



Technical Manual

About GraphingData

By Alan Gómez Mireles

20130820

Correo: alanyahir63@gmail.com

Contact: 8714153150

Graphing data from a database

Advanced programming topics

April 6th 2022

INDEX

Contenido

INTRODUCTION.....	2
OBJECTIVE	2
SYSTEM REQUIREMENTS	2
Software requirements	2
REQUERIMENTS	2
PROCESSES	2
3	
CLASSES.....	4
CÓDES.....	4
Suggestions	24
REFERENCES.....	24

INTRODUCTION

The following manual arises from the need to explain the structure and operation of a program for the generation of graphs from data obtained from a database.

OBJECTIVE

Inform and specify to the user the structure and conformation of the program so that they can make support, modifications and updates to the program in general.

SYSTEM REQUIREMENTS

Hardware requirements

- Equipment, keyboard, mouse, monitor
- Memoria RAM 2 GB (equipment and mobile device)
- LAN Network y/o Wireless.
- Processor 1.4 GHz.

Software requirements

- Operating system (Windows 7 onwards).
- Java 8.0.

REQUERIMENTS

- Have SQL Server 2019 installed (user: sa , password:12345678)
- Netbeans 8.2 or higher versions must be downloaded.
- Have downloaded the IDE (Java)
- Available memory space (15 MB minimum)

The Java programming language is an object-oriented development tool, it was designed to not rely on many implementations, which allows developers to run on any device without the need to recompile the code, which is considered cross-platform.

PROCESSES

Entry to the program

To enter the GraphingData program code, the first thing to do is to download the corresponding file. Usually the folder has a rar or zip format.

 20130820_GraphingDataAlanGM		06/04/2022 10:06 a. m.	Archivo WinRAR Z...	5,939 KB
---	---	------------------------	---------------------	----------

Once there, unzip the file and its corresponding folder, with the same name, in the breakdown of the folders, we will find some like build, dist, nbproject, src, store, and the technical and user manuals.

Nombre	Estado	Fecha de modificación
build		09/02/2022 02:47 a. m.
dist		09/02/2022 02:47 a. m.
nbproject		08/02/2022 11:12 p. m.
src		09/02/2022 02:46 a. m.
store		09/02/2022 02:40 a. m.
build		09/02/2022 02:40 a. m.
manifest.mf		08/02/2022 11:12 p. m.
Manual de usuario		09/02/2022 03:55 a. m.
Manual técnico		09/02/2022 04:00 a. m.

Of these folders, we will find that what interests us most is the src folder, where we will find the packages with the classes, and the store folder, where we will find the jar file (executable).

📁 Clases	🕒	05/04/2022 11:46 p. m.	Carpeta de archivos
📁 Frames	🕒	06/04/2022 01:50 a. m.	Carpeta de archivos

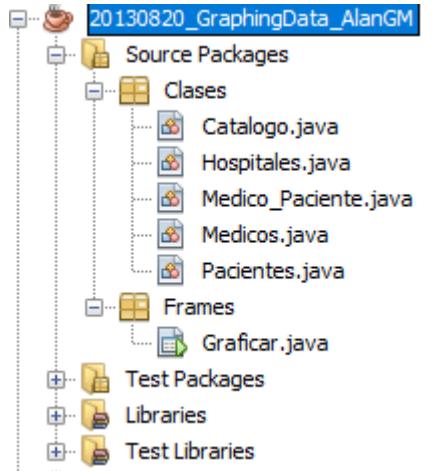
📄 Catalogo	🕒	06/04/2022 08:57 a. m.	Archivo JAVA	3 KB
📄 Hospitales	🕒	06/04/2022 08:57 a. m.	Archivo JAVA	3 KB
📄 Medico_Paciente	🕒	06/04/2022 08:57 a. m.	Archivo JAVA	3 KB
📄 Medicos	🕒	06/04/2022 08:57 a. m.	Archivo JAVA	4 KB
📄 Pacientes	🕒	06/04/2022 08:57 a. m.	Archivo JAVA	4 KB

📄 Graficar.form	🕒	06/04/2022 09:13 a. m.	Archivo FORM	37 KB
📄 Graficar	🕒	06/04/2022 09:13 a. m.	Archivo JAVA	31 KB

📁 lib	🕒	06/04/2022 09:18 a. m.	Carpeta de archivos
📅 20130820_GraphingDataAlanGM	🕒	06/04/2022 09:18 a. m.	Executable Jar File
📄 README	🕒	06/04/2022 09:18 a. m.	Documento de te...

CLASSES

The program consists of 5 classes, which refer to each table that we get from our SQL SERVER query. And on the other hand, our graphical interface that refers to our frame.



Program development

For the development of this program the following steps are performed:

1. Open Netbeans
2. Create a new Java Application type project.
3. Create the package of classes, each one named Classes, which contain Catalog, Hospitals, Patient_Physician, Doctors and Patients. On the side of the frame package, we have the Graph frame.
4. The mssql-jdbc-8.2.2.2 library for Netbeans 8.2 is required for the SQL SERVER connection to work.
5. For the operation of the graphs, it is necessary the libraries jfreechart-1.0.19 and the library jcommon-1.0.23.
6. Below is the coding of each class (CODES):

CÓDES

The code of the Catalog class is:

```
/*
 * Alan Yahir Japhet Gómez Mireles 20130820
 */
package Clases;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author alany
 */
//Clase catalogo
public class Catalogo {
    //Atributos
    private Connection conexion;

    //Conectar a la base de datos
    public Connection conectar()
    {
        //Modificar servidor, usuario y password
        String conexionURL = "jdbc:sqlserver://localhost:1433; database = ProyectoTopicosG; "
                            + "user =sa; password =12345678; loginTimeout = 30;"
```

```
31         try
32     {
33         conexion = (Connection) DriverManager.getConnection(conexionURL);
34         return conexion;
35     }
36     catch(SQLException e)
37     {
38         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
39         return null;
40     }
41 }
42
43 // Método de Desconexión
44 public void disconnect()
45 {
46     try
47     {
48         conexion.close();
49     }
50     catch (SQLException e)
51     {
52         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
53     }
54 }
55
56 // Método para Mostrar Datos del Catalogo
57 public void mostrarDatosCatalogo(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
```

```
56 // Método para Mostrar Datos del Catalogo
57 public void mostrarDatosCatalogo(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
61     modelo.setColumnCount(0);
62
63     conectar();
64
65     modelo=(DefaultTableModel)tabla.getModel();
66     modelo.addColumn("Id_cts");
67     modelo.addColumn("TipoSangre");
68     modelo.addColumn("PorcentajeRareza");
69
70     tabla.setModel(modelo);
71
72     String datos[] = new String[3];
73
74     try{
75         Statement st;
76         st=(Statement)conexion.createStatement();
77
78         ResultSet resSet;
79         resSet=st.executeQuery("SELECT * FROM Catalogo");
80
81         while(resSet.next())
82         {
83             datos[0]=resSet.getString(1);
84             datos[1]=resSet.getString(2);
```

```
74
75     try{
76         Statement st;
77         st=(Statement)conexion.createStatement();
78
79         ResultSet resSet;
80         resSet=st.executeQuery("SELECT * FROM Catalogo");
81
82         while(resSet.next())
83         {
84             datos[0]=resSet.getString(1);
85             datos[1]=resSet.getString(2);
86             datos[2]=resSet.getString(3);
87
88             modelo.addRow(datos);
89         }
90         tabla.setModel(modelo);
91     }
92     catch(SQLException e)
93     {
94         JOptionPane.showMessageDialog(null, "Error " + e.getMessage());
95     }
96     //Desconecta
97     disconnect();
98 }
```

The code of the Hospitals class:

```
1  /*  
2  * Alan Yahir Japhet Gómez Mireles 20130820  
3  */  
4  package Clases;  
5  
6  import java.sql.Connection;  
7  import java.sql.DriverManager;  
8  import java.sql.ResultSet;  
9  import java.sql.SQLException;  
10 import java.sql.Statement;  
11 import javax.swing.JOptionPane;  
12 import javax.swing.JTable;  
13 import javax.swing.table.DefaultTableModel;  
14  
15 /**  
16 *  
17 * @author alany  
18 */  
19 //Clase Hospitales  
20 public class Hospitales {  
21     //Atributos  
22     private Connection conexion;  
23  
24     //Conectar a la base de datos  
25     public Connection conectar()  
26     {  
27         //Modificar servidor, usuario y password  
28         String conexionURL = "jdbc:sqlserver://localhost:1433; database = ProyectoTopicosG; "  
29             + "user = sa; password = 12345678; loginTimeout = 30;";  
30  
31         try
```

```
31 |         try
32 |         {
33 |             conexion = (Connection) DriverManager.getConnection(conexionURL);
34 |             return conexion;
35 |         }
36 |         catch(SQLException e)
37 |         {
38 |             JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
39 |             return null;
40 |         }
41 |     }
42 |
43 |     // Método de Desconexión
44 |     public void disconnect()
45 |     {
46 |         try
47 |         {
48 |             conexion.close();
49 |         }
50 |         catch (SQLException e)
51 |         {
52 |             JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
53 |         }
54 |     }
55 |
56 |     // Método para Mostrar Datos de los hospitales
57 |     public void mostrarDatosHospitales(JTable tabla)
58 |     {
59 |         DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60 |         modelo.setRowCount(0);
61 |         modelo.setColumnCount(0);
```

```
63     conectar();
64
65     modelo=(DefaultTableModel)tabla.getModel();
66     modelo.addColumn("Id_hospital");
67     modelo.addColumn("NombreHospital");
68     modelo.addColumn("Dirección");
69     modelo.addColumn("AñosOperando");
70
71     tabla.setModel(modelo);
72
73     String datos[]={new String[4];
74
75     try{
76         Statement st;
77         st=(Statement)conexion.createStatement();
78
79         ResultSet resSet;
80         resSet=st.executeQuery("SELECT * FROM Hospitales");
81
82         while(resSet.next())
83         {
84             datos[0]=resSet.getString(1);
85             datos[1]=resSet.getString(2);
86             datos[2]=resSet.getString(3);
87             datos[3]=resSet.getString(4);
88
89             modelo.addRow(datos);
90         }
91         tabla.setModel(modelo);
```

```
82         while(resSet.next())
83     {
84         datos[0]=resSet.getString(1);
85         datos[1]=resSet.getString(2);
86         datos[2]=resSet.getString(3);
87         datos[3]=resSet.getString(4);
88
89         modelo.addRow(datos);
90     }
91     tabla.setModel(modelo);
92 }
93 catch(SQLException e)
94 {
95     JOptionPane.showMessageDialog(null, "Error " + e.getMessage());
96 }
97 //Desconectar
98 disconnect();
99 }
100 }
```

The code of the class Patient_Physician is:

```
1  */
2   * Alan Yahir Japhet Gómez Mireles 20130820
3   */
4   package Clases;
5
6   import java.sql.Connection;
7   import java.sql.DriverManager;
8   import java.sql.ResultSet;
9   import java.sql.SQLException;
10  import java.sql.Statement;
11  import javax.swing.JOptionPane;
12  import javax.swing.JTable;
13  import javax.swing.table.DefaultTableModel;
14
15  /**
16   *
17   * @author alany
18   */
19 //Clase Medico_Paciente (Relación)
20 public class Medico_Paciente {
21     //Atributos
22     private Connection conexion;
23
24     //Conectar a la base de datos
25     public Connection conectar()
26     {
27         //Modificar servidor, usuario y password
28         String conexionURL = "jdbc:sqlserver://localhost:1433; database = ProyectoTopicosG; "
29                     + "user = sa; password = 12345678; loginTimeout = 30;";
30
31         try
```

```
31     try
32     {
33         conexion = (Connection) DriverManager.getConnection(conexionURL);
34         return conexion;
35     }
36     catch(SQLException e)
37     {
38         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
39         return null;
40     }
41 }
42
43 // Método de Desconexión Autor
44 public void disconnect()
45 {
46     try
47     {
48         conexion.close();
49     }
50     catch (SQLException e)
51     {
52         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
53     }
54 }
55
56 // Método para Mostrar Datos de la relacion Medico_Paciente
57 public void mostrarDatosMed_Pac(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
61     modelo.setColumnCount(0);
```

```
56 // Método para Mostrar Datos de la relacion Medico_Paciente
57 public void mostrarDatosMed_Pac(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
61     modelo.setColumnCount(0);
62
63     conectar();
64
65     modelo=(DefaultTableModel)tabla.getModel();
66     modelo.addColumn("Id_medico");
67     modelo.addColumn("Id_paciente");
68
69     tabla.setModel(modelo);
70
71     String datos[] = new String[2];
72
73     try{
74         Statement st;
75         st=(Statement)conexion.createStatement();
76
77         ResultSet resSet;
78         resSet=st.executeQuery("SELECT * FROM Medico_Paciente");
79
80         while(resSet.next())
81         {
82             datos[0]=resSet.getString(1);
83             datos[1]=resSet.getString(2);
84
85             modelo.addRow(datos);
86         }
87     }
88 }
```

```
73
74     try{
75         Statement st;
76         st=(Statement)conexion.createStatement();
77
78         ResultSet resSet;
79         resSet=st.executeQuery("SELECT * FROM Medico_Paciente");
80
81         while(resSet.next())
82         {
83             datos[0]=resSet.getString(1);
84             datos[1]=resSet.getString(2);
85
86             modelo.addRow(datos);
87         }
88         tabla.setModel(modelo);
89     }
90     catch(SQLException e)
91     {
92         JOptionPane.showMessageDialog(null, "Error " + e.getMessage());
93     }
94     //Desconectar
95     disconnect();
96 }
```

The code of the class Doctors is:

```
1  /*
2  * Alan Yahir Japhet Gómez Mireles 20130820
3  */
4  package Clases;
5
6  import java.sql.Connection;
7  import java.sql.DriverManager;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import java.sql.Statement;
11 import javax.swing.JOptionPane;
12 import javax.swing.JTable;
13 import javax.swing.table.DefaultTableModel;
14
15 /**
16 *
17 * @author alany
18 */
19 //Clase Medicos
20 public class Medicos {
21     //Atributos
22     private Connection conexion;
23
24     //Conectar a la base de datos
25     public Connection conectar()
26     {
27         //Modificar servidor, usuario y password
28         String conexionURL = "jdbc:sqlserver://localhost:1433; database = ProyectoTopicosG; "
29                     + "user = sa; password = 12345678; loginTimeout = 30;";
30
31         try
```

```
31         try
32     {
33         conexion = (Connection) DriverManager.getConnection(conexionURL);
34         return conexion;
35     }
36     catch(SQLException e)
37     {
38         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
39         return null;
40     }
41 }
42
43 // Método de Desconexión
44 public void disconnect()
45 {
46     try
47     {
48         conexion.close();
49     }
50     catch (SQLException e)
51     {
52         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
53     }
54 }
55
56 // Método para Mostrar Datos de los Medicos
57 public void mostrarDatosMedicos(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
61     modelo.setColumnCount(0);
```

```
56 // Método para Mostrar Datos de los Medicos
57 public void mostrarDatosMedicos(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
61     modelo.setColumnCount(0);
62
63     conectar();
64
65     modelo=(DefaultTableModel)tabla.getModel();
66     modelo.addColumn("Id_medico");
67     modelo.addColumn("Nombre");
68     modelo.addColumn("ApPaternoM");
69     modelo.addColumn("ApMaternoM");
70     modelo.addColumn("Edad");
71     modelo.addColumn("AñosExperiencia");
72     modelo.addColumn("FamiliasAtiende");
73     modelo.addColumn("Id_hospital");
74
75     tabla.setModel(modelo);
76
77     String datos[]=new String[8];
78
79     try{
80         Statement st;
81         st=(Statement)conexion.createStatement();
82
83         ResultSet resSet;
84         resSet=st.executeQuery("SELECT * FROM Medicos");
```

```
79     try{
80         Statement st;
81         st=(Statement)conexion.createStatement();
82
83         ResultSet resSet;
84         resSet=st.executeQuery("SELECT * FROM Medicos");
85
86         while(resSet.next())
87         {
88             datos[0]=resSet.getString(1);
89             datos[1]=resSet.getString(2);
90             datos[2]=resSet.getString(3);
91             datos[3]=resSet.getString(4);
92             datos[4]=resSet.getString(5);
93             datos[5]=resSet.getString(6);
94             datos[6]=resSet.getString(7);
95             datos[7]=resSet.getString(8);
96
97             modelo.addRow(datos);
98         }
99         tabla.setModel(modelo);
100    }
101 catch(SQLException e)
102 {
103     JOptionPane.showMessageDialog(null, "Error " + e.getMessage());
104 }
105 //Desconectar
106 disconnect();
107 }
108 }
```

The code of the Patients class is:

```
1  /*
2  * Alan Yahir Japhet Gómez Mireles
3  */
4  package Clases;
5
6  import java.sql.Connection;
7  import java.sql.DriverManager;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import java.sql.Statement;
11 import javax.swing.JOptionPane;
12 import javax.swing.JTable;
13 import javax.swing.table.DefaultTableModel;
14
15 /**
16 *
17 * @author alany
18 */
19 //Clase Pacientes
20 public class Pacientes {
21     //Atributos
22     private Connection conexion;
23
24     //Conectar a la base de datos
25     public Connection conectar()
26     {
27         //Modificar servidor, usuario y contraseña
28         String conexionURL = "jdbc:sqlserver://localhost:1433; database = ProyectoTopicosG; "
29                     + "user = sa; password = 12345678; loginTimeout = 30;";
30
31         try
```

```
31     try
32     {
33         conexion = (Connection) DriverManager.getConnection(conexionURL);
34         return conexion;
35     }
36     catch(SQLException e)
37     {
38         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
39         return null;
40     }
41 }
42
43 // Método de Desconexión
44 public void disconnect()
45 {
46     try
47     {
48         conexion.close();
49     }
50     catch (SQLException e)
51     {
52         JOptionPane.showMessageDialog(null, "Error" + e.getMessage());
53     }
54 }
55
56 // Método para Mostrar Datos de los Pacientes
57 public void mostrarDatosPacientes(JTable tabla)
58 {
59     DefaultTableModel modelo = (DefaultTableModel)tabla.getModel();
60     modelo.setRowCount(0);
61     modelo.setColumnCount(0);
```

```
63     conectar();
64
65     modelo=(DefaultTableModel) tabla.getModel();
66     modelo.addColumn("Id_paciente");
67     modelo.addColumn("Nombre");
68     modelo.addColumn("ApPaterno");
69     modelo.addColumn("ApMaterno");
70     modelo.addColumn("Edad");
71     modelo.addColumn("Sexo");
72     modelo.addColumn("Telefono");
73     modelo.addColumn("Id_cts");
74     tabla.setModel(modelo);
75
76     String datos[]={new String[8];
77
78     try{
79         Statement st;
80         st=(Statement)conexion.createStatement();
81
82         ResultSet resSet;
83         resSet=st.executeQuery("SELECT * FROM Pacientes");
84
85         while(resSet.next())
86         {
87             datos[0]=resSet.getString(1);
88             datos[1]=resSet.getString(2);
89             datos[2]=resSet.getString(3);
90             datos[3]=resSet.getString(4);
91             datos[4]=resSet.getString(5);
```

```

85         while(resSet.next())
86     {
87         datos[0]=resSet.getString(1);
88         datos[1]=resSet.getString(2);
89         datos[2]=resSet.getString(3);
90         datos[3]=resSet.getString(4);
91         datos[4]=resSet.getString(5);
92         datos[5]=resSet.getString(6);
93         datos[6]=resSet.getString(7);
94         datos[7]=resSet.getString(8);
95
96         modelo.addRow(datos);
97     }
98     tabla.setModel(modelo);
99 }
100 catch(SQLException e)
101 {
102     JOptionPane.showMessageDialog(null, "Error " + e.getMessage());
103 }
104 //Desconectar
105 disconnect();
106 }
107

```

All the codes shown above can be accessed from the rar file. This same project can also be imported from java for modification and use,

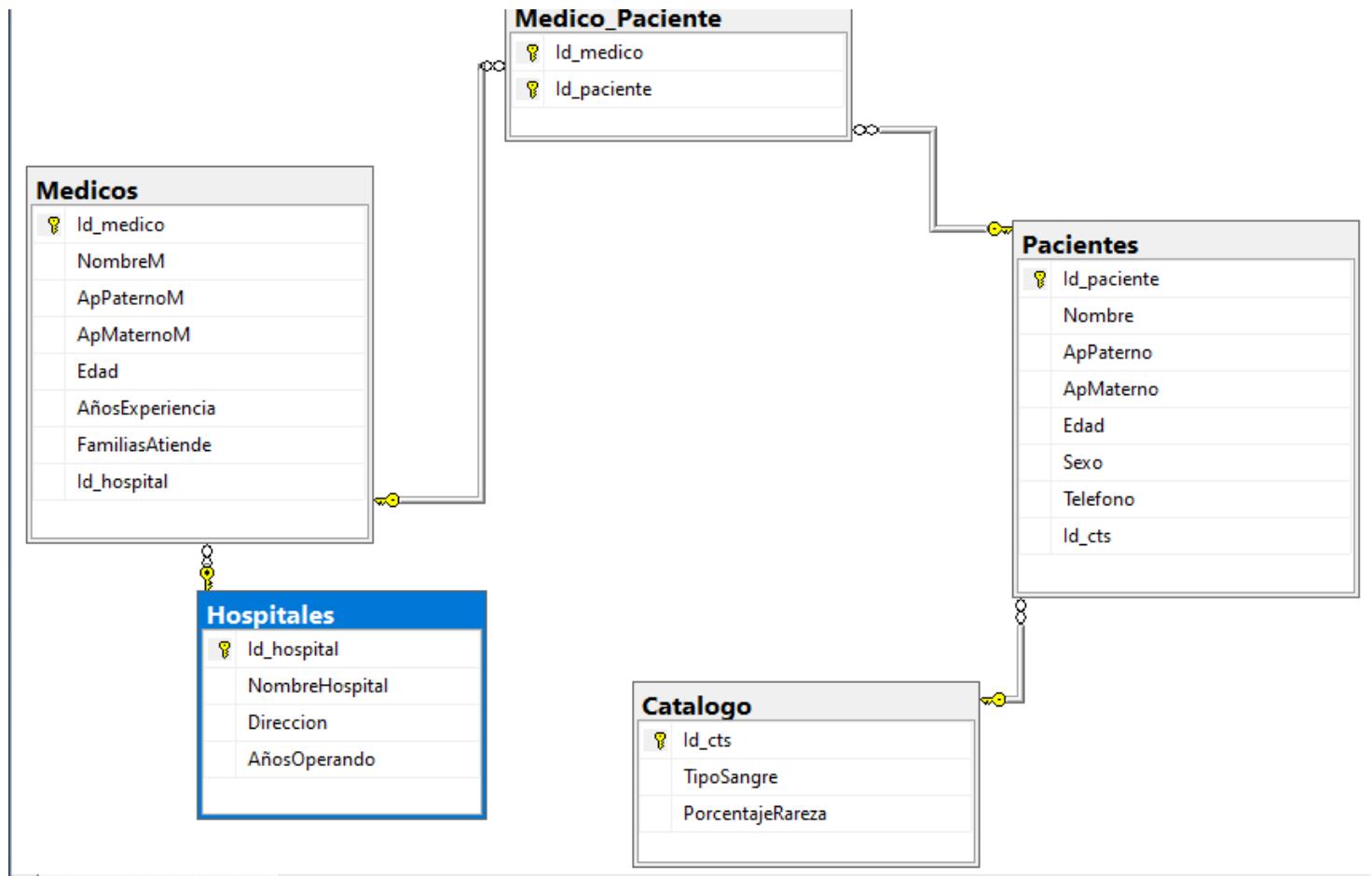
It can occur the error that the libraries are not imported, therefore, it is necessary to import them directly, giving right click in libraries and add.

Suggestions

Some suggestions that we can give for the correct operation of the program are:

- Configure correctly the SQL SERVER profile, this has to have the general profile sa, and having a static password, we will have to use the password:12345678, in the localhost server.
- Make sure that our jar program is correctly downloaded.
- To print, it is recommended that the position is horizontal, and also use legal size paper.
- Install correctly the libraries for the connection to sql, the jfreechart and the jcommon, they can be accessed from the dist folder.

Class Diagram



REFERENCES

The following are some websites that were used as resources for the development of this program; these examples were modified in order to make a more complete program.

- JFreeChart. (s. f.). JFreeChart. Recuperado 6 de abril de 2022, de <https://www.jfree.org/jfreechart/>
- P. (2017, 22 octubre). Visualizar datos y generar gráficas en Java con JFreeChart. Blog Bitix. Recuperado 6 de abril de 2022, de <https://picodotdev.github.io/blog-bitix/2017/10/visualizar-datos-y-generar-graficas-en-java-con-jfreechart/>