Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Sistema para la realización del proceso de limpieza de datos

Modalidad: Proyecto Tecnológico

Reporte Final

Trimestre 2017 Invierno

Alumno:

Emmanuel Vega Callejas

Matrícula: 207204584

al207204584@alumnos.azc.uam.mx

Asesores:

M. en C. María de Lourdes Sánchez Guerrero
Profesora Titular
Departamento de Sistemas
lsg@correo.azc.uam.mx

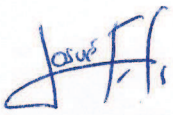
M. en C. Josué Figueroa González
Profesor Asociado
Departamento de Sistemas
jfgo@correo.azc.uam.mx

26 de abril de 2017

Declaratoria

Nosotros, María de Lourdes Sánchez guerrero y Josué Figueroa González, declaramos que aprobamos el contenido del presente Reporte de Proyecto de Integración y damos nuestra autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

María de Lourdes Sánchez Guerrero
Firma de la Asesora



Josué Figueroa González
Firma del Asesor

Yo, Emmanuel Vega Callejas, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Emmanuel Vega Callejas
Firma del alumno

Resumen

En los últimos años la tecnología se ha vuelto esencial dentro de la vida cotidiana, lo cual provoca en diversas ocasiones que se generen grandes masas de información que muchas veces no tienen sentido, uno de esos casos lo tenemos en la aplicación de encuestas, al guardar los datos muchas veces estos carecen de sentido a simple vista lo cual hace complicado el estudio y tratamiento de la información.

El problema radica en que al intentar procesar tal información dicho proceso no es tan sencillo y rápido como se desearía, esto genera una necesidad de crear nuevas herramientas y teorías computacionales que nos ayuden a darle sentido a la información a partir de enormes volúmenes de datos (en este caso generados por encuestas).

Para darle solución a este problema, se desarrolló una aplicación que nos ayuda a darle sentido a toda esa información mediante la creación y aplicación de reglas, las cuales deben ser creadas en base a la información registrada así como de los requerimientos y mediante las cuales dicha información va tomando sentido.

La aplicación consiste básicamente en 3 etapas:

1. **Selección de datos.**

En esta etapa se determinan las fuentes de datos y el tipo de información a utilizar. Se cargan los datos.

2. **Limpieza y Preprocesamiento de datos.**

Esta etapa consiste en la preparación de los datos y en la creación de las reglas para posteriormente ser aplicadas.

3. **Transformación.**

Consiste en el tratamiento preliminar de los datos, se aplican las reglas creadas para transformar la información. Se obtiene el resultado final.

Tabla de contenido

Introducción

En la actualidad existe la urgente necesidad de crear nuevas teorías y herramientas computacionales que ayude a las personas a extraer información útil a partir de enormes volúmenes de datos, estos datos se han ido acumulando con el paso del tiempo y crecen de manera exponencial por lo cual se necesitan estas herramientas para darle sentido a toda esa información.

El término Knowledge Discovery in Databases (Descubrimiento de conocimiento en bases de datos "KDD") se refiere al proceso de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información [1].

Este proceso consta de 5 fases: Selección, Preprocesamiento, Transformación, Minería de datos e Interpretación. Las fases que se contemplan para la realización de este proyecto son las siguientes:

1. **Selección de datos.**

En esta etapa se determinan las fuentes de datos y el tipo de información a utilizar. Es la etapa donde los datos relevantes para el análisis son extraídos desde la o las fuentes de datos.

2. **Limpieza y Preprocesamiento de datos.**

Esta etapa consiste en la preparación y limpieza de los datos extraídos desde las distintas fuentes en una forma manejable. En esta etapa se utilizan diversas estrategias para manejar datos faltantes o en blanco, datos inconsistentes o que están fuera de rango, obteniéndose al final una estructura de datos adecuada para su posterior transformación.

3. **Transformación.**

Consiste en el tratamiento preliminar de los datos, transformación y generación de nuevas variables a partir de las ya existentes con una estructura de datos apropiada. Aquí se realizan operaciones de agregación o normalización, consolidando los datos de la mejor manera para su uso posterior.

Antecedentes

Proyectos de Integración o Terminales

Correlacionador de Bitácoras de equipos en una LAN [2]

Los dos proyectos son muy diferentes y se aplican en áreas muy distintas pero tienen una similitud y es que ambos trabajan con cantidades muy grandes de datos buscando información relevante una aplicándolo a redes para tener mayor seguridad y la otra para darle sentido a datos que a simple vista no lo tienen.

Aplicación de Distintas Técnicas de Minería de Datos para el Tratamiento de Información [3]

Ambos proyectos son muy similares ya que los dos trabajan con grandes masas de información y realizan limpieza de datos uno mediante algoritmos de minería de datos y el propio es un sistema para automatizar la limpieza de información sin considerar la minería de datos.

Sistema de Recuperación de Información Semántico [4]

Este proyecto tiene como objetivo explorar y aplicar técnicas que permitan realizar búsquedas más exactas mediante reglas las cuales indagan en una infinidad de resultados mostrando solo los más importantes, esto es lo que le da parecido al proyecto aquí desarrollado ya que también se tiene una gran cantidad de información de la cual se busca solo la información más relevante.

Externos

Data Quality Services (DQS) [5]

Ambos servicios ofrecen una opción para realizar la limpieza de datos de forma más eficiente por medio de la creación de categorías para procesar los datos de manera más sencilla y rápida.

Predicting Student Performance: A Statistical and Data Mining Approach [6].

Este proyecto estudia los factores que influyen en el funcionamiento de los estudiantes mediante la aplicación de encuestas con las cuales se recaba una gran cantidad de información que es difícil de digerir y entender, la idea principal es categorizar cada posible respuesta y crear rangos con su respectivo dominio de valores para facilitar su estudio.
En el proyecto que se desarrollará se pretende crear un modelo similar de limpieza de datos haciendo uso de categorías y dominios para cada posible respuesta dependiendo del tipo de pregunta.

Preventing Student Dropout in Distance Learning Using Machine Learning Techniques [7].

La recolección de datos de los estudiantes a menudo es costosa y consume tiempo, ambos proyectos tienen como objetivo reducir estos costos y tiempos dándole más sentido a los datos creando grupos de valores y atributos para hacer más sencillo su manejo y su comprensión.

Justificación

Es evidente la necesidad de que existan herramientas que nos ayuden a extraer de forma confiable y de manera exacta información relevante de grandes cantidades de datos, la importancia de la limpieza de la información radica en esa necesidad de tener datos más concretos y exactos de forma fácil y rápida. Este sistema fue hecho para realizar este proceso de manera más eficiente, se buscó crear una aplicación en la que es posible crear categorías, convertir valores continuos a discretos, agrupar categorías, todo con una interfaz gráfica sencilla de utilizar.

Objetivos

Objetivo General:

Diseñar e implementar un sistema que permita realizar el proceso de limpieza de datos de manera eficiente.

Objetivos Específicos:

- Diseñar e implementar un módulo para cargar los diferentes tipos de archivos a procesar.
- Diseñar e implementar un módulo que nos permitirá desplegar los datos.
- Diseñar e implementar un módulo que permitirá registrar las reglas que se aplicaran a los datos.
- Diseñar e implementar un módulo que permitirá aplicar las reglas a los datos y generará un archivo de salida.

Marco teórico

KDD: Proceso de Extracción de conocimiento *[9]*

La Extracción de conocimiento está principalmente relacionado con el proceso de descubrimiento conocido como Knowledge Discovery in Databases (KDD), que se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información. No es un proceso automático, es un proceso iterativo que exhaustivamente explora volúmenes muy grandes de datos para determinar relaciones. Es un proceso que extrae información de calidad que puede usarse para dibujar conclusiones basadas en relaciones o modelos dentro de los datos.

La ***Figura 1*** ilustra las etapas del proceso KDD:

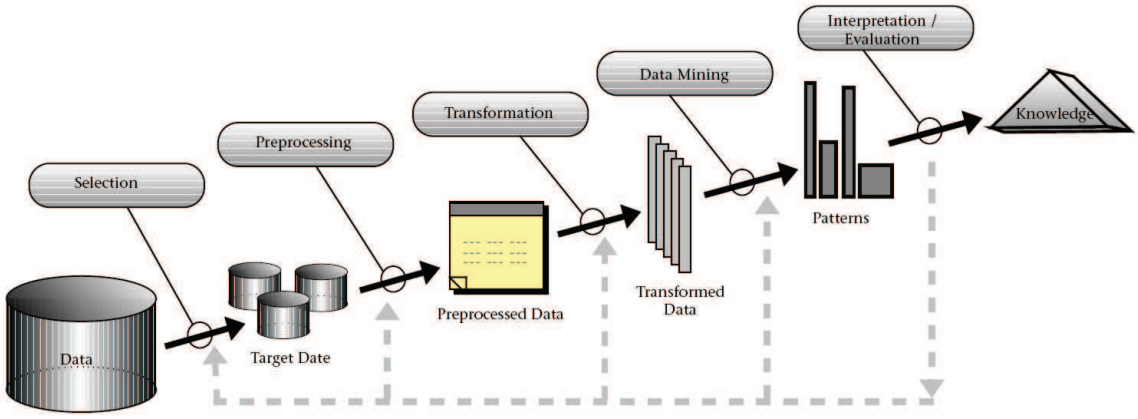


Figura 1. Etapas del proceso KDD

Proceso KDD

Como muestra la figura anterior, las etapas del proceso KDD se dividen en 5 fases y son:

1. **Selección de datos**. En esta etapa se determinan las fuentes de datos y el tipo de información a utilizar.
2. **Preprocesamiento**. Esta etapa consiste en la preparación y limpieza de los datos extraídos desde las distintas fuentes de datos en una forma manejable, necesaria para las fases posteriores.
3. **Transformación**. Consiste en el tratamiento preliminar de los datos, transformación y generación de nuevas variables a partir de las ya existentes con una estructura de datos apropiada.
4. **Data Mining**. Es la fase de modelamiento, en donde métodos inteligentes son aplicados con el objetivo de extraer patrones previamente desconocidos, válidos, nuevos, potencialmente útiles y comprensibles y que están contenidos u "ocultos" en los datos.
5. **Interpretación y Evaluación**. Se identifican los patrones obtenidos y que son realmente interesantes, basándose en algunas medidas y se realiza una evaluación de los resultados obtenidos.

Cabe mencionar que en este proyecto no son contempladas todas etapas antes mencionadas, las etapas que se usaron para la realización del proyecto fueron:

1. **Selección de datos, 2. Preprocesamiento y 3. Transformación.**

Limpieza de datos

La limpieza de datos es el proceso mediante el cual se analiza la calidad de los datos. Este proceso permite identificar datos incompletos, incorrectos, inexactos, etc. Y luego sustituir, modificar o eliminar estos datos sucios.

El objetivo final de cualquier acción de limpieza de datos es mejorar la confianza de la organización de sus datos. Para llevar a cabo una acción de limpieza de datos es necesario seguir las fases mostradas en la ***Figura 2***:

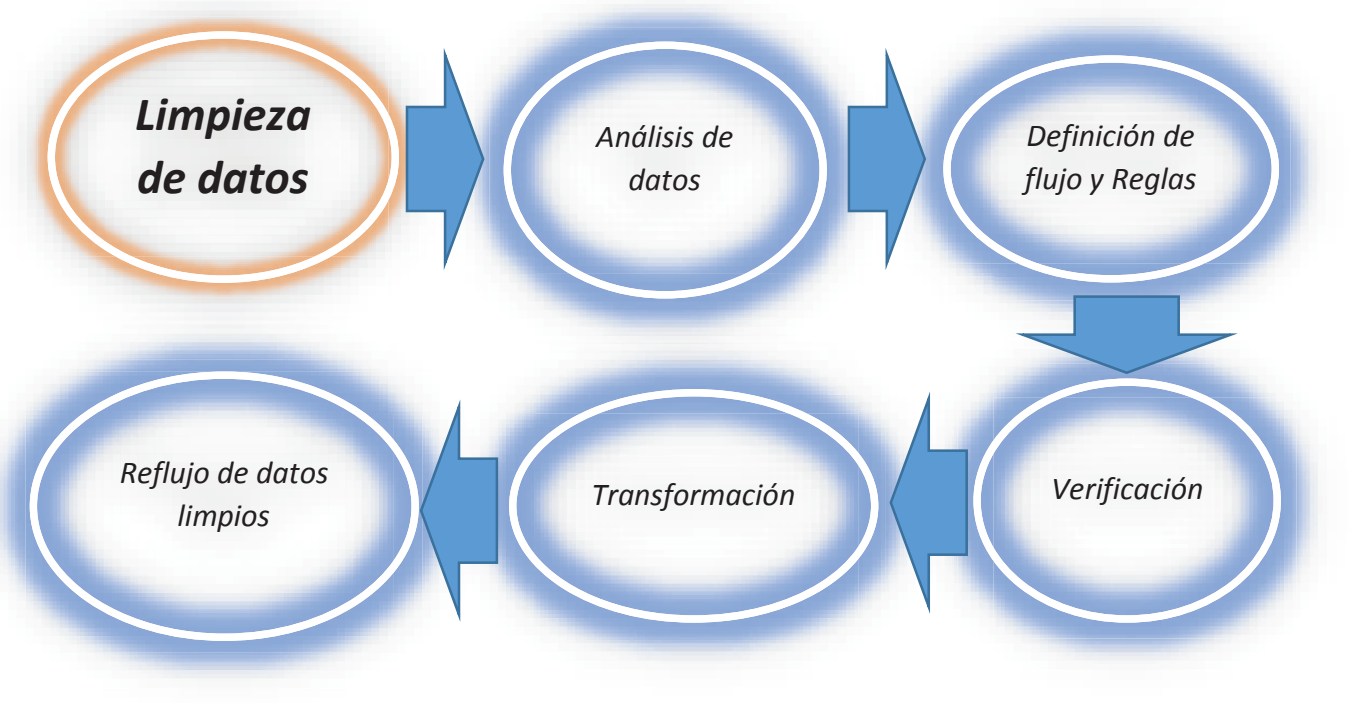


Figura 2. Fases necesarias para una limpieza de datos exhaustiva.

A continuación se describe cada una de las fases *[8]*:

1. Análisis de datos: su misión es determinar qué tipo de errores e inconsistencias deben ser eliminados. Además de una inspección manual de las muestras de datos, es necesaria la automatización, en otras palabras, la incorporación de programas que actúen sobre los metadatos para detectar problemas de calidad de datos que afecten a sus propiedades.

2. Definición del flujo de transformación y reglas de mapeo: dependiendo del número de fuentes de origen de datos, su heterogeneidad y la previsión de problemas de calidad de los datos, será necesario ejecutar más o menos pasos en la etapa de transformación y adecuación.
3. Verificación: el nivel de adecuación y la efectividad de una acción de transformación debe siempre ser testado y evaluado; uno de los principios de la limpieza de datos.
4. Transformación: consiste en proceder a ejecutar determinadas reglas de limpieza y actualización de los datos.
5. Reflujo de datos limpios: una vez se han eliminado los errores de calidad, los datos "limpios" deben reemplazar a los que no lo están en las fuentes originales.

Desarrollo del proyecto

La aplicación requiere de dos Bases de Datos.

1.- proyecto: que es donde se genera un modelado de los datos que son cargados desde el archivo que contiene la información.

2.- registro: aquí se almacena el modelado de las reglas que se van creando, dependiendo el tipo de regla esta es almacenada en su respectiva tabla.

En la ***Figura 3*** se muestra el diagrama entidad relación de las tablas en las bases de datos.

Figura 3. Diagrama Entidad-Relación

En un principio no se han cargado datos en la base de datos “proyecto” por lo que el modelo central no existe, esta se crea en el módulo de Carga el cual se explicara más adelante. Tanto la tabla de reglas_directo como la de reglas_por_rango comienzan sin datos, únicamente se definen las columnas como se muestran en el diagrama anterior y se van llenando conforme se guardan las reglas en el módulo “Reglas” que será detallado posteriormente, estas dos tablas guardan el nombre de la tabla y la columna con la cual se trabajara, los valores originales y la regla que se aplicara para cada valor o un rango de estos, es posible guardar tantas reglas como se desee y de diferentes tablas.

La aplicación consta de los siguientes módulos.

- Carga
- Reglas
- Procesar

En la ***Figura 4*** se muestra la secuencia de actividades desarrollados para el proyecto.

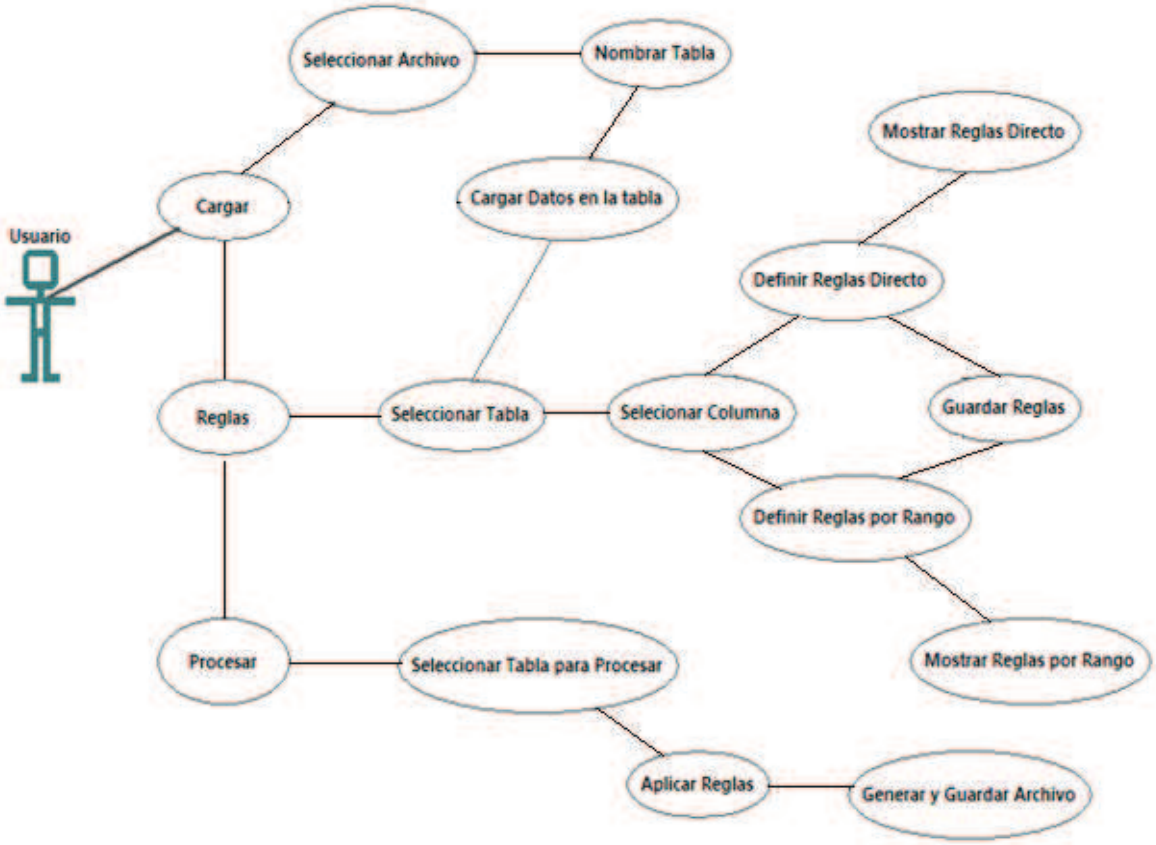


Figura 4. Diagrama de secuencia de actividades

A continuación se describe el desarrollo de cada uno de los módulos.

Carga

En el módulo de carga se selecciona el archivo que contiene los datos y se cargan en la Base de Datos “proyecto” creando una nueva tabla que es nombrada aquí mismo, esta tabla se genera desde cero y el número de columnas así como el número de datos depende del tamaño del archivo seleccionado. En la ***Figura 5*** se muestra este módulo.

Figura 5. Pantalla de Carga del archivo y nombramiento de la Tabla

Se crea la tabla con la que se va a trabajar, para ello se debe seleccionar el archivo que contiene los datos, posteriormente se debe especificar un nombre para la tabla que se creara y se guarda en la base de datos “proyecto” en la cual no existe ninguna tabla en un principio. Los nombres de los campos o columnas son desplegados.

Reglas

En el módulo que se muestra en la ***Figura 6*** se selecciona tanto la tabla (que son cargadas desde la Base de Datos "proyecto"), como la columna con la cual se establecerán las reglas, aquí también se crean y guardan dichas reglas con su respectiva tabla y columna a la que pertenecen, dependiendo el tipo de regla esta es guardada en la tabla correspondiente. Aquí manejamos dos tipos de reglas: reglas directo y por rango cada una con su respectiva tabla.

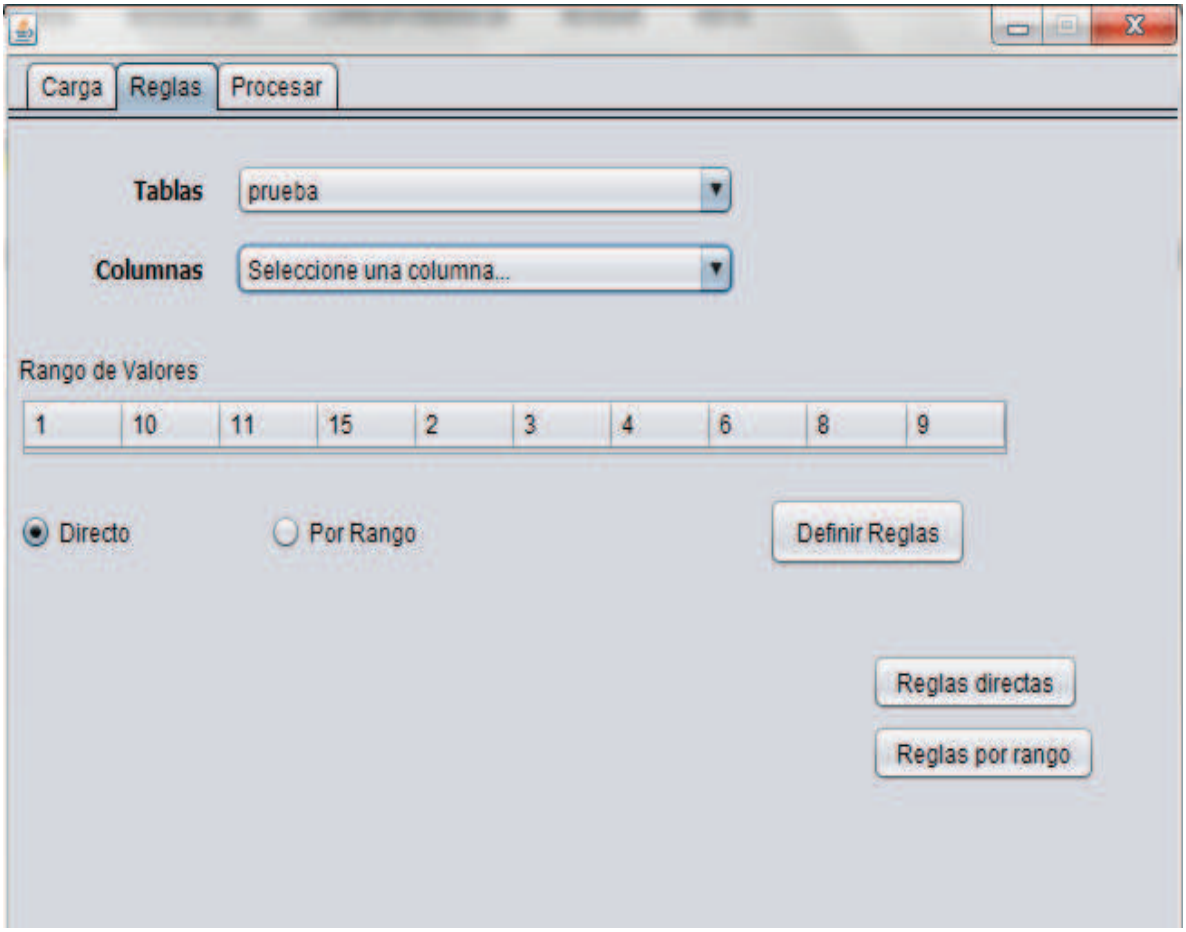


Figura 6. Pantalla de Reglas se selecciona tabla y columna.

Después de seleccionar una tabla y una columna que contenga esa tabla, automáticamente mostrara el rango de valores que existan en esa columna, con esto se puede decidir si se quiere trabajar con ella y de qué modo se quiere crear las reglas, los dos tipos de reglas son:

- Directo.- Los valores son cambiados directamente, por ejemplo: 1 → si, 2 → no, etc.

- Por Rango.- Se crean rangos y se asigna un solo valor para cada rango, por ejemplo: del 1 al 5 → nada, del 6 al 10 → mucho, etc.

Procesar

En este módulo lo primero que se debe hacer es seleccionar la tabla a la que se le aplicaran las reglas (esto se hace porque se puede trabajar con varias tablas a la vez), una vez seleccionada la tabla se aplican las reglas y se guarda el resultado en la ruta que también es posible especificar. La ***Figura 7*** muestra este módulo.

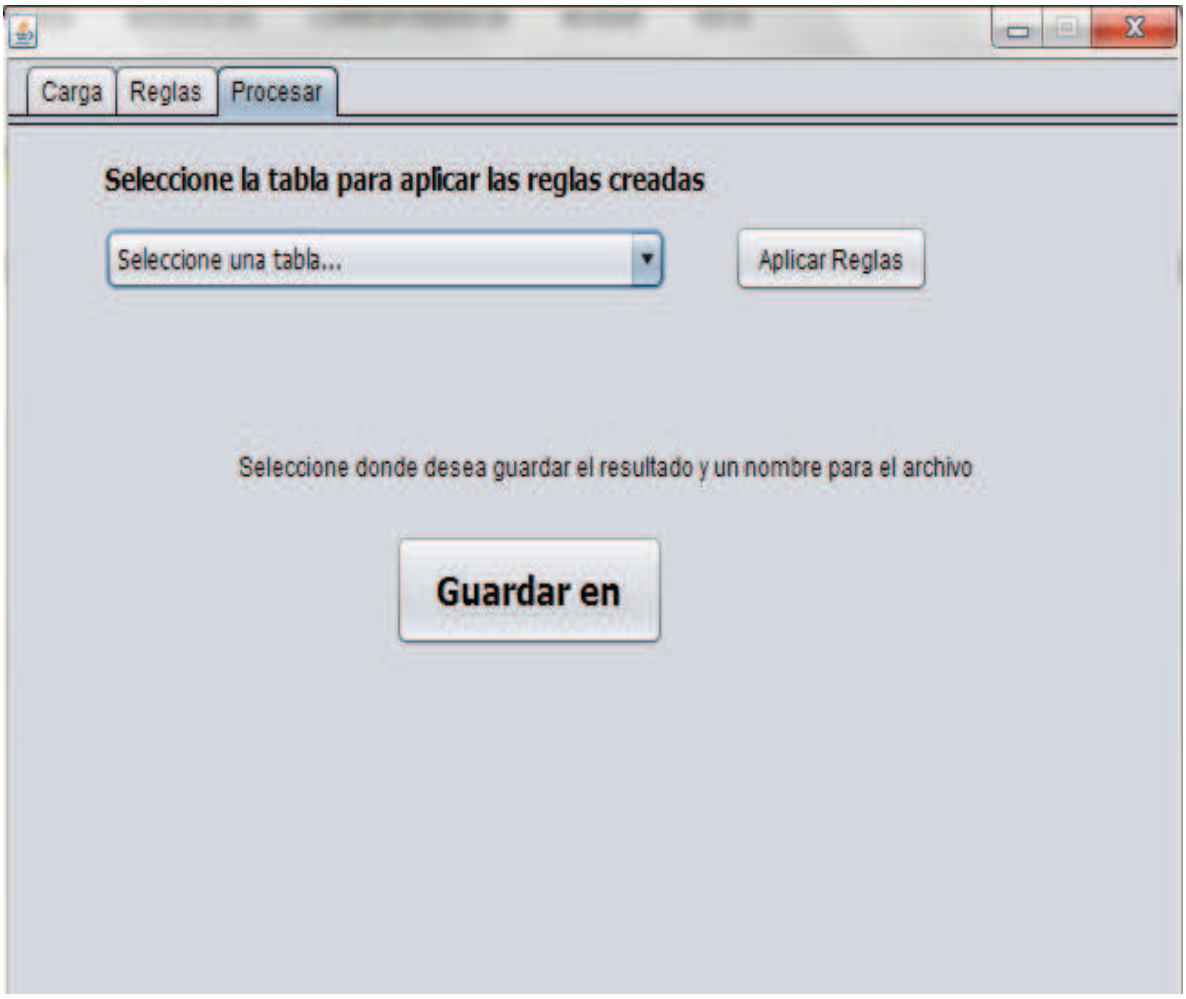


Figura 7. Interfaz gráfica del módulo procesar, aplicar reglas y guardar resultado.

Una vez seleccionada la tabla se crea un respaldo de esta para ser editada con los nuevos valores, al aplicar las reglas los valores son actualizados y al guardar el resultado se nos pide especificar donde se desea guardar con esto se concluye y se crea el archivo de salida con el nombre dado.

Resultados

Para las pruebas se utilizó un archivo con extensión .CSV con datos similares a lo que se pretende que procese la aplicación. El archivo contiene solo una pequeña muestra con la finalidad de poder hacer más concreto su estudio en este reporte, aunque cabe mencionar que el sistema tiene la capacidad de trabajar con archivos con una cantidad de datos mucho mayor.

Para estas pruebas fue necesaria una conexión con la Base de Datos mediante JDBC y la existencia de dos Bases de Datos una llamada “proyecto” que es donde se guarda el modelado de los datos que son cargados desde el archivo y la otra llamada “registro” que contiene las dos tablas donde se guardan las reglas. Estas dos tablas deben existir y se crearon con las siguientes sentencias desde MYSQL.

reglas_directo: CREATE TABLE reglas_directo (id int not null primary key auto_increment, tabla varchar(20), columna varchar(40), valor varchar(20), regla varchar(20));

reglas_por_rango: CREATE TABLE reglas_por_rango (id int not null primary key auto_increment, tabla varchar(20), columna varchar(40), valor_inicial varchar(20), valor_final varchar(20), regla varchar(20));

Después de realizar varias pruebas, los resultados que se obtuvieron al finalizar la aplicación se describen a continuación.

El diagrama que se muestra en la ***Figura 8*** representa el sistema que se desarrolló.

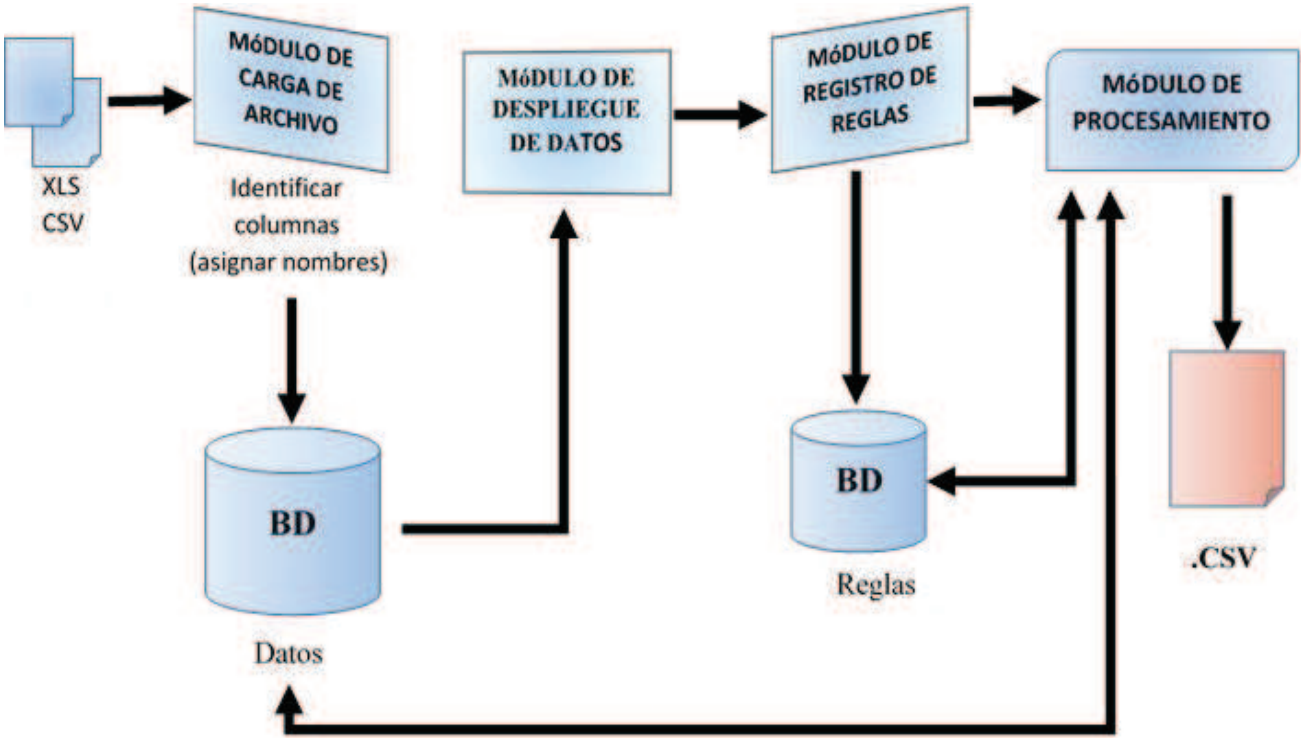


Figura 8. Representación gráfica del sistema.

A continuación se describe cada módulo en base a los resultados.

Módulo de carga de archivo.

Se logró que la aplicación cargara un archivo con la información a la Base de Datos identificando los nombres de las columnas y desplegándolos en una tabla.

En la ***Figura 9*** se muestra el resultado de cargar el archivo "prueba" a la Base de Datos, se puede observar como despliega el nombre de las columnas, esto es parte del módulo de despliegue de datos.

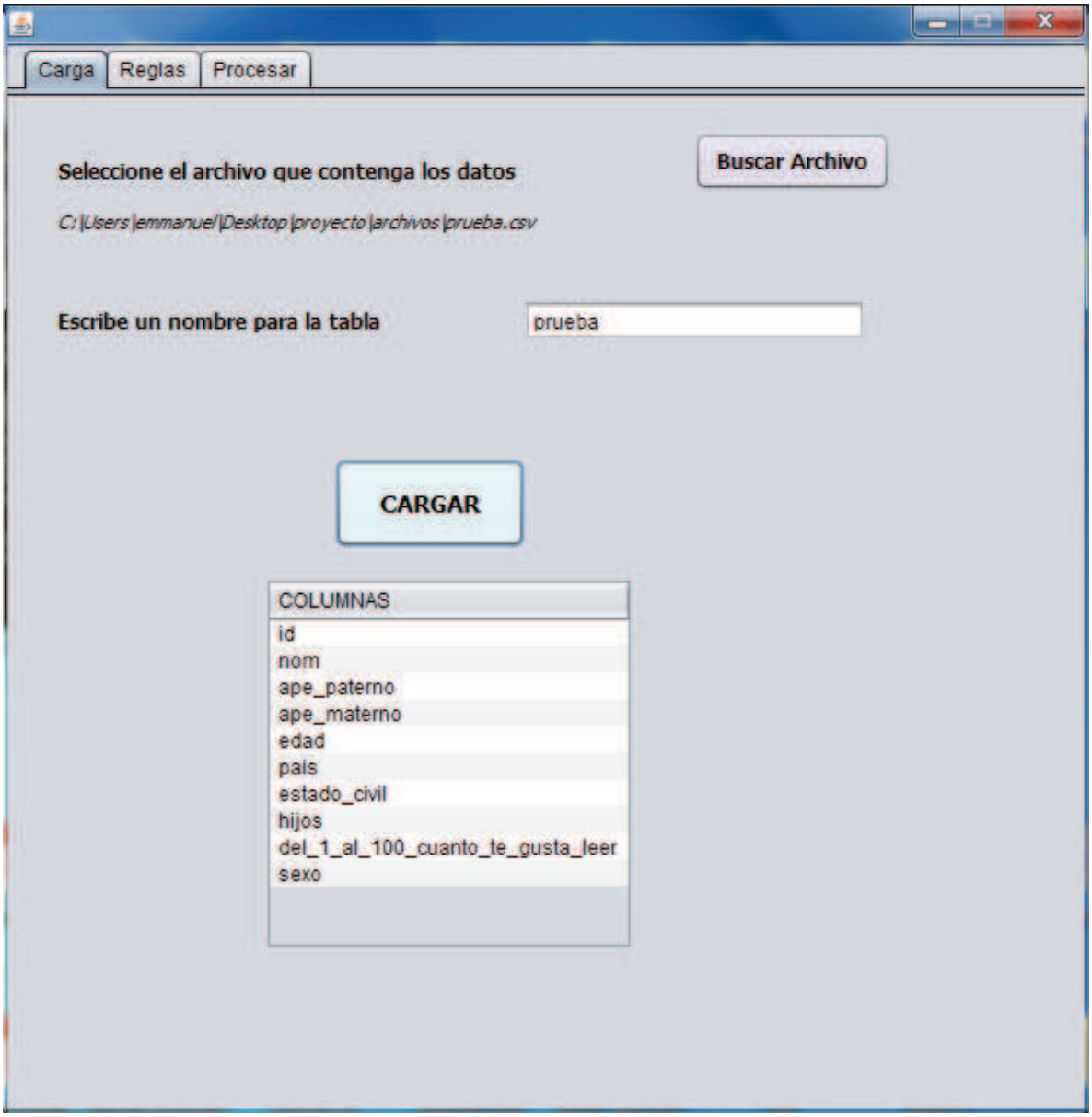


Figura 9. Interfaz de carga de archivo.

Una vez cargado el archivo en la Base de Batos la información está lista para ser procesada y podemos pasar al siguiente módulo.

Módulo de registro de reglas.

Después de que se generó el modelado de la información en la base de datos todo está listo para la creación de las reglas como se muestra en la ***Figura 10 y 11***. Lo primero que se hizo fue seleccionar una tabla para posteriormente seleccionar una columna de dicha tabla, al hacer esto automáticamente se despliegan el rango de valores de la columna seleccionada para decidir si se quiere crear alguna regla para esos datos. Se logró que guardara todas las reglas que fueron hechas tanto "directas" como "por rango".

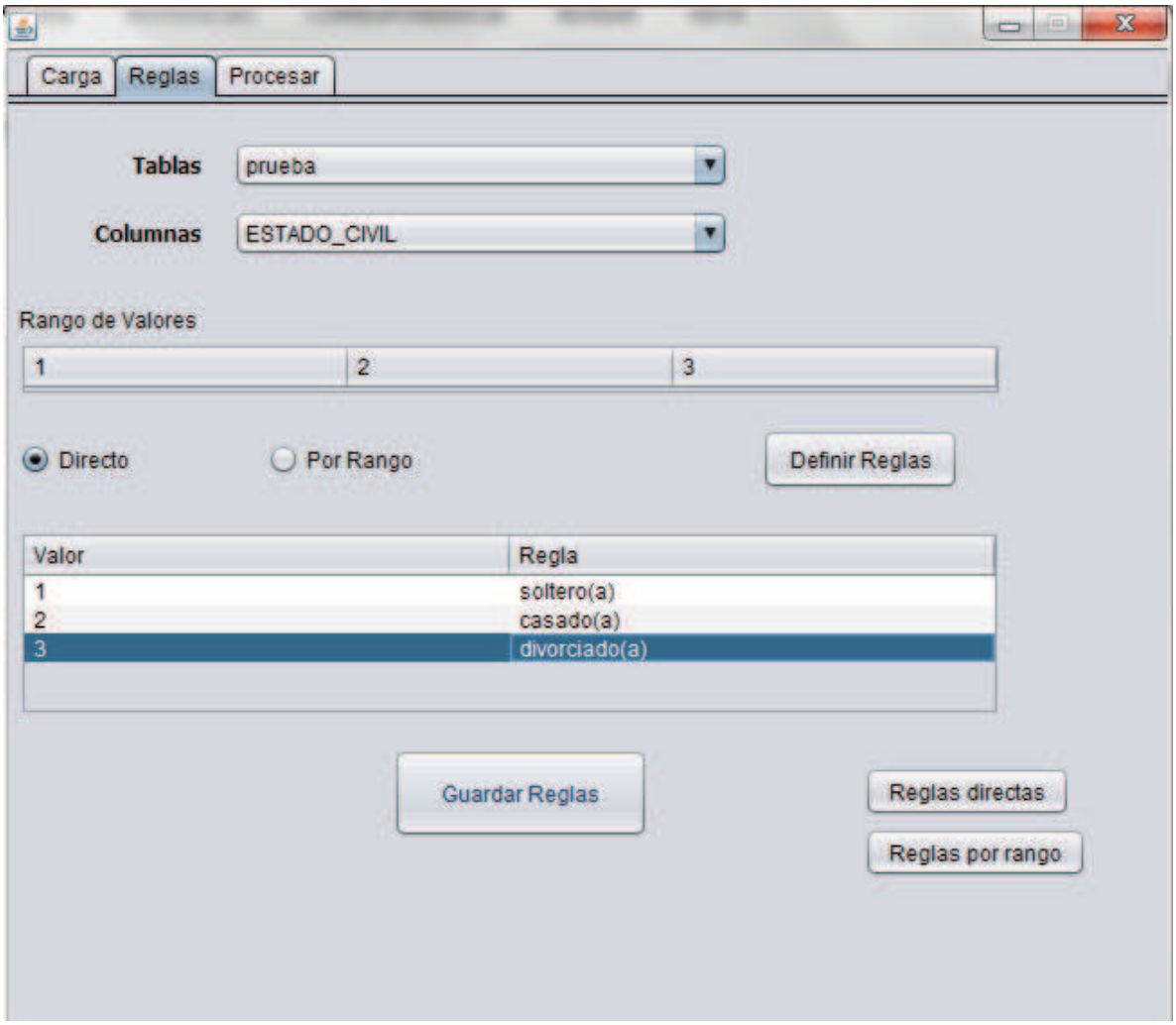


Figura 10. Interfaz Reglas. Guardar reglas directo

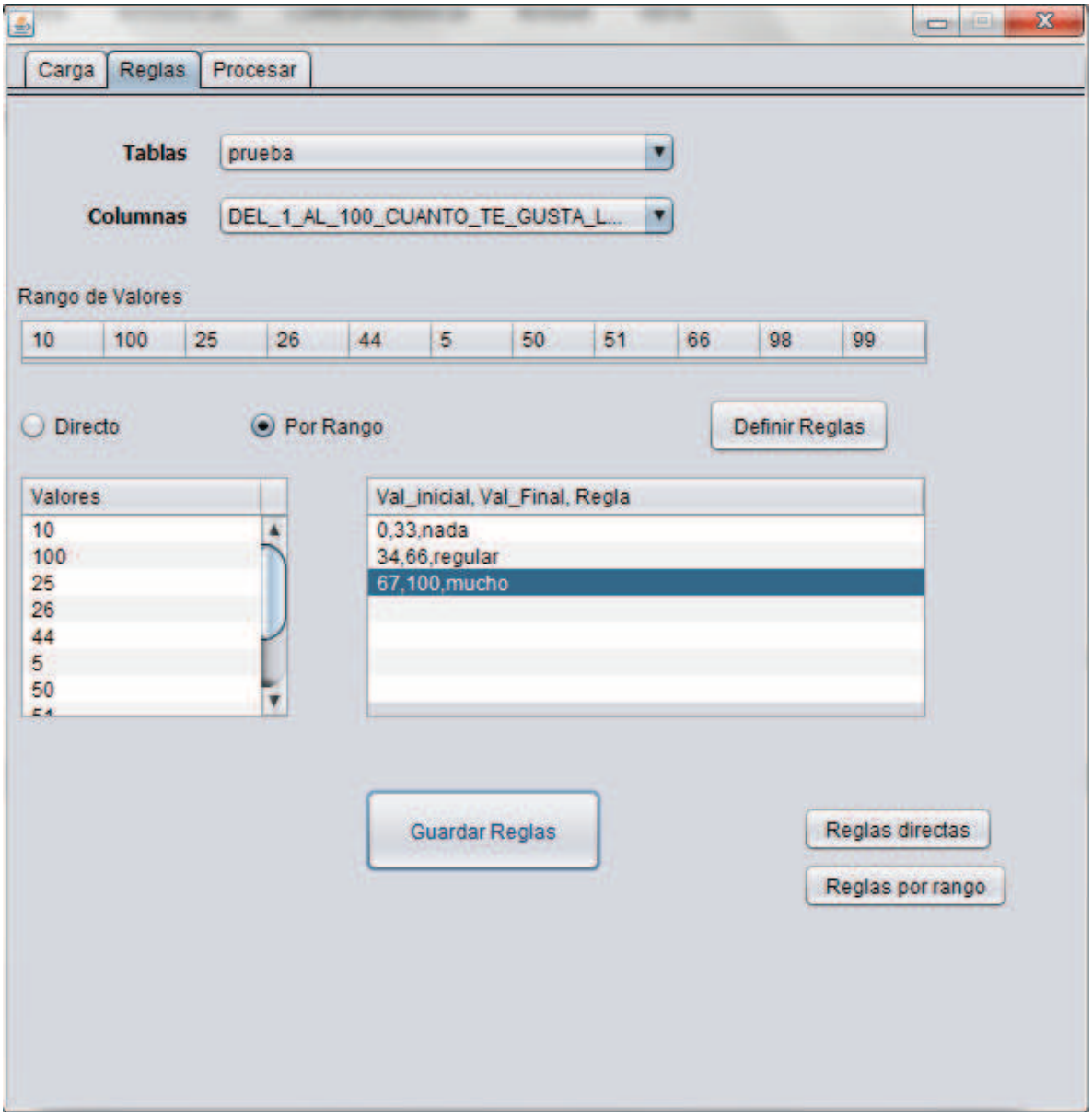


Figura 11. Interfaz Reglas. Guardar reglas por rango

Una vez generadas y guardadas las reglas están listas para ser aplicadas y procesar la información en el siguiente modulo.

Módulo de procesamiento.

Lo primero en este módulo fue seleccionar la tabla a la que se le aplican las reglas. Se logró aplicar las reglas generadas en el módulo anterior a nuestra tabla seleccionada para posteriormente generar el archivo de salida (archivo .CSV). Como se muestra en la ***Figura 12*** se logró aplicar las reglas a los datos y generar un archivo de salida el cual es posible abrir con tan solo un clic.

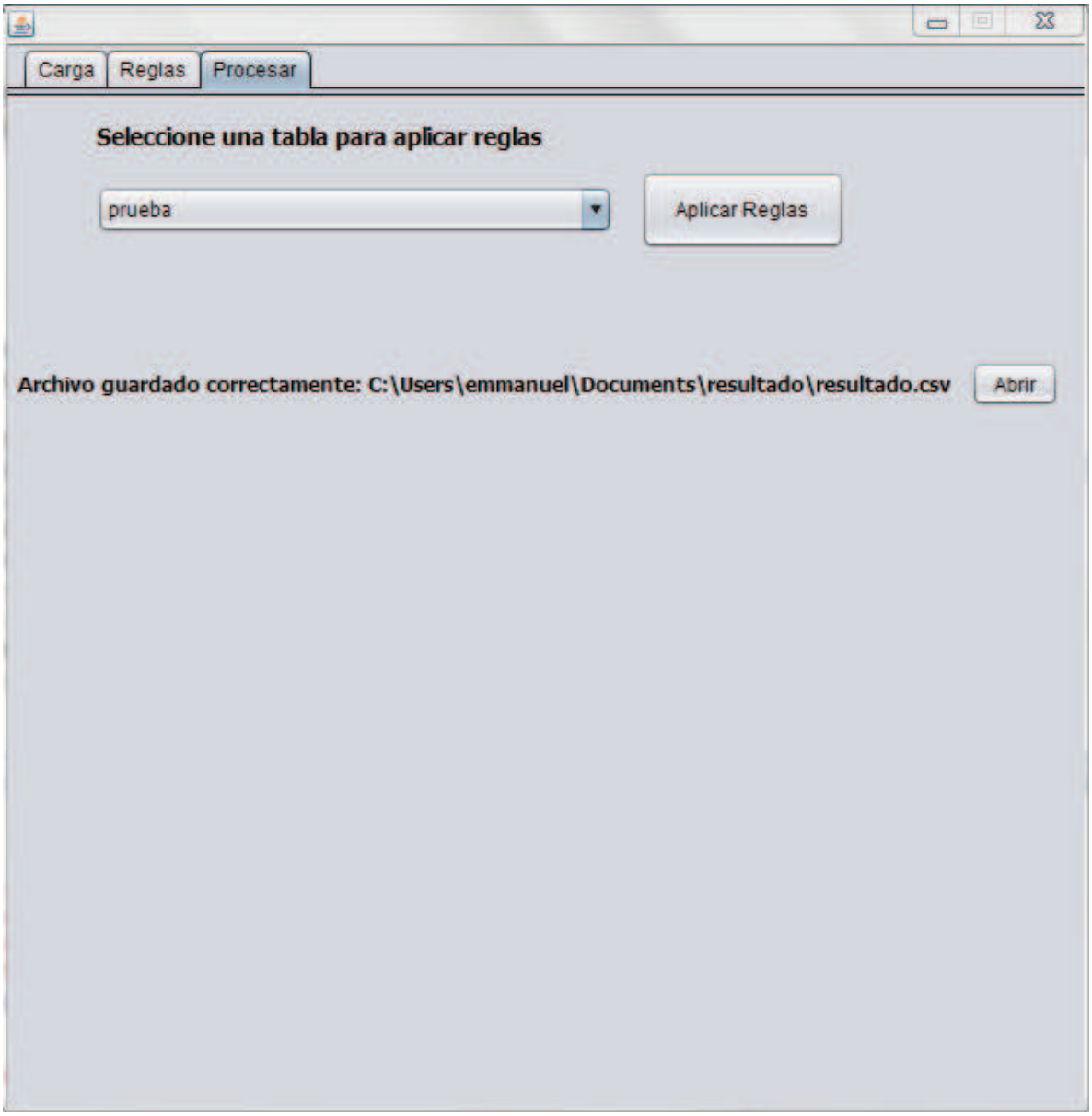


Figura 12. Interfaz procesar. Reglas aplicadas y archivo generado.

Con este método se concluye y el archivo de salida queda listo para que con tan solo un clic se pueda abrir.

Conclusiones

Las conclusiones acerca de la realización de este proyecto final de carrera, a pesar de algunas dificultades que se fueron presentando durante el desarrollo, son ampliamente positivas.

El hecho de realizar una aplicación en la cual se engloba un servicio con conexión a Base de Datos y manejo de archivos así como la utilización de una interfaz gráfica amigable e intuitiva contribuyo positivamente a reforzar y ampliar conocimientos sobre estos temas los cuales fueron parte de mi formación durante el transcurso de la carrera.

La primera dificultad que se presentó en la realización de este proyecto fue la carga del archivo y la conexión con la base de datos. Esto fue debido a la falta de conocimiento en el manejo de ambos temas juntos ya que solo había trabajado por separado el manejo de archivos y la conexión con la base de datos. El problema se resolvió con un poco de investigación lo que contribuyó a ampliar mi conocimiento en este ámbito.

Otra dificultad que se presento fue en el manejo de los datos ya que no se sabe qué tipo de datos que contiene el archivo estos datos son guardados todos de un mismo tipo esto generaba conflictos a la hora de aplicar las reglas sobre todo cuando eran reglas por rango ya que todos los datos los leía como String en un arreglo de datos ordenados de menor a mayor se mostraba lo siguiente: 1, 10, 11, 177, 2, 27, 30, 301, 49, 5, 50, 70.
Al intentar poner un rango por ejemplo del 0 al 35 me tomaba en cuenta los siguientes: 1, 10, 11, 177, 2, 27, 30 y lo que esperaba era que tomara lo siguiente: 1, 2, 5, 10, 11, 27, 30.

Solo queda reiterar que el aprendizaje del conocimiento obtenido en la realización de este proyecto ha sido enriquecedor en muchos sentidos, desde la investigación y redacción hasta la ampliación del conocimiento en cuanto a estos temas y a otros que fueron investigados.

El resultado final puede evaluarse como un sistema funcional que tiene como propósito realizar el proceso de limpieza de datos más eficientemente y casi automáticamente.

Referencias bibliográficas

[1] Fayyad, U. and Uthurusamy, R. (1996). Data mining and knowledge discovery in databases. Communications of the ACM, 39(11), pp.24-26.

[2] D.D. Mejía Montes, *"Correlacionador de Bitácoras de equipos en una LAN",* proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana, México, 2010.

[3] N. Guzmán González, *"Aplicación de Distintas Técnicas de Minería de Datos para el Tratamiento de Información",* proyecto tecnológico, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana, México, 2010.

[4] S.M. Ugalde Chávez, "*Sistema de Recuperación de Información Semántico*", proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana, México, 2012.

[5] Msdn.microsoft.com. (2016). Data Quality Services. [online] Available at: https://msdn.microsoft.com/en-us/library/ff877925.aspx [Accessed 11 Nov. 2016].

[6] Ramesh, V., Parkavi, P. and Ramar, K. (2013). Predicting Student Performance: A Statistical and Data Mining Approach. International Journal of Computer Applications, 63(8), pp.35-39.

[7] Kotsiantis, S., Pierrakeas, C. and Pintelas, P. (2004). Preventing Student Dropout in Distance Learning Using Machine Learning Techniques. Applied Artificial Intelligence, 18(5), pp.411-426.

[8] Blog.es.logicalis.com. (2017). *Data cleansing y sus fases: contra los problemas de calidad de datos*. [online] Available at: https://blog.es.logicalis.com/analytics/data-cleansing-y-sus-fases-contra-los-problemas-de-calidad-de-datos [Accessed 19 Apr. 2017].

[9] Webmining.cl. (2017). KDD Proceso de Extracción de conocimiento | WebMining. [online] Available at: http://www.webmining.cl/2011/01/proceso-de-extraccion-de-conocimiento/ [Accessed 26 Apr. 2017].

Entregables

- Código fuente de la aplicación en un archivo comprimido.
- Script de generación de las Bases de datos y las tablas que se utilizaran.
- Manual de usuario en archivo PDF.
- Reporte final en archivo PDF.

Anexo A. Código Fuente

Interfaz y Clase principal

207204584\proyecto\src\interfaces\principal.java

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Importamos las librerías que se utilizaran */

package interfaces;

import clases.*;
import java.awt.Desktop;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.*;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.event.TableModelEvent;
import javax.swing.event.TableModelListener;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author emmanuel
 */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Instanciamos variables globales*/

public class principal extends javax.swing.JFrame {
```

```
    @SuppressWarnings("FieldMayBeFinal")
    private FileNameExtensionFilter filter = new FileNameExtensionFilter("archivo de
texto","txt","csv");
    String ruta ="";
    String Result_Ruta ="";
    String Tab = "";
    String tabla = "ejemplo";
    int num,tok;
    int numDatos;
    String reglasRangoSQL[] = new String [100];

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Los componentes son ocultados en un principio */

    public principal() {
        initComponents();
        labelRuta.setVisible(false);
        labelTabla.setVisible(false);
        txtNomTabla.setVisible(false);
        btnCargar.setVisible(false);
        paneReglasDirecto.setVisible(false);
        btnGuardar.setVisible(false);
        paneRango.setVisible(false);
        paneValRango.setVisible(false);
        paneReglasRango.setVisible(false);
        panelColumnas.setVisible(false);
        jLabel5.setVisible(false);
        rbPorRango.setVisible(false);
        rbDirecto.setVisible(false);
        btnDefinir.setVisible(false);
        RutaGuardar.setVisible(false);
        Guardar.setVisible(false);
        btnProcesar.setVisible(false);
        btnAbrir.setVisible(false);

        buttonGroup1.add(rbDirecto);
        buttonGroup1.add(rbPorRango);

        cargarTablas();
        this.setLocationRelativeTo(null);

        setResizable(false);

    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    // <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
    private void initComponents(434 lines…) {
}// </editor-fold>

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* Creamos las variables para la conexion a la bd*/

    private final String driver = "com.mysql.jdbc.Driver";
    private final String user = "root";
    private final String url = "jdbc:mysql://localhost/proyecto";
    private final String pass = "1234";

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* En este método las tablas son cargadas desde el archivo seleccionado hacia la base de
datos*/

    public void cargarTablas(){

        String SQL = "show full tables from proyecto";
        try {
            Class.forName(driver);
            Connection con = DriverManager.getConnection(url, user, pass);
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(SQL);
            cmbTablas.removeAllItems();
            cmbTablas.addItem("Seleccione una tabla...");
            cmbTablas1.removeAllItems();
            cmbTablas1.addItem("Seleccione una tabla...");
            while(rs.next()){
                cmbTablas.addItem(rs.getString(1));
                cmbTablas1.addItem(rs.getString(1));

            }
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*En este método se cargan en los combos las columnas de la tabla seleccionada haciendo
una consulta a la base de datos*/
    public void cargarColumnas(){

        if (cmbTablas.getSelectedIndex()>0){
            try {
                Class.forName(driver);
                Connection con = DriverManager.getConnection(url, user, pass);
                Statement st = con.createStatement();
```

```
            ResultSet rs = st.executeQuery("show columns from
"+cmbTablas.getSelectedItem().toString());
            cmbColumnas.removeAllItems();
            cmbColumnas.addItem("Seleccione una columna...");
            rs.next(); //omitimos el id
            while(rs.next()){
               cmbColumnas.addItem(rs.getString(1));

            }
         } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
         }
      }

   }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Dependiendo la columna seleccionada este método muestra los valores de dicha
columna*/
   public void cargarValores(){

      if (cmbColumnas.getSelectedIndex()>0){
         tok = 0;
         num = 0; //se crea la variable num para contar el numero de valores
         String SQL1 = "select distinct "+cmbColumnas.getSelectedItem().toString();
         String SQL2 = SQL1 + " from "+cmbTablas.getSelectedItem().toString();
         String SQL = SQL2 + " order by "+cmbColumnas.getSelectedItem().toString()+"
asc";
         Variables.valorGuardadoB = SQL;
         try{
            Class.forName(driver);
            Connection con = DriverManager.getConnection(url, user, pass);
            Statement st = con.createStatement();
            ResultSet rst = st.executeQuery(SQL);
            ResultSetMetaData rsMD = rst.getMetaData();
            int numColumnas = rsMD.getColumnCount();
            DefaultTableModel Modelo = new DefaultTableModel();
            this.jTable2.setModel(Modelo);


            while(rst.next()){

               Modelo.addColumn(rst.getObject(numColumnas));
               num ++;

            }

      } catch (ClassNotFoundException | SQLException ex) {
```

```
            Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
        }
        }

    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Dependiendo la tabla seleccionada este método llama al método cargarColumnas para
que el combo columnas muestre las opciones*/
    private void cmbTablasActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        repaint();
        if(cmbTablas.getSelectedIndex()>-1){
            cmbColumnas.removeAllItems();
            cargarColumnas();
            paneValRango.setVisible(false);
            paneReglasRango.setVisible(false);
            paneReglasDirecto.setVisible(false);
            btnGuardar.setVisible(false);
            jLabel5.setVisible(false);
            paneRango.setVisible(false);
            rbPorRango.setVisible(false);
            rbDirecto.setVisible(false);
            btnDefinir.setVisible(false);

        }
    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Este método llama al método cargarValores para mostrar los valores de la columna
seleccionada*/
    private void cmbColumnasActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(cmbColumnas.getSelectedIndex()>-1){

            paneRango.setVisible(true);
            cargarValores();
            paneValRango.setVisible(false);
            paneReglasRango.setVisible(false);
            paneReglasDirecto.setVisible(false);
            btnGuardar.setVisible(false);
            jLabel5.setVisible(true);
            rbPorRango.setVisible(true);
            rbDirecto.setVisible(true);
            btnDefinir.setVisible(true);
            repaint();
        }
    }
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Se muestra la tabla reglas_por_rango*/
    private void btnRegRangoActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        MostrarReglasPorRango obj = new MostrarReglasPorRango();
        obj.setVisible(true);
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Se muestra la tabla reglas_directo*/
    private void btnRegDirectoActionPerformed(java.awt.event.ActionEvent evt) {
         // TODO add your handling code here:
        MostrarReglasDirecto obj = new MostrarReglasDirecto();
        obj.setVisible(true);
    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Dependiendo la opción seleccionada de los radio button se realiza una acción, si se
selecciona por rango se manda llamar la clase conectar y se manda como parámetro el
script que guarda cada regla, en caso de elegir directo aquí mismo se genera el script y se
manda a ejecutar a la clase conectar*/
    private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(rbPorRango.isSelected()){
            for (int i=0 ; i < tok  ; i++){
                    Conectar conec = new Conectar();
                    conec.main(reglasRangoSQL[i]);
            }
        }

        else{
            String colum = cmbColumnas.getSelectedItem().toString();
            String tab = cmbTablas.getSelectedItem().toString();
            String sql2[]=new String[1000];

            //System.out.println(num);
            for(int i=0; i < num ;i++){

                String valoriginal = (String) tbDirecto.getValueAt(i, 0);
                String valnuevo = (String) tbDirecto.getValueAt(i, 1);
                String sql = "INSERT reglas_directo (tabla, columna, valor, regla) ";
                sql2[i] = sql+ "VALUES ('"+tab+"', '"+colum+"', '"+valoriginal+"',
'"+valnuevo+"');";

                Conectar conec = new Conectar();
                conec.main(sql2[i]);
            }
```

```
    }
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Este botón actúa dependiendo la opción elegida, si es por rango muestra dos tablas una con los valores y otra vacía la cual se utiliza para definir las reglas*/

    private void btnDefinirActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        repaint();
        btnGuardar.setVisible(true);
        if (rbPorRango.isSelected()){
            paneReglasDirecto.setVisible(false);
            paneValRango.setVisible(true);
            paneReglasRango.setVisible(true);


            try{
                String SQL = Variables.valorGuardadoB;
                Class.forName(driver);
                Connection con = DriverManager.getConnection(url, user, pass);
                Statement stm = con.createStatement();
                ResultSet rst = stm.executeQuery(SQL);
                ResultSetMetaData rsMD = rst.getMetaData();
                int numFilas = rsMD.getColumnCount();

                DefaultTableModel Model = new DefaultTableModel();
                this.tabValoresRango.setModel(Model);

                Model.addColumn("Valores");
                while(rst.next()){

                    Object [] fila = new Object [numFilas];

                    for (int y = 0; y < numFilas; y++){
                        fila [y] = rst.getObject(y+1);
                    }

                    Model.addRow(fila);

                }

                DefaultTableModel Modelo = new DefaultTableModel();
                this.tbRegRango.setModel(Modelo);
                Modelo.addColumn("Val_inicial, Val_Final, Regla");
                Object [] fila = new Object [5];
                Modelo.addRow(fila);
                Modelo.addRow(fila);
```

```
            Modelo.addRow(fila);
            Modelo.addRow(fila);
            Modelo.addRow(fila);
            Modelo.addRow(fila);
            Modelo.addRow(fila);

            Modelo.addTableModelListener(new TableModelListener(){
               String tabla = cmbTablas.getSelectedItem().toString();
               String colum = cmbColumnas.getSelectedItem().toString();

            @Override
            public void tableChanged(TableModelEvent e) {
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Aquí se registran los eventos que ocurren en la tabla vacia y se guarda su contenido para
posteriormente extraer tres valores de cada evento los cuales son valor inicial, valor final y
regla. Después simplemente son guardados en su respectivo campo*/
               if(e.getType() == TableModelEvent.UPDATE ){

                  int fil = e.getFirstRow();
                  int i = 0;
                  String temp2[] = new String[100];
                  temp2[i] = (String) tbRegRango.getValueAt(fil, 0);
                  StringTokenizer token = new StringTokenizer (temp2[0]);
                  // bucle por todas las palabras
                  while (token.hasMoreTokens()){
                     temp2[i] = temp2[0].replace(" ", "");
                     temp2[i] = token.nextToken(",");
                     i++;
                  }

                  String aux = "INSERT reglas_por_rango (tabla, columna, valor_inicial,
valor_final, regla) ";
                  String sql = aux + "VALUES ('"+tabla+"', '"+colum+"', '"+temp2[0]+"',
'"+temp2[1]+"', '"+temp2[2]+"');";
                  reglasRangoSQL[tok] = sql;
                  tok++;
               }
            }
         });

         } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
         }
      }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*En este caso se muestra una tabla con dos columnas una con el valor original y la otra
vacía que es donde va la regla */
```

```
        else {
            paneValRango.setVisible(false);
            paneReglasRango.setVisible(false);
            paneReglasDirecto.setVisible(true);

            try{
               String SQL = Variables.valorGuardadoB;
               Class.forName(driver);
               Connection con = DriverManager.getConnection(url, user, pass);
               Statement stm = con.createStatement();
               ResultSet rst = stm.executeQuery(SQL);
               ResultSetMetaData rsMD = rst.getMetaData();
               int numFilas = rsMD.getColumnCount();

               DefaultTableModel Modelo = new DefaultTableModel();
               this.tbDirecto.setModel(Modelo);
               Modelo.addColumn("Valor");
               Modelo.addColumn("Regla");

               while(rst.next()){
                  Object [] fila = new Object [numFilas];
                  for (int y = 0; y < numFilas; y++){
                     fila [y] = rst.getObject(y+1);
                  }
                  Modelo.addRow(fila);
               }
            }
            catch(ClassNotFoundException | SQLException e){
               //e.printStackTrace();
               JOptionPane.showMessageDialog(this, "Error");
            }
         }
   }

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Este método permite seleccionar el archivo con el que se desea trabajar, se guarda el
nombre del archivo y la ruta en la que se encuentra*/
   private void btnSelecArchivoActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      labelTabla.setVisible(true);
      txtNomTabla.setVisible(true);
      JFileChooser filechoose = new JFileChooser();
      filechoose.setFileFilter(filter);
      int opcion = filechoose.showOpenDialog(this);

      if (opcion==JFileChooser.APPROVE_OPTION){
         String nombre_archivo = filechoose.getSelectedFile().getName();
         ruta = filechoose.getSelectedFile().toString();
```

```
            labelRuta.setText(ruta);
        }

        labelRuta.setVisible(true);
        btnCargar.setVisible(true);

    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Aquí se manda llamar a la clase registrarArchi la cual se encarga de cargar los datos a la
tabla creada y muestra las columnas generadas de dicha tabla*/
    private void btnCargarActionPerformed(java.awt.event.ActionEvent evt) {
        repaint();
        panelColumnas.setVisible(true);
        registrarArchi();
        Variables.valorGuardadoA = txtNomTabla.getText();

        try{
            Class.forName(driver);
            Connection con = DriverManager.getConnection(url, user, pass);
            Statement stm = con.createStatement();
            ResultSet rst = stm.executeQuery("show columns from
"+Variables.valorGuardadoA);
            ResultSetMetaData rsMD = rst.getMetaData();
            numDatos = rsMD.getColumnCount();

            DefaultTableModel Modelo = new DefaultTableModel();
            this.tbColumnas.setModel(Modelo);
            Modelo.addColumn("COLUMNAS");
            while(rst.next()){
                Object [] fila = new Object [numDatos];

                for (int y = 0; y < numDatos; y++){
                    fila [y] = rst.getObject(y+1);
                }
                Modelo.addRow(fila);
            }
        } catch (ClassNotFoundException ex) {
            JOptionPane.showMessageDialog(this, "Error ");

            System.out.println(ex);
        } catch (SQLException ex) {
            System.out.println(ex);
            JOptionPane.showMessageDialog(this, "Error al conectar...");
                        Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null,
ex);

        }
```

```
        cargarTablas();
    }

    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Al igual que cmbTablas esta muestra las tablas existentes y crea un respaldo de la tabla seleccionada para ser editada sin restricciones*/
    private void cmbTablas1ActionPerformed(java.awt.event.ActionEvent evt) {
            Conectar cone = new Conectar();
            if(cmbTablas1.getSelectedIndex()>0){
                btnProcesar.setVisible(true);
                Tab = cmbTablas1.getSelectedItem().toString();
                String backup = "CREATE TABLE registro."+Tab+" select * from proyecto."+Tab+";";
                cone.main(backup);
            try{
            Class.forName(driver);
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/registro", user, pass);
            Statement stm = con.createStatement();
            ResultSet rst = stm.executeQuery("select distinct columna from reglas_por_rango");
            ResultSetMetaData rsMD = rst.getMetaData();
            numDatos = rsMD.getColumnCount();

                while(rst.next()){

                    String agregar = (String) rst.getObject(1);
                    String sql = "ALTER table "+Tab+" add "+agregar+"_2 integer;";
                    cone.main(sql);
                    sql = "update "+Tab+" set "+agregar+"_2 = "+agregar+" where "+agregar+" is not null;";
                    cone.main(sql);

                }
            }   catch (ClassNotFoundException ex) {
                    Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
                } catch (SQLException ex) {
                    Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
                }
            };
    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Aquí es donde se verifican todas las reglas que existen para la tabla seleccionada y son aplicadas dichas reglas, todo esto se realiza en la tabla creada como respaldo*/
    private void btnProcesarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Conectar cone = new Conectar();
```

```
        RutaGuardar.setVisible(true);
        Guardar.setVisible(true);
        String temp[]=new String[100];
        String ta = cmbTablas1.getSelectedItem().toString();
        String select_rango = "select columna,valor_inicial,valor_final,regla from
reglas_por_rango where tabla='"+ta+"'";
        String select_directo = "select columna,valor,regla from reglas_directo where
tabla='"+ta+"'";

            try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/registro",
"root", "1234");
            Statement stm1 = con.createStatement();
            Statement stm2 = con.createStatement();
            ResultSet rst_Rango = stm1.executeQuery(select_rango);
            ResultSet rst_Directo = stm2.executeQuery(select_directo);
            ResultSetMetaData rsMD_Directo = rst_Directo.getMetaData();
            ResultSetMetaData rsMD_Rango = rst_Rango.getMetaData();
            int numColumnas = rsMD_Directo.getColumnCount();
            int numColum = rsMD_Rango.getColumnCount();


            while(rst_Directo.next()){
               Object [] fila = new Object [numColumnas];
               String val_filas="";
               String col,va,re="";
               for (int y=0; y<numColumnas; y++){
                  fila [y] = rst_Directo.getObject(y+1);

                  val_filas += rst_Directo.getObject(y+1)+",";

               }
               col = (String) rst_Directo.getObject(1);
               va = (String) rst_Directo.getObject(2);
               re = (String) rst_Directo.getObject(3);

               //update prueba set estado_civil = replace(estado_civil, "1", "soltero");
               String sql_update = "update "+ta+" set "+col+"=replace("+col+", \""+va+"\",
\""+re+"\")";

               cone.main(sql_update);
            }

            String col="";
            int cont = 0;

            while(rst_Rango.next()){
```

```
            String v_inicial,v_final,re="";

            col = (String) rst_Rango.getObject(1);
            v_inicial = (String) rst_Rango.getObject(2);
            v_final = (String) rst_Rango.getObject(3);
            re = (String) rst_Rango.getObject(4);
            String sql = "update "+ta+" set "+col+" = \""+re+"\" where "+col+"_2 >=
'"+v_inicial;
            String sql_update = sql+"' and "+col+"_2 <= '"+v_final+"';";
            temp[cont] = col;
            cont ++;
            cone.main(sql_update);
          }

          for (int x=0 ; x<cont ; x++){
            String Borrar_columna = "alter table "+ta+" drop column "+temp[x]+"_2;";
            cone.main(Borrar_columna);
          }

      } catch (ClassNotFoundException ex) {
          Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
      } catch (SQLException ex) {
          Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
      }
    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Con este método se genera el archivo de salida, el cual contiene los datos actualizados
con las reglas aplicadas*/
    private void GuardarActionPerformed(java.awt.event.ActionEvent evt) {

      JFileChooser filechoose = new JFileChooser();
      filechoose.setFileFilter(filter);
      int opcion = filechoose.showSaveDialog(this);

      if (opcion==JFileChooser.APPROVE_OPTION){
          String nombre_archivo = filechoose.getSelectedFile().getName();
          Result_Ruta = filechoose.getSelectedFile().toString();
          RutaGuardar.setText("Archivo guardado correctamente: "+Result_Ruta);
      }

      try{
          Class.forName(driver);
          Connection con = DriverManager.getConnection(url, user, pass);
          Statement stm = con.createStatement();
          ResultSet rst = stm.executeQuery("select * from registro."+Tab);
          ResultSetMetaData rsMD = rst.getMetaData();
          int numColumnas = rsMD.getColumnCount();
```

```
            String columnas_nuevas="";

            for (int x=2; x<=numColumnas; x++){
               columnas_nuevas += rsMD.getColumnLabel(x)+",";
            }

            String columnas_txt = columnas_nuevas.substring(0,columnas_nuevas.length() -1);
            crearTxt(columnas_txt, Result_Ruta);

            while(rst.next()){
               String val_filas="";
               for (int y=1; y<numColumnas; y++){
                  val_filas += rst.getObject(y+1)+",";
               }
               String filas_txt = val_filas.substring(0,val_filas.length() -1);
               crearTxt(filas_txt, Result_Ruta);
            }
         }
         catch(ClassNotFoundException e){
            //JOptionPane.showMessageDialog(this, "Error al buscar.");
         }
         catch(SQLException se){
            //JOptionPane.showMessageDialog(this, "Error en la lectura de la tabla");
         }
         Conectar cone = new Conectar();
         String Borrar_aux = "drop table "+Tab+";";
         cone.main(Borrar_aux);
         Guardar.setVisible(false);
         btnAbrir.setVisible(true);
      }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Con este método al presionar el botón Abrir nos abrirá el archivo creado*/
      private void btnAbrirActionPerformed(java.awt.event.ActionEvent evt) {

         try {
            File objeto = new File (Result_Ruta+".csv");
            Desktop.getDesktop().open(objeto);
         } catch (IOException ex) {
//             System.out.println(ex);
//             JOptionPane.showMessageDialog(this, "El archivo no existe o fue borrado");
         }

      }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Este método se utiliza para mandar los parámetros a la clase Cargar para que registre los
datos del archivo en la base de datos*/
```

```
    private void registrarArchi(){

        Cargar car = new Cargar();

        String archivo = ruta;
        String nomtabla = txtNomTabla.getText();
        car.main(archivo, nomtabla);
            }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Con este método se crea y llena el archivo de salida que es nuestro resultado de la
transformación de los datos aplicando las reglas*/
    public static void crearTxt (String txt, String Ruta){
        try {
            File archivo = new File(Ruta+".csv");
            try (
                    FileWriter escribir = new FileWriter(archivo,true)) {
                    escribir.write(txt +"\r\n");
            }
        }
        catch(Exception e)
        {
        JOptionPane.showMessageDialog(null,"Error al escribir");
        }
    }

    /**
     * @param args the command line arguments
     */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Método principal*/

    public static void main(String args[]) {

        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(principal.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
```

```
        }
        //</editor-fold>

        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new principal().setVisible(true);
            }
        });
    }

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*declaración de variables java swing */
    // Variables declaration - do not modify
    private javax.swing.JButton Guardar;
    private javax.swing.JLabel RutaGuardar;
    private javax.swing.JButton btnAbrir;
    private javax.swing.JButton btnCargar;
    private javax.swing.JButton btnDefinir;
    private javax.swing.JButton btnGuardar;
    private javax.swing.JButton btnProcesar;
    private javax.swing.JButton btnRegDirecto;
    private javax.swing.JButton btnRegRango;
    private javax.swing.JButton btnSelecArchivo;
    private javax.swing.ButtonGroup buttonGroup1;
    public static javax.swing.JComboBox cmbColumnas;
    private javax.swing.JComboBox cmbTablas;
    private javax.swing.JComboBox cmbTablas1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTable jTable2;
    private javax.swing.JLabel labelRuta;
    private javax.swing.JLabel labelTabla;
    private javax.swing.JScrollPane paneRango;
    private javax.swing.JScrollPane paneReglasDirecto;
    private javax.swing.JScrollPane paneReglasRango;
    private javax.swing.JScrollPane paneValRango;
```

```
    private javax.swing.JScrollPane panelColumnas;
    private javax.swing.JRadioButton rbDirecto;
    private javax.swing.JRadioButton rbPorRango;
    private javax.swing.JTable tabValoresRango;
    private javax.swing.JTable tbColumnas;
    private javax.swing.JTable tbDirecto;
    private javax.swing.JTable tbRegRango;
    private javax.swing.JTextField txtNomTabla;
    // End of variables declaration
}
```

==========

Clase Cargar

==========

207204584\proyecto\src\clases\Cargar.java

```
/*
 En esta clase es donde se carga el archive hacia la base de datos
 */
package clases;

    import java.io.*;
    import java.util.*;
    import java.sql.*;
    import java.sql.Connection;
    import java.sql.Statement;
    import java.util.logging.Level;
    import java.util.logging.Logger;
import javax.swing.JOptionPane;

public class Cargar {

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Se crean las variables para la conexión a la base de datos*/
        static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
        static final String DB_URL = "jdbc:mysql://localhost/proyecto";
        static final String USER = "root";
        static final String PASS = "1234";


        @SuppressWarnings("null")
/*En este modulo se recibe la ruta del archivo que se utilizara y se recibe también el
nombre para la tabla los datos son cargados en el buffer para posteriormente guardarlos en
la base de datos*/
```

```
    public static void main(String archivo, String tabla){//se recive el nombre del archivo
      // Cargamos el buffer con el contenido del archivo
       BufferedReader br = null;
       try {
          br = new BufferedReader (new FileReader (archivo));
       } catch (FileNotFoundException ex) {
          Logger.getLogger(Cargar.class.getName()).log(Level.SEVERE, null, ex);
       }

       String temp[]=new String[1000];
       String bfRead;

       int contador = 0;
       try {
          while((bfRead = br.readLine())!=null){
          //haz el ciclo, mientras bfRead tiene datos
             temp[contador] = bfRead;

             contador ++;
          }

       } catch (IOException ex) {
          Logger.getLogger(Cargar.class.getName()).log(Level.SEVERE, null, ex);
       }

/*con los datos leídos se crean los script SQL para ser ejecutados*/

       // Leemos la primera linea
          int tok=0;
          String pos[] = new String[1000];
          int cont = contador-1;
          String Sau="", otra="", sqlFinal[]=new String[1000];

          for (int i=1;i<contador;i++){
             temp[i] = temp[i].replace(",", "',"); //reemplazamos las , por ','
             temp[i] = "VALUES ('"+temp[i]+"')"; //creamos la cadena
          }

          StringTokenizer token = new StringTokenizer (temp[0]);
          // bucle por todas las palabras
          while (token.hasMoreTokens()){
             pos[tok] = token.nextToken(",");
             tok++;
          }

          String Saux="", Sfinal="";
          String S1="CREATE TABLE "+tabla+" (id INT NOT NULL PRIMARY KEY
AUTO_INCREMENT, ";
```

```
            for(int z = 0; z < tok; z++){
               Saux += pos[z] + " VARCHAR(100),";
               Sau += pos[z] + ",";
            }

            String S2 = "INSERT "+tabla+" (";
            Sfinal = (S1+Saux);
            String nuevaCadena = Sfinal.substring(0,Sfinal.length() -1) + ");";
            otra = S2 + Sau.substring(0 , Sau.length() -1) + ")";

            for (int j = 1 ; j < cont + 1 ; j++){
               sqlFinal[j] = otra + "\n" + temp[j] + ";";
            }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Se ejecutan los Script generados para llenar la tabla creada*/
            Connection conn = null;
            Statement stmt = null;
            try{
               //STEP 2: Registrar JDBC driver
               Class.forName(JDBC_DRIVER);

               //STEP 3: Abrir la conexion
               //System.out.println("Conectando a la Base de Datos...");
               conn = DriverManager.getConnection(DB_URL, USER, PASS);
               //System.out.println("Conexion realizada...");

               //STEP 4: Ejecutar query
               //System.out.println("Creando tabla en la Base de datos...");
               Variables.valorGuardadoA = tabla;
               stmt = conn.createStatement();
               String sql = nuevaCadena;
               stmt.executeUpdate(sql);

               for (int j=1 ; j < cont + 1; j++){
                  stmt = conn.createStatement();
                  String sql2 = sqlFinal[j];
                  stmt.executeUpdate(sql2);
               }

               //System.out.println("La tabla "+tabla+" se creo correctamente...");

            }catch(SQLException se){
            JOptionPane.showMessageDialog(null, "Error");
         }catch(ClassNotFoundException e){
            //System.out.println( "Error al crear la conexion" );
            conn = null;
```

```
        }finally{
            try{
                if(stmt!=null)
                conn.close();
            }catch(SQLException se){
            }// do nothing
            try{
                if(conn!=null)
                conn.close();
            }catch(SQLException se){
                se.printStackTrace();
            }
        }
        //System.out.println("Goodbye!");
    }//end main

}//
```

==========

Clase conectar

==========

207204584\proyecto\src\clases\Conectar.java

```
/*
En este módulo se manda a ejecutar cada update (no regresa nada solo se ejecuta la
sentencia SQL)
 */
package clases;

import interfaces.principal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author emmanuel
 */
```

```
/*se crea la conexión y se recibe la sentencia SQL para ser ejecutada*/

public class Conectar {

    public static Connection conn;
    public static final String driver = "com.mysql.jdbc.Driver";
    public static final String user = "root";
    public static final String password = "1234";
    public static final String url = "jdbc:mysql://localhost:3306/registro";

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*Se ejecuta cada sentencia SQL que llega a esta clase*/

    public static void main(String sql){//
        try {
                Class.forName(driver);
                conn = DriverManager.getConnection(url, user, password);
                Statement stm = conn.createStatement();
                stm.executeUpdate(sql);

            }
//        }
//        });
            catch (SQLException | ClassNotFoundException ex) {
                Logger.getLogger(principal.class.getName()).log(Level.SEVERE, null, ex);
            }
    }

    //este metodo nos retorna la conexion
    public Connection getConnection(){
        return conn;

    }

    //con este metodo nos desconectamos de la BD
    public void desconectar(){
        conn = null;
        if(conn == null){
            System.out.println("conexion finalizada");
        }
    }

}
```

==========

Clase variables

207204584\proyecto\src\clases\Variables.java

```
/*
 Esta clase solo fue creada para almacenar variables que son utilizadas en
distintas clases
 */
package clases;
/**
 *
 * @author emmanuel
 */
public class Variables {

    public static String valorGuardadoA = "";
    public static String valorGuardadoB = "";
    public static String valorGuardadoC = "";

    public static void main(String tabla){//


    }
}
```

Interfaz MostrarReglasDirecto

207204584\proyecto\src\interfaces\MostrarReglasDirecto.java

```
/*
 Esta interfaz muestra las reglas directas que se han creado
 */
package interfaces;

import java.io.BufferedWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
```

```
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.io.File;
import java.io.FileWriter;

/**
 *
 * @author emmanuel
 */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*se establece la conexión y se realiza una consulta para mostrar el resultado en una tabla*/
public class MostrarReglasDirecto extends javax.swing.JFrame {


    String columnas_nuevas="";
    public MostrarReglasDirecto() {
        initComponents();
        this.setLocationRelativeTo(null);
        setResizable(false);

        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/registro",
"root", "1234");
            Statement stm = con.createStatement();
            ResultSet rst = stm.executeQuery("select * from reglas_directo");
            ResultSetMetaData rsMD = rst.getMetaData();
            int numColumnas = rsMD.getColumnCount();

            DefaultTableModel Modelo = new DefaultTableModel();
            this.reglasDirecto.setModel(Modelo);

            for (int x=1; x<=numColumnas; x++){
                Modelo.addColumn(rsMD.getColumnLabel(x));
            }

            while(rst.next()){
                Object [] fila = new Object [numColumnas];

                for (int y=0; y<numColumnas; y++){
                    fila [y] = rst.getObject(y+1);

                }

                Modelo.addRow(fila);
            }
```

```
        }
        catch(ClassNotFoundException e){
            JOptionPane.showMessageDialog(this, "Error al buscar.");

        }
        catch(SQLException se){
            JOptionPane.showMessageDialog(this, "La tabla no existe");
        }
    }


    @SuppressWarnings("unchecked")

    private void initComponents() {…74 lines}

/*Se Cierra la ventana */
    private void CerrarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MostrarReglasDirecto.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MostrarReglasDirecto.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MostrarReglasDirecto.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(MostrarReglasDirecto.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MostrarReglasDirecto().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton Cerrar;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable reglasDirecto;
    // End of variables declaration
}
```

==========

Interfaz MostrarReglasPorRango

==========

207204584\proyecto\src\interfaces\MostrarReglasPorRango.java

```
/*
Esta interfaz muestra las reglas por rango que se han creado
*/

package interfaces;

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

/**
 *
```

```
 * @author emmanuel
 */

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*se establece la conexión y se realiza una consulta para mostrar el resultado en una tabla*/
public class MostrarReglasPorRango extends javax.swing.JFrame {

    /** Creates new form MostrarReglasPorRango */
    public MostrarReglasPorRango() {
        initComponents();
        this.setLocationRelativeTo(null);
        setResizable(false);

        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/registro",
"root", "1234");
            Statement stm = con.createStatement();
            ResultSet rst = stm.executeQuery("select * from reglas_por_rango");
            ResultSetMetaData rsMD = rst.getMetaData();
            int numColumnas = rsMD.getColumnCount();

            DefaultTableModel Modelo = new DefaultTableModel();
            this.ReglasPorRango.setModel(Modelo);

            for (int x=1; x<=numColumnas; x++){
                Modelo.addColumn(rsMD.getColumnLabel(x));
            }


            while(rst.next()){
                Object [] fila = new Object [numColumnas];

                for (int y=0; y<numColumnas; y++){
                    fila [y] = rst.getObject(y+1);
                }

                Modelo.addRow(fila);

            }

        }
        catch(ClassNotFoundException e){
            JOptionPane.showMessageDialog(this, "Error al buscar.");

        }
        catch(SQLException se){
            JOptionPane.showMessageDialog(this, "La tabla no existe");
```

```
        }

    }

    @SuppressWarnings("unchecked")

    private void initComponents() {…68 lines}// </editor-fold>

/*Cerramos la ventana*/
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MostrarReglasPorRango.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MostrarReglasPorRango.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MostrarReglasPorRango.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MostrarReglasPorRango.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MostrarReglasPorRango().setVisible(true);
```

```
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JTable ReglasPorRango;
private javax.swing.JButton jButton1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
// End of variables declaration

}
```