedad de aplicaciones de Java, incluyendo JEE o aplicaciones Web, para generar contenido dinámico.

### Requerimientos de JasperReports

Las siguientes librerías junto con la de JasperReports deben incluirse en el proyecto en que se desee incluir esta herramienta para generar reportes.

- Jakarta Commons Digester Component
- Jakarta Commons BeanUtils Component
- Jakarta Commons Collections Component
- Jakarta Commons Logging Component
- Generador de PDF – iText
- XLS Jakarta POI

### FUNCIONAMIENTO DE JASPERREPORTS

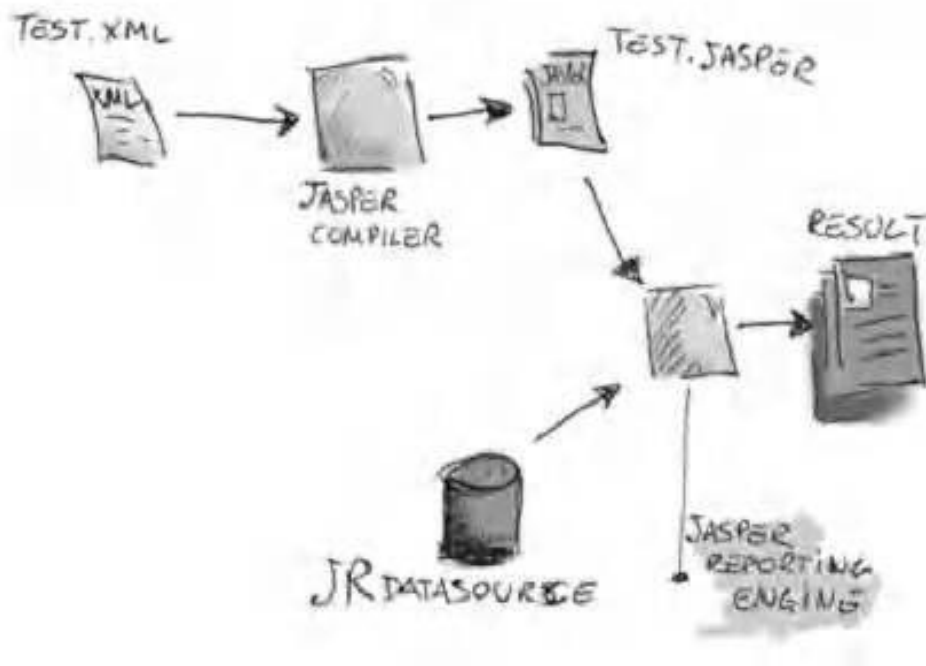
JasperReports trabaja en forma similar a un compilador y a un intérprete. El usuario diseña el reporte codificándolo en XML de acuerdo a las etiquetas y atributos definidos en un archivo llamado jasperreports.dtd (parte de JasperReports). Usando XML el usuario define completamente el reporte, describiendo donde colocar texto, imágenes, líneas, rectángulos, cómo adquirir los datos, como realizar ciertos cálculos para mostrar totales, etc.

Este archivo fuente XML debe ser compilado para obtener un reporte real. La versión compilada del fuente es nombrada "**archivo jasper**" (este termina con .jasper). Un Archivo jasper es el compilado de un código fuente. Cuando tenemos un archivo jasper, necesitamos otra cosa para producir un reporte: necesitamos datos. Esto no siempre es cierto. En algunos casos querríamos generar un reporte que no mostrara datos dinámicos, solo texto estático por ejemplo, pero esto puede simplificarse a un reporte que tiene solamente un registro vacío. Para proporcionar estos registros al

“jasper engine” necesitamos presentarlos usando una interfaz especial específica llamada JRDataSource.

Una fuente de datos + un Archivo jasper = un “archivo print”.

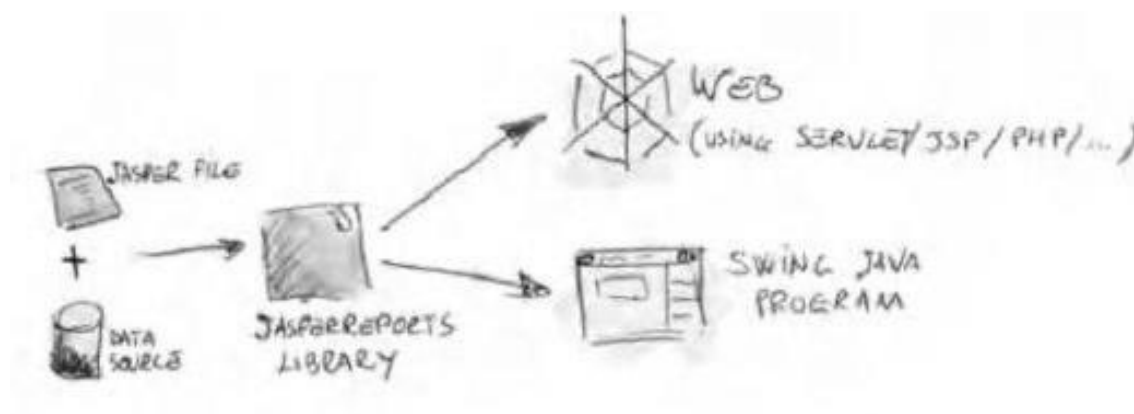
Un “archivo print” puede exportarse en muchos formatos como PDF, HTML, RTF, XML, XLS, CVS, etc. La exportación se puede realizar utilizando clases especiales para implementar exportadores específicos.



## Compilación, exportación de reportes de JasperReports

Para un principiante, diseñar y crear el archivo jasper es la tarea más dura. Cuando se haya diseñado y compilado el archivo jasper, se puede utilizar la librería JasperReports para llenar dinámicamente el reporte en varios entornos como una aplicación web (Usando un servlet de Java por ejemplo, pero también funciona para generar reportes PDF desde un script PHP).

Jasper tiene disponible un visualizador especial para desplegar la vista previa de un reporte; diseñado para aplicaciones tradicionales de Java basadas en Swing.



## IREPORT

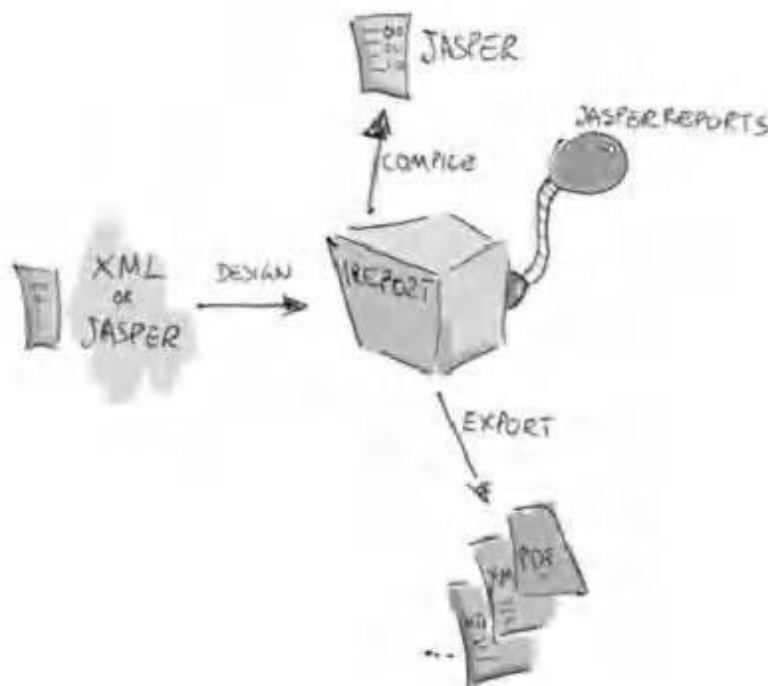
iReport es un diseñador visual de código libre para JasperReports escrito en Java. Es un programa que ayuda a los usuarios y desarrolladores que usan la librería JasperReports para diseñar reportes visualmente. A través de una interfaz rica y simple de usar, iReport provee las funciones más importantes para crear reportes amenos en poco tiempo.

iReport puede ayudar a la gente que no conoce la sintaxis XML para generar reportes de JasperReports.

### Funcionamiento de iReport

iReport provee a los usuarios de JasperReports una interfaz visual para construir reportes, generar archivos “jasper” y “print” de prueba. iReport nació como una herramienta de desarrollo, pero puede utilizarse como una herramienta de oficina para adquirir datos almacenados en una base de datos, sin pasar a través de alguna otra aplicación.

iReport puede leer y modificar ambos tipos de archivo, XML y jasper. A través de JasperReports, es capaz de compilar XML a archivos jasper y “ejecutar reportes” para llenarlos usando varios tipos de fuentes de datos (JRDataSource) y exportar el resultado a PDF, HTML, XLS, CSV,...



## Configuración de la conexión a una base de datos.

Para establecer una conexión entre iReport y una base de datos, se debe proporcionar el driver JDBC correspondiente. Las últimas versiones iReport ya proporcionan drivers JDBC para establecer conexiones con bases de datos como MySQL. Para nuestro ejemplo se usará una conexión con una base de datos Access, para la cual se ha configurado un origen de datos con nombre DSN.

Suponiendo que se ha configurado un origen de datos nombrado para una base de datos Access con los siguientes valores:

Nombre de driver: PDRV Nombre de inicio de sesión: cvazquez Contraseña: vazquez

Procedemos a configurar iReport para establecer la conexión con la base de datos, para ello debe ir a menú->Fuente de datos->Conexiones/Fuente de datos. En la pantalla Connections/Datasources oprima el botón new para agregar una conexión.

La pantalla de conexión debe llenarse tal como se muestra a continuación:



A continuación oprima el botón Test para probar la conexión. Si la conexión fue exitosa, oprima finalmente el botón Save para guardar esta conexión.



## Creación del Reporte.

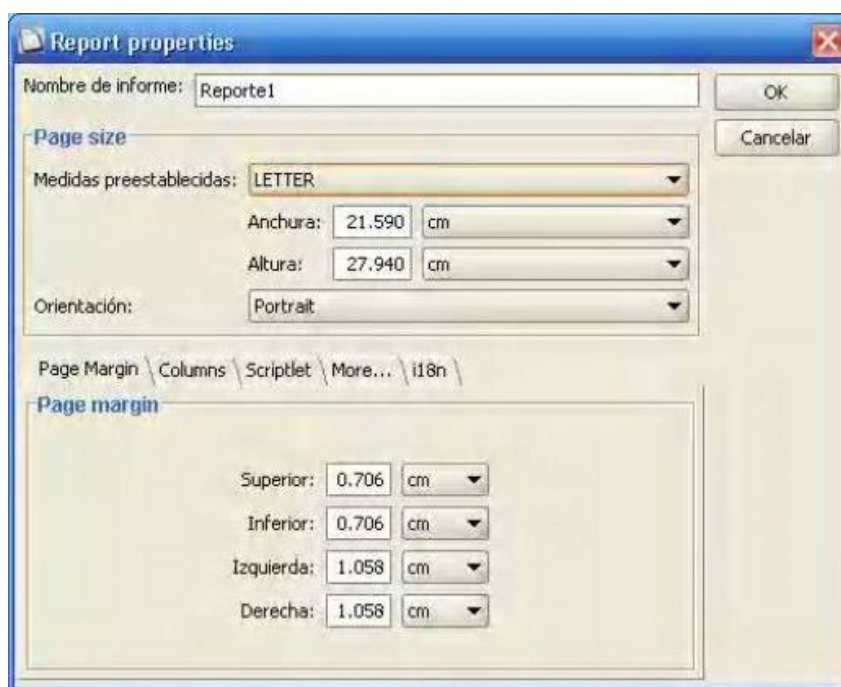
En iReport, se tiene la opción para trabajar por proyecto, el cual puede contener varios reportes, en nuestro caso no se creará un proyecto, se creará solo un reporte de la siguiente manera:

Seleccione nuevo documento del menú Fichero o bien oprima el botón new report de la barra de herramientas, aparecerá una pantalla de propiedades del nuevo reporte que queremos crear:

En esta pantalla podemos configurar las propiedades del reporte, en nuestro caso, le llamaremos Reporte1, oprimir el botón OK para crearlo.

Seleccionar la opción Guardar como... del menú fichero o bien el botón Save report de la barra de herramientas, debe seleccionar el nombre y el directorio en que se guardara el reporte. El reporte se guardará con la extensión .xml. Por defecto los archivos de salida de la compilación se crearán en el directorio de instalación de iReport si no especificó uno.





## Secciones de un Reporte en iReport.

A continuación se explicará de manera breve, las secciones que componen a un reporte en iReport

	title	
	pageHeader	
	columnHeader	
	detail	
	columnFooter	
	pageFooter	
	lastPageFooter	
	summary	

- **title.** El título de nuestro reporte debe escribirse en esta sección. Solo se mostrará en la primera página del reporte.
- **pageHeader.** Aparece en la parte superior de cada página. Puede contener información adicional del reporte, descripciones, etc.

- **columnHeader.** En esta sección se muestran los nombres de los campos que se van a presentar
- **detail.** En esta sección se despliegan los valores correspondientes a los nombres de los campos definidos en la sección anterior. Estos datos pueden obtenerse mediante consultas SQL a una base de datos por ejemplo.
- **columnFooter.** Puede presentar información de totales para algunos de los campos de la sección detail. Por ejemplo "Total de Empleados: 220"
- **pageFooter.** Aparece en la parte inferior de cada página. Este parte puede presentar, la fecha, número de página del reporte.
- **summary.** Esta sección puede presentar totales de campos de la sección detail. Si se desea incluir algún gráfico en el reporte, debe hacerse en esta sección.

En el diseño de su reporte pueden omitirse algunas de las secciones o bandas mencionadas, en nuestro caso solo usaremos las secciones title, PageHeader, ColumHeader, detail, y Pagefooter. Para omitir las secciones del reporte que no se usaran, debe oprimir el botón bands de la barra de herramientas, o bien haciendo click con el botón secundario del ratón sobre el diseño del reporte y seleccionando la opción band properties del menú contextual. En la pantalla de propiedades de las bandas, debe seleccionar las bandas no deseadas y colocar su propiedad band height igual a cero como se muestra en la siguiente figura.



## Diseño del Reporte.

Se muestran a continuación los botones principales para el diseño del reporte de la barra de herramientas:

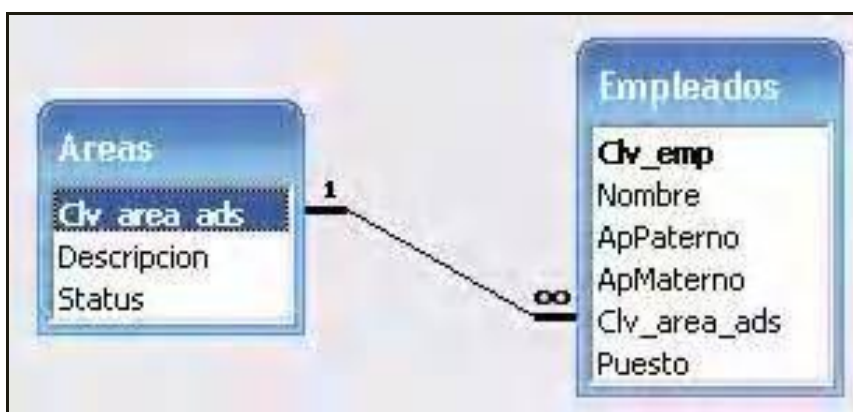


Agreguemos en primer lugar el título de nuestro reporte, para ello seleccione de la barra de herramientas el objeto Static text tool y dibuje una caja sobre la banda title. Haciendo doble click sobre la caja dibujada se mostrará la pantalla de propiedades de la caja de texto estático. Aquí puede configurar el tamaño, ubicación de la caja de texto, tipo de letra para el texto, entre otros; seleccionando la pestaña Static Text puede ingresar el título que desee para el reporte, el resultado debe ser parecido al de la siguiente figura:

En el encabezado de página, pageHeader, podemos colocar una descripción del reporte utilizando también el objeto Static Text tool.

Ahora se agregarán los nombres de los campos que pretendemos mostrar en el reporte, en este caso se recuerda que se configuró una conexión con una base de datos Access a través de un driver u origen de datos con nombre DSN.

Para cuestiones de prueba he agregado las siguientes tablas con los siguientes campos a la base de datos:

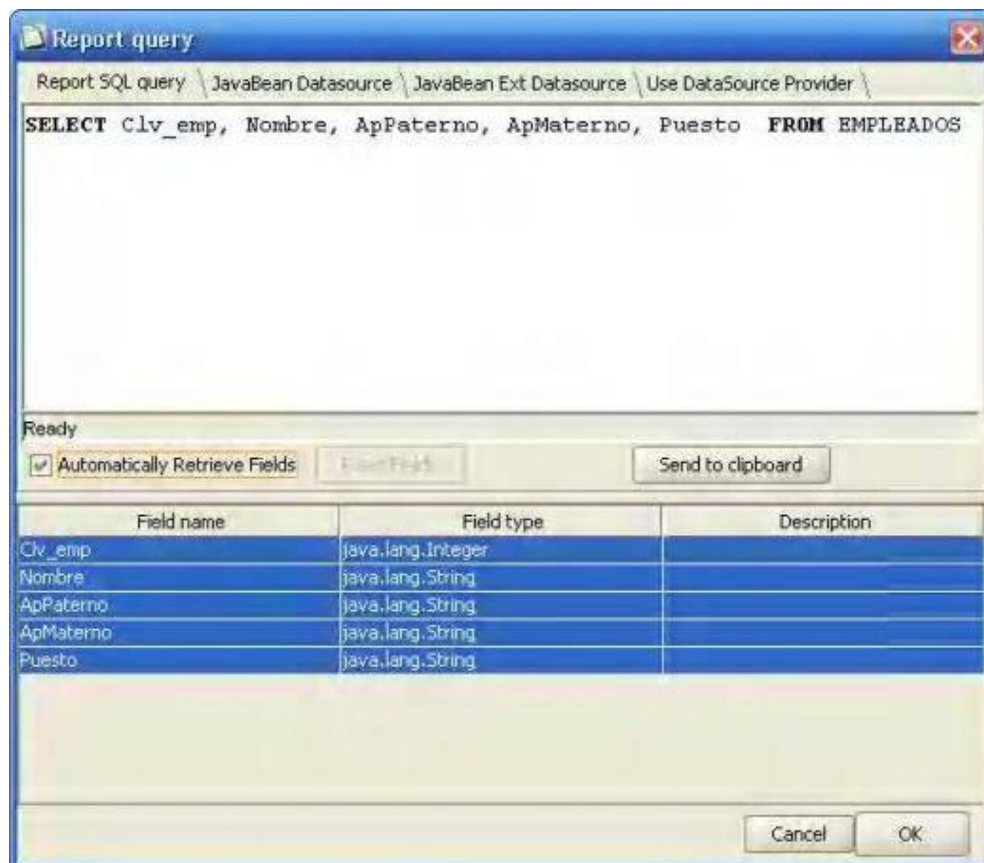


La relación de las tablas como se observa es “uno a muchos”. En este contexto, un empleado puede pertenecer a solo una área de trabajo y una área puede relacionarse con uno o muchos empleados.

Se realiza por ahora una consulta sencilla para el reporte a la tabla de empleados de la siguiente manera:

```
SELECT Clv_emp, Nombre, ApPaterno, ApMaterno, Puesto FROM EMPLEADOS
```

Antes, de agregar los nombres y campos a nuestro reporte, se establecerá la consulta anterior para el reporte. Vaya a la barra de herramientas y seleccione el botón Database, en la pantalla Report Query y pestaña Report SQL query, puede escribirse la sentencia SQL. Si se encuentra seleccionado el check box Automatically Retrieve Fields, nos mostrará automáticamente los campos que se obtiene la consulta, el tipo y una descripción de estos si es que cuentan con ella. Si la consulta es incorrecta mostrará un mensaje de error. La pantalla debe lucir como sigue:



Debe hacer click en el botón OK para guardar esta consulta.

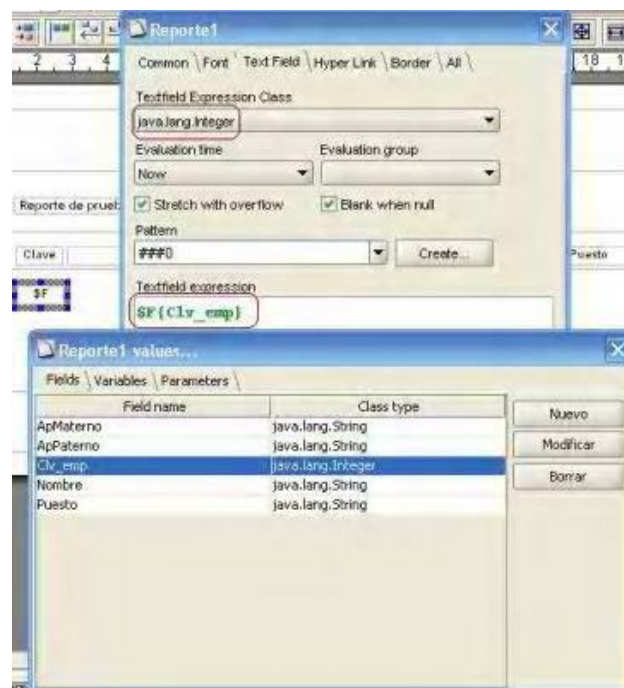
De la misma manera en que se agregó el título al reporte, deberá agregar los nombres de los campos en la banda columnHeader utilizando objetos Static Text tool. El reporte debe lucir hasta ahora como se muestra a continuación:

Mi primer reporte				
Reporte de prueba en iReport, muestra los datos de empleados				
Clave	Nombre	Apellido Paterno	Apellido Materno	Puesto
Detail				
pageFooter				

Ahora solo resta colocar en la sección detail, los campos que se mostrarán en el reporte. Para esto se usará el objeto Text Field, las cajas se pintarán de manera similar a las cajas de texto estático realizadas en la banda columnHeader, sin embargo cada campo debe configurarse de acuerdo al tipo de dato que se quiere mostrar.

A continuación se mostrará la manera de configurar un Text Field. Una vez colocado un campo en la sección detail, haga doble click sobre este para abrir su ventana de propiedades y sitúese en la pestaña Text Field. Vaya enseguida al menú Ver y seleccione el item Campos de informe, esto desplegará la pantalla values con los campos de nuestro reporte, los cuales se generaron al establecer la consulta SQL. Esta pantalla muestra adicionalmente los parámetros y variables del reporte, cada uno se distinguirá con la siguiente notación: Campos:  $\$F\{\text{Campo}\}$  Variables:  $\$V\{\text{valor}\}$  Parámetros:  $\$P\{\text{Parámetro}\}$

Utilice esta pantalla para auxiliarse al configurar un Text Field.





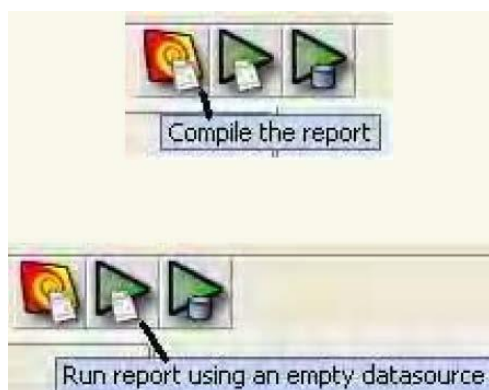
La figura anterior muestra como debe configurarse el campo “clave de empleado”. En la ventana de propiedades ponga especial atención en seleccionar el tipo correcto del campo en el combo Textfield ExpressionClass (Integer en este caso). En la sección Textfield expression de la misma ventana, cambie la expresión `$F{Field}` por el nombre correcto del campo `$F{Clv_emp}`. Configure así cada uno de los campos restantes.

Adicionalmente, se agregará una línea al inicio de la sección detail para separar los registros con el objeto line tool de la barra de herramientas. Debe reducir el tamaño de la banda detail al alto de las cajas de los campos para evitar demasiado espacio entre los registros y listo, con ligeras adecuaciones a los campos, el reporte final debería lucir de la siguiente manera:

Mi primer reporte				
Reporte de prueba en iReport, muestra los datos de empleados				
Clave	Nombre	Apellido Paterno	Apellido Materno	Puesto
\$F	\$F(Nombre)	\$F(ApPaterno)	\$F(ApMaterno)	\$F(Puesto)

## Compilación y Ejecución del Reporte

Las siguientes figuras muestran los botones de la barra de herramientas necesarios para compilar, y ejecutar el reporte con o sin conexión.



## JAVA CLIENTE - SERVIDOR

Antes que nada, seleccione la vista para el Reporte, vaya al menú Construir y seleccione el ítem vista previa en JRViewer (Vista previa en el Viewer de Jasper). En este menú, puede seleccionar la vista previa para distintos formatos de archivo, siempre y cuando haya configurado los programas externos como se explicó en la primera parte del artículo.

Compile el reporte, el resultado de la compilación aparecerá en la parte inferior de la pantalla. Los errores más frecuentes de compilación se relacionan con los tipos de los campos que pretenden mostrarse. Si la compilación resultó sin errores, está listo para ver su reporte, es recomendable probarlo primero sin usar una conexión a una base de datos. Finalmente, ejecute el reporte ocupando la conexión a la base de datos que se configuró. El resultado dependiendo de sus datos en las tablas debe ser parecido al siguiente:



The screenshot shows the 'Report JasperViewer' window. The title bar says 'Report JasperViewer'. The toolbar includes icons for opening, saving, printing, and zooming, along with a zoom percentage of 100%. The main content area displays a report titled 'Mi primer reporte' with a subtitle 'Reporte de prueba en iReport, muestra los datos de empleados'. Below the subtitle is a table with 5 columns: Clave, Nombre, Apellido Paterno, Apellido Materno, and Puesto. The table contains 18 rows of employee data. At the bottom right of the window, it says 'Page 1 of 95'.

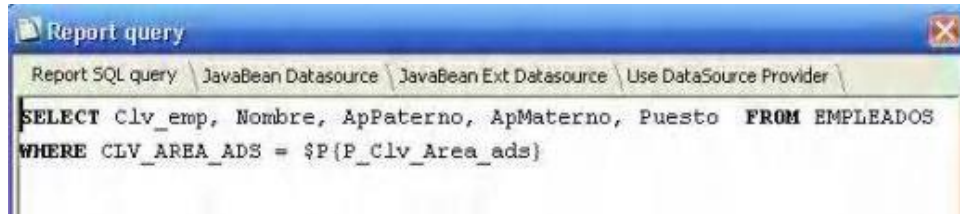
Clave	Nombre	Apellido Paterno	Apellido Materno	Puesto
144	BENITO	BAHENA Y	LOME	CAJERO GENERAL
2210	LUIS	LOPEZ	VARGAS	AUXILIAR ADMIVO. "A"
10877	FELIPE	RAYON	BELTRAN	OPERARIO 1º TALLER
10884	URBANO	RAMIREZ	CORTES	CHOFER NOCTURNO TALLER
10907	GABINO	BENITO	AGUIRRE	OPERADOR DE TROLEBUS
10917	JORGE DE JESUS	HERNANDEZ	ORTIZ	CHECADOR DE TIEMPO
10922	ARTURO	MENDOZA	LORENZO	OPERADOR DE TROLEBUS
10923	J. DOMINGO	PEREZ	MEDINA	OPERARIO 1RA. "B" CABO
10936	JOSE	SANCHEZ	FLORES	OPERARIO 1º TALLER
10967	ADOLFO	GARCIA	LARA	SUPERVISOR "B" DE OPERACION
10969	ALFONSO ARTEMIO	ROSAS	VILLALVA	OPERADOR DE TROLEBUS
10971	ANTONIO	CASILLAS	CORTES	OPERADOR DE TROLEBUS
10972	JOSE LUIS	RAMIREZ	AYALA	OPERADOR DE TROLEBUS
10978	HUMBERTO	CORDOBA	VEGA	OPERADOR DE TROLEBUS
11014	MARIO	OSORIO	ARENAS	SUPERV. DE TALLER AUTOMOTRIZ
11018	GUILLERMO	SOTO	GARCIA	OPERADOR DE TROLEBUS
11023	MANUEL	VEGA	CAZARES	OPERADOR DE TROLEBUS

El Viewer de JasperReports, muestra en su barra de herramientas la posibilidad de enviar el reporte directamente a la impresora o bien de guardar el reporte en algún formato de archivo específico: PDF, HTML, XLS, RTF entre otros. La funcionalidad de iReport en este caso es de oficina, obteniendo los datos almacenados en una base de datos y mostrándolos sin pasar a través de ninguna otra aplicación.

Llenar el reporte dinámicamente desde una aplicación Swing.

Algo que resultaría más interesante, es llenar el reporte dinámicamente desde alguna aplicación Java. En este caso, debe hacerse uso del archivo \*.jasper generado de la compilación del archivo xml.

Oprima el botón Database de la barra de herramientas. En la ventana Report Query, modifique el Query que se muestra en la pestaña Report SQL query, como se muestra a continuación:



Se ha modificado el query para que se muestre en el reporte, solo a aquellos empleados que pertenezcan a determinada área, dicha área se pasará como parámetro desde una aplicación Swing para llenar el reporte.

Debe agregarse este parámetro al reporte, para esto, oprima el botón Parameters de la barra de herramientas o bien, desde el menú ver, seleccione Parámetros de informe. En la pantalla valores, asegúrese de estar ubicado en la pestaña Parameters y oprima el botón Nuevo. Agregue el nuevo parámetro del reporte, como se muestra en la siguiente figura.



El tipo de parámetro se estableció como String aún cuando se sabe de las tablas que debería ser entero, en realidad esto funciona bien, pasando desde la aplicación java al reporte un String. El tipo de parámetro del reporte debe ser del mismo tipo al que se vaya a pasar desde sus aplicaciones en Java, de lo contrario obtendrá errores que le darán muchos dolores de cabeza. Recompile el proyecto, si se muestra un mensaje de error de compilación mencionando la ausencia del parámetro, vuelva a agregarlo y



abra la ventana Report query para asegurarse que se muestran los campos de la consulta. Por alguna razón, la aplicación algunas veces no detecta el nuevo parámetro. El siguiente hilo es capaz de llenar y exportar el reporte realizado, solo ha de proporcionarse una conexión a la base de datos y la ruta del archivo jasper.

```
import java.util.*; import java.sql.Connection;
import java.awt.event.*;

/*Librerías necesarias para Jasper Reports*/
import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.view.*;

public class cExport_thread extends Thread {

    cConnection conexion;

    public cExport_thread(String Clv_area) {
    }

    /**
     * Método del hilo */
    public void run(){
        try {
            //Ruta de Archivo Jasper
            String fileName="C:\\proyecto\\Reporte1.jasper";
            //Obtner una conexión a la base de datos
            conexion = new cConnection();
            Connection con = conexion.mkConection();
            //Pasamos parametros al reporte Jasper.
            Map parameters = new HashMap();
            parameters.put("P_Clv_Area_ads",Clv_area);
            //Preparacion del reporte (en esta etapa llena el diseño de reporte)
            //Reporte diseñado y compilado con iReport
            JasperPrint jasperPrint =
                JasperFillManager.fillReport(fileName,parameters,con);
            //Se lanza el Viewer de Jasper, no termina aplicación al salir
            JasperViewer jviewer = new JasperViewer(jasperPrint,false);
            jviewer.show();
        } catch (Exception j) {
            System.out.println("Mensaje de Error:"+j.getMessage());
        } finally{
            conexion.closeConecction();
        }
    }
}
```

La finalidad del hilo, en mi caso particular, fue para liberar de carga al evento de un botón en una aplicación Swing, dado que la obtención de la conexión y el llenado del reporte puede ser tardado. El hilo debe lanzarse de la siguiente manera:

```
cExport_thread thread_exp = new cExport_thread();  
thread_exp.start();
```

Hasta esta fecha, han salido bastantes versiones de JasperReports e iReport, si utiliza versiones diferentes a las aquí utilizadas, asegúrese que coincidan las librerías de Jasper tanto en iReport como en su proyecto java con el que pretende llenar el reporte.

---

## CREACIÓN DE GRÁFICOS EN IREPORT

---

JasperReports no maneja directamente gráficos: estos deben crearse independientemente como imágenes, incluso utilizando una de las numerosas librerías de código libre disponibles para la creación de gráficos. La imagen producida será mostrada usando un componente de imagen. La idea es realmente simple, pero la creación de un gráfico en tiempo de ejecución requiere de un buen conocimiento de la programación de JasperReports, y muchas veces es necesario utilizar scriptlets capaces de coleccionar los datos que se mostrarán en el gráfico.

A partir de la versión 0.4.0, iReport proporciona una herramienta para simplificar la construcción de un gráfico. Esta herramienta permite crear un gráfico configurando propiedades y datos principales que serán vistos de manera simple por el usuario final.

La creación de un gráfico se basa en una librería muy conocida de código libre llamada JFreeCharts desarrollada por David Gilbert de Object Refinery Limited. iReport soporta por ahora solamente un pequeño número de tipos de gráficos presentados en JFreeCharts, y pueden modificarse solamente algunas de las propiedades del gráfico, pero es posible aún así, crear gráficos limpios con un gran impacto a la vista.

Fuente: Documentación y Tutoriales de iReport en: <http://ireport.sourceforge.net/>

Después de la pequeña introducción, se mostrará la manera de crear un grafico en iReport.

La versión de iReport aquí utilizada será la 0.5.1, para versiones más recientes, la creación de los gráficos puede ser diferente; deben referirse a la documentación correspondiente de su versión.

Supondremos que ya se cuenta con una configuración correcta del iReport y sus librerías, especialmente la librería JFreecharts, la versión que se incluye con iReport 0.5.1, es la jfreechart-1.0.0-rc1.

Además, se necesitará tener configurada una conexión a una base de datos, y por supuesto datos para poder establecer una consulta que alimentará con datos al gráfico.

A partir de los datos de las siguientes tablas, se creará el gráfico en cuestión.



Las tablas anteriores muestran las actividades realizadas en una empresa. Cada una de las actividades se relaciona con un solo servicio. Cada actividad pertenece a un trabajador, que por simplicidad este campo no se muestra en este diseño de tablas.

Supongamos que se desea saber cuantas actividades se realizaron por servicio en un determinado rango de fechas, digamos en un mes. Se debe proceder a crear la consulta SQL correspondiente para obtener estos datos, más aún, podemos reflejarlos gráficamente, la consulta sería como la siguiente:

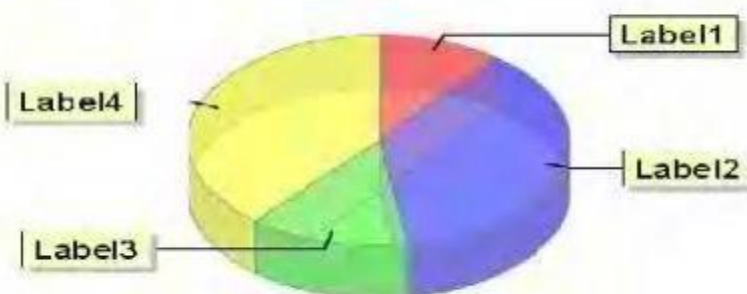
```

Report query
Report SQL query \ JavaBean Datasource \ JavaBean Ext Datasource \ Use DataSource Provider \

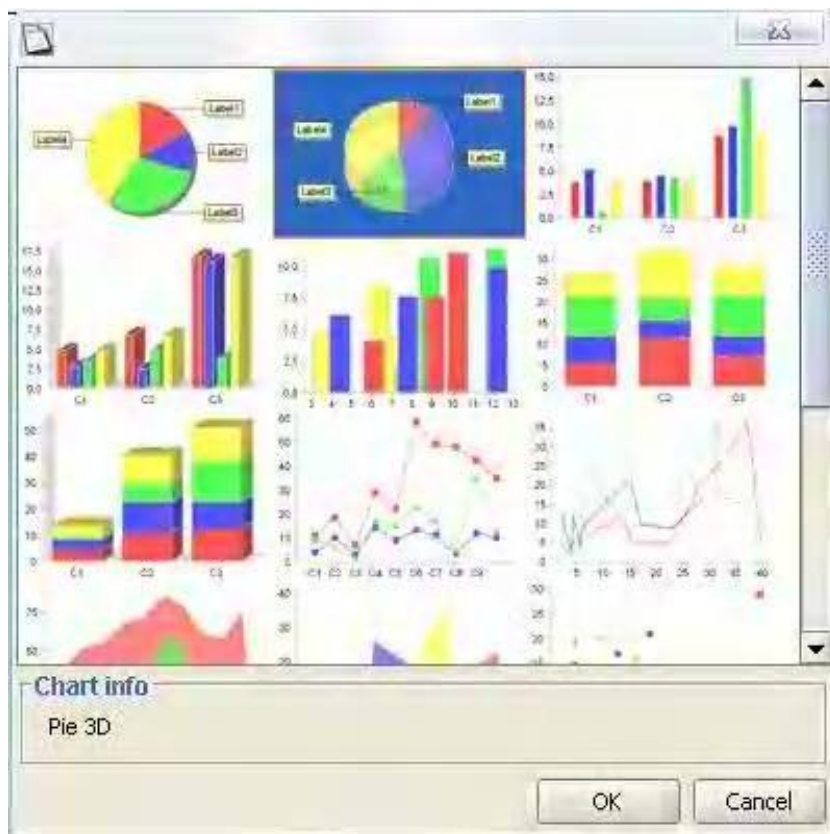
select Servicios.Tipo_Servicio,count(*) As Total
from Actividades,Servicios
where Actividades.IdServicio = Servicios.IdServicio
And Actividades.Fecha_ter >= #1/5/2005#
And Actividades.Fecha_ter <= #25/5/2005#
Group By Servicios.Tipo_Servicio
    
```

La consulta anterior agrupará las actividades realizadas por tipo de Servicio que se realizaron del 1 de mayo al 25 de mayo del 2005.

El diseño de reporte es el siguiente:

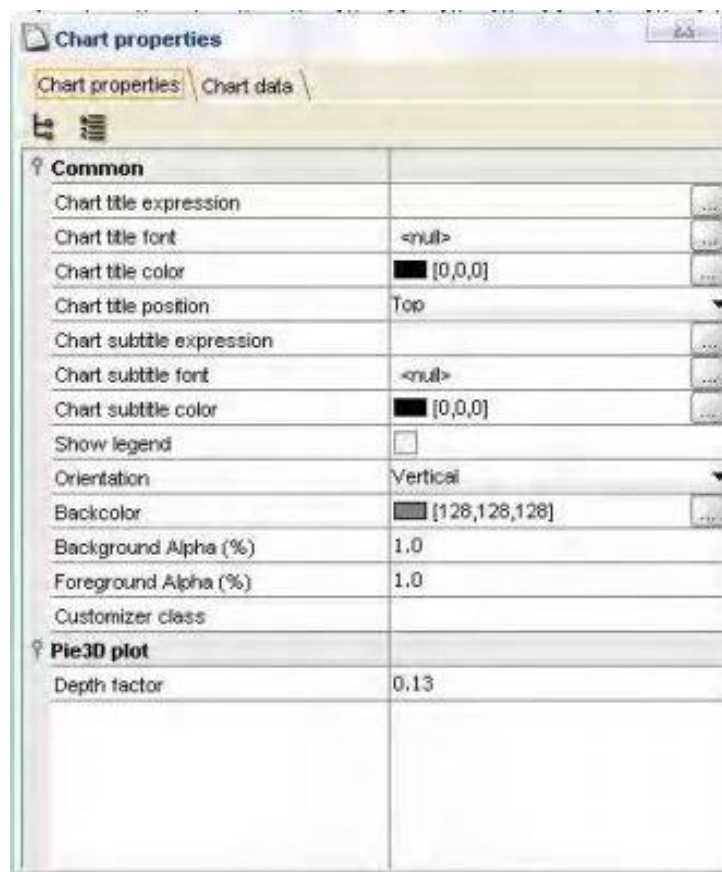
Sistema de Adquisición y Mantenimiento de Reporte de Actividades. Servicio de Transportes Eléctricos del D.F.	
Resumen de Actividades por Tipo de Servicio	
Rango de Fechas: de <input type="text" value="\$P{date1}"/> a <input type="text" value="\$P{date2}"/>	
Tipo de Servicio	Número
<input type="text" value="\$F(Tipo_Servicio)"/>	<input type="text" value="\$F(Total)"/>
new Date() <input "="" \$v{page_number}="" \$v{page_total}"="" +="" de="" type="text" value="Pág. "/>	
Total de Actividades: <input type="text" value="\$V{EXP_1}"/>	
<div></div>	

Para insertar un gráfico en un reporte de iReport, seleccione el objeto Chart tool de la barra de herramientas de iReport y dibuje el gráfico sobre la sección summary del reporte:



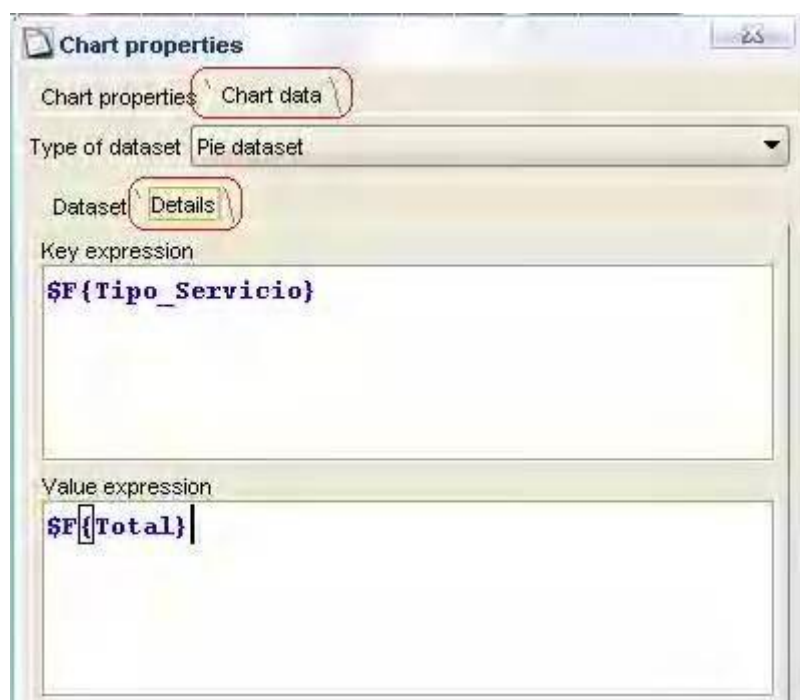
A continuación deberá seleccionar el tipo de gráfico deseado, para este ejemplo seleccionamos el tipo Pie 3D. Oprimir el botón OK y el componente que contendrá al gráfico ya ha sido creado, lo que resta es configurar las propiedades del mismo y los datos que mostrará.

Haciendo doble click sobre el componente del gráfico, se mostrará la pantalla de propiedades de este, sitúese en la pestaña Chart, a continuación oprima el botón Edit chart properties, deberá aparecer una pantalla como la siguiente:



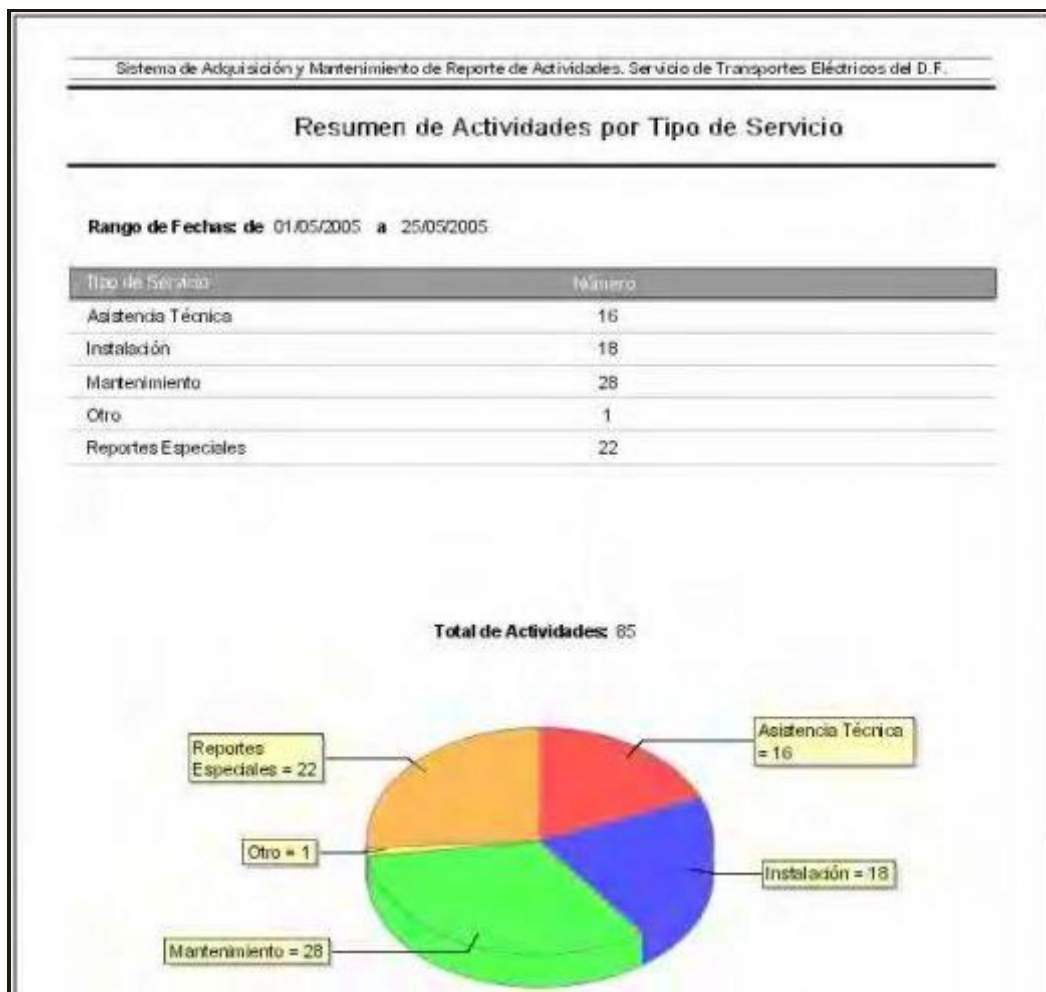
En la pestaña Chart properties de esta pantalla, se encuentran las propiedades que pueden modificarse para el gráfico en turno, como el nombre del gráfico, las fuentes, leyendas, ect. Puede dejar las propiedades actuales por ahora y modificarlas posteriormente. Ahora centraremos la atención en la manera de configurar los datos para el gráfico.

Debe situarse en la pestaña Chart data de la pantalla anterior y enseguida en la pestaña Details. En el apartado Key expression, coloque el campo `$F{Tipo_Servicio}` y en el apartado Value expresión coloque el campo `$F{Total}`. Estos campos son precisamente los que se obtienen de la consulta que se estableció anteriormente, estos a su vez se mostrarán en la sección detail del reporte.



Listo se han configurado los datos que necesita el gráfico. Al compilar y ejecutar el reporte con iReport el resultado puede ser parecido al siguiente:





Si se desea mostrar el gráfico anterior en tiempo de ejecución desde una aplicación Java, se procede de manera semejante a como se muestran otros reportes. En este caso deben determinarse los parámetros que se pasarán desde la aplicación java al reporte, por lo que la consulta se modifica como se muestra a continuación:

```
Report query
Report SQL query \ JavaBean Datasource \ JavaBean Ext Datasource \ Use DataSource Provider \
select Servicios.Tipo_Servicio,count(*) As Total
from Actividades,Servicios
where Actividades.IdServicio = Servicios.IdServicio
And Actividades.Fecha_ter >= $P{date1}
And Actividades.Fecha_ter <= $P{date2}
Group By Servicios.Tipo_Servicio
```

Lo que se pretende con la consulta anterior, es pasar como parámetros las fechas para las que se desea obtener el total de actividades agrupadas por servicio.

Para la versión 0.5.1 de iReport, solo debe compilarse el reporte para obtener el archivo Jasper que será llenado por la aplicación. En versiones anteriores o quizá



recientes, es posible que se necesite incluir un scriptlet para coleccionar los datos del gráfico.

Contando con el archivo Jasper del reporte, la manera en que se manda llenar desde una aplicación Java es la misma que para otros reportes, como se muestra en el siguiente fragmento de código:

```
//Ruta de Archivo Jasper
String fileName="C:\\proyecto\\Grafico.jasper";
//Obtner una conexión a la base de datos
conexion = new cConnection(); Connection con = conexion.mkConection();
//Pasamos parametros al reporte Jasper.
Map parameters = new HashMap(); parameters.put("date1",p_date1);
parameters.put("date2",p_date2);
//Preparacion del reporte (en esta etapa llena el diseño de reporte) //Reporte
diseñado y compilado con iReport
JasperPrint jasperPrint = JasperFillManager.fillReport(fileName,parameters,con);
//Se lanza el Viewer de Jasper, no termina aplicación al salir
JasperViewer jviewer = new JasperViewer(jasperPrint,false); jviewer.show();
...
```

Puede exportar el reporte a diferentes formatos de archivo directamente desde la aplicación sin pasar por el JasperViewer, para esto puede referirse a las API's de su versión correspondiente.



# Capítulo 10

## PRACTICAS DE LABORATORIO

### PRACTICA 01

#### Objetivo

1. Creación del esquema SCOTT.
2. Creación del esquema EUREKA.

#### Actividades Previas

1. Verifique que Oracle Database se encuentre instalado.
2. De no estar instalado, proceda a su instalación. Le recomiendo que revise el siguiente enlace:

<http://gcoronelc.blogspot.pe/2012/03/instalacion-de-oracle-11g-r2.html>

#### Creación del Esquema SCOTT

1. Proceda a crear el esquema SCOTT.
2. Revise su catálogo.
3. Revise el contenido de cada una de sus tablas.

#### Creación del Esquema EUREKA

1. Proceda a crear el esquema EUREKA. Le recomiendo que revise el siguiente enlace:

<http://gcoronelc.blogspot.pe/2014/11/creacion-del-esquema-eureka.html>

2. Revise su catálogo.
3. Revise el contenido de cada una de sus tablas.

## PRACTICA 02

### Objetivo

Aplicar JDBC con bases de datos Oracle para desarrollar consultas básicas.

### Proyecto 1

Haciendo uso del esquema SCOTT de la base de datos Oracle desarrolle una aplicación que permita consultar empleado en base a su nombre.

A continuación tenemos la interfaz que debe desarrollar.

Consultar Empleados

Nombre

Consultar

Resultado

Codigo	Nombre	Cargo	Salario

### Proyecto 2

Haciendo uso de la base de datos EurekaBank, desarrolle una aplicación que permita a los empleados consultar los movimientos de una cuenta.

El empleado debe previamente iniciar sesión.

## PRACTICA 03

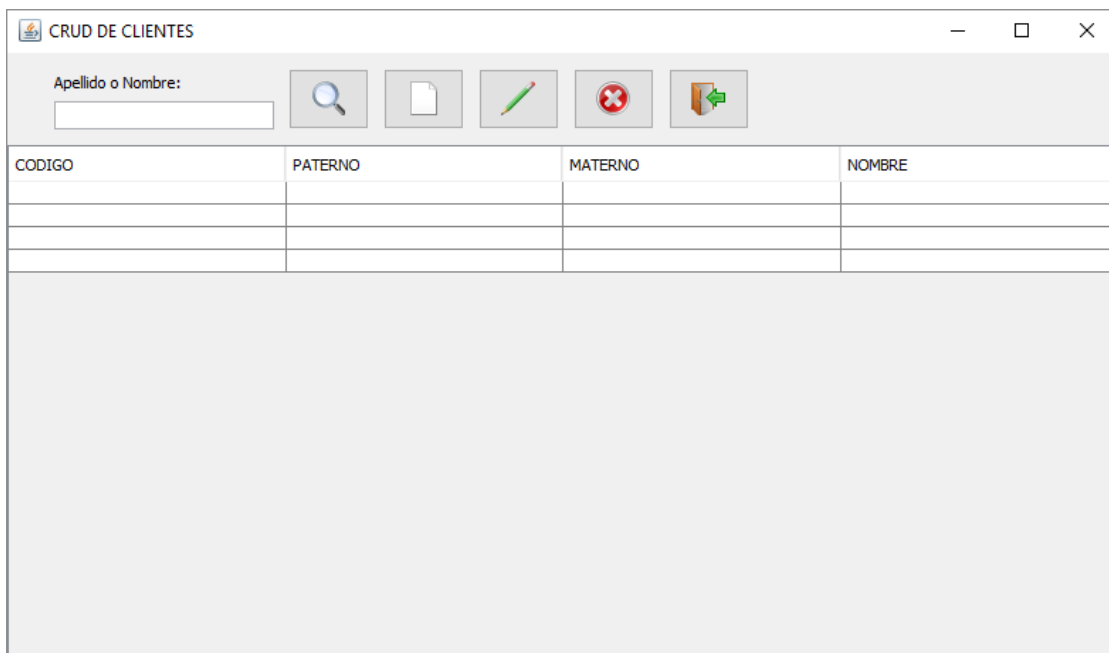
### Objetivo

Desarrollar el mantenimiento de una tabla.

### Proyecto






Desarrolle el mantenimiento de la tabla CLIENTE del esquema EUREKA.

A continuación se le sugiere una interfaz de usuario:



CRUD DE CLIENTES

Apellido o Nombre:

CODIGO	PATERNO	MATERNO	NOMBRE

## PROYECTO 04

### Objetivo

Programar procesos de negocio.

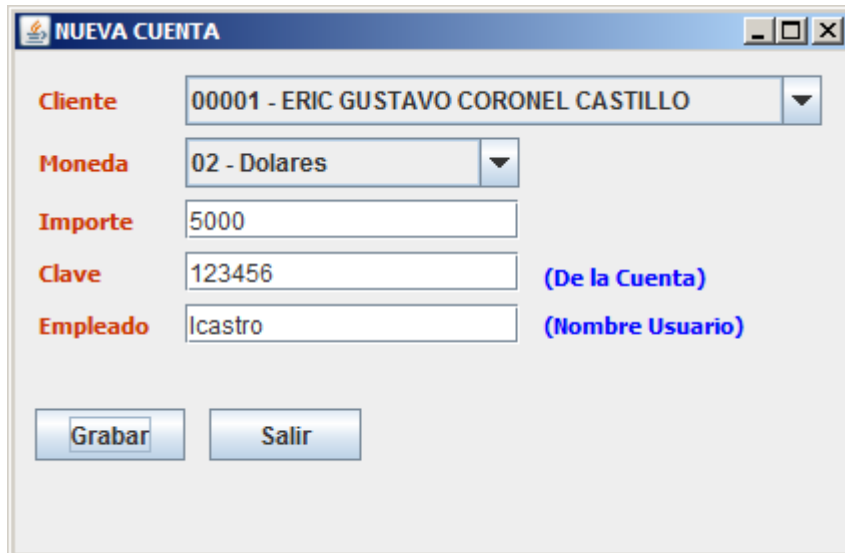
### Proyecto

Haciendo uso del esquema EUREKA, desarrolle una aplicación que permita crear una nueva cuenta.

El empleado que crea la cuenta debe ingresar su nombre de usuario y luego la aplicación debe solicitar su clave antes de realizar la transacción.

#### Paso 1: Ingreso de Datos

En esta interfaz se ingresan los datos de la nueva cuenta.



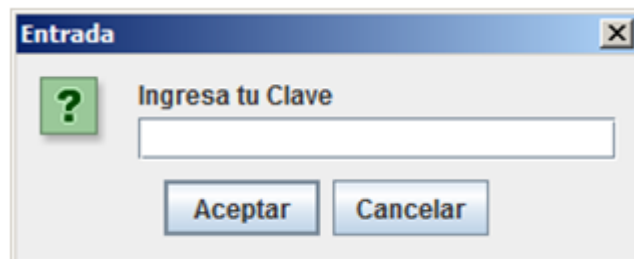
The screenshot shows a window titled "NUEVA CUENTA" with the following fields and values:

Label	Value	Notes
Cliente	00001 - ERIC GUSTAVO CORONEL CASTILLO	
Moneda	02 - Dolares	
Importe	5000	
Clave	123456	(De la Cuenta)
Empleado	lcastro	(Nombre Usuario)

At the bottom, there are two buttons: "Grabar" and "Salir".

#### Paso 2: Solicitar Clave

En esta interfaz se solicita la clave del empleados que está creando la cuenta.

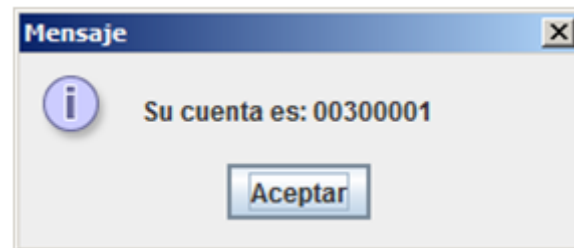


The screenshot shows a dialog box titled "Entrada" with a green question mark icon. The text "Ingresa tu Clave" is displayed above a text input field. Below the input field are two buttons: "Aceptar" and "Cancelar".

Si no es correcto debe mostrar un mensaje de error.

### Paso 3: Confirmar Transacción

Si la transacción se ejecuta correctamente debe mostrar el número de cuenta.



## PRACTICA 05

### Objetivo

- Aplicar CallableStatement para ejecutar procedimientos almacenados.
- Programar transacciones de base de datos.

### Proyecto

Haciendo uso del esquema EUREKA, desarrolle una aplicación que permita:

- Registrar un depósito
- Registrar un retiro
- Consultar los movimientos de una cuenta

El empleado debe primero iniciar sesión.

Todos los procesos se deben desarrollar utilizando procedimientos almacenados.

## PROYECTO 06

### Objetivo

Aplicar la librería JasperReport para construir reportes.

### Proyecto

Desarrollar una aplicación que permita obtener los siguientes reportes:

1. Reporte de cuentas por sucursal o todas las cuentas.

El usuario debe poder elegir la sucursal o todas las sucursales.

De cada cuenta debe mostrarse los siguientes datos:

- Número de cuenta
- Nombre del cliente
- Moneda
- Saldo
- Fecha de último movimiento

2. Reporte de movimientos de una cuenta

En la cabecera del reporte debe mostrar los siguientes datos:

- Número de cuenta
- Nombre del cliente
- Moneda
- Saldo
- Fecha de último movimiento

En el detalle del reporte debe mostrar los siguientes datos de cada movimiento:

- Número de movimiento
- Empleado que registró el movimiento
- Fecha de movimiento
- Tipo de movimiento
- Importe del movimiento



## PROYECTO 07

### Objetivo

Integración de todas las partes de una solución.

### Proyecto

En esta oportunidad debe integrar todas las partes desarrolladas de la solución y desarrollar las que aún no se han implementado.

Debe aplicar la programación en capas, patrones de software, principios SOLID, DRY, etc.

La solución debe contemplar los siguientes procesos:

- Inicio de sesión

La solución debe tener una ventana de inicio de sesión, a través de esta interfaz se debe identificar mediante su usuario y contraseña el empleado que quiera utilizar el sistema.

- Mantenimiento de tablas

El sistema debe permitir darle mantenimiento a las principales tablas del sistema: CLIENTE, SUCURSAL, MONEDA, PARAMETROS, etc.

- Procesos de negocio

El sistema debe implementar los principales procesos de negocio, como son:

- Creación de cuenta
- Registrar depósito
- Registrar retiro
- Registrar transferencia
- Cierre de cuenta

- Consultas y reportes

El sistema debe implementar las principales consultas y reportes que sean útiles a la para la buena gestión de la empresa.

- Procesos adicionales

Adicionalmente de permitir que el empleado pueda cambiar su contraseña.