

INFORME

-FINAL-

Escuela Politecnica Nacional
Facultad Ingeniería de Sistemas
Programacion II
Proyecto
Informe final



Integrantes:

1. Danny Ñaguazo
2. Joseph Jiménez
3. David Puga

Curso: G1CC

Grupo: 3

Profesora: Maritzol Tenemaza

Índice

OBJETIVOS:	2
METODOLOGIA:	2
1. CREACIÓN DE LA BASE DE DATOS.....	2
1.1. Instalación de MySQL.....	2
1.2. Creación de la base de datos.	13
2. CONEXIÓN DE LA BASE DE DATOS	17
3. CREACIÓN DE LAS TABLAS Y SUS RELACIONES	21
3.1. Creación de Tablas	21
3.2. Relación de tablas.....	23
3.3. Creación Del Crud Para La Tabla “Vuelo”	26
3.4. Resultados del CRUD.....	31
4. AFINACIÓN DEL PROGRAMA.....	31
4.1. Afinación De La Base De Datos.....	31
4.2. Creación De Nuevas Clases Para El Crud.....	33
4.3. Mejora Visual De Las Interfaces	34
4.4. Preparación De La Interfaz Para La Venta De Boletos	36
CONCLUSIONES Y RECOMENDACIONES:	43
BIBLIOGRAFIA:	44

OBJETIVOS:

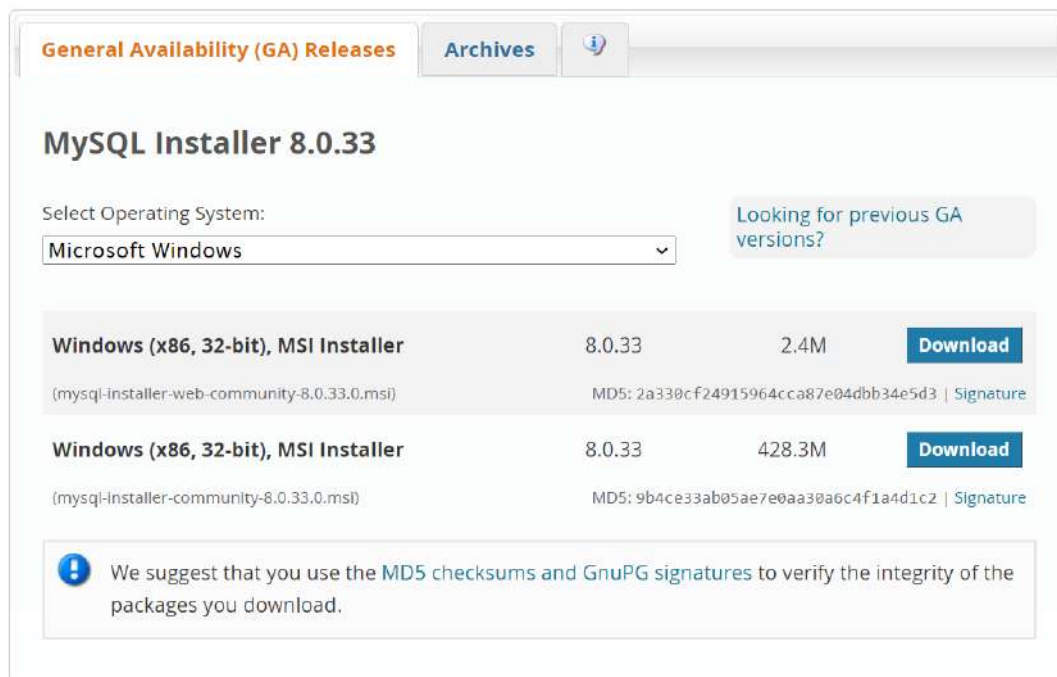
- Desarrollar un programa de gestión de vuelos para una aerolínea que satisfaga las necesidades tanto de la parte administrativa como de los clientes, esto a través de la creación de una interfaz atractiva y de fácil navegación.
- Este proyecto implica la gestión de una base de datos en MySQL mediante la herramienta Apache NetBeans. Este proceso se basará en el conocimiento adquirido a lo largo del periodo académico, respaldado por un trabajo de investigación.

METODOLOGIA:

1. CREACIÓN DE LA BASE DE DATOS

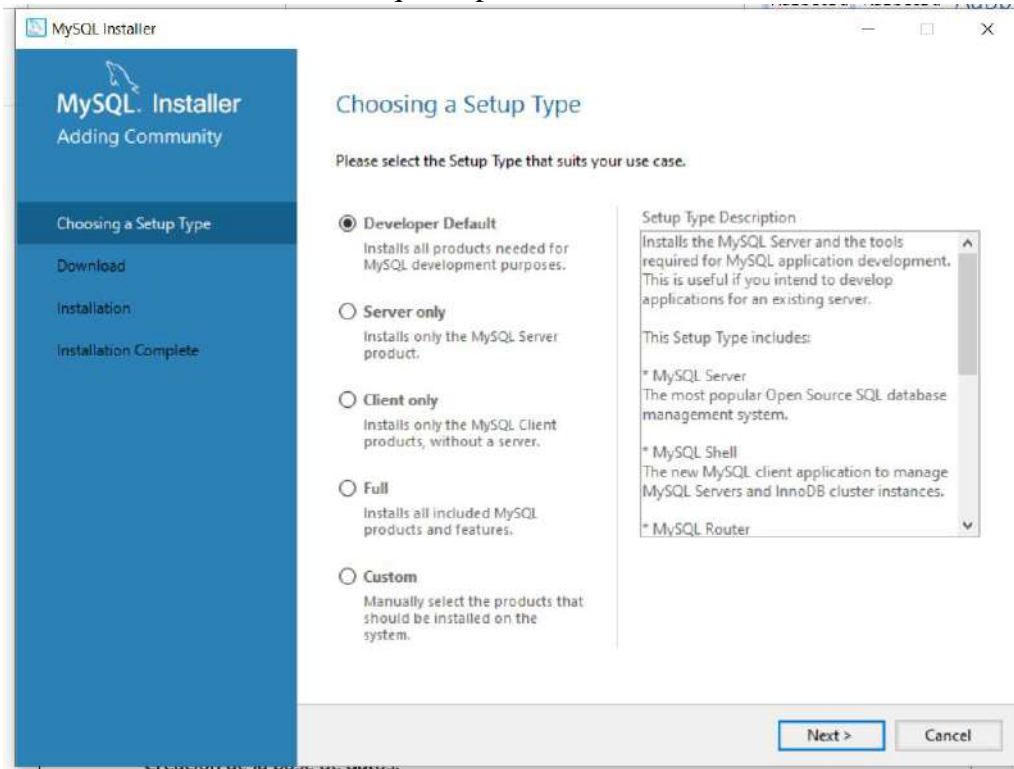
1.1.Instalación de MySQL

Para poder instalar MySQL debemos ingresar al navegador y buscar el link de descargar para la instalación del mismo con su actualización mas reciente que en este caso es 8.0.33.



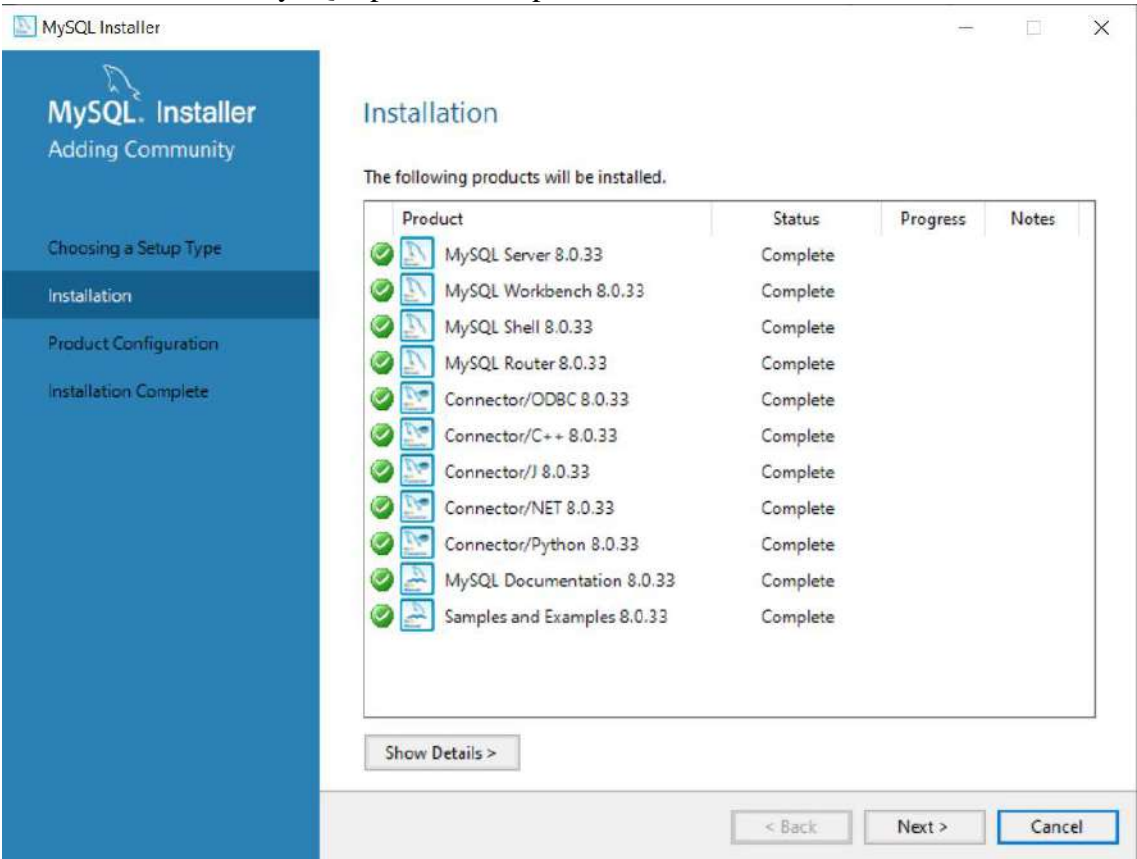
Tras ejecutar el instalador nos muestra una ventana para continuar con la misma nos

muestra diferentes servidores que se pueden instalar.

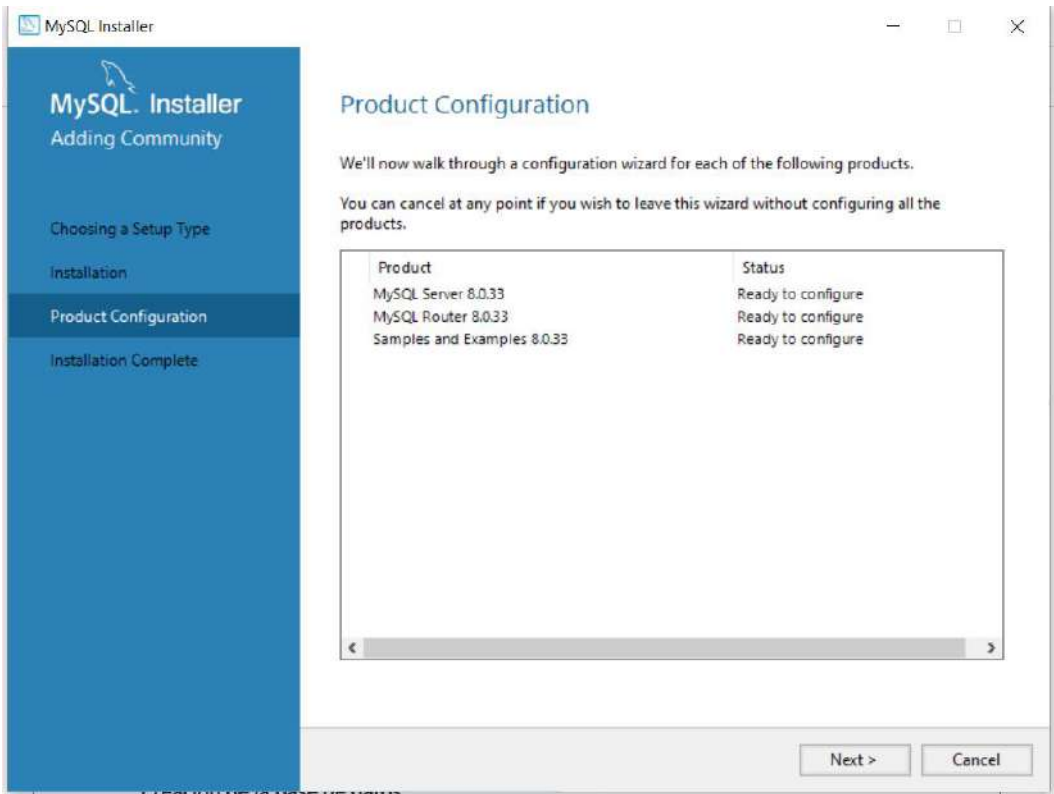


Se instalara el caso Developer Default porque incluye en su mayoría todos los recursos que nos otorga los otros casos: Server only, Client only y Full, la opción Custom simplemente nos permite añadir diferentes productos a elección.

En la pestaña de Installation nos pedira que descarguemos los recursos para el funcionamiento de MySQL, por ellos aceptamos e instalamos.



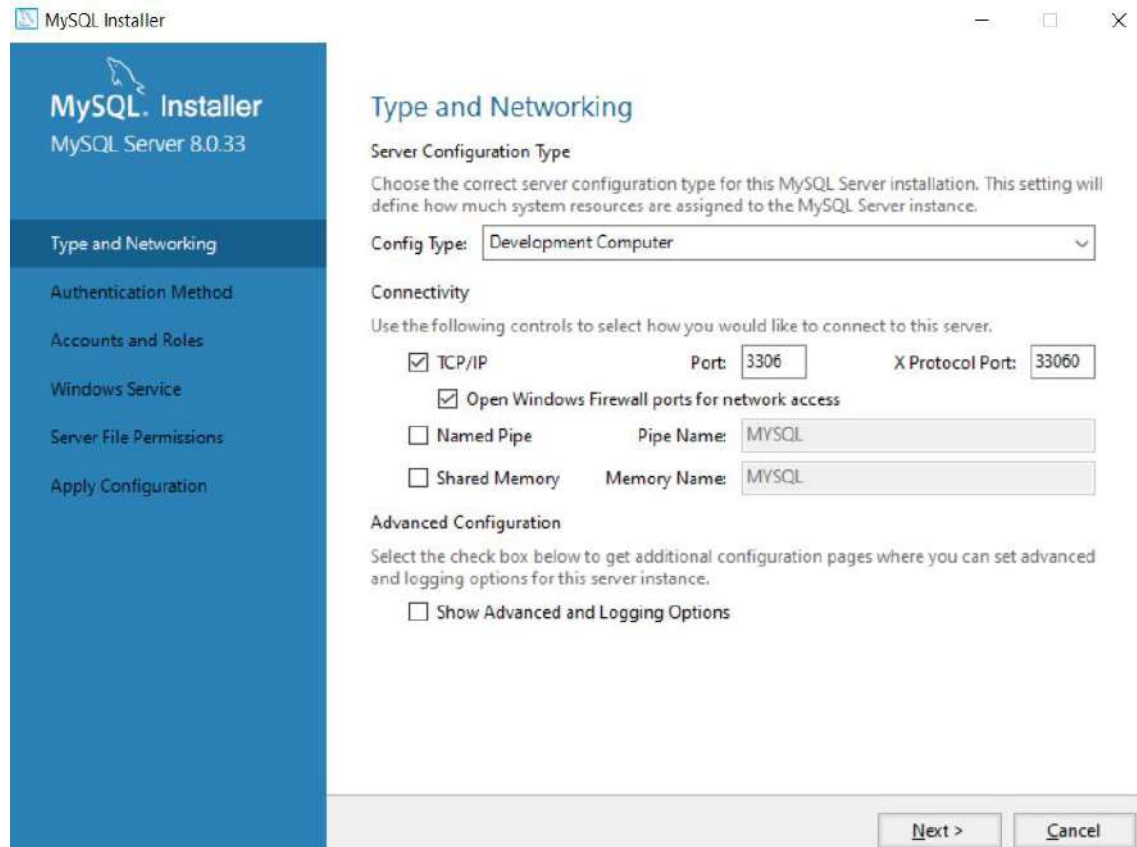
En la pestaña Product Configuration configuraremos el Servidor y comprobaremos su conexión.



El Servidor tiene diferentes pestañas a configurar Type and Networking, Authentication Method, Accounts and Roles, Windows Service, Server File Permissions y Apply Configuration.

Type and Networking.-

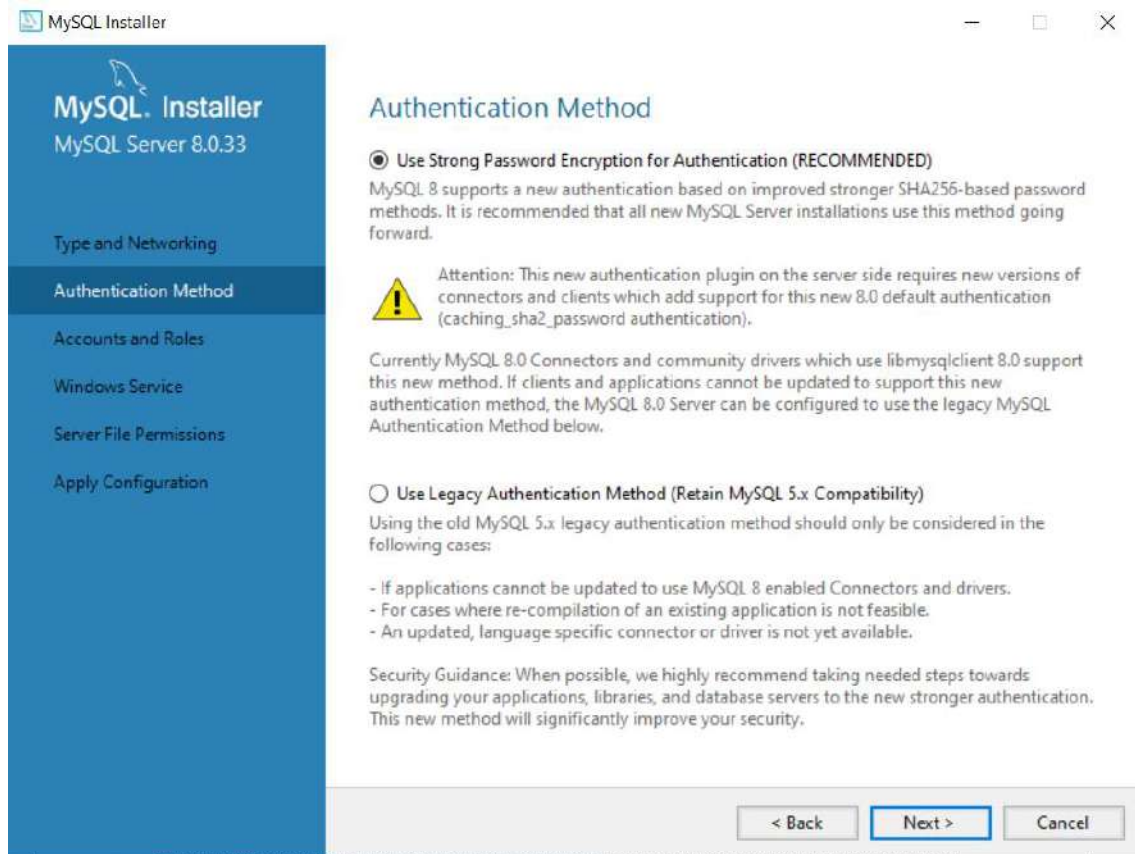
Con el Config Type lo mantendremos en Development Computer, en Connectivity marcamos TCP/IP, el Port por default es 3306, si ya se tiene un servidor con este port se recomienda continuar con el siguiente por ejemplo 3307 pero como no es el caso lo mantendremos igual, X Protocol Port por default tiene 33060, lo dejamos tal como esta y procedemos con el siguiente.



Authentication Method.-

En la autenticidad para la conexión del servicio, este nos recomienda el “Strong Password Encryption for Authentication” ya que sus características son soportadas por MYSQL 8, mientras que la opción “Legacy Authentication Method (Retain MYSQL 5.x Compatibility)” para la conectividad con MYSQL 8 y versiones anteriores y no se la recomienda por el hecho de que al tener los recursos mas actualizados es mejor ir por la

opcion recomendada por lo tanto la utilizaremos.



Accounts and Roles.-

Se define una contraseña para el usuario root por defecto que tiene MYSQL y como grupo decidemos la contraseña “grupo3ijpr”, se podría añadir mas usuario pero

usaremos por el momento el usuario root.

The screenshot shows the 'MySQL Installer' window for 'MySQL Server 8.0.33'. The left sidebar contains a list of steps: 'Type and Networking', 'Authentication Method', 'Accounts and Roles' (which is highlighted), 'Windows Service', 'Server File Permissions', and 'Apply Configuration'. The main area is titled 'Accounts and Roles' and contains the following sections:

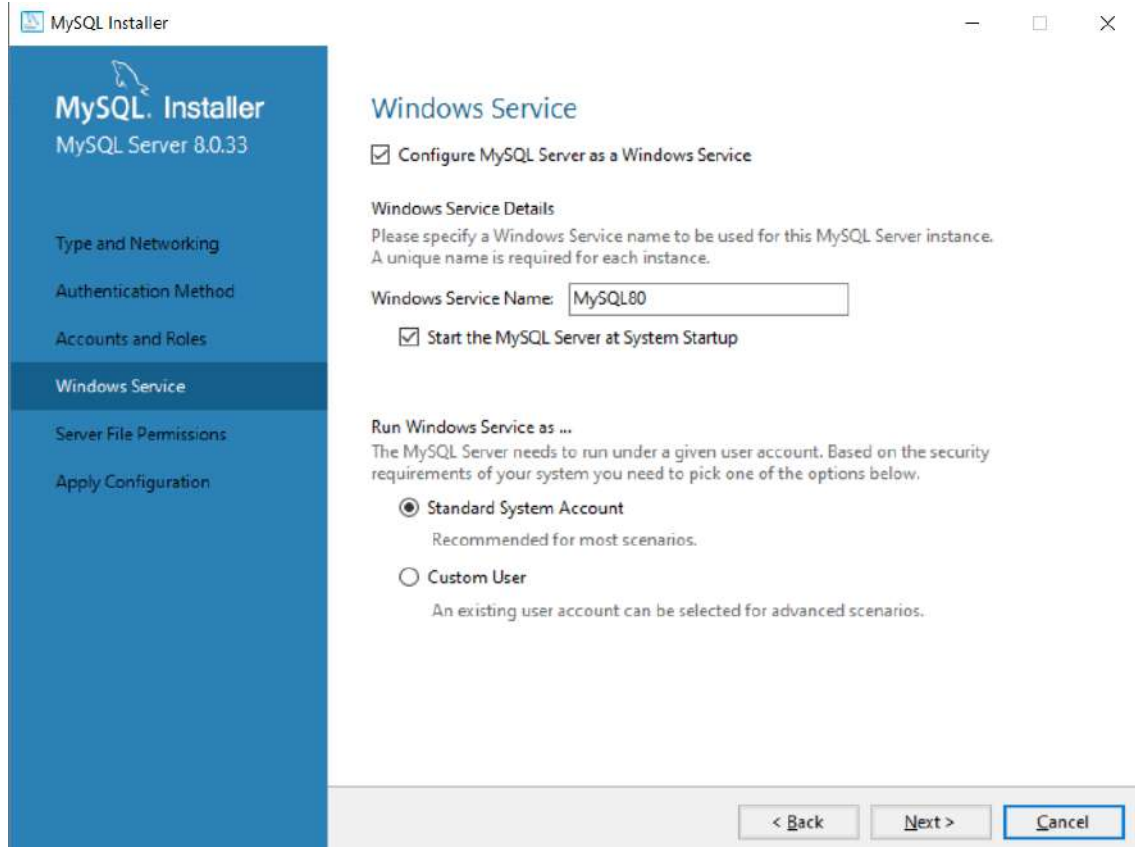
- Root Account Password:** A section with the instruction 'Enter the password for the root account. Please remember to store this password in a secure place.' It includes two password input fields: 'MySQL Root Password:' and 'Repeat Password:'. Below these fields, the 'Password strength' is indicated as 'Weak' in red text.
- MySQL User Accounts:** A section with the instruction 'Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.' It features a table with three columns: 'MySQL User Name', 'Host', and 'User Role'. To the right of the table are three buttons: 'Add User', 'Edit User', and 'Delete'.

At the bottom of the window, there are three navigation buttons: '< Back', 'Next >', and 'Cancel'.

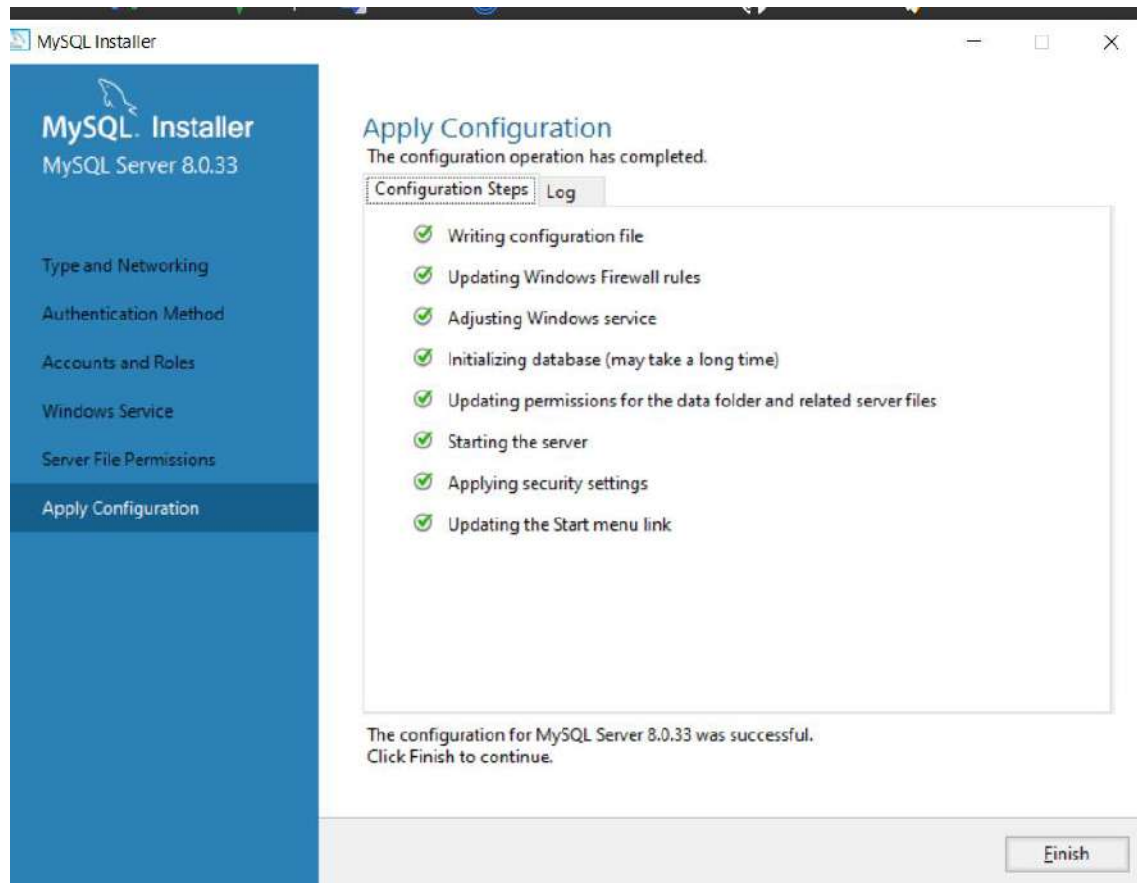
Windows Service.-

Mantendremos que se configure MYSQL Server para el servicio de Windows, este nos otorgo un nombre por defecto llamado MySQL80 esto lo podemos cambiar pero lo

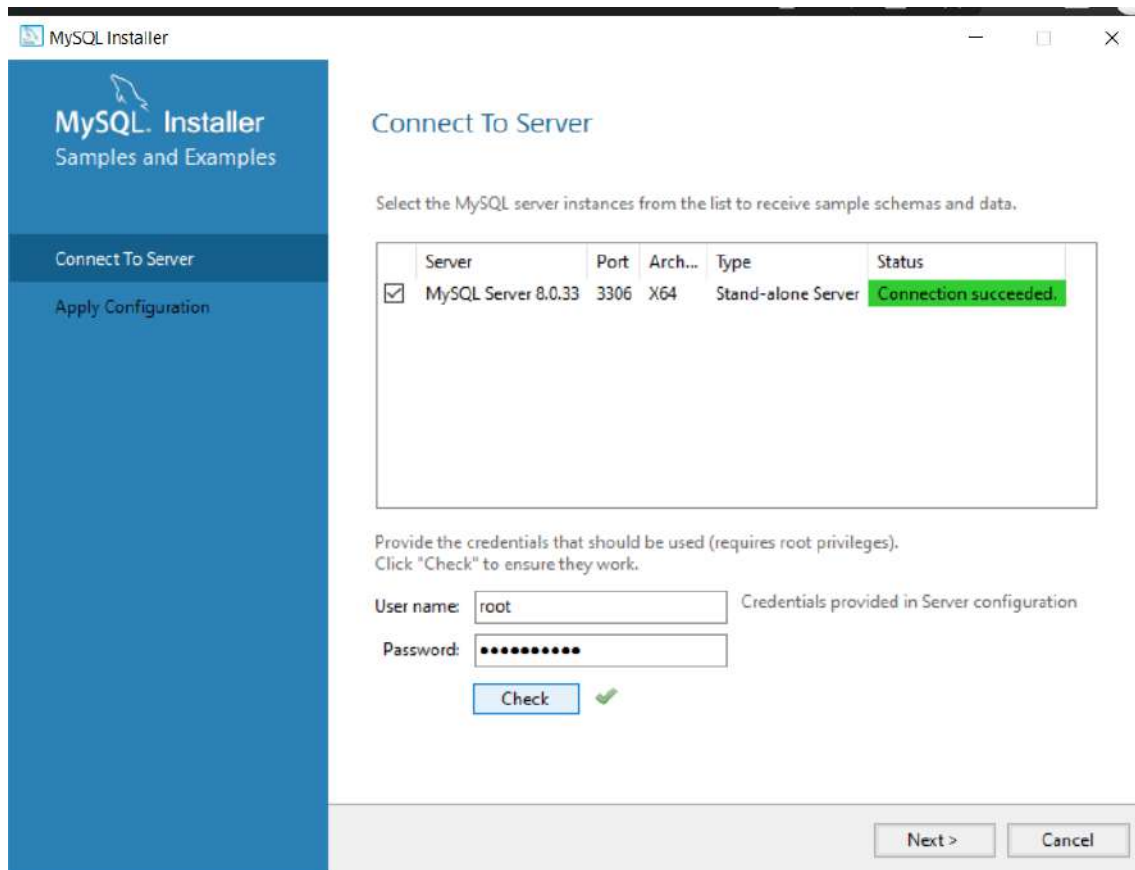
mantedremos por el momento a su vez la opcion de manera estándar.



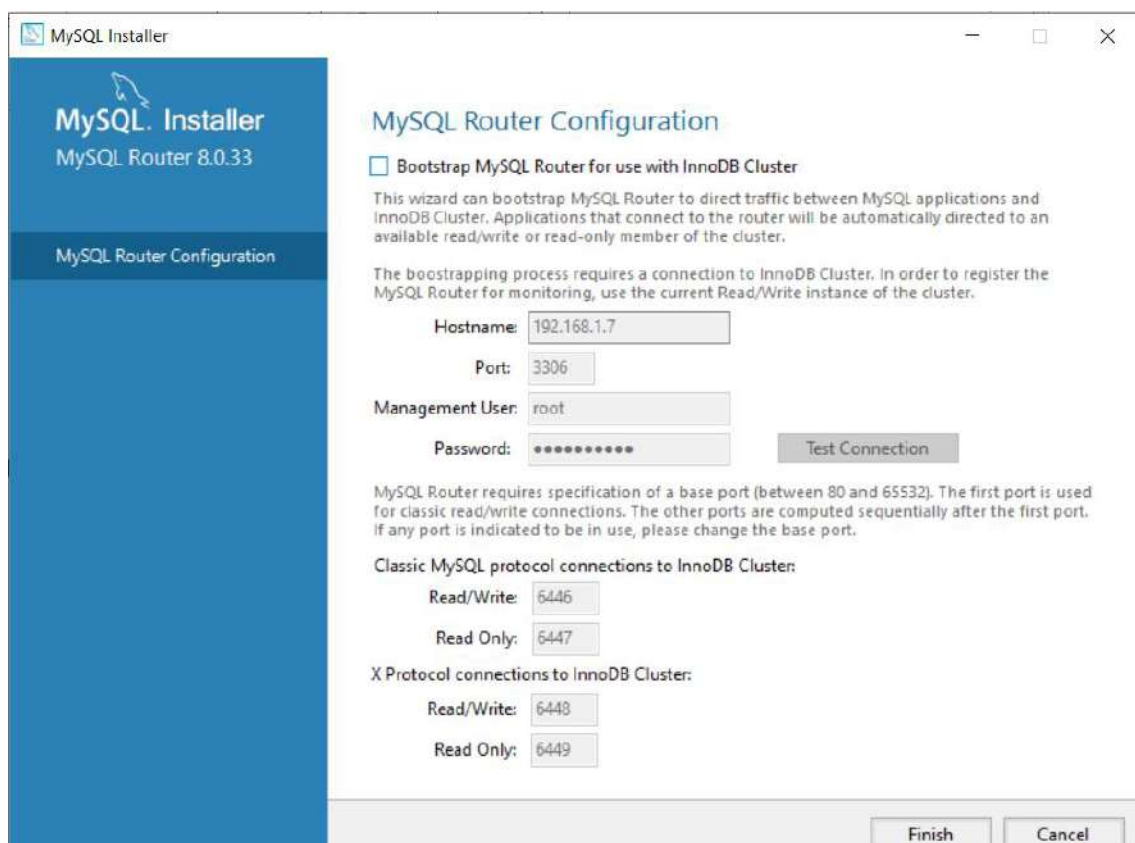
En la opción "Apply Configuration", presionamos "EXECUTE" para activar todos los recursos de trabajo.



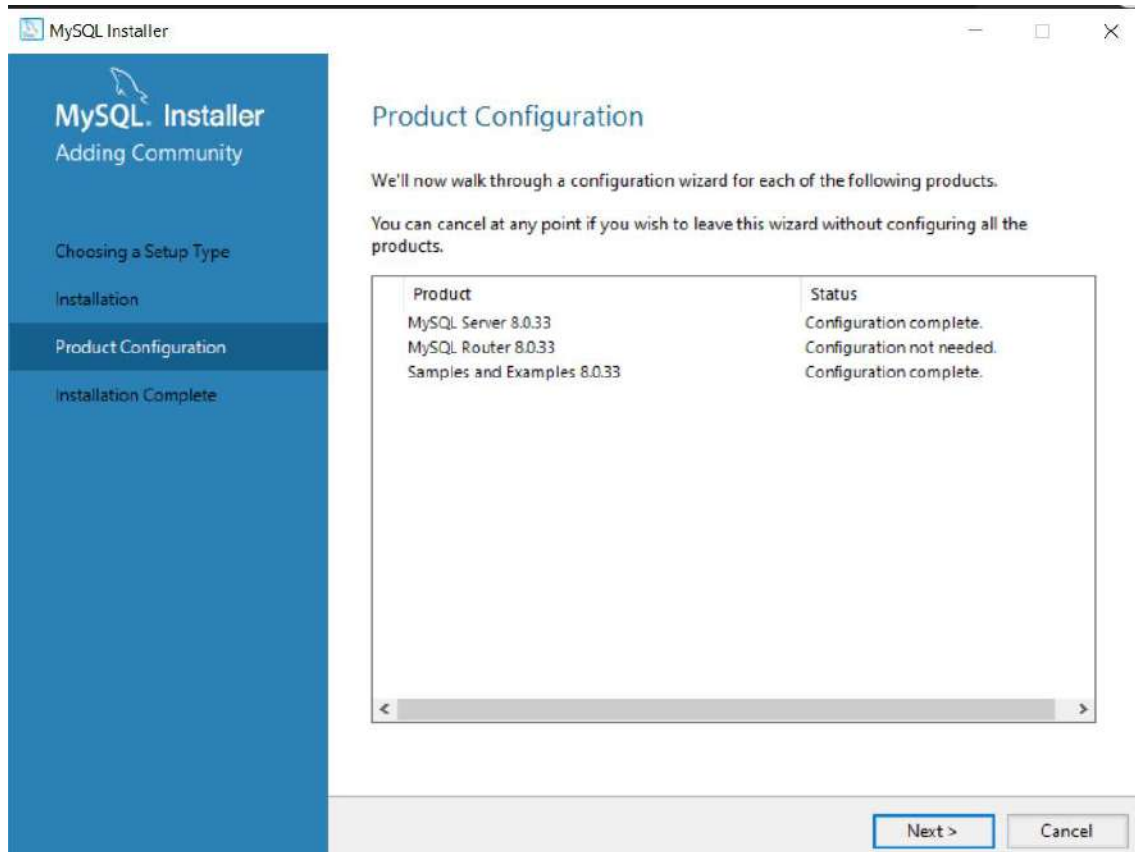
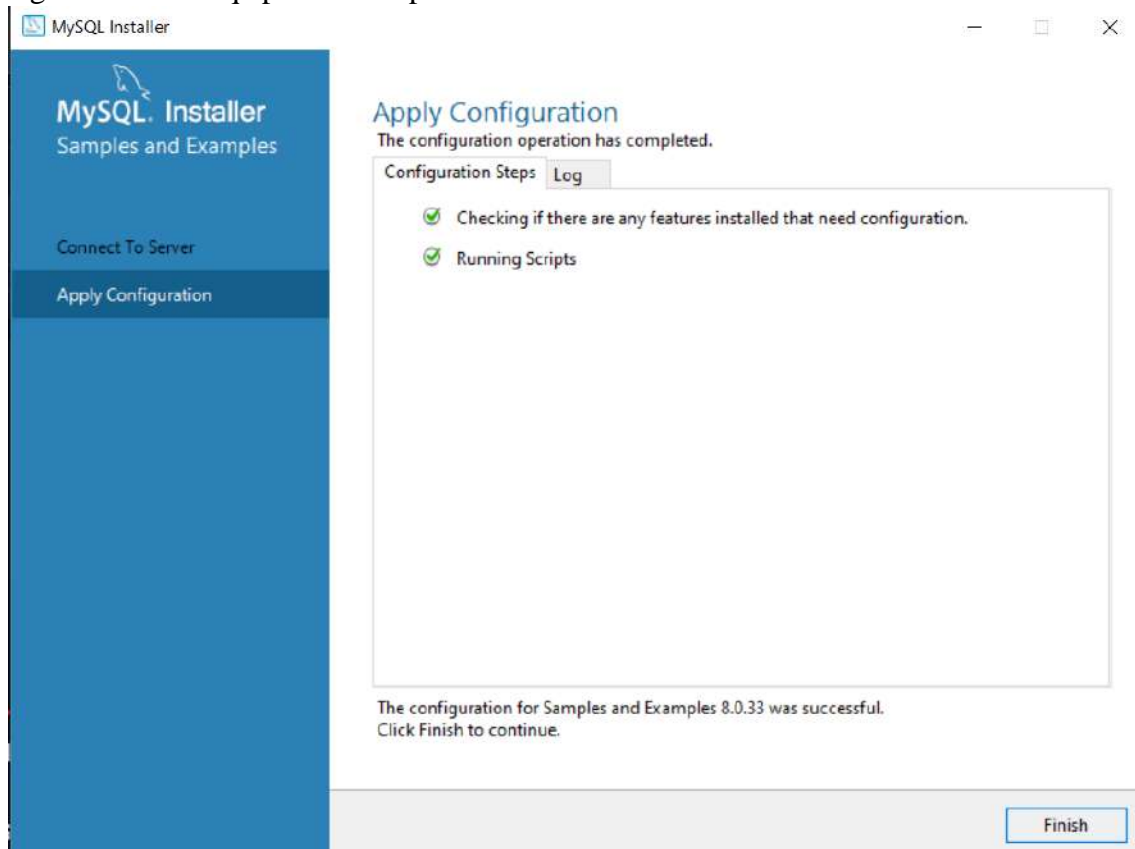
Y precionanos Finish ,posterior nos saldra la sigueinte ventana



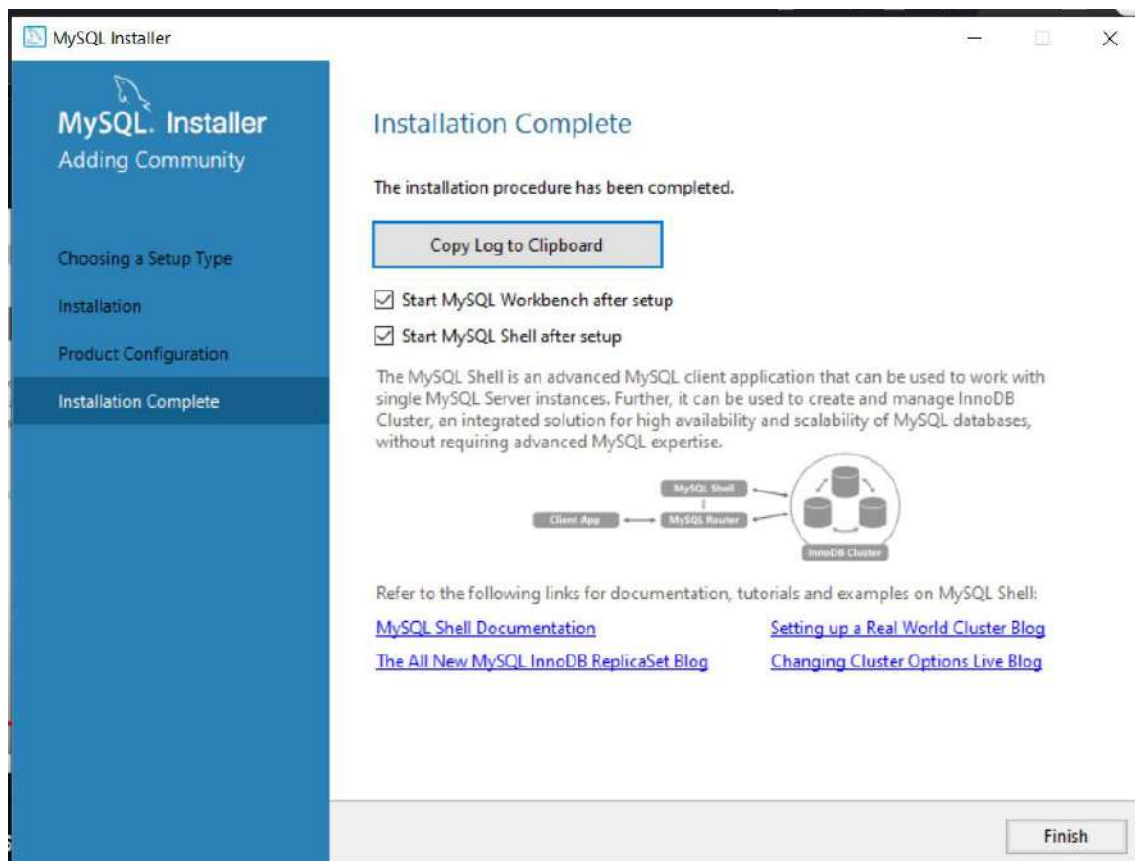
se nos solicitara el host la contraseña ,llenamos y presionamos next



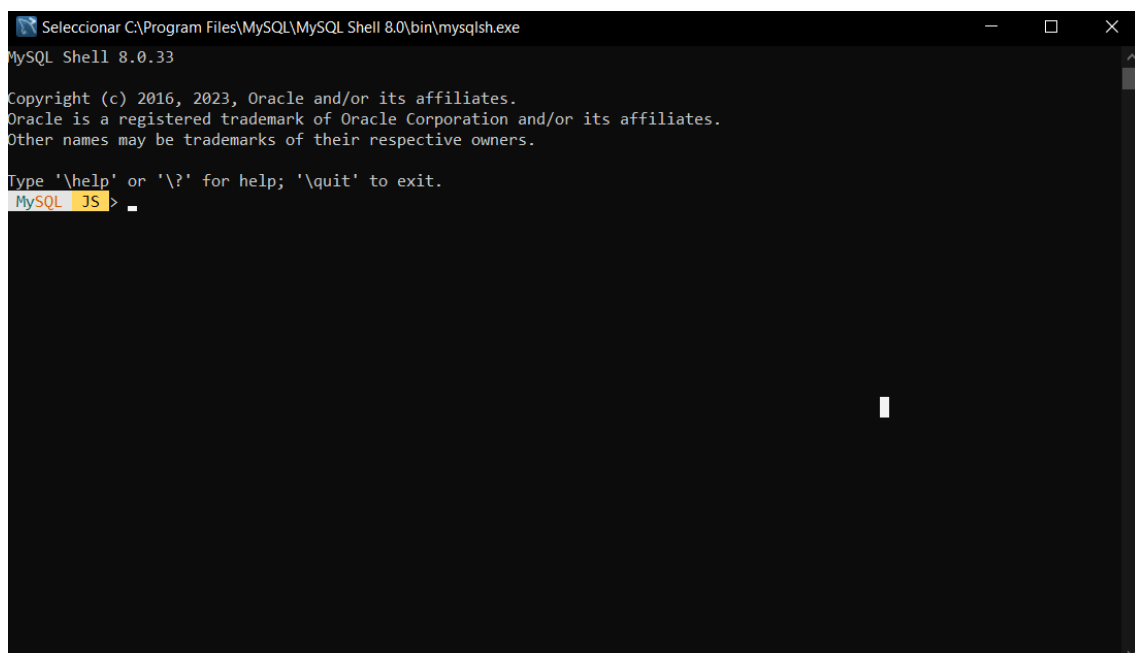
Luego aparecera una nueva ventana y precionamos execute y posterior finalizar,a la sigiente ventana q aparecera le ponemos next

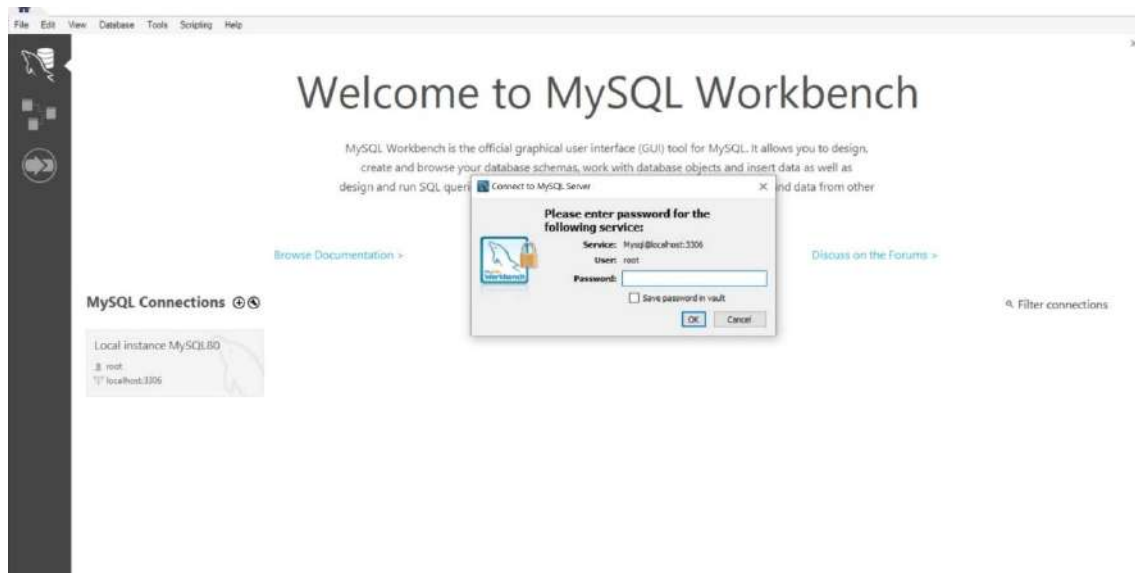


Y por ultimo finalizar



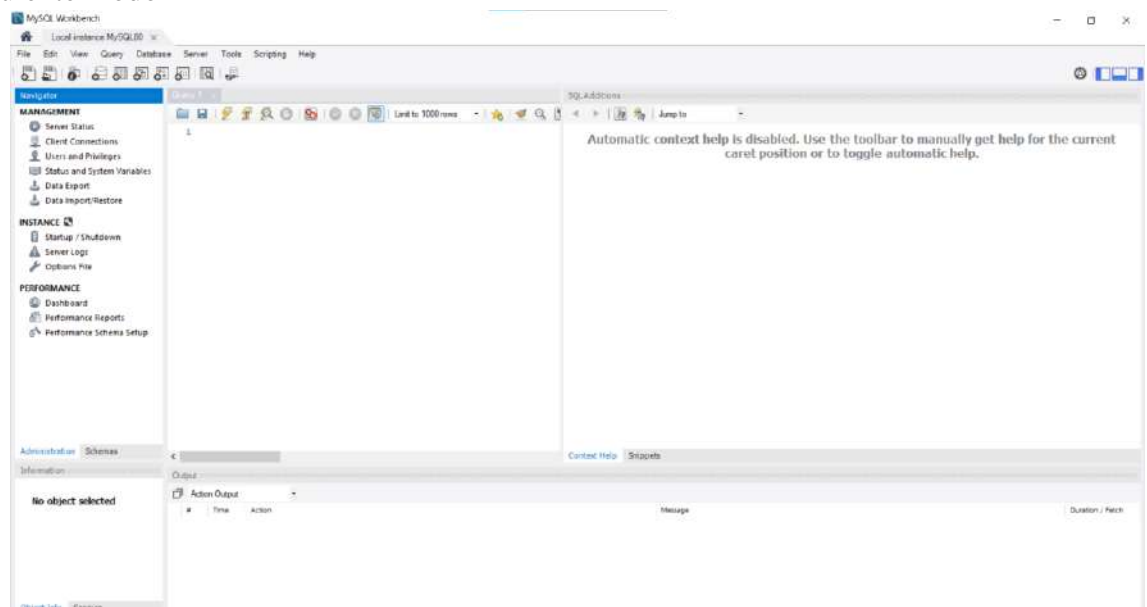
Y ya tenemos instalado MySQL por consola y workbains



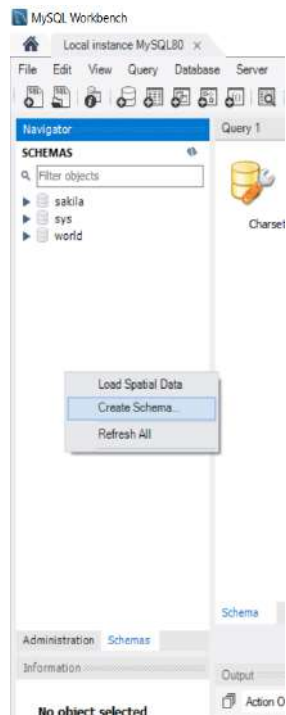


1.2.Creación de la base de datos.

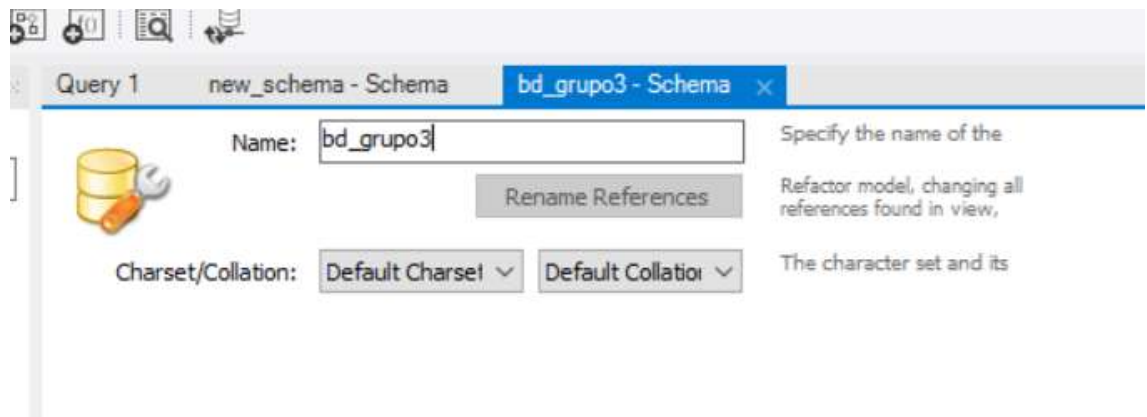
Para crear la base de datos de nuestro proyecto ingresamos al work bench e ingresamos la contraseña que creamos al momento de instalar MySQL,después la pantalla se mirara del siguiente modo



Luego vamos a “schemas” y en un espacio en blanco damos click derecho y seleccionamos “créate schema”

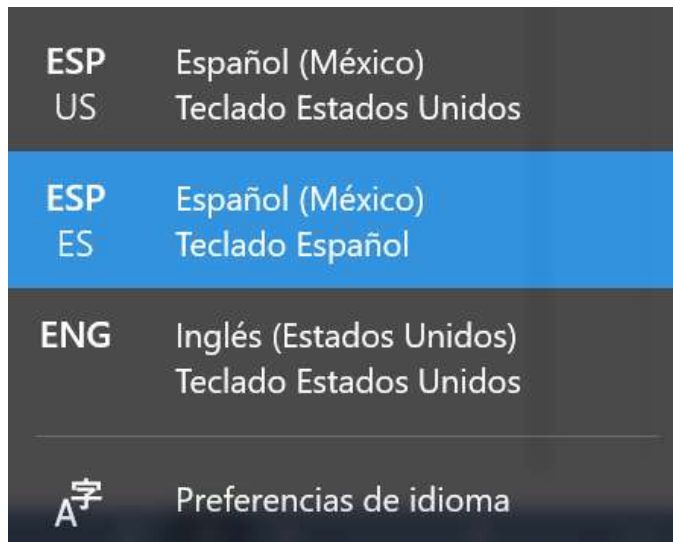


Luego aparecerá lo siguiente ,y renombraremos a esquema

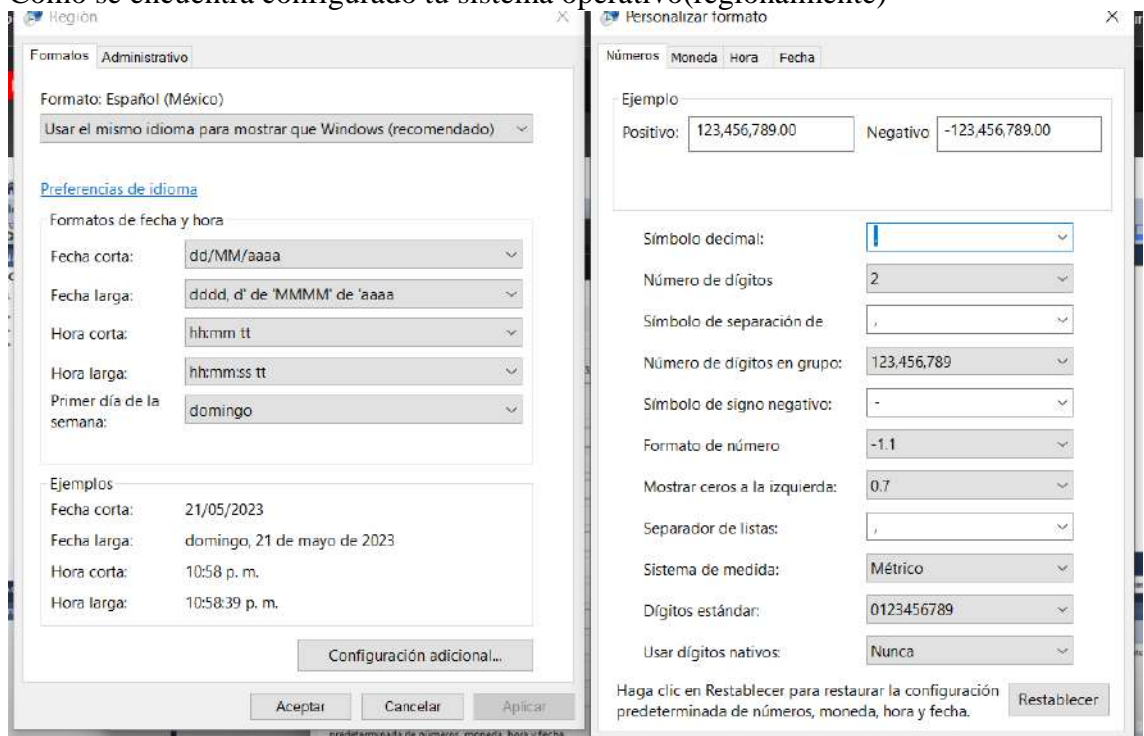


Antes de presionar aplicar se debe tomar en cuenta lo siguiente:

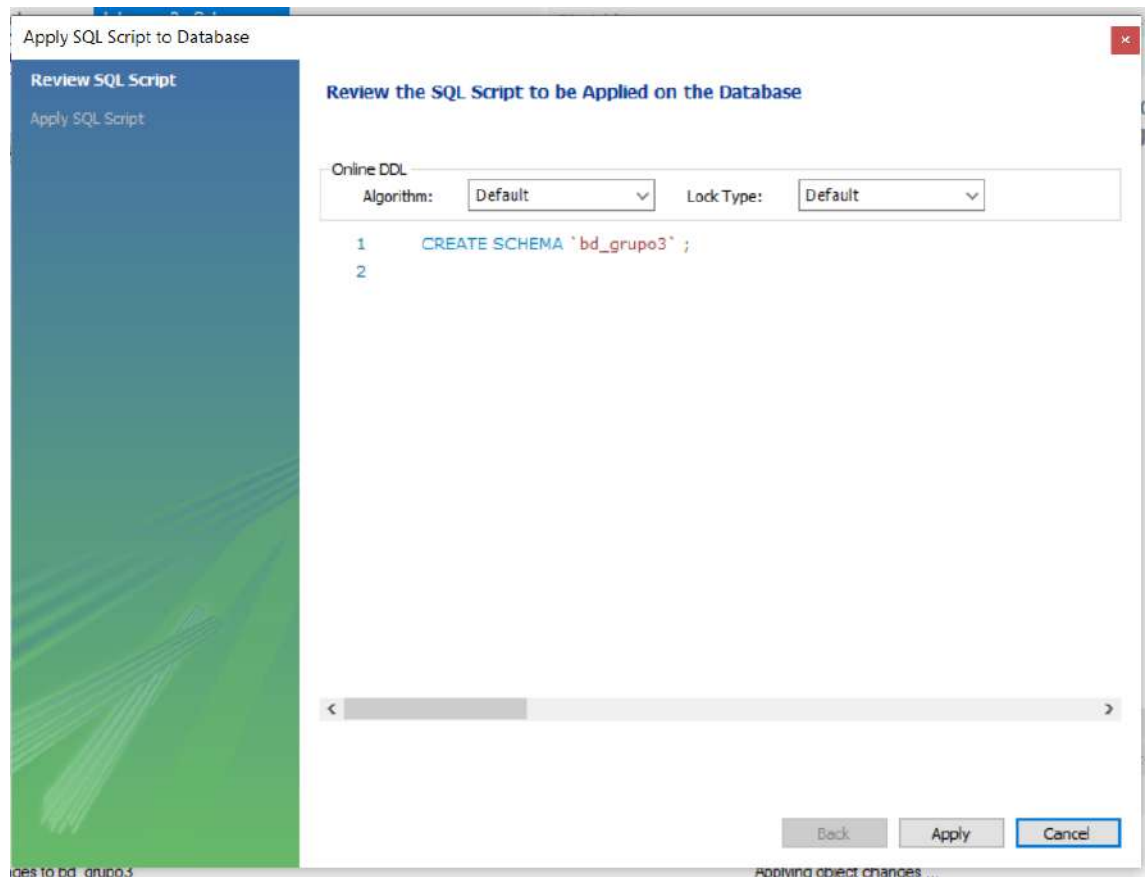
- El idioma en el q se encuentra su pc



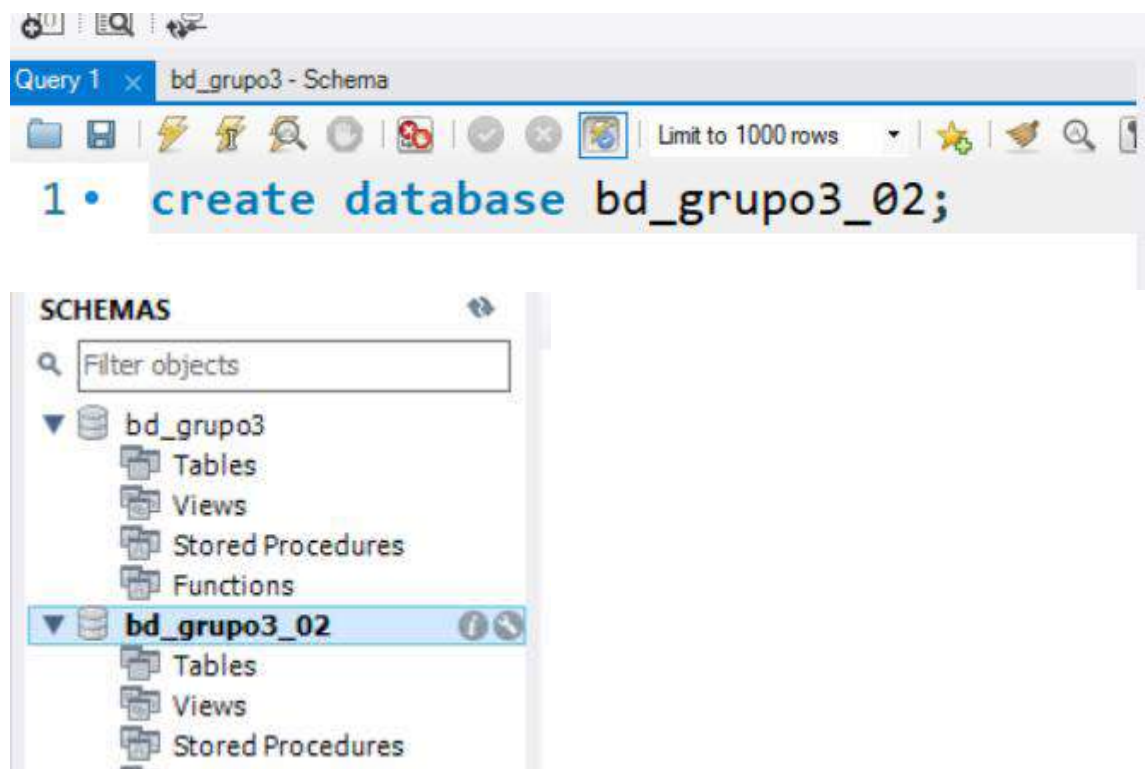
- Como se encuentra configurado tu sistema operativo(regionalmente)



Luego presionamos aplicar, en la ventana q aparecerá presionaremos apli nuevamente y por último “finish”.



Ahora procederemos a crear nuestra base de datos a través de los siguientes comandos al escribirlos los ejecutamos y nos podremos dar cuenta cómo va quedando nuestra estructura de base de datos



Y así tendremos lista para utilizar nuestra base de datos que es básicamente un esquema.

2. CONEXIÓN DE LA BASE DE DATOS

2.2.Crear un nuevo proyecto.

Este proyecto contendrá el código y a parte el uso de una librería externa usada para conectar a Netbeans con la base de datos de MySQL.

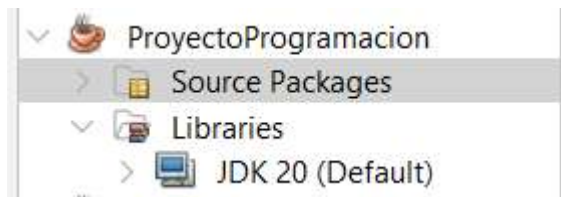


Figura 1.

Dentro de las carpetas del proyecto, añadiremos una nueva librería dentro de “Libraries” como se muestra en la figura 2 y 3.

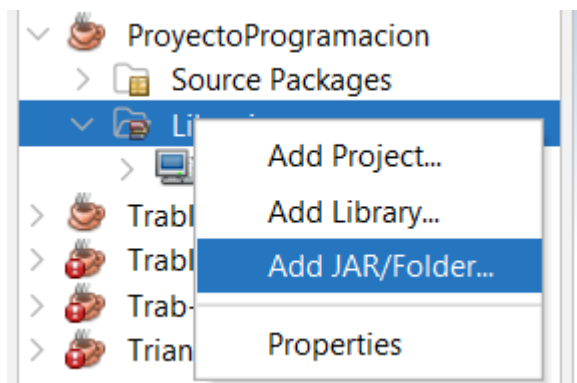


Figura 2.

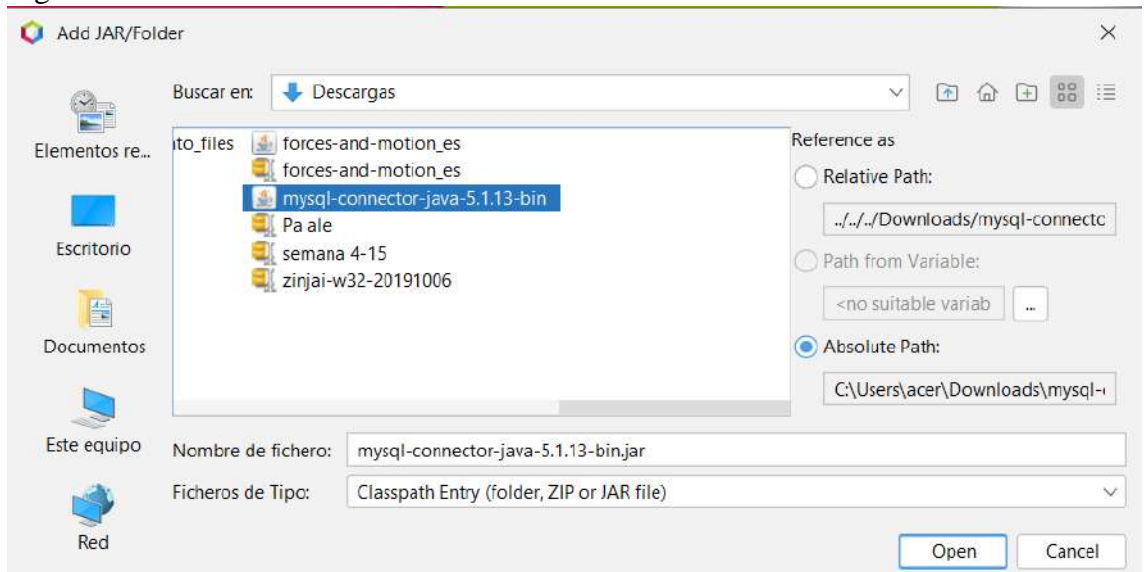


Figura 3.

2.2.Crear un paquete denominado “Vista”.

Dentro de Source Packages guardamos Vista.

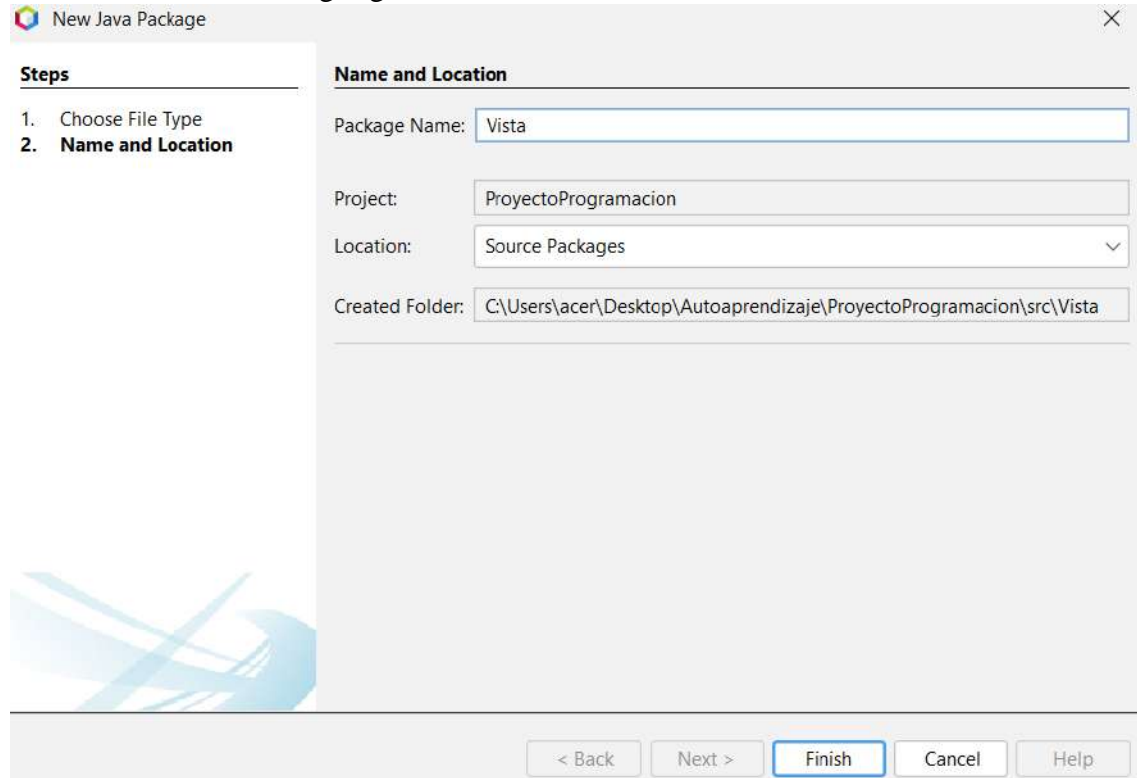


Figura 4.

En Vista creamos un JFrame denominado “JFconexion”.

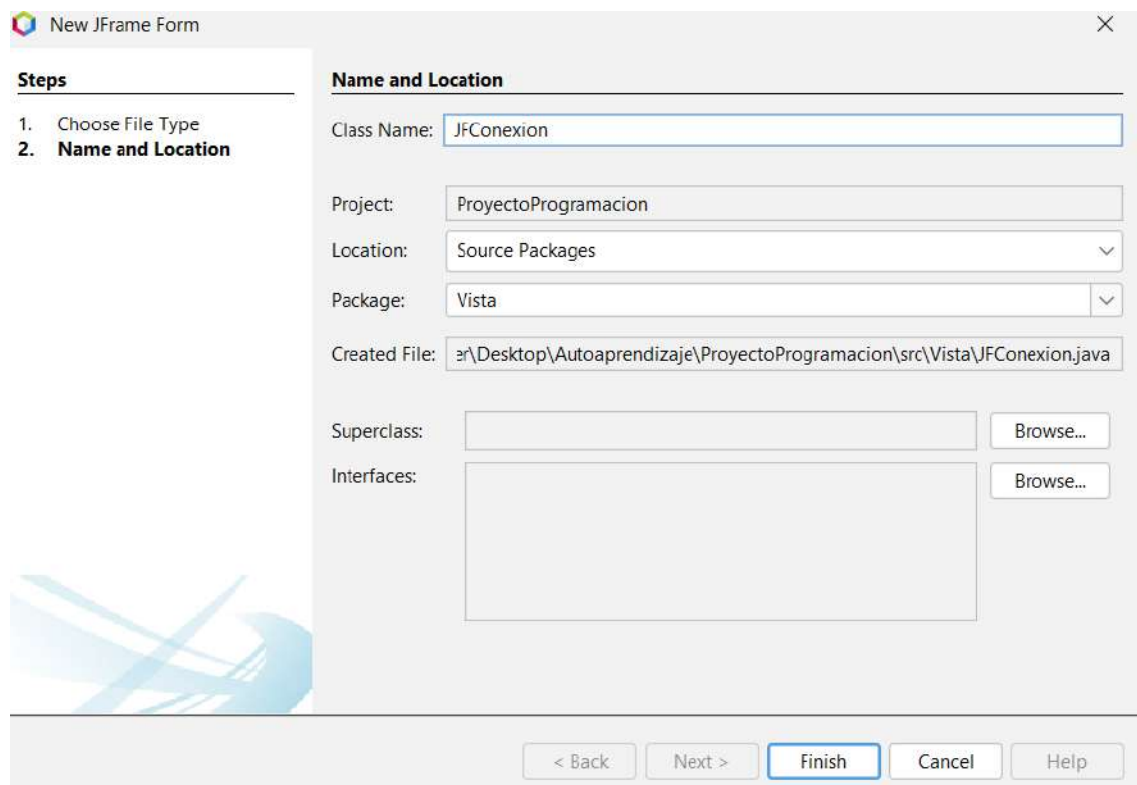
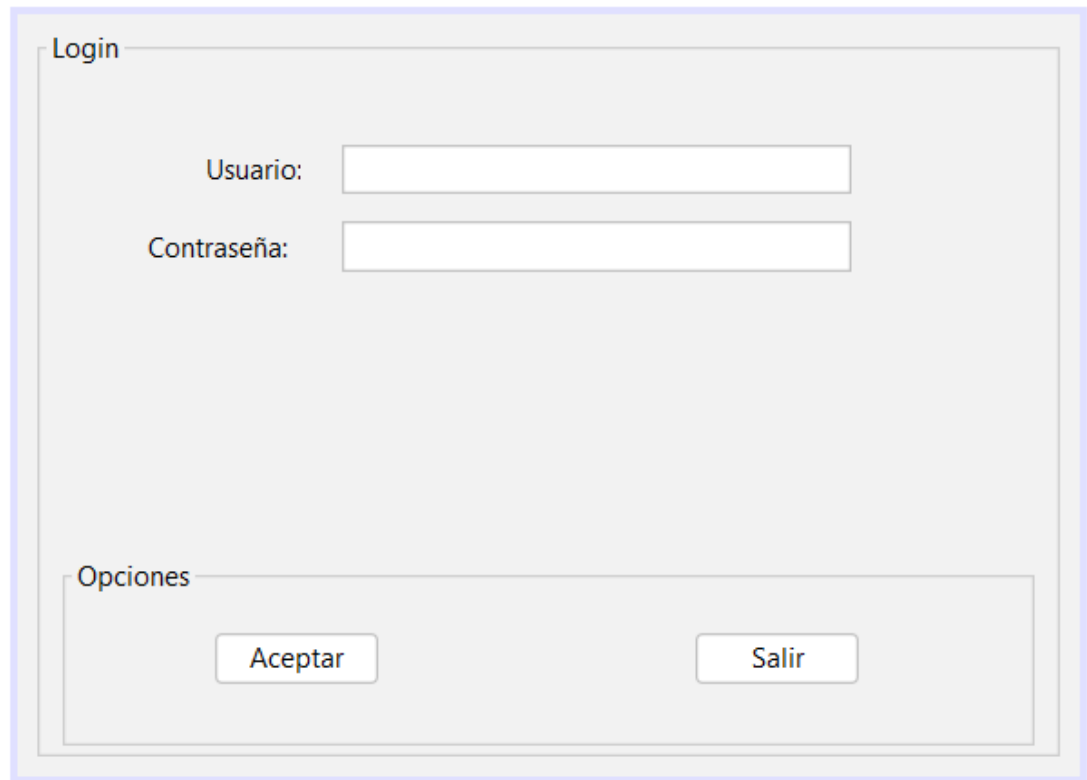


Figura 5.

2.3. Interfaz del programa y el código a usar.

Este es la interfaz del login que usaremos para conectar Netbeans con la base de datos.



The image shows a Java Swing window titled "Login". It contains two text input fields: "Usuario:" and "Contraseña:". Below these fields is a section titled "Opciones" which contains two buttons: "Aceptar" and "Salir". The window has a light gray background and a blue border.

Figura 6.

Ahora en el código, declararemos una variable de tipo Connection dentro de una clase que medirá la conexión importada de una de las librerías mencionada anteriormente.

```

    * @author acer
    */
    public class JFConexion extends javax.
        public static Connection con;

```

Figura 7.

Estos son todas las librerías a usar (figura 8).

```

import com.mysql.jdbc.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

```

Figura 8.

Finalmente, el código queda de esta manera, creamos datos de tipo String. El primer denotado bd que tiene el nombre de la base de datos y puerto de la base de datos, dos String más para el usuario y contraseña. Después creamos una función llamada

conector, que contiene un try and catch para un posible resultado erróneo al momento de ingresar datos incorrectos.

DriverManager tiene una función que recibe como parámetros 3 String, se localizaran la url, usuario y contraseña, estos dos últimos son ingresados por el usuario. El dato con se llenada de null si la conexión fue exitosa y desplegara el respectivo mensaje, si no se completó, dentro del catch desplegara otro mensaje de error.

```
Connection conector = null;
String bd = "bd_grupo3";
String ip = "localhost";
String puerto = "3306";

String cadena = "jdbc:mysql://" + ip + ":" + puerto + "/" + bd;

public Connection establecerConexion(String user, String password){
    try{
        Class.forName("com.mysql.jdbc.Driver");
        conector = DriverManager.getConnection(cadena, user, password);
        JOptionPane.showMessageDialog(null, "Conexion exitosa ");
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, "No se pudo realizar la conexion con la base de datos. Error: " + e.toString());
    }
    return conector;
}
```

Figura 9.

2.4.Ejecución del programa.

Al momento de la ejecución, aparecerá esta interfaz (figura 10).

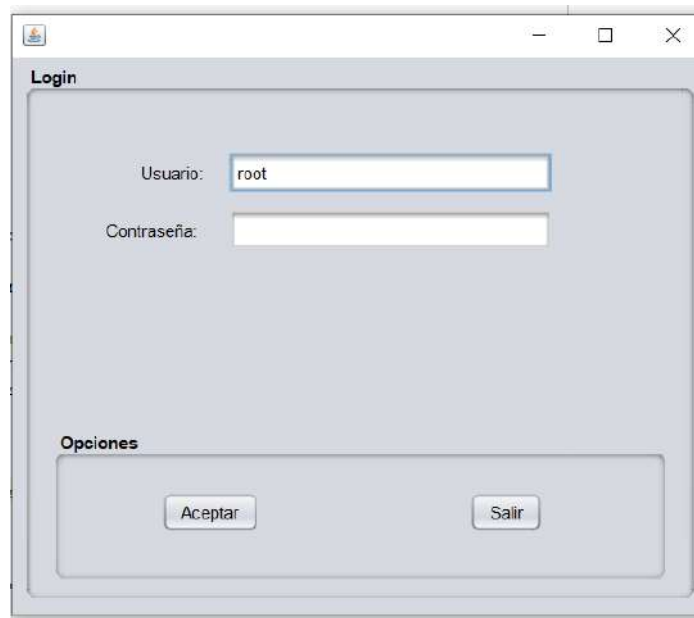


Figura 10.

Si se ingresan credenciales incorrectas, aparecerá un mensaje de error.

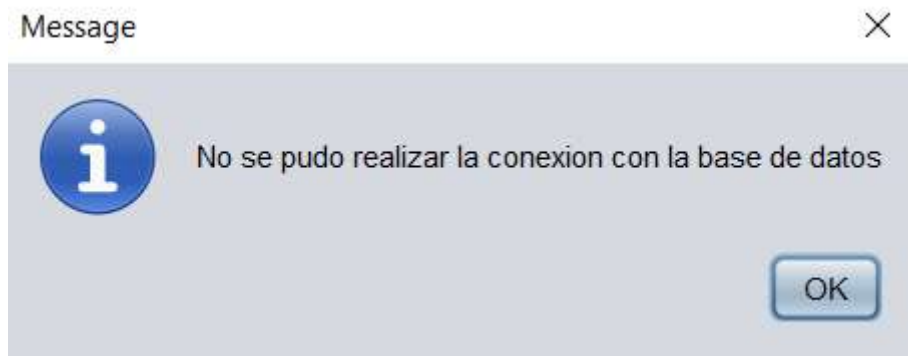


Figura 11.

Si se ingresa las credenciales correctas (figura 12).



Figura 14.

3. CREACIÓN DE LAS TABLAS Y SUS RELACIONES

3.1.Creación de Tablas

En My SQL primero se seleccionó la base de datos en con el comando “use” seguido por el nombre de la base de datos.

```
use bd_grupo3;
```

Para crear una tabla en My SQL se usó el comando “CREATE TABLE” de la siguiente forma.

```
CREATE TABLE Nombre_Tabla
(
  Nombre_Columna1  Tipo_de_Dato
(longitud),
  Nombre_Columna2  Tipo_de_Dato
(longitud),
  Nombre_Columna3  Tipo_de_Dato
(longitud),
  ....
);
```

Para la creación de la tabla es necesario elegir una columna como la primary key. La primary key (clave primaria) se utiliza para identificar de manera única cada registro en una tabla. Es una columna o un conjunto de columnas que garantiza la unicidad y la integridad de los datos en la tabla.

Para nuestra tabla “Vuelo” la primary key es “IdVuelo” y le colocamos la propiedad “not null” que obliga insertar el registro a esa columna. Nuestro código para la tabla “Vuelo” quedaría de la siguiente manera:

```
CREATE TABLE Vuelo
(
  IdVuelo int primary key not null,
  Vuelo_Origen Varchar (50)not null,
  Vuelo_Destino Varchar (50)not null,
  Vuelo_Hora Varchar (10)not null,
  Model_Avion Varchar (25)not null
);
```

Se uso un código similar para crear las tablas “Asientos” y “Clientes”:

```

CREATE TABLE Asiento
(
    IdAsiento int primary key not null,
    Ubicacion Varchar (50)not null,
    Clase Varchar (50)not null,
    Vueloid int not null
);

CREATE TABLE Cliente
(
    Vueloid int not null,
    Client

e_Cedula int primary key not null,
Cliente_Nombres Varchar (50)not
null,
Vueloid int not null
);

```

3.2. Relación de tablas

La relación entre tablas en MySQL se establece mediante el uso de claves foráneas (foreign keys) y claves primarias (primary keys). Estas claves permiten establecer vínculos entre los registros de diferentes tablas y mantener la integridad referencial en la base de datos. Por lo que se añadió una columna nueva a las tablas para que sean la Foreign key.

Para añadir una nueva columna a una tabla existente se usa el comando:

```

ALTER TABLE nombre de la tabla
Add          Nombre_Columna
Tipo_de_Dato (longitud);

```

De esta forma se añadió una nueva columna con el nombre “Vueloid” a las tablas “Cliente” y “Asiento”. Esta nueva columna va a ser la Foreign key que nos ayudara a relacionar la tabla “Vuelo” con las tablas.


```
ALTER TABLE Cliente  
Add Vueloid int (50) not null ;
```

```
ALTER TABLE Asiento  
Add Vueloid int (50) not null ;
```

Se uso una Relación uno a uno (One-to-One): En esta relación, un registro en una tabla está relacionado con exactamente un registro en otra tabla, y viceversa.

Para establecer esta relación, se modificó la tabla que va a contener la Foreign key en este caso la tablas “Asiento” y “Cliente”. Se usó el siguiente comando:

```
ALTER TABLE Nombre de la tabla  
ADD CONSTRAINT fk_nombre de la relación  
FOREIGN KEY (nombre del fk)  
REFERENCES nombre de la tabla que se va  
a relacionar (llave primaria);
```

En Nuestro caso nuestro código es:

```
Alter Table Asiento  
ADD CONSTRAINT fk_Asiento_Vuelo  
FOREIGN KEY (Vueloid)  
REFERENCES `vuelo`(IdVuelo);  
  
ALTER TABLE cliente  
ADD CONSTRAINT fk_Cliente_Vuelo  
FOREIGN KEY (Vueloid)  
REFERENCES vuelo(IdVuelo);
```

Captura de Ejecución

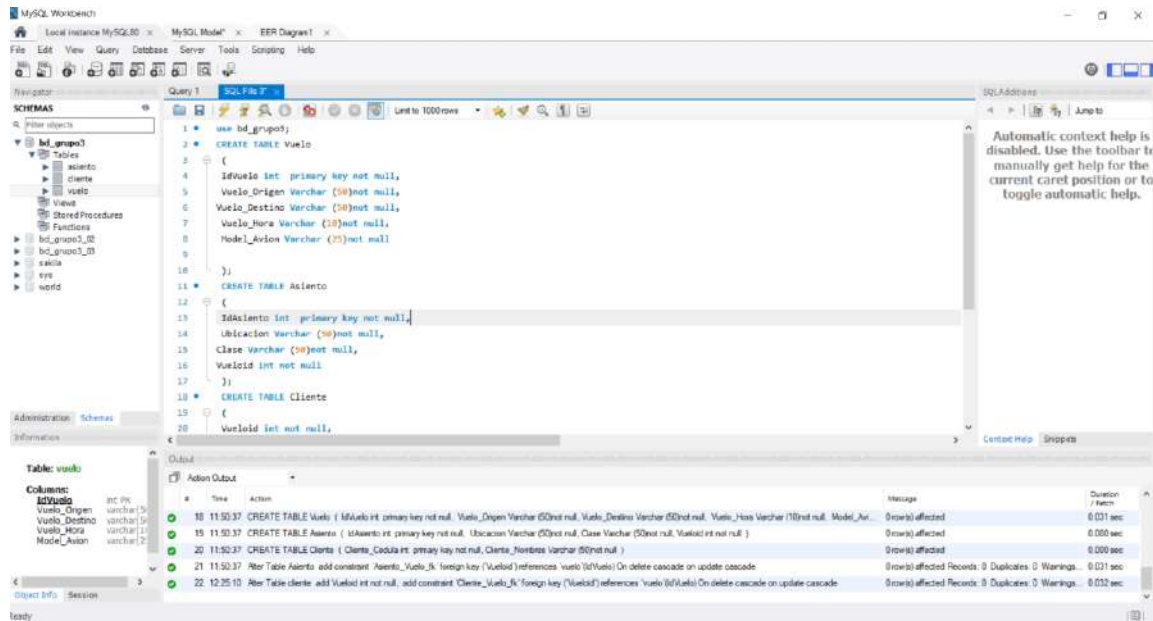
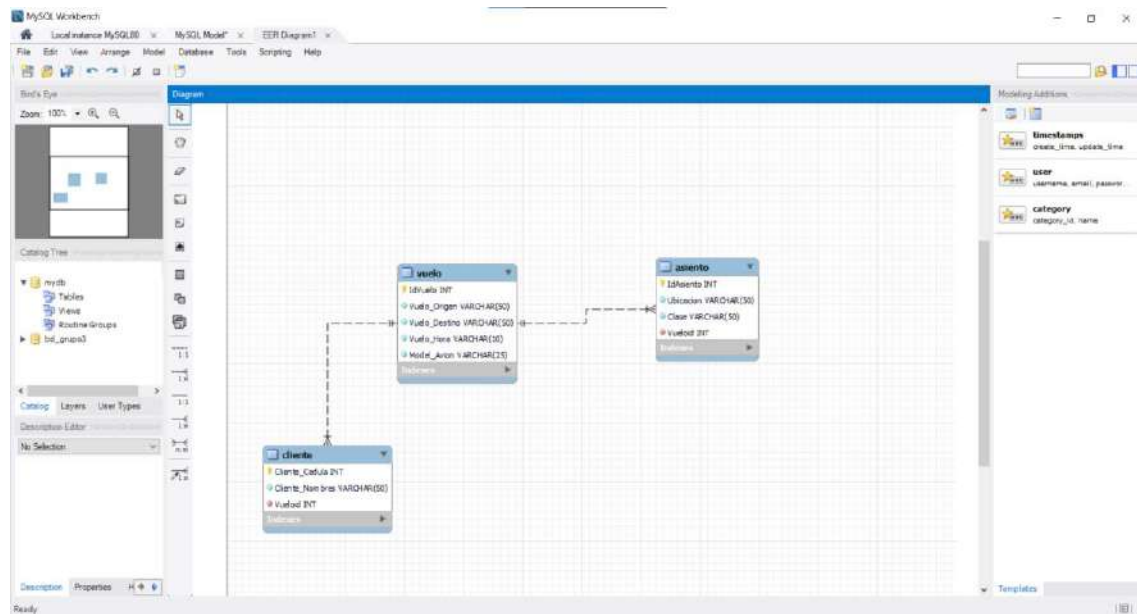


Diagrama de las Tablas



3.3.Creación Del Crud Para La Tabla “Vuelo”

Lo primero que se realizó fue el método de mostrar la tabla. Resumiendo lo que se realizó es:

Se define una variable String llamada "sql" para almacenar la consulta SQL que se utilizará más adelante.

Se crea un objeto DefaultTableModel para definir las columnas de la tabla y su modelo de datos.

Se agregan las columnas al modelo de datos utilizando el método addColumn().

Se establece el modelo de datos en el componente "visor" (que parece ser algún tipo de tabla o visor de datos).

Se verifica si el parámetro "id" está vacío. Si es así, se construye una consulta SQL para seleccionar todos los registros de la tabla especificada. De lo contrario, se construye una consulta SQL para seleccionar los registros que coincidan con el IdVuelo proporcionado.

Se crea un array de Strings llamado "datos" para almacenar los valores de cada columna de cada registro.

Se ejecuta la consulta SQL utilizando el objeto Statement y se obtiene un objeto ResultSet llamado "rs" que contiene los resultados de la consulta.

Se asignan los valores de cada columna del registro actual a las posiciones correspondientes del array "datos".

Se agrega el array "datos" al modelo de datos utilizando el método addRow().

Código:

```

public void mostrar (String tabla, String id){
    String sql = "";
    Statement st;
    DefaultTableModel model = new DefaultTableModel();
    model.addColumn(columnName: "ID");
    model.addColumn(columnName: "Origen");
    model.addColumn(columnName: "Destino");
    model.addColumn(columnName: "Hora de vuelo");
    visor.setModel(dataModel:model);
    if(id.equals(anObject: "")){
        sql="Select*from " + tabla;
    }else{
        sql="Select*from " + tabla + " where IdVuelo like'%" + id + "%'";
    }
    String [] datos = new String [4];
    try {
        st= conexion.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while (rs.next()){
            datos[0]=rs.getString(columnIndex: 1);
            datos[1]=rs.getString(columnIndex: 2);
            datos[2]=rs.getString(columnIndex: 3);
            datos[3]=rs.getString(columnIndex: 4);
            model.addRow(rowData: datos);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Conexion fallida");
    }
}

```

Luego se creó el Agregar. Primero se ejecuta una consulta en la tabla "vuelo" para verificar si ya existe un registro con el mismo valor de IdVuelo ingresado en el campo de texto "JTFid". Se utiliza Integer.parseInt() para convertir el valor del campo de texto a un número entero.

Si se encuentra algún resultado en el conjunto de resultados (resultado.next()), se muestra un mensaje emergente que indica que la ID no puede repetirse. Esta parte del código valida la unicidad de la clave primaria (IdVuelo) antes de insertar un nuevo registro.

Se crea un objeto PreparedStatement para realizar una inserción en la tabla "vuelo". El valor de cada columna se establece utilizando los métodos setInt() y setString() según corresponda.

Se ejecuta la instrucción de inserción utilizando el método executeUpdate().

Código:

```

private void JBagregarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Statement set=conexion.createStatement();
        ResultSet resultado=set.executeQuery("SELECT * FROM vuelo WHERE IdVuelo LIKE '"+Integer.parseInt(a: this.JTFid.getText())+"'");
        if(resultado.next()){
            getToolkit().beep();
            JOptionPane.showMessageDialog(parentComponent: null,message: "La ID no puede repetirse");
            this.JTFid.requestFocus();//comprobacion de la primari key
        }
        else {
            PreparedStatement pps = conexion.prepareStatement(sql:"INSERT INTO vuelo VALUES(?,?,?,?)");
            pps.setInt(parameterIndex: 1, x: Integer.parseInt(a: this.JTFid.getText()));
            pps.setString(parameterIndex: 2, x: this.JTForigen.getText());
            pps.setString(parameterIndex: 3, x: this.JTFdestino.getText());
            pps.setString(parameterIndex: 4, x: this.JTFhora.getText());
            pps.executeUpdate();
            this.limpiar();
        }
    }
    catch (SQLException ex ) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "No se pudo ingresar los datos correctamente, existen espacios en blanco");
        Logger.getLogger(name: JFVuelo.class.getName()).log(level: Level.SEVERE, msg:null, thrown:ex);
    }
    mostrar(tabla: "vuelo",id: "");
    // TODO add your handling code here:
}

```

Para Buscar un dato se usa el método mostrar creado anteriormente.

```

private void JTFbuscarKeyReleased(java.awt.event.KeyEvent evt) {
    this.mostrar(tabla: "vuelo", id: this.JTFbuscar.getText());
    // TODO add your handling code here:
}

```

Para modificar se crearon 2 botones el primero “modificar” y el botón “actualizar”

En el botón Se obtiene la fila seleccionada en el componente "visor" (que es la tabla) utilizando el método `getSelectedRow()`.

Se realiza una serie de acciones para configurar la interfaz de usuario de acuerdo con la acción de modificación. Por ejemplo, se ocultan ciertos botones (JBagregar, JBmodificar, JBeliminar, JBregresar) y se muestra el botón JBactualizar. Además, se establecen los valores de los campos de texto (JTFid, JTForigen, JTFdestino, JTFhora) utilizando los valores de la fila seleccionada en el componente "visor".

También se establece la propiedad editable del campo de texto JTFid en falso para evitar su edición.

Si no se selecciona ninguna fila, se muestra un mensaje emergente indicando que se debe seleccionar una fila para usar la función de modificación. El segundo botón “Actualizar” es la confirmación de la modificación

Se crea un objeto PreparedStatement para preparar una consulta de actualización en la tabla "Vuelo" utilizando los valores ingresados en los campos de texto (JTForigen, JTFdestino, JTFhora) y el valor de JTFid.

Se establecen los valores de los parámetros de la consulta utilizando los métodos `setString()` y `setInt()`.

Se ejecuta la consulta de actualización utilizando el método `executeUpdate()`.

Se muestra un mensaje emergente indicando que los datos se han actualizado correctamente.

Se realizan una serie de acciones para restablecer la interfaz de usuario a su estado inicial. Se muestran nuevamente los botones ocultos anteriormente, se oculta el botón JBactualizar y se restablecen los valores de los campos de texto.

Se llama al método "mostrar ()" para actualizar la visualización de la tabla "Vuelo" en la interfaz de usuario.

Código:

```
private void JBmodificarActionPerformed(java.awt.event.ActionEvent evt) {  
    int fila = visor.getSelectedRow();  
    if(fila >= 0){  
        this.JBagregar.setVisible(aFlag: false);  
        this.JBmodificar.setVisible(aFlag: false);  
        this.JBeliminar.setVisible(aFlag: false);  
        this.JBregresar.setVisible(aFlag: false);  
        JTFid.setText(t: visor.getValueAt(row: fila, column: 0).toString());  
        JTForigen.setText(t: visor.getValueAt(row: fila, column: 1).toString());  
        JTFdestino.setText(t: visor.getValueAt(row: fila, column: 2).toString());  
        JTFhora.setText(t: visor.getValueAt(row: fila, column: 3).toString());  
        this.JBactualizar.setVisible(aFlag: true);  
        this.JTFid.setEditable(b: false);  
    } else {  
        JOptionPane.showMessageDialog(parentComponent: null, message: "Busque la fila a modificar y haga un click en ella para usar esta funcion");  
    }  
    // TODO add your handling code here:  
}
```

```
private void JBactualizarActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        PreparedStatement pps = conexion.prepareStatement(  
            sql: "UPDATE Vuelo SET Vuelo_Origen = ?, Vuelo_Destino = ?, Vuelo_Hora = ? WHERE IdVuelo = ?");  
        pps.setString(parameterIndex: 1, x: this.JTForigen.getText());  
        pps.setString(parameterIndex: 2, x: this.JTFdestino.getText());  
        pps.setString(parameterIndex: 3, x: this.JTFhora.getText());  
        pps.setInt(parameterIndex: 4, x: Integer.parseInt(s: this.JTFid.getText()));  
        pps.executeUpdate();  
        JOptionPane.showMessageDialog(parentComponent: null, message: "Datos actualizados");  
        this.JBagregar.setVisible(aFlag: true);  
        this.JBmodificar.setVisible(aFlag: true);  
        this.JBeliminar.setVisible(aFlag: true);  
        this.JBregresar.setVisible(aFlag: true);  
        this.JTFid.setEditable(b: true);  
        this.JBactualizar.setVisible(aFlag: false);  
        this.mostrar(tabla: "vuelo", id: "");  
        this.limpiar();  
    } catch (SQLException ex) {  
        Logger.getLogger(JFVuelo.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);  
    }  
}
```

Finalmente se programó el botón de borrar.

Primero se obtiene la fila seleccionada en el componente "visor" utilizando el método `getSelectedRow()`.

Se muestra un cuadro de diálogo de confirmación utilizando `JOptionPane.showConfirmDialog()`, preguntando al usuario si desea eliminar el registro con el ID obtenido de la fila seleccionada.

Si el usuario elige "Sí" en el cuadro de diálogo de confirmación (`op == JOptionPane.YES_OPTION`), se ejecuta el código dentro del bloque "if".

Se verifica si se ha seleccionado una fila válida (`fila >= 0`). Dentro del bloque "if" y en caso de haber seleccionado una fila válida, se crea un objeto `PreparedStatement` para preparar una consulta de eliminación en la tabla "vuelo" utilizando el valor del ID obtenido de la fila seleccionada.

Se establece el valor del parámetro de la consulta utilizando el método `setInt()`.

Se ejecuta la consulta de eliminación utilizando el método `executeUpdate()`.

Y Se muestra un mensaje emergente indicando que el registro se ha eliminado correctamente.

Se llama al método "`mostrar()`" para actualizar la visualización de la tabla "vuelo" en la interfaz de usuario.

Código:

```
private void JeliminarActionPerformed(java.awt.event.ActionEvent evt) {  
    int fila = visor.getSelectedRow();  
    int op = JOptionPane.showConfirmDialog(parentComponent, "Desea eliminar el registro ID: " + visor.getValueAt(row:fila, column:0).toString()  
    + "?", title: "Eliminar", optionType: JOptionPane.YES_NO_OPTION);  
    if (op == JOptionPane.YES_OPTION) {  
        if (fila >= 0) {  
            try {  
                PreparedStatement pps = conexion.prepareStatement(sql: "DELETE FROM vuelo WHERE idVuelo = ?");  
                pps.setInt(parameterIndex: 1, x: Integer.parseInt(s: visor.getValueAt(row:fila, column:0).toString()));  
                pps.executeUpdate();  
                JOptionPane.showMessageDialog(parentComponent, null, message: "Registro eliminado correctamente");  
                mostrar(tabla: "vuelo", sql: "");  
            } catch (SQLException ex) {  
                Logger.getLogger(name: JFVuelo.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);  
                JOptionPane.showMessageDialog(parentComponent, null, message: "Ocurrió un error al intentar eliminarlo");  
            }  
        } else {  
            JOptionPane.showMessageDialog(parentComponent, null, message: "Busque la fila a modificar y haga un click en ella para usar esta funcion");  
        }  
    }  
}
```

3.4.Resultados del CRUD

The image shows two side-by-side screenshots of a web application interface for managing flights. Both windows are titled 'Vuelo'.

Left Screenshot: The form is empty. It has input fields for 'ID (5 dígitos):', 'Origen:', 'Destino:', and 'Hora de Vuelo:'. The 'Hora de Vuelo' field has a format hint 'Formato: HH:MM (23:59)'. Below the form are four buttons: 'Agregar', 'Modificar', 'Eliminar', and 'Regresar'.

Right Screenshot: The form is filled with data: 'ID (5 dígitos):' is '12345', 'Origen:' is 'Ecuador', 'Destino:' is 'Rusia', and 'Hora de Vuelo:' is '12:00'. The 'Actualizar' button is visible. Below the form is a table titled 'Tabla de Vuelos'.

ID	Origen	Destino	Hora de vuelo
12345	Ecuador	Rusia	12:00
47856	Mexico	Canada	17:35
58369	Africa	China	18:40
87524	Hawaii	Filipinas	14:48
97856	Estados Unidos	Inglaterra	7:30

Below the table is a search field labeled 'Buscar por ID de vuelo:'.

4. AFINACIÓN DEL PROGRAMA

4.1. Afinación De La Base De Datos

Se crearon nuevas tablas en la base de datos para almacenar los siguientes datos: Avión (Modelo del avión), boletos y recorrido del vuelo. Luego se realizó su respectiva relación con otras tablas. Se usaron los siguientes comandos.

Comando de avión

```
CREATE TABLE IF NOT EXISTS Avion
(
    Avionid int not null auto_increment,
    Compania varchar (50) not null,

    PRIMARY KEY (Avionid)
)ENGINE = InnoDB;
```

Comando de creación de la tabla Boleto


```

CREATE TABLE IF NOT EXISTS Boleto
(
    BoletoId int not null auto_increment,
    ClienteId int not null,
        CONSTRAINT ClienteId FOREIGN KEY (ClienteId)
REFERENCES Cliente (ClienteId),
    VueloIdB int not null,
        CONSTRAINT VueloIdB FOREIGN KEY (VueloIdB)
REFERENCES Vuelo (VueloId),
    AsientoIdB int not null,
        CONSTRAINT AsientoIdB FOREIGN KEY (AsientoIdB)
REFERENCES Asiento (AsientoId),

    PRIMARY KEY (BoletoId)
)ENGINE = InnoDB;

```

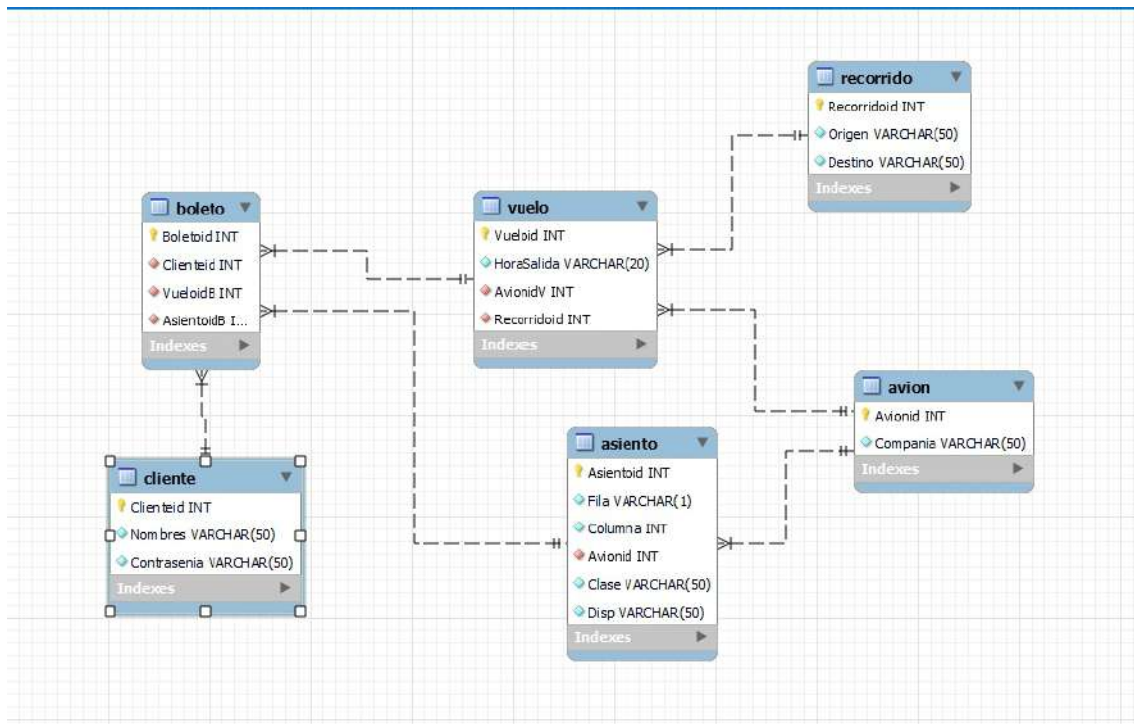
Comando para la creación de la tabla Recorrido

```

CREATE TABLE IF NOT EXISTS Recorrido
(
    RecorridoId int not null auto_increment,
    Origen varchar (50) not null,
    Destino varchar (50) not null,
    PRIMARY KEY (RecorridoId)
)ENGINE = InnoDB;

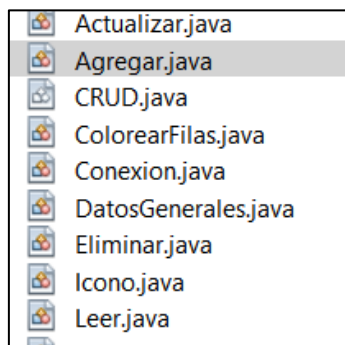
```

Nuevo Diagrama de la Base de datos



4.2.Creación De Nuevas Clases Para El Crud

Antes todo el código relacionado con el CRUD se encontraba en las interfaces del programa. Para no caer en la repetición de código. Se crearon nuevas clases para almacenar el CRUD.



4.3.Mejora Visual De Las Interfaces

Se realizaron mejoras visuales a todas las interfaces del cliente y el administrador.

Interfaz Login para Administración



The screenshot shows a login interface for 'AIRSCAPE Administracion'. It features a central white box with a teal border containing two input fields: 'Usuario' (with a placeholder 'Usuario aquí') and 'Contraseña' (with a placeholder 'XXXXXXXX'). Below these fields are three buttons: 'ACEPTAR', 'SALIR', and 'INGRESAR COMO CLIENTE'. The background is a light blue gradient.

Cliente



The screenshot shows a web application window titled 'Cliente'. The main content area has a teal header with the title 'CLIENTE'. Below the header are three input fields labeled 'NOMBRE:', 'CEDULA:', and 'CONTRASEÑA:'. Underneath these fields is a row of five buttons: 'AGREGAR', 'MODIFICAR', 'ELIMINAR', 'ACTUALIZAR', and 'REGRESAR'. Below the buttons is a table titled 'TABLA DE CLIENTES' with four columns labeled 'Title 1', 'Title 2', 'Title 3', and 'Title 4'. The table body is currently empty. At the bottom of the interface is a search bar labeled 'BUSCAR POR CEDULA:' with an input field.

Asiento

Asiento

AVION

COMPañIA:

ID AVION:

AGREGARMODIFICARELIMINARACTUALIZAR

TABLA DE AVIONES

Title 1Title 2Title 3Title 4

ASIENTOS

ID AVION:

ASIGNAR AUTOMATICAMEN...

TABLA DE ASIENTOS

Title 1Title 2Title 3Title 4

REGRESAR

Interfaz Asiento-avión.

Asiento

AVION

COMPañIA:

ID AVION:

AGREGARMODIFICARELIMINARACTUALIZAR

TABLA DE AVIONES

Title 1Title 2Title 3Title 4

ASIENTOS

ID AVION:

ASIGNAR AUTOMATICAMEN...

TABLA DE ASIENTOS

Title 1Title 2Title 3Title 4

REGRESAR

Interfaz Vuelo-Boleto

4.4.Preparación De La Interfaz Para La Venta De Boletos

Se empezó por la creación del J Frame de interfaz del cliente

Se importaron las clases del CRUD para hacer uso sus respectivos métodos

```
package VistaCliente;

import Negocio.Actualizar;
import Negocio.Agregar;
import Negocio.ColorearFilas;
import Negocio.DatosGenerales;
import Negocio.Icono;
import Negocio.Leer;
import VistaAdministracion.JFLoginAdministracion;
import javax.swing.JOptionPane;
```

Diseño de las interfaces para la compra del boleto.

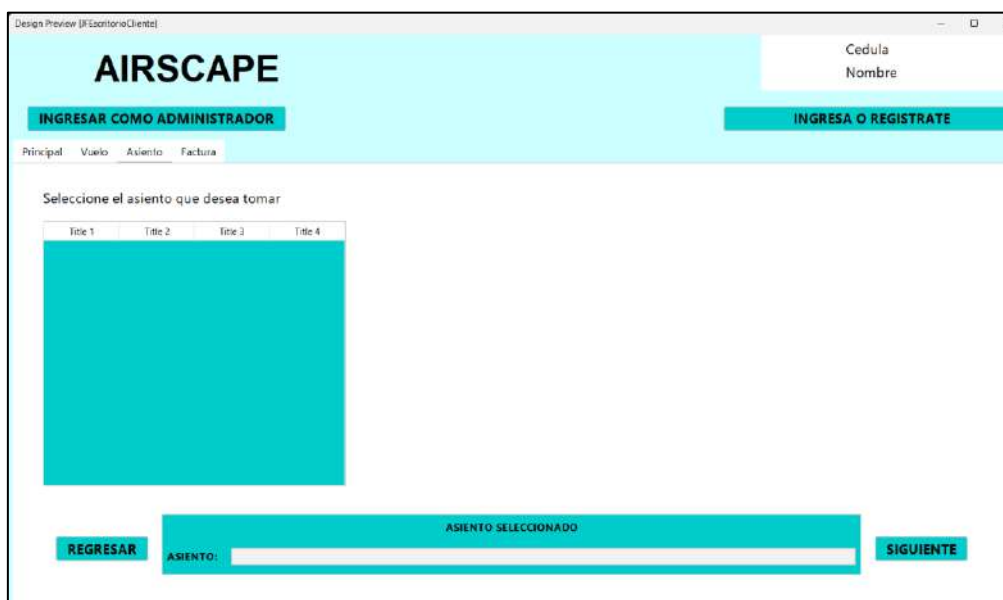
Pestaña principal



Pestaña para la selección del horario del vuelo



Pestaña para la selección del Asiento



Pestaña Para el muestreo de la factura.



Programación del Botón de Compra del menú principal.

```
private void jBAcomprarActionPerformed(java.awt.event.ActionEvent evt) {
    this.aux = jTrecorridos.getSelectedRow();

    if(this.aux >= 0){
        this.origen = jTrecorridos.getValueAt(row: this.aux, column:1).toString();
        this.destino = jTrecorridos.getValueAt(row: this.aux, column:2).toString();
        this.aux = Integer.parseInt(jTrecorridos.getValueAt(row: aux, column:0).toString());
        this.tabla.diseñarTabla(nombreColumnas: "Num. Vuelo, Hora de salida, Numero de avion");
        this.tabla.mostrarTabla(tablaModificar: jTvuelos, nombreTabla: "Vuelo", idOpcional: String.valueOf(this.aux),
            columnasMostrarOpcionales: "Vueloid, HoraSalida, AvionidV", key: "Recorridoid");
        this.jTabbedPane.setSelectedIndex(index: 1);
    } else {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Seleccione en la tabla el recorrido");
    }
    //jTabbedPane.setEnabledAt(0, false);
    //jTabbedPane.setEnabledAt(1, true);
}
```

Cuando se hace clic en un botón, se captura la fila seleccionada de una tabla, se obtienen ciertos valores de las columnas, se configura y muestra otra tabla en función de estos valores, y se cambia a una pestaña diferente(vuelo) en una interfaz con pestañas.

Programación del botón para la selección del vuelo

```
private void jBsiguientevActionPerformed(java.awt.event.ActionEvent evt) {
    this.aux = jTvuelos.getSelectedRow();
    if(this.aux >= 0){
        this.horasalida = jTvuelos.getValueAt(row: aux, column:1).toString();
        this.Vueloid = jTvuelos.getValueAt(row: aux, column:0).toString();
        this.aux = Integer.parseInt(jTvuelos.getValueAt(row: aux, column:2).toString());
        this.Avionid = String.valueOf(this.aux);
        this.tabla.diseñarTabla(nombreColumnas: "Id Asiento, Fila, Columna, Clase, Disponibilidad");
        this.tabla.mostrarTabla(tablaModificar: jTasientos, nombreTabla: "Asiento", idOpcional: String.valueOf(this.aux),
            columnasMostrarOpcionales: "Asientoid, Fila, Columna, Clase, Disp", key: "Avionid");
        this.jTasientos.setDefaultRenderer(columnClass: jTasientos.getColumnClass(column:0), new ColorearFilas());
        this.jTabbedPane.setSelectedIndex(index: 2);
    } else {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Seleccione en la tabla el recorrido");
    }
    // TODO add your handling code here:
}
```

Cuando se hace clic en un botón, se captura la fila seleccionada de una tabla, se obtienen ciertos valores de las columnas, se configura y muestra otra tabla en función de estos valores, y cambia a una pestaña diferente (Asiento).

Programación del botón para la selección del asiento

```
private void jBsiguienteActionPerformed(java.awt.event.ActionEvent evt) {
    this.aux = jTasientos.getSelectedRow();
    if(this.aux >= 0){
        if(this.jLcedula.getText().equals("")){
            JOptionPane.showMessageDialog(parentComponent, null, message: "Cuenta no ingresada, ingrese como cliente o registrese");
            this.jBlogin.doClick();
        } else {
            this.AsientoId = jTasientos.getValueAt(row: aux, column: 0).toString();
            String comprobador = this.tabla.leerDatosAlguienExacto(nombreTabla: "Asiento", idObligatorio: this.AsientoId,
                columnasAMostrarOpcional: "Disp", key: "AsientoId")[0];
            if(comprobador.equals("")){
                JOptionPane.showMessageDialog(parentComponent, null,
                    message: "El asiento que ha sido elegido no se encuentra disponible, por favor, elija uno en color verde (Disponible)");
            } else {
                this.aux = JOptionPane.showConfirmDialog(parentComponent, null, "Esta seguro de adquirir el boleto con las siguientes caracteris
                    + this.origen + " ----> " + this.destino
                    + "\n--Hora de salida: " + this.horaSalida
                    + "\n--Numero de avion: " + this.Avionid
                    + "\n--Ubicacion del asiento: " + this.asiento
                    + "\n--Clase: " + this.claseAsiento
                    + "\nAl confirmar usted acepta nuestros Terminos y Condiciones, ademas se cargara valores a pagar a su cuenta",
                    title: "Confirmacion", optionType: JOptionPane.YES_NO_OPTION);
                if(this.aux == JOptionPane.YES_OPTION){
                    this.imprimirBoleto();
                    this.agregar.agregarDatos(nombreTabla: "Boleto", primaryKey: "BoletoId", idDelDestinoIngresar: "",
                        "default," + this.jLcedula.getText() + "," + this.VueloId + "," + this.AsientoId);
                    this.actualizar.actualizarDatos(nombreTabla: "Asiento", nombreFiltroDelDestinoIngresar: "Disp", datosActualizados: "NO", primaryKey: "AsientoId",
                        this.jTabbedPane1.setSelectedIndex(index: 3);
                }
            }
        }
    }
}
```

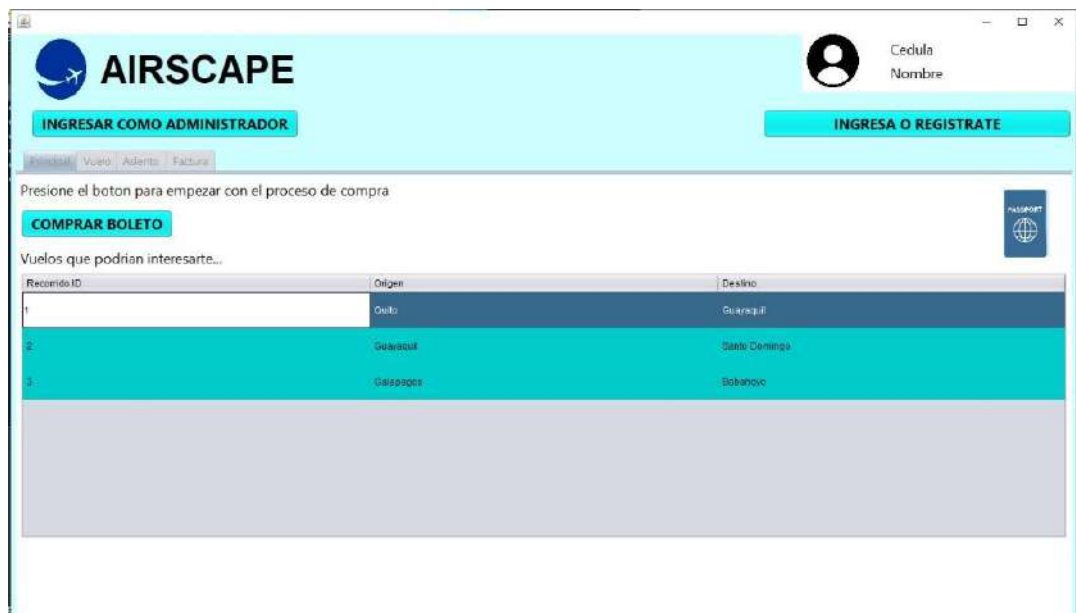
Cuando se hace clic en un botón, se captura la fila seleccionada en una tabla de asientos, se realizan comprobaciones relacionadas con la disponibilidad y la información del usuario, se muestra una confirmación de compra y, en caso afirmativo, se realizan acciones como imprimir el boleto y actualizar los datos en las tablas.

Método para la impresión del boleto.

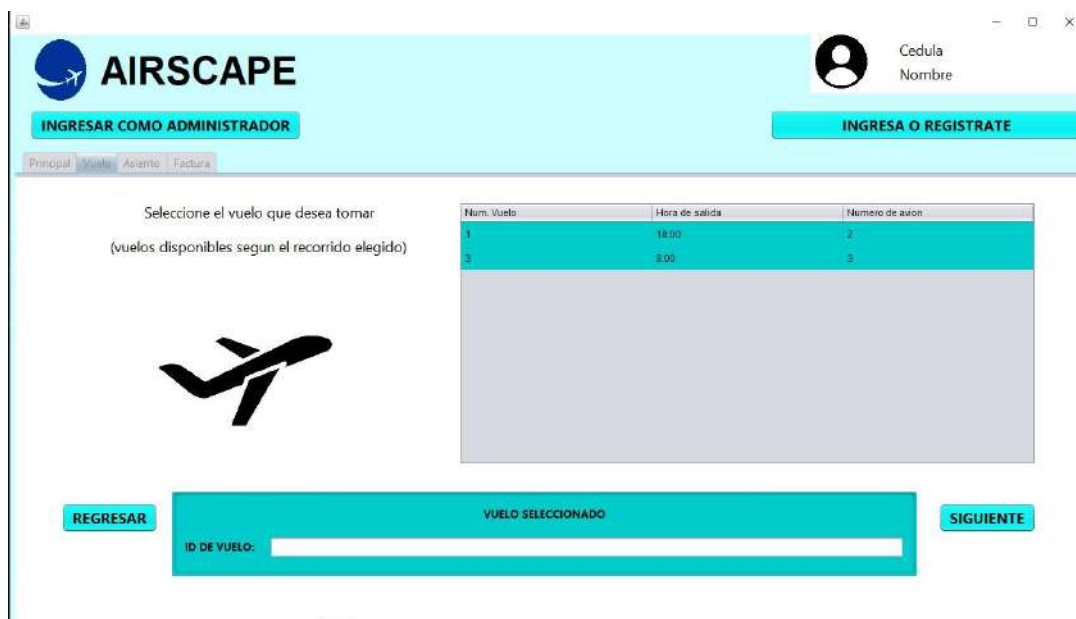
```
private void imprimirBoleto(){
    String salida = "\tInformacion Boleto";
    salida += "\nCedula del usuario: " + this.jLcedula.getText();
    salida += "\nNombre del usuario: " + this.jLnombre.getText();
    salida += "\n\n ■ Datos del viaje desde " + this.origen + " hasta " + this.destino;
    salida += "\nHora de salida: " + this.horaSalida;
    salida += "\nNumero de avion: " + this.Avionid;
    this.companiaA = this.tabla.leerDatosAlguienExacto(nombreTabla: "Avion", idObligatorio: this.Avionid,
        columnasAMostrarOpcional: "Compania", key: "Avionid")[0];
    salida += "\nCompañia del avion: " + this.companiaA;
    salida += "\n\n ■ Datos del asiento: ";
    salida += "\nUbicacion del asiento: " + this.asiento;
    salida += "\nClase: " + this.claseAsiento;
    salida += "\n\nTotal a pagar por boleto: USD $$$" + (this.claseAsiento.equals(anObject: "VIP")?"540":
        (this.claseAsiento.equals(anObject: "Turista")?"350":"235"));
    this.jTafactura.setText(text: salida);
}
```

RESULTADOS

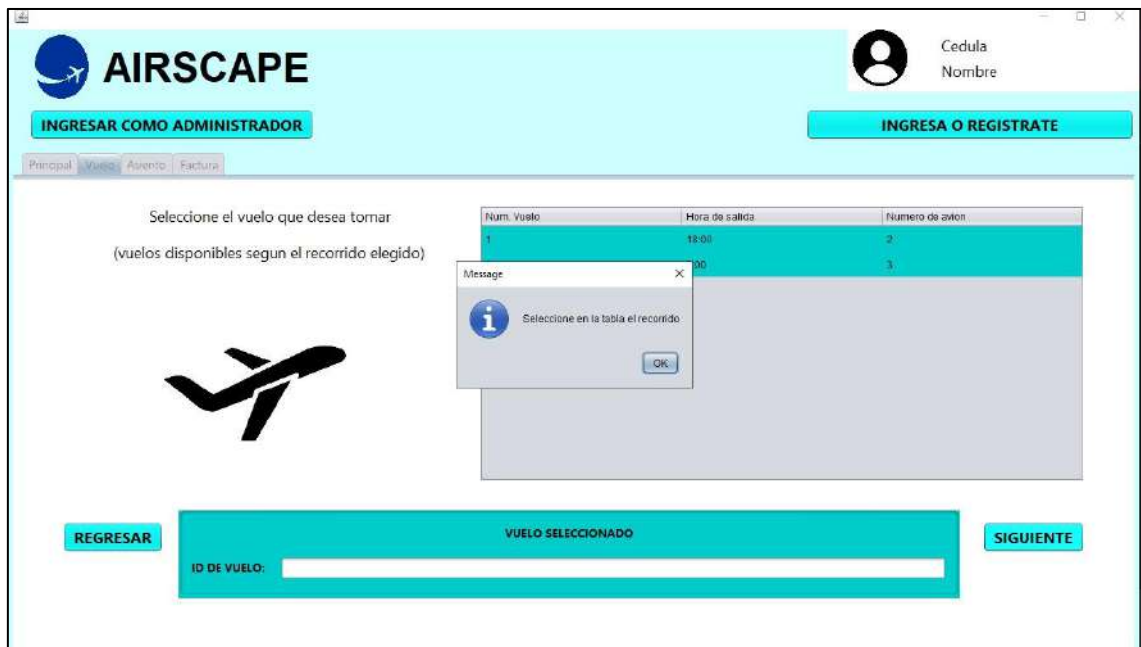
Inicio con la muestra el menú para el Cliente con los vuelos disponibles.



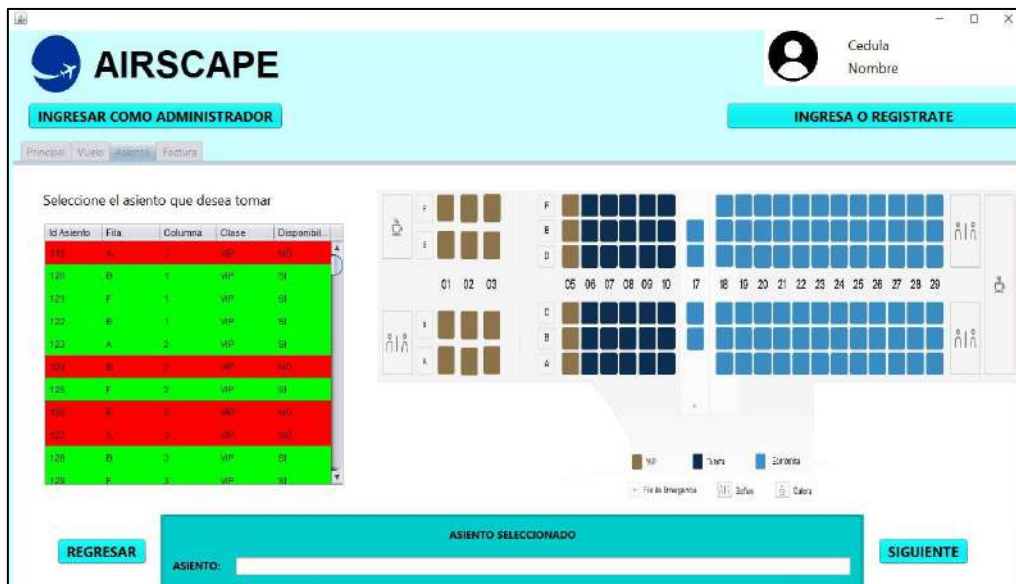
Una vez seleccionado el vuelo se muestra los horarios disponibles



Si no se selecciona una fila aparecerá el siguiente mensaje



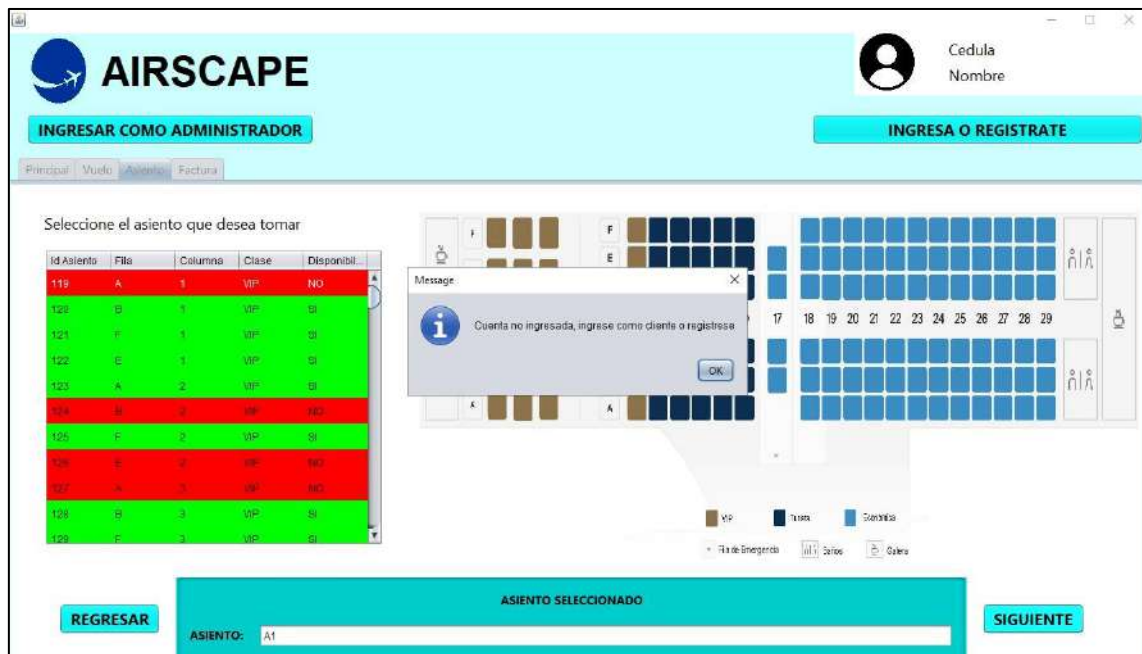
Una vez seleccionado el horario se mostrarán los asientos disponibles (de color verde) y no disponibles (rojo).



Se puede seleccionar por medio de la tabla o escribiendo manualmente la ubicación del asiento.



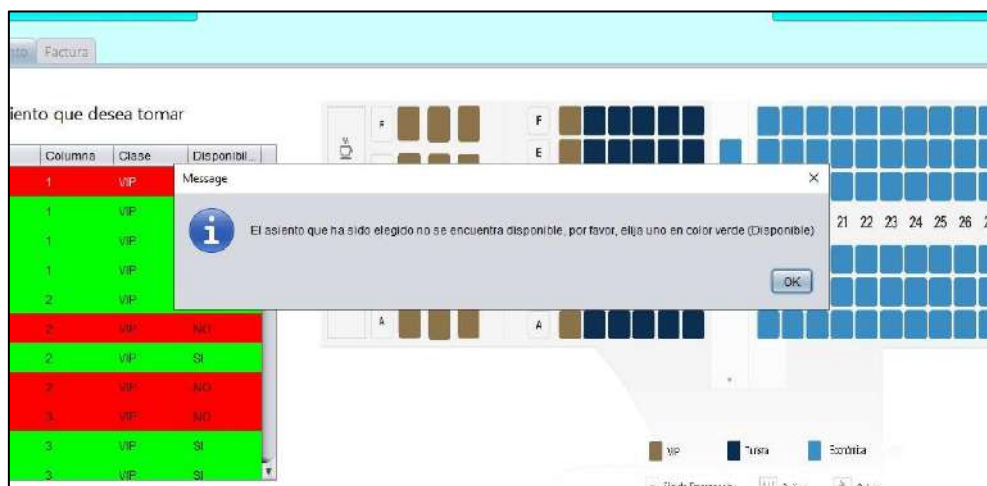
Si no se ingresa una cuenta mostrara el siguiente mensaje hasta que el usuario ingrese.



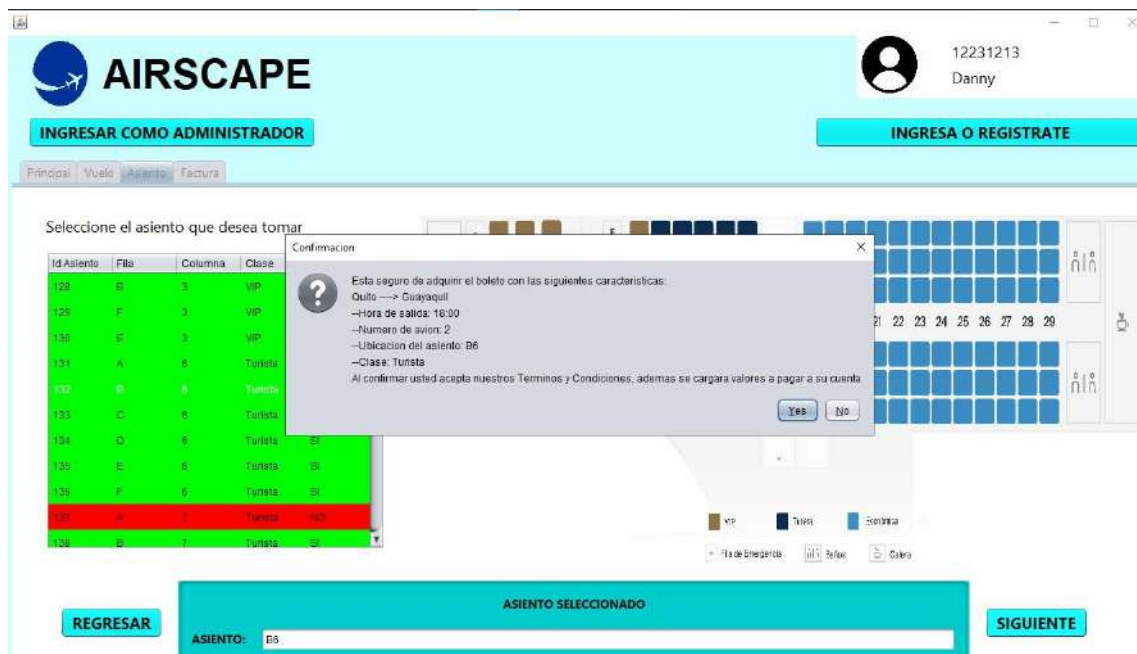
Una vez ingresado se mostrará en la parte de arriba.



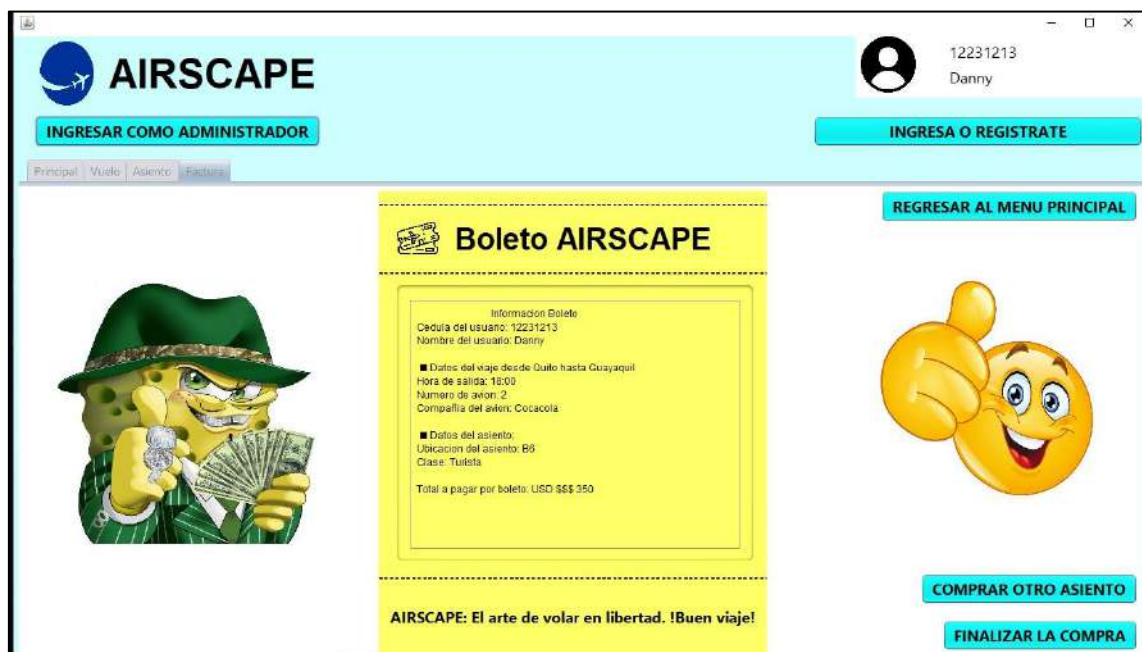
Si no escogemos un asiento libre, nos mostrara el siguiente mensaje de advertencia.



Una vez seleccionado el asiento deseado mostrara un texto de confirmación para comprar el boleto.



Una vez confirmada la compra mostrara en la información del boleto digital.



Finalizada la compra mostrara la opción de regresar al menú (inicio), comprar otro asiento (asiento) o finalizar la compra (salir del programa).

CONCLUSIONES Y RECOMENDACIONES:

Crear un programa en Java para la venta de boletos de avión es un proyecto interesante y desafiante. Por esto mismo recomendamos lo siguiente:

- Diseña la arquitectura de la aplicación para que pueda manejar un aumento en el número de usuarios y transacciones.
- Divide tu programa en módulos y componentes más pequeños y manejables. Esto facilitará el desarrollo, la depuración y el mantenimiento.

- Implementa validaciones de entrada para garantizar que los datos ingresados por los usuarios sean correctos y coherentes.
- Mantente actualizado con las últimas tecnologías y mejores prácticas en desarrollo de software para asegurarte de que tu programa sea eficiente y esté a la vanguardia.

BIBLIOGRAFIA:

- Ramos Cemarza, V. (2022, Agosto 1). *Descarga e Instalación de MySQL Server/ Curso de Base de Datos MySQL Server* [mp4]. Victor Ramos.
<https://youtu.be/hc9c7dwFDoI>
- Ramos Cemarza, V. (2022, Agosto 2) *Crear Schema o Database en MySQL / Curso de Base de Datos MySQL Server* [mp4]. Victor Ramos.
<https://youtu.be/S85fXk7WbyQ>
- Sin Rueda Tecnológica. (2021, Julio 30). *¿Cómo conectar Java (Apache Netbeans) con MYSQL?* [mp4].
https://www.youtube.com/watch?v=ESY_pEvT9TY
- Franklin García. (2020, Julio 27). *Crear TABLAS en MYSQL* [mp4].
<https://www.youtube.com/watch?v=qBI9VIk8IVE&t=506s>
- AVR. (2021, Septiembre 12). *Modificar y relacionar tablas en MySQL* [mp4].
<https://www.youtube.com/watch?v=qBI9VIk8IVE&t=506s>