

# Manual de Usuario

**Sistema de reservación de agencia de viajes.**

**Versión 1.1**

# Table of Contents

I.	Introducción .....	2
II.	Implementación del sistema. ....	2
III.	Ingresando al sistema.....	3
IV.	ANEXOS. ....	47

# Introducción

---



*El propósito de este manual es facilitar al usuario la operación de las diferentes pantallas de captura y consulta de la información que se administra en el sistema de reservación de viaje*

## Implementación del sistema.

---

### **a) Requerimientos de hardware.**

- Computadora personal
  - Memoria RAM de 4GB.
  - Disco duro de 220 GB (mínimamente).
  - Procesador a Intel de 4th generación hacia delante

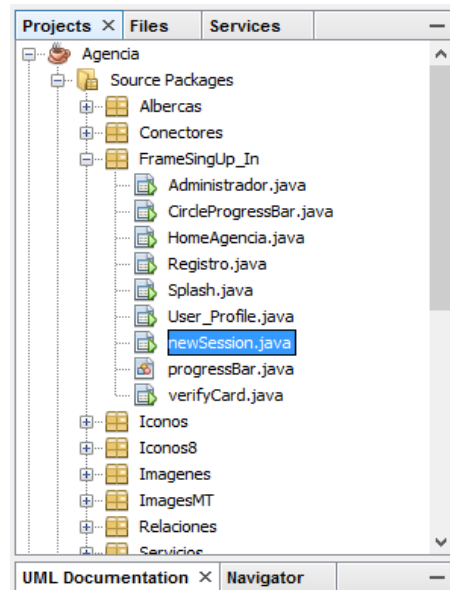
### **b) Requerimientos de Software.**

- Sistema Operativo Windows 8,8.1,10 a x32 o x64 bits.
- Java versión 1.8.0.121 mínimamente.
- Permiso de acceso como administrador.

**Note:** Los requerimientos de hardware y software son los mínimos. En versiones adelantadas el sistema se comportaría de manera similar.

## Ingresando al sistema.

En la pantalla principal del entorno de desarrollo de Netbeas, hay que hacer doble clic en la clase de newSession para ejecutar el programa principal



Inmediatamente después, el sistema solicita correo y contraseña, datos que previamente han sido registrados.



Figura 2.1 Inicio de sesión

Una vez que el usuario dé en el botón de Iniciar sesión, éste ejecutará el siguiente código.


### Botón **Iniciar sesión** (código)

```
private void JBTN_iniciaSesionActionPerformed(java.awt.event.ActionEvent
evt) {
    Cliente newSession = new Cliente();

    //Verificar campos llenos
    if (JTXT_email.getText().trim().equals("") &&
    JTXT_password.getText().trim().equals(""))
    {
        JOptionPane.showMessageDialog(null, "Campos incompletos");
        limpiar();
    } else {
        //Establecer conexión
        connectionMySQL conectarSesion = new connectionMySQL();
        Connection succesfulConnection;
        succesfulConnection= conectarSesion.getConnection();

        //Obtener datos ingresados
        Controlador_ClienteEmail cn = new Controlador_ClienteEmail();
        newSession.setEmail(JTXT_email.getText());
        newSession.setPassword(JTXT_password.getText());

        //Realizar consulta del cliente para acceder cuenta
        cn.accederCuenta(newSession, succesfulConnection);
        reserva.VentanaReserva goReserva;
        goReserva = new reserva.VentanaReserva();
        this.setVisible(false);
        goReserva.setVisible(true);
        //Limpiar los campos
        limpiar();
    }
}
```



Dentro de éste evento del botón antes mencionado en la clase **newSession**, en el círculo **A** se muestra la instancia del objeto **Cliente**<sup>1</sup>, el cual se utilizará para albergar cada uno de los campos que le corresponden para que el usuario pueda iniciar sesión. En donde se verifica que ninguno de los campos de texto de la interfaz de la venta esté vacíos; En caso de que estén vacíos se le manda un mensaje al usuario con la leyenda: “Campos incompleto”, y se limpian las cajas correspondientes para ingresar los datos con el método **limpiar**.

```
public void limpiar() {
    JTXT_email.setText("");
    JTXT_password.setText("");
}
```

Método limpiar de la clase newSession

Posteriormente se lleva a cabo la conexión con la base de datos (como se muestra en el círculo B) utilizando el método **getConnection** de la clase **connectionMySQL**.

<sup>1</sup> La clase **Cliente** (su código) se describirá al final de este manual dentro de los anexos.

### Clase connectionMYSQL y método getConnection (Código).

```
String Protocol = "jdbc:mysql://localhost:3306/";
String DataBase = "Agencia";
String Password = "root1";
String User = "root";

public Connection getConnection()
{
    Connection conexion = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conexion = DriverManager.getConnection(Protocol+DataBase,User,Password);
    } catch (ClassNotFoundException | SQLException e) {
        System.out.println(e.getMessage());
    }
    return conexion;
}
```

Una vez que se ha establecido la conexión con la base de datos y se ha verificado que los campos no están vacíos del cliente, se realiza la búsqueda de la cuenta (del cliente) que desea ingresar al sistema. Para ello se instancia un objeto de la clase **Controlador\_ClienteEmail**, para a continuación asignar los datos de las cajas de texto a los respectivos atributos del cliente (como se muestra en la circunferencia C).

Finalmente, gracias a la instancia del objeto de Controlador\_ClienteEmail (cn), hacemos uso del método **accederCuenta**.

### Método accederCuenta (código).

```
public void accederCuenta(Cliente user, Connection conexion) {
    String SQL_Login = "SELECT * FROM emailCliente WHERE email = '"
        + user.getEmail() + "'" + "AND CONTRASEÑA = '"
        + user.getPassword() + "'";
    String email = "", psw = "";

    try {
        Statement st = conexion.createStatement();
        ResultSet rs = st.executeQuery(SQL_Login);
        while (rs.next()) {
            email = rs.getString("email");
            psw = rs.getString("contraseña");
            idCliente = rs.getString("idCliente");
        }
        System.out.println(idCliente);
        if ((user.getEmail().equals(email)) &&
            (user.getPassword().equals(psw))) {
            progressCircle();
        } else {
            JOptionPane.showMessageDialog(null, "Compruebe sus da-
tos");
        }

    } catch (SQLException ex) {
        ex.getMessage();
    }
    VentanaReserva a = new VentanaReserva(user);
}
```

El cual realiza una consulta para verificar el correo y contraseña en la base de datos de agencia del cliente, para posteriormente ser ejecutada con el método `executeQuery`. Y mostrar un progress bar de llamando al método `progressCircle`.

```
public void progressCircle()
{
    new Thread(new Runnable() { /*Instancia del hilo con su respectivo
                                método abstracto de la interfaz Runnable*/
        @Override
        public void run() { //Abre método run del hilo
            CircleProgressBar circle = new CircleProgressBar();
            circle.setVisible(true);
            for (int num = 1; num <= 100; num++) { //Inicio de for
                try {
                    circle.getCirclePB().UpdateProgress(num);
                    circle.getCirclePB().repaint();
                    Thread.sleep(35);
                } catch (InterruptedException ex) {
                    ex.printStackTrace();
                }
            } //Cierra for
            circle.setVisible(false);
        } //Cierra método run del hilo
    }).start(); //Inicia el hilo
}
```

Finalmente, hacemos visible la ventana de Home y “escondemos” la ventana del inicio de sesión, para posteriormente limpiar los campos.

**Nota:** La descripción completa de cada código se encuentra en sus clases respectivas.

En dado caso de que el usuario sea el administrador, entonces éste iniciará sesión de manera “normal”, a excepción de que sus tarjetas de acceso serán las siguientes. User: admin, Password: admin, como se muestra en la siguiente figura.



## 2.2 Inicio de sesión de administrador

Una vez que el usuario/administrador ha ingresado sus datos, al momento de presionar el botón iniciar sesión, se ejecutará el siguiente código.

```
if (newSession.getEmail().equals("admin") &&
    newSession.getPassword().equals("admin"))
{
    Administrador admin = new Administrador();
    setVisible(false);
    admin.setVisible(true)
}
```

Este segmento de código nos dirigirá hacia la ventana del administrador en la que harás las modificaciones pertinentes. Dicha ventana se muestra a continuación.



The screenshot shows the 'Administrador' window with the following elements highlighted by numbered circles:

- 1:** 'Base de datos:' dropdown menu showing 'information\_s...'.
- 2:** 'Consultar' button.
- 3:** A large empty rectangular area, likely for displaying table data.
- 4:** 'Eliminar' button.
- 5:** '+ Nuevo usuario' button.
- 6:** 'Refrescar' button.
- 7:** 'Insertar' button.
- 8:** 'Cerrar sesión' button in the top right corner.

The form includes fields for 'Nombre:', 'Apellido paterno:', 'Apellido materno:', 'Correo:', 'Contraseña:', 'Tarjeta:', 'CVV:', 'Tipo' (with a 'Crédito' dropdown), and 'Expiración:' (with a 'junio' dropdown and a year selector set to '2017').

### 2.2.1 Ventana del administrador

Dentro de esta interfaz, el administrador tendrá la facilidad de escoger las distintas bases de datos con las cuales esté trabajando el sistema. En este caso, simplemente se trabajará con la base de datos Agencia.

Una vez que hemos especificado la base de datos, en el apartado del Combo box, frente la etiqueta Relaciones, se mostrarán (y podrán desplegar) las distintas tablas existentes en la base. Desde ahí se pondrá realizar operaciones de consulta, así como de eliminación.

idCliente	nombre	apPaterno	apMaterno
1	JESUS KAIMORTS	DÍAZ	MEDINA
2	ANGELICA ELVIRA	MEDINA	FRAGOSO
3	NELLY ARLET	BAUTISTA	HERNÁNDEZ
4	ALEJANDRO	VAZQUEZ	SANCHEZ
5	ROBERTO	TECLA	PARRA
6	NATALY KIMBERLY	DÍAZ	CÁRDENAS
7	KEITH ESTEFAN	DÍAZ	MEDINA
8	OLIVER FERNANDO	DÍAZ	MEDINA

### 2.2.2 Consulta de tablas (Tabla Cliente)

Una vez que se ha hecho clic en el botón de consultar, dentro de la codificación se ejecutará lo siguiente.

```
private void BTN_ConsultarActionPerformed(java.awt.event.ActionEvent evt)
{
    consultar(CB_Relacion.getSelectedItem().toString().trim());
}
```

El cuál, dentro de la clase Administrador (que es donde todo ocurre), habrá un código llamado consultar, que implementa el siguiente código.

```

public void consultar(String tabla) {
    try {
        String columnas, SQL_QUERY;
        connectionMYSQL cn = new connectionMYSQL();
        Connection conexion = cn.getConnection();
        Statement st = conexion.createStatement();
        ResultSet rs = null;
        ResultSetMetaData rsmd = null;
        columnas = "DESC " + tabla;
        SQL_QUERY = "SELECT * FROM " + tabla;
        rs = st.executeQuery(SQL_QUERY);
        rsmd = rs.getMetaData();
        /*Obtener encabezado de la consulta*/
        int col = rsmd.getColumnCount();
        DefaultTableModel modelo = new DefaultTableModel();

        for (int i = 1; i <= col; i++) {
            modelo.addColumn(rsmd.getColumnLabel(i));
        } //Ciera encabezado de consulta

        while (rs.next()) {
            String filas[] = new String[col];
            for (int j = 0; j < col; j++) {
                filas[j] = rs.getString(j + 1);
            }
            modelo.addRow(filas);
        }
        TABLE_DATA.setModel(modelo);
        rs.close();
        conexion.close();
    } catch (SQLException e) {
        e.getMessage();
    }
}

```

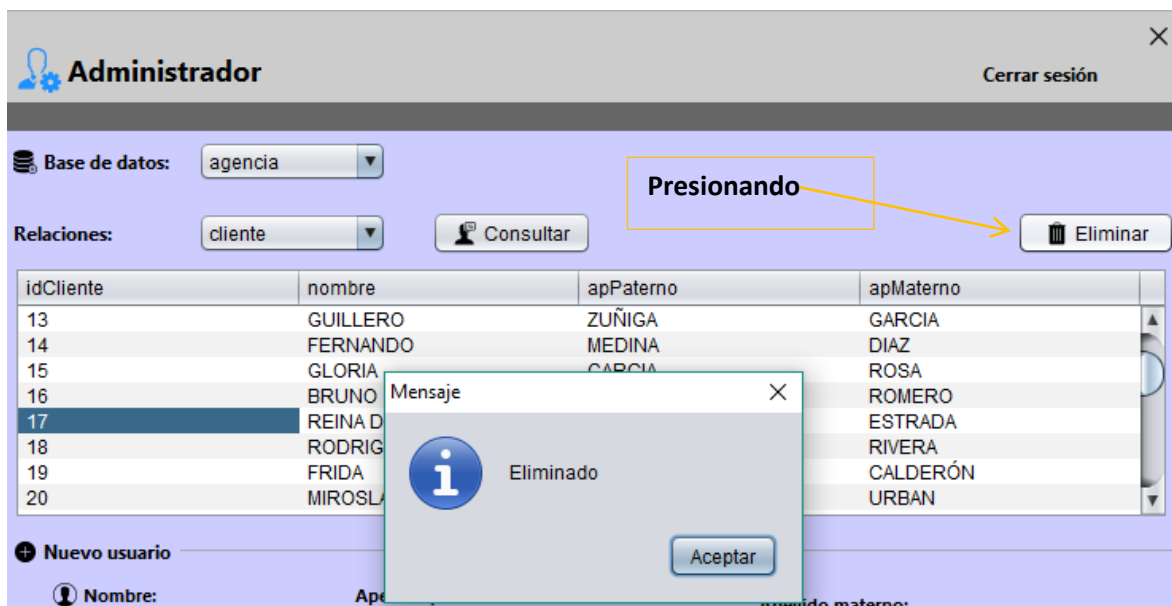
Lo que anteriormente se realiza es una consulta de todos los registros de la tabla según seleccionada a parte del combo box. Una vez que se ha hecho la consulta, establecemos un modelo para especificar el tamaño y descripción de nuestra tabla mostrada en la interfaz del administrador, la cual se adaptará al modelo (cardinalidad y grado) de cada relación. Ésta será dinámica y amigable con el administrador.

Cabe destacar que para obtener las tablas y las bases de datos se usaron dos métodos más, los cuales son cargarBD y cargarTablas<sup>2</sup>, además de que el administrador puede editar al momento los datos algún cliente de manera física en la interfaz, pero no de manera interna en la base de datos.

Si en algún momento el administrador tiene la necesidad de eliminar algún registro, esté lo hará a partir de la siguiente manera (sólo puede eliminar por el momento registros correspondientes a los clientes, así como los que estén referenciados a él: correo electrónico y tarjeta).

---

<sup>2</sup> En los anexos se muestra la codificación de ambos métodos.



### 2.2.3 Eliminación de un cliente

Asimismo, la interfaz del administrador proporciona la opción de agregar un nuevo cliente. En la parte inferior aparece un formulario generalizado para el proceso de registro. Éste se adaptó con el fin de agilizar las inserciones.

**+ Nuevo usuario**

Nombre: \_\_\_\_\_ Apellido paterno: \_\_\_\_\_ Apellido materno: \_\_\_\_\_  
 Correo: \_\_\_\_\_ Contraseña: \_\_\_\_\_ Tarjeta: \_\_\_\_\_ CVV: \_\_\_\_\_  
 Tipo: Crédito Expiración: junio 2017  
 Refrescar  
 Insertar Actualizar

#### 2.2.4 Formulario de registro de nuevo usuario

Una vez que el administrador tenga que registrar a algún cliente directamente, omitiendo el proceso de registro que se realizará y detallará más adelante, el botón insertar generará el evento para insertar en la base de datos.

**Administrador** Cerrar sesión

Base de datos: agencia

Relaciones: cliente Consultar Eliminar

idCliente	nombre	apPaterno	apMaterno
76	NATALIA	LAFOURCADE	CUEVA
77	CESAR FERNANDO	BARCENAS	HERNANDEZ
78	RICARDO ABDALA	CONTRERAS	FUENTES
79	MITZI BRISEIDAH	BRITO	ROJAS
80	VALERY LIZBETH	LEDÓN	TORRES
81	JADEE HASANI	DIAZ	CARDENAS
82	JUAN CARLOS	MARTINEZ	GONZALEZ
83	JOSE MANUEL	ROSALES	VAZQUEZ

**+ Nuevo usuario**

Nombre: RODOLFO RODRIGO Apellido paterno: mijares Apellido materno: dorantes  
 Correo: o\_ro@gmail.com Contraseña: 5tgb Tarjeta: 5322432323232 CVV: 3142  
 Tipo: Crédito Expiración: diciembre 2020  
 Refrescar  
 Insertar Actualizar

**Presiona insertar**

idCliente	nombre	apPaterno	apMaterno
77	CESAR FERNANDO	BARCENAS	HERNANDEZ
78	RICARDO ABDALA	CONTRERAS	FUENTES
79	MITZI BRISEIDAH	BRITO	ROJAS
80	VALERY LIZBETH	LEDÓN	TORRES
81	JADEE HASANI	DIAZ	CARDENAS
82	JUAN CARLOS	MARTINEZ	GONZALEZ
83	JOSE MANUEL	ROSALES	VAZQUEZ
84	RODOLFO RODRIGO	MIJARES	DORANTES

**+ Nuevo usuario**

Nombre: **RODOLFO RODRIGO**    Apellido paterno: **mijares**    Apellido materno: **dorantes**

Correo: **o\_ro@gmail.com**    Contraseña: **5tgb**    Tarjeta: **5322432323232**    CVV: **3142**

Tipo: **Crédito**    Expiración: **diciembre** **2020**

**Refrescar**

### 2.2.5 Registro de nuevo usuario (Ventana Administrador)

Para poder llevar a cabo este proceso, el botón de insertar ejecuta el siguiente código<sup>3</sup>.

<sup>3</sup> Los métodos resaltados: getNoCard, getFecha, insertar, insertCard, son descritos en el apartado de registro.

```

private void BTN_InsertarActionPerformed(java.awt.event.ActionEvent evt)
{
    //Declaración de objetos a utilizar
    Cliente userA = new Cliente();
    Tarjeta cardA = new Tarjeta();
    Controlador_ClienteEmail verifyUser = new Controlador_ClienteEmail();
    Controlador_Tarjeta verifyCard = new Controlador_Tarjeta();
    connectionMySQL cn = new connectionMySQL();
    Connection conexion = cn.getConnection();

    //Asignación de datos al objeto Cliente.
    userA.setNombre(TXT_Name.getText().toUpperCase().trim());
    userA.setA_Paterno(TXT_apPat.getText().toUpperCase().trim());
    userA.setA_Materno(TXT_apMat.getText().toUpperCase().trim());
    userA.setEmail(TXT_Correo.getText().trim());
    userA.setPassword(TXT_PSS.getText().trim());

    //Asignación de datos al objeto Tarjeta
    cardA.setNoTarjeta(getNoCard().trim());
    cardA.setCVV(Integer.parseInt(TXT_CVV.getText()));
    cardA.setTipo(CB_TipoCard.getSelectedItem().toString().trim());
    cardA.setFecha(getFecha().trim());

    //Verificación de prueba y llamada a metodos insertCard e insertar.
    if (verifyUser.verifyRegister(userA) && verifyCard.verifyCard(cardA))
    {
        System.out.println(userA.getNombre()+"si tiene datos");
        System.out.println(cardA.getNoTarjeta()+"si tiene tarjeta");
        verifyUser.insertar(userA, conexion, "Cliente");
        verifyCard.insertCard(cardA);
    }
}

```

Por último, para limpiar cada una de las casillas del registro, procedemos a presionar el botón refrescar.

The image shows two screenshots of a web form titled "Nuevo usuario". The top screenshot shows the form filled with data: Nombre: RODOLFO RODRIGO, Apellido paterno: mijares, Apellido materno: dorantes, Correo: o\_ro@gmail.com, Contraseña: 5tgb, Tarjeta: 5322432323232, CVV: 3142, Tipo: Crédito, Expiración: diciembre 2020. The bottom screenshot shows the same form with all fields empty, and the "Refrescar" button highlighted.

## 2.2.6 Limpiar formulario

Para realizar esta operación, solamente se ejecuta el siguiente código.

```
private void BTN_ActualizarActionPerformed(java.awt.event.ActionEvent  
evt) {  
    // TODO add your handling code here:  
    limpiar();  
}
```

Que a su vez ejecuta el siguiente código, el cuál “limpia” todo aquello que contengas los TextField.

```
public void limpiar()  
{  
    TXT_Name.setText("");  
    TXT_apPat.setText("");  
    TXT_apMat.setText("");  
    TXT_Correo.setText("");  
    TXT_PSS.setText("");  
    TXT_Tarjeta.setText("");  
    TXT_CVV.setText("");  
}
```

Generalizando, podemos establecer una tabla de la figura 2.2.1 para describir la ventana del administrador.

Figura	Componente	Descripción
1	Combo box (2)	<ul style="list-style-type: none"><li>• Es un componente desplegable que muestra las distintas bases de datos en el computador, así como sus determinadas tablas en ella.</li><li>• La base de datos usada es la de Agencia.</li></ul>
2	Botón de consulta	<ul style="list-style-type: none"><li>• Realiza la búsqueda pertinente con respecto a la tabla seleccionada.</li></ul>
3	Tabla	<ul style="list-style-type: none"><li>• A partir de la búsqueda realizada, se muestran los metadatos pertinentes con respecto a la tabla de la base de datos.</li></ul>
4	Botón de eliminar.	<ul style="list-style-type: none"><li>• Una vez que se ha visualizado la tabla y sus registros pertinentes, se realiza la eliminación del registro seleccionado.</li><li>• Por ahora solamente se pueden eliminar clientes de la base de datos con su respectivo email y tarjeta asignada</li></ul>
5	Formulario (Campos de texto)	<ul style="list-style-type: none"><li>• Solamente está adaptado para la inserción de clientes.</li><li>• Los campos no pueden quedar vacíos.</li><li>• Se verifica cada uno de los datos en la base de datos y se controla el manejo de errores pertinente.</li></ul>



6	Botón de refrescar	<ul style="list-style-type: none"> <li>• Limpia todos los campos de entradas del formulario de registro de usuario.</li> </ul>
7	Botón de inserción	<ul style="list-style-type: none"> <li>• Realiza las consultas necesarias para insertar a un nuevo usuario y se visualiza en la tabla.</li> </ul>
8	Etiqueta de cierre de sesión.	<ul style="list-style-type: none"> <li>• Finaliza la sesión del administrador y muestra la ventana de inicio de sesión de forma normal</li> </ul>

De la misma manera, podemos establecer la siguiente tabla de forma global de la imagen 2.1

Figura	Componente	Descripción.
1	Botón de inicio de sesión	Se ejecuta el código respectivo para ingresar al sistema, una vez que los datos hayan sido validados.
2	Campos de entrada	<p>Se solicitan los datos para iniciar la sesión.</p> <ul style="list-style-type: none"> <li>• El email es único e irrepetible en cada una de las cuentas.</li> <li>• No puede haber dos clientes distintos con un mismo email.</li> <li>• La contraseña tiene está adaptada para “ocultar” lo escrito en la caja de texto.</li> </ul>
3	Etiqueta de fondo.	Se establece un fondo para la ventana, el cual solamente puede ser modificado a partir del código.

En dado caso de que el cliente no tenga una cuenta para ingresar al sistema, este tiene la facilidad de registrarse

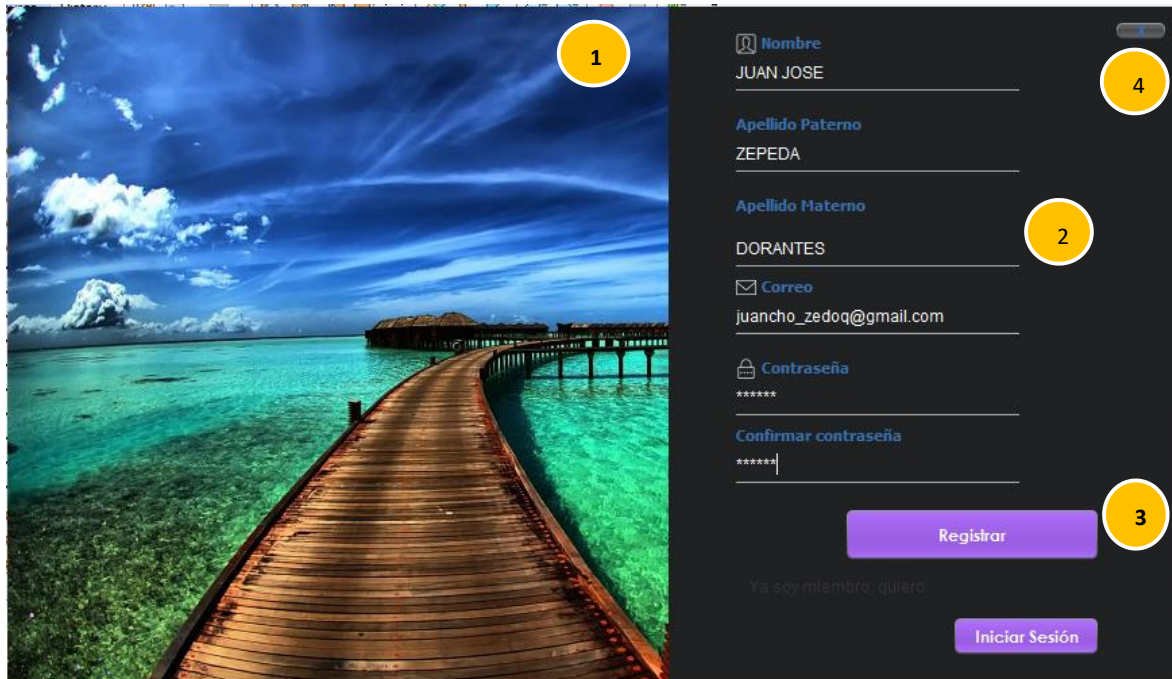


Figura 2.3 Acceso a registro de usuario

El cuál ejecutará el siguiente código al hacer clic en el texto Registrarme

```
private void JLBL_RegisterGoMouseClicked(java.awt.event.MouseEvent evt) {  
    setVisible(false);  
    Registro register = new Registro();  
    register.setVisible(true);  
}
```

Éste simplemente oculta la venta del inicio sesión para dar pauta a que aparezca la ventana de registro.



## 2.4 Ventana de registro

Una vez que se está en la ventana de Registro, se procede a llenar cada uno los campos (siendo todos obligatorios), para que de esta manera una vez que se seleccione el botón Registrar, se ejecute el siguiente código.

```
private void BTN_RegistrarActionPerformed(java.awt.event.ActionEvent evt)  
{  
    desbloquear();  
    Cliente user = new Cliente();  
    Controlador_ClienteEmail registro = new Controlador_ClienteEmail();  
    verifyCard addCard = new verifyCard();  
    /*Inserción de los datos a los atributos del objeto Cliente  
    Se utiliza el método toUpperCase() y trim() para establecer un  
    formato en la base de datos respecto al nombre y para evitar que  
    haya espacios vacíos después del nombre*/  
    user.setNombre(TXTname.getText().toUpperCase().trim());  
    user.setA_Paterno(TXTap.getText().toUpperCase().trim());  
    user.setA_Materno(TXTam.getText().toUpperCase().trim());  
    user.setEmail(TXTemail.getText().trim());  
    user.setPassword(TXTpassword.getText().trim());  
    user.setConfirmPassword(TXT_confirmPassword.getText().trim());  
    .....  
    .....  
}
```

Posteriormente se verifica que cada uno de los campos no estén vacíos con el método, de la clase Controlador\_ClienteEmail, **verifyRegister**: Éste se encarga de validar que ninguno de los datos ingresados esté en blanco.

```

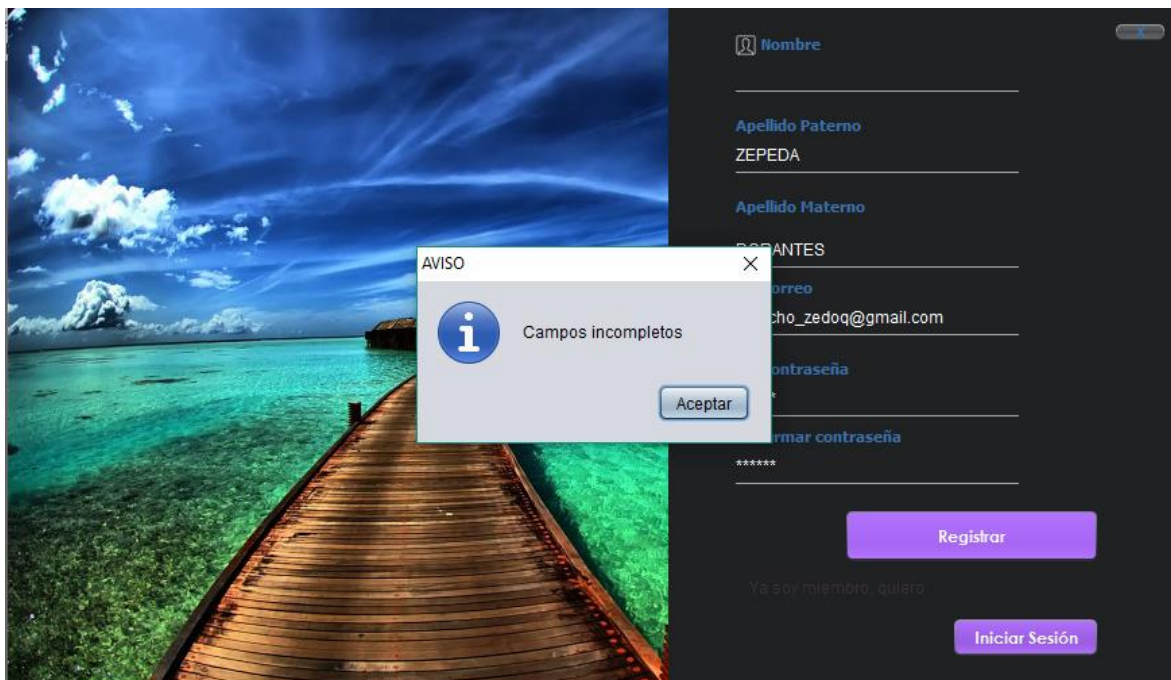
if (registro.verifyRegister(user)) {

    connectionMySQL conexion = new connectionMySQL();
    String nomTable = "Cliente";
    .....

public boolean verifyRegister(Cliente user) {
    boolean continuar;
    if ("".equals(user.getNombre()) ||
        "".equals(user.getA_Paterno()) ||
        "".equals(user.getA_Materno()) || "".equals(user.getEmail()) ||
        "".equals(user.getConfirmPassword()) ||
        "".equals(user.getPassword())) {
        JOptionPane.showMessageDialog(null, "Campos incompletos",
            "AVISO", javax.swing.JOptionPane.INFORMATION_MESSAGE);
        continuar = false;
    } else {
        continuar = true;
    }
    return continuar;
}
}

```

En caso de que los campos, o alguno de ellos, estén vacíos aparecerá un mensaje con la leyenda de “campos incompletos”. El cual desaparecerá una vez que se dé clic en aceptar.

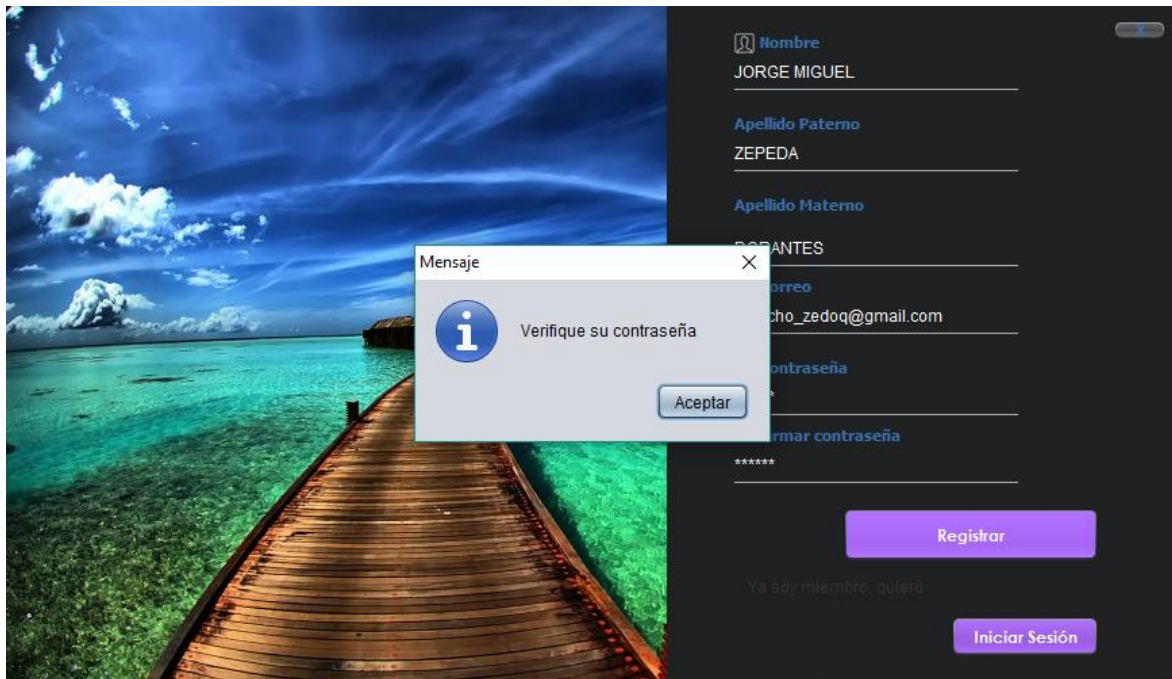


2.5 Campos incompletos (Ventana de registro)

Mientras los campos de registro no estén completos. El proceso de registro será detenido y el usuario candidato para el registro, no podrá ser dado de alta en el sistema.

En llegado caso de que todos los campos estén llenos, pero la contraseña determinada por el usuario sea distinta con la que se controla al confirmar, se mostrará un mensaje con la leyenda: “Verifique su contraseña”.

```
//Verifica que la contraseña sea la misma
if ((user.getPassword()).equals(user.getConfirmPassword())) {
    //Realiza conexión con la Base de datos de Agencia.
    Connection conectar = conexion.getConnection();
    if (conectar != null) {
        String url = "jdbc:mysql://localhost:3306/", nameBD = "Agencia";
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection cn = DriverManager.getConnection(url + nameBD,
                                                        "root", "root1");
            .....
        }
    }
}
```



## 2.6 Verificación de contraseña

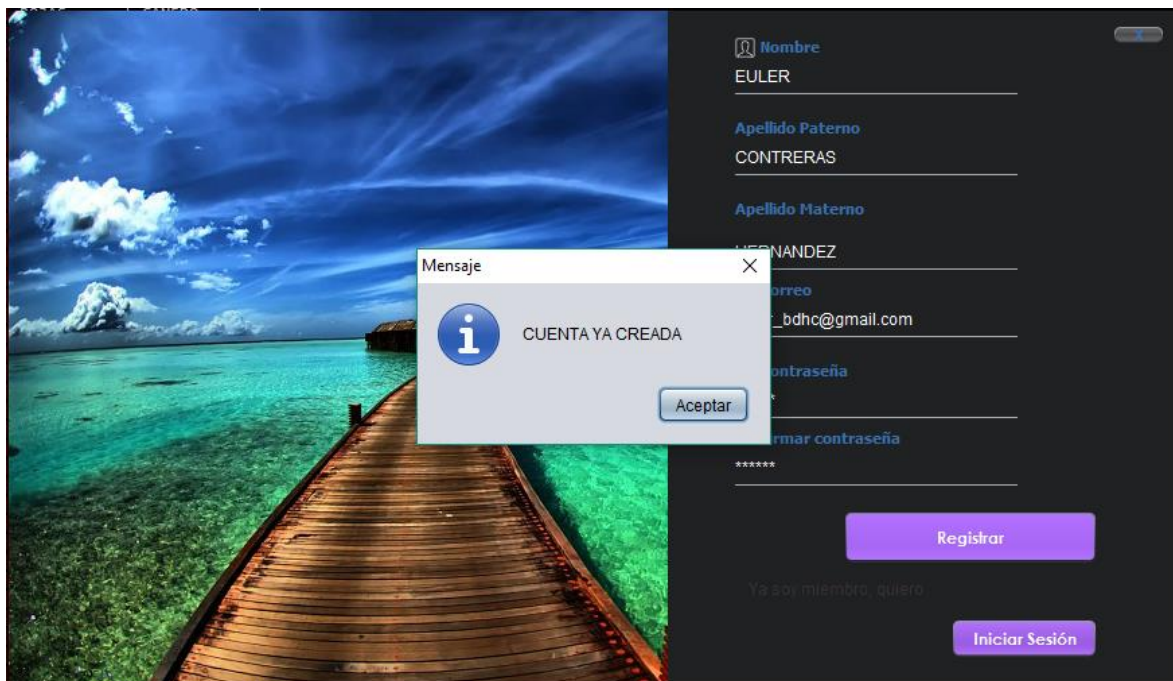
Asimismo, si en un determinado momento un cliente ya registrado desea volver a darse de alta, pero con un correo distinto, entonces el sistema propondrá una simple actualización del correo (y contraseña) respetando su cuenta.

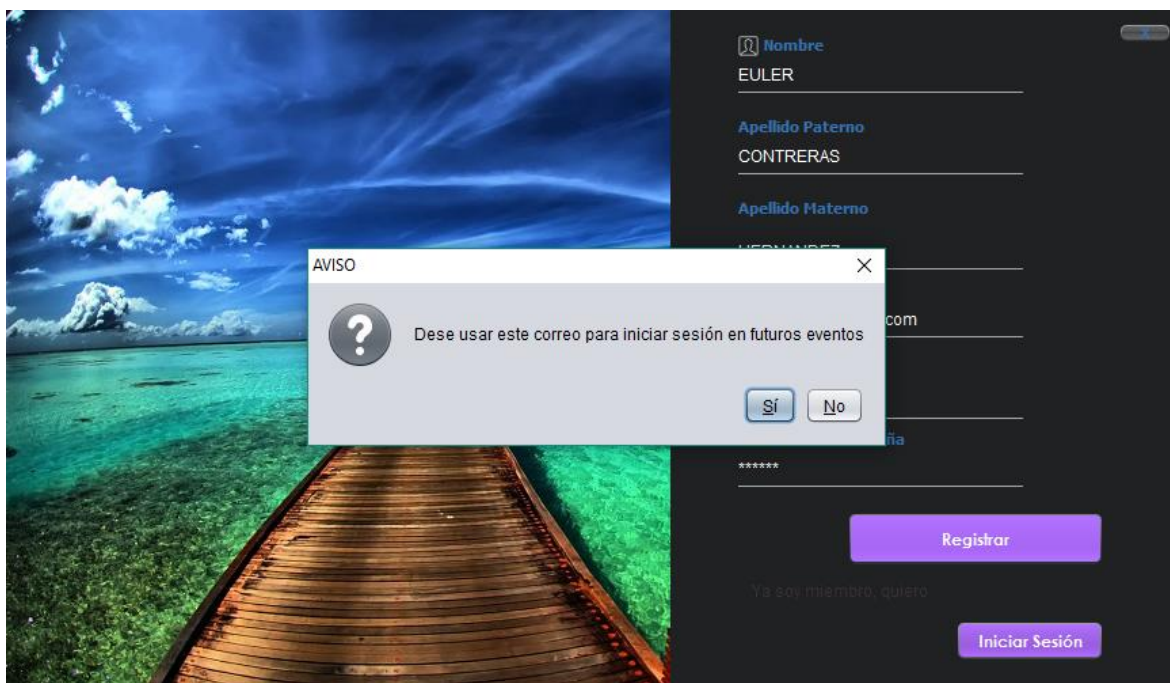


```

//Verifica si el cliente ya ha sido registrado con anterioridad
if (registro.buscarCliente_Repetido(user, conectar)) {
    /*Verifica si el proyecto de cliente no está usando
    un correo ya establecido en otra cuenta*/
    if (registro.buscarCorreoRepetida(user, conectar)==false) {
        JOptionPane.showMessageDialog(null,
            "Usuario previamente registrado");
        limpiar();
    } else {
        JOptionPane.showMessageDialog(null, "CUENTA YA CREADA");
        int res = JOptionPane.showConfirmDialog(null,
            "Dese usar este correo para iniciar sesión en futuros eventos",
            "AVISO", JOptionPane.YES_NO_OPTION);
        if (res == JOptionPane.YES_OPTION) {
            //System.out.println("dijo si");
            registro.justInsertedEmail(user, conectar);
            reserva.VentanaReserva a = new reserva.VentanaReserva();
            this.setVisible(false);
            a.setVisible(true);
        } else {
            if (res == JOptionPane.NO_OPTION) {
                //System.out.println("Dijo no");
                limpiar();
            }
        }
    }
}
}
}
}

```





## 2.7 Actualización de correo con cuenta ya registrada

En llegado caso de que el cliente decline la propuesta, se limpiarán todos los campos para que el proceso de reinicie, empero, si el cliente desea actualizar su correo y contraseña, se posicionará a continuación en la ventana para registrar la tarjeta de débito o crédito.

Cada uno de estos casos es controlado en el mismo evento del botón Registrar, a partir de las condiciones mostradas.

En caso ideal, en donde el cliente no ha sido registrado, el correo es distinto al de todos los demás registros y las contraseñas coinciden, entonces se ejecuta el siguiente código.

```
else {
    //Realiza inserción del cliente en la Base de datos.
    registro.insertar(user, cn, nomTable);
    /*Bloquea cada uno de los campos (TextFields)ya ingresados*/
    //Oculta la ventana
    this.setVisible(false);
    //Establece un intervalo de 1 segundo antes de abrir la nueva mentana
    Thread.sleep(1000);
    /*Se proyecta la ventana para agregar los datos respectivos
    a la tarjeta del usuario*/
    addCard.setVisible(true);
}
```

Cabe mencionar que los datos del cliente y, su correo electrónico y contraseña se encuentran en relaciones separadas dentro de la base de datos. Esto con el fin de haber normalizado para un mejor control.

De manera general, de la figura 2.3, podemos establecer la siguiente tabla.

Figura	Componentes	Descripción
1	Fondo de ventana	Se muestra una imagen dentro de una etiqueta, la cual está contenida en un panel. Esta se puede cambiar solamente en la parte del administrador del sistema.
2	Campos de entrada	<ul style="list-style-type: none"> <li>• Cada una de las cajas de texto deben de estar llenas. En caso de que una de ellas no lo esté, se limpiarán los campos completos.</li> <li>• Al ingresar la contraseña, esta se mostrará como asteriscos por seguridad del usuario.</li> </ul>
3	Botón de registro.	<ul style="list-style-type: none"> <li>• El botón registrar establece el dado de alta del cliente en la base de datos.</li> <li>• En caso de que el usuario ya exista, el botón iniciar sesión le permite ir a la ventana de inicio de sesión.</li> </ul>

Una vez que el registro se ha hecho, aparece un process bar que muestra (simula) la carga del registro.

Posteriormente, se procede a ir a la ventana del ingreso de la tarjeta de crédito/débito con la cual se realizarán las reservaciones pertinentes.



Figura 2.8 Process bar

 A screenshot of a web application's "Forma de pago" (Payment Method) section. The form is titled "DETALLES DE FORMA DE PAGO" and includes a shopping cart icon. It contains fields for "Número de tarjeta" (Card Number) and "CVV", a "Tipo" (Type) dropdown menu set to "Crédito", and an "Expiración" (Expiration) date selector set to "junio 2017". A green "Siguiente" (Next) button is at the bottom right. Numbered callouts (1, 2, 3, 4) point to the card number field, the type dropdown, the expiration date, and the next button respectively.

2.9 Registro de tarjeta.

En dado caso de que el cliente no ingrese algún número de tarjeta (la cual es obligatoria), se mostrará un mensaje con la leyenda de que la tarjeta es inválida. Para ello se utiliza el método, dentro de la clase `verifyCard`, `getNoCard`.

```
public String getNoCard() {
    String noCardF = "";
    String noCard01, noCard02, noCard03, noCard04;

    if (TXT_Card01.getText().equals("") || TXT_Card02.getText().equals("")
        || TXT_Card03.getText().equals("") || TXT_Card04.getText().equals(""))
    {
        JOptionPane.showMessageDialog(null, "Numero de tarjeta no válido");
        limpiar();
    } else {
        noCard01 = TXT_Card01.getText();
        noCard02 = TXT_Card02.getText();
        noCard03 = TXT_Card03.getText();
        noCard04 = TXT_Card04.getText();
        noCardF = noCard01 + noCard02 + noCard03 + noCard04;
    }
    return noCardF;
}
```



2.10 Número de tarjeta inválido.

Si el cliente ha ingresado un número de tarjeta válido, pero éste tiene una fecha de vigencia ya expirada, entonces el sistema tiene control del mismo manejo de error, mostrando un mensaje con la frase: "Tarjeta expirada". Este evento fue controlado gracias a el método, en la clase `verifyCard`, `getFecha`.



```

public String getFecha() {
    Calendar month = new GregorianCalendar();
    int monthP = month.get(Calendar.MONTH);
    String fecha = "", mes, año;
    if ((jYearChooser1.getYear()) == 2017) {
        if (jMonthChooser1.getMonth() <= monthP) {
            JOptionPane.showMessageDialog(null, "Tarjeta expirada");
            limpiar();
        } else {
            mes = Integer.toString(jMonthChooser1.getMonth() + 1);
            año = Integer.toString(jYearChooser1.getYear());
            if ((jMonthChooser1.getMonth() + 1) >= 10) {
                fecha = "" + año + "-" + mes + "-01";
            } else {
                fecha = "" + año + "-0" + mes + "-01";
            }
        }
    } else if (jYearChooser1.getYear() > 2017) {
        mes = Integer.toString(jMonthChooser1.getMonth() + 1);
        año = Integer.toString(jYearChooser1.getYear());
        if ((jMonthChooser1.getMonth() + 1) >= 10) {
            fecha = "" + año + "-" + mes + "-01";
        } else {
            fecha = "" + año + "-0" + mes + "-01";
        }
    } else {
        JOptionPane.showMessageDialog(null, "Tarjeta expirada");
        limpiar();
    }
    return fecha;
}

```

Datos	Forma de pago
 <h3>DETALLES DE FORMA DE PAGO</h3>	
<b>Número de tarjeta</b> <div>1387 - 7283 - 8732 - 3872</div>	<b>CVV</b> <div>3287</div>
<b>Tipo</b> <div>Crédito</div>	<b>Expiración</b> <div> <div>mayo</div> <div>2017</div> </div>
<div> <div>  <b>Tarjeta expirada</b> </div> <div> <div>Aceptar</div> </div> </div>	
<div>Siguiente</div>	

## 2.11 Tarjeta expirada

Cabe mencionar que cada vez que avancen los meses y años, gracias al uso de la librería de `java.util.GregorianCalendar`, se actualizará.

En mejor de los casos, en donde se llena correctamente el número de tarjeta y la tarjeta no está expirada, dará pauta a visualizar la ventana principal de Home.



**Datos** | **Forma de pago**

**DETALLES DE FORMA DE PAGO**

 **Número de tarjeta** **CVV**

8328 - 8832 - 3276 - 7637 3276

**Tipo** **Expiración**

Crédito octubre 2020

**Siguiete**

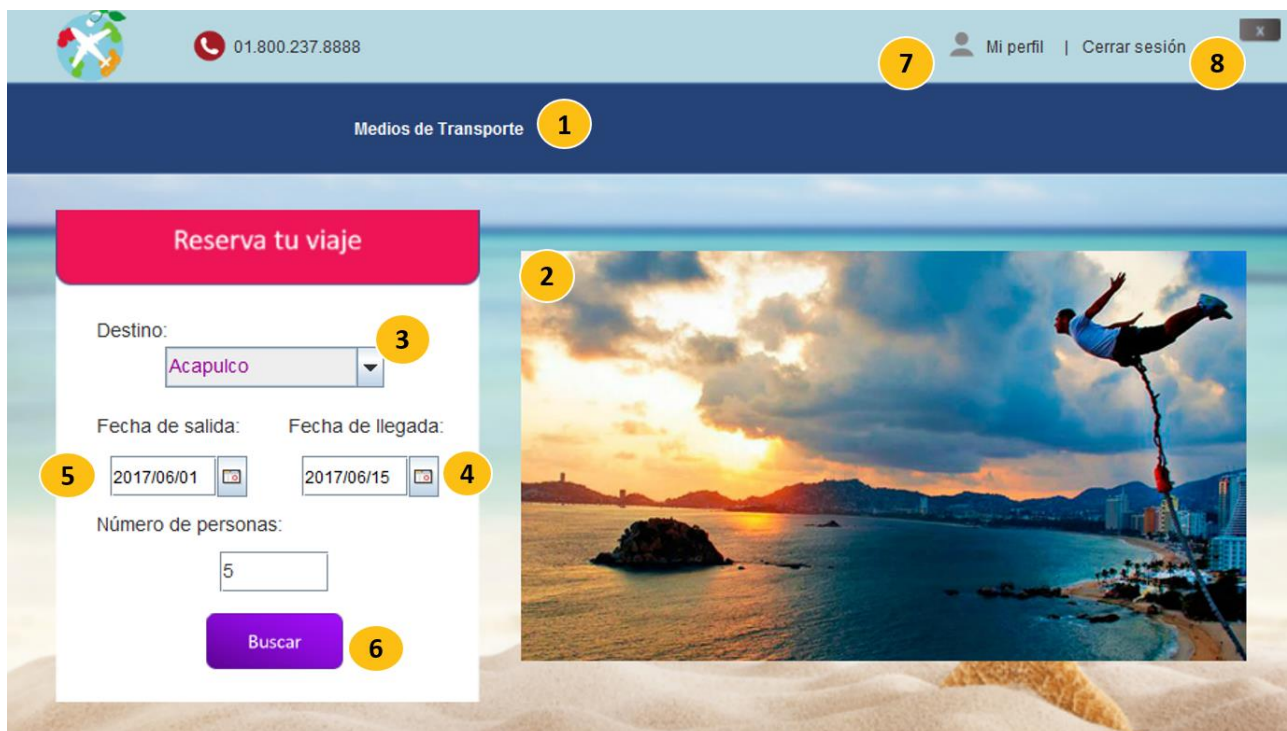
## 2.12 Registro de tarjeta exitoso

De esta manera, podemos establecer la siguiente tabla generalizada para la ventana mostrada en la imagen 2.9.

Figura	Componente(s).	Descripción.
1	Entrada de textos	Se registra de manera particular las debidas tarjetas de crédito asignadas a cada cliente, así como su clave interbancaria. <ul style="list-style-type: none"> <li>Se ha adaptado para que cada 4 dígitos, se posicione en la siguiente casilla las cajas de texto.</li> </ul>
2	Combo box	<ul style="list-style-type: none"> <li>Permite establecer qué tipo de tarjeta es la que está registrando el cliente para realizar sus debidas reservaciones.</li> </ul>
3	JMonthChooser y JYearChooser	<ul style="list-style-type: none"> <li>La opción del mes se va actualizando con respecto a la fecha que tenga el equipo del usuario.</li> <li>Se maneja el control de errores para establecer que la fecha no sea menor o igual a la actual.</li> <li>Si la fecha llega a ser menor o igual a la actual, se limpian los datos para ser registrados otros nuevos y vigentes.</li> </ul>

4	Botón de siguiente	<ul style="list-style-type: none"> <li>• Da acceso a la ventana de Home (principal) para realizar su respectiva reservación.</li> </ul>
---	--------------------	---

Al dar clic en el botón iniciar sesión y ser válida la información ingresada, o bien al haberse registrado con éxito el registro de un nuevo usuario, el sistema mostrará la ventana siguiente, en la cual el viajero podrá buscar hoteles y paquetes para cualquiera de los destinos que ofrece la Agencia de Viajes dentro del país.



**3.1 Pantalla de inicio (home)**

Figura	Componentes	Descripción
1	Botón	El botón "Medios de Transporte" lleva a una nueva ventana desde la cuál es posible adquirir algún boleto de avión o de camión.
2	Etiqueta	<ul style="list-style-type: none"> <li>• La etiqueta muestra fotografías de varios de los destinos turísticos que ofrece la agencia de viajes.</li> <li>• Las imágenes cambian cada 2 segundos.</li> </ul>
3	Cuadro combinado	<ul style="list-style-type: none"> <li>• Al dar clic sobre este cuadro se despliega un listado de lugares turísticos a los que puedes viajar.</li> <li>• Al seleccionar el destino, aparecerá el nombre como primera opción.</li> <li>• Este componente es clave para realizar las búsquedas de hoteles y servicios registrados en la base de datos.</li> </ul>

<b>4</b>	Calendario	<ul style="list-style-type: none"> <li>• Al dar clic sobre este elemento aparece un calendario que facilita la selección de una fecha.</li> <li>• Se valida que la fecha de llegada no sea anterior a la fecha de salida.</li> </ul>
<b>5</b>	Calendario	<ul style="list-style-type: none"> <li>• Al dar clic sobre este elemento aparece un calendario que facilita la selección de una fecha.</li> <li>• Se valida que la fecha de salida no sea siguiente a la fecha de llegada.</li> </ul>
<b>6</b>	Botón	Después de llenar los 4 campos se debe dar clic en el botón "Buscar", en este momento el sistema buscará la información en la base de datos, la consulta retornará en las ventanas siguientes: nombre, categoría y precios que ofrece cada una de las cadenas hoteleras en el destino elegido por el usuario.
<b>7</b>	Botón	"Mi perfil" permite ir a una nueva ventana donde se muestra información del usuario.
<b>8</b>	Botón	Cuando el usuario no desee realizar más transacciones puede terminar su sesión dando clic en este botón, el sistema guardará los cambios y lo devolverá a la primer pantalla, donde podrá iniciar una nueva sesión si así lo desea.

Al dar clic en el **Botón 6 (Buscar)**, donde previamente deben estar llenados los campos, de alguna otra manera, no se podrá continuar con la reservación del cliente, una vez llenados estos, el botón buscar tendrá un evento en el cual, validaremos las fechas, tanto en la que llegó como en la que selecciona de salida, de la siguiente forma:

```
//COMPARAMOS LAS FECHAS

Calendar cal1 = Calendar.getInstance();
Calendar cal2 = Calendar.getInstance();
cal1.setTime(date); //date = fecha de llegada
cal2.setTime(date2); //date2 = fecha de salida

//-----
//dias
int fLdia = date.getDate();
int fIdia = date2.getDate();

//meses
int fLmes = date.getMonth();
int fImes = date2.getMonth();

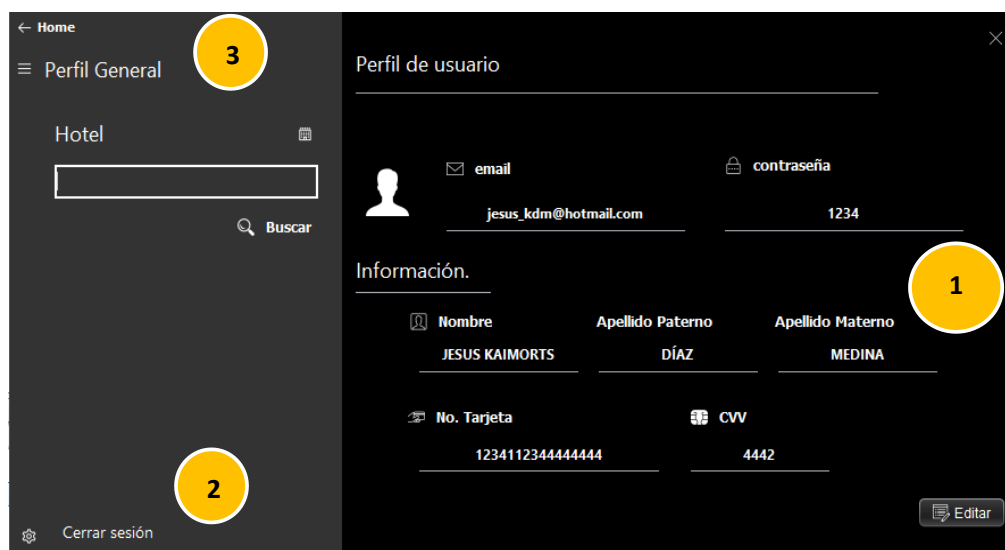
//con esto podemos controlar los días
int diaT = fIdia - fLdia;

//obtenemos un true si la fecha de ida es despues que la de
llegada
boolean fechaDespues = cal2.after(cal1);

//verificamos que la fecha de llegada no sea antes que la de ida
if (fechaDespues == false) {
    JOptionPane.showMessageDialog(null, "Ingresa una fecha
valida");
}
```

De esta manera, podemos controlar las fechas para poder mostrar un mensaje en pantalla para que ingrese una fecha Valida. Esto lo hace en base a que el día de la fecha de llegada no sea después que la fecha de salida.

Asimismo, en el botón 7 de “Mi perfil”, al momento de seleccionarla se muestra la siguiente ventana.



3.2 Ventana del “Mi perfil”.

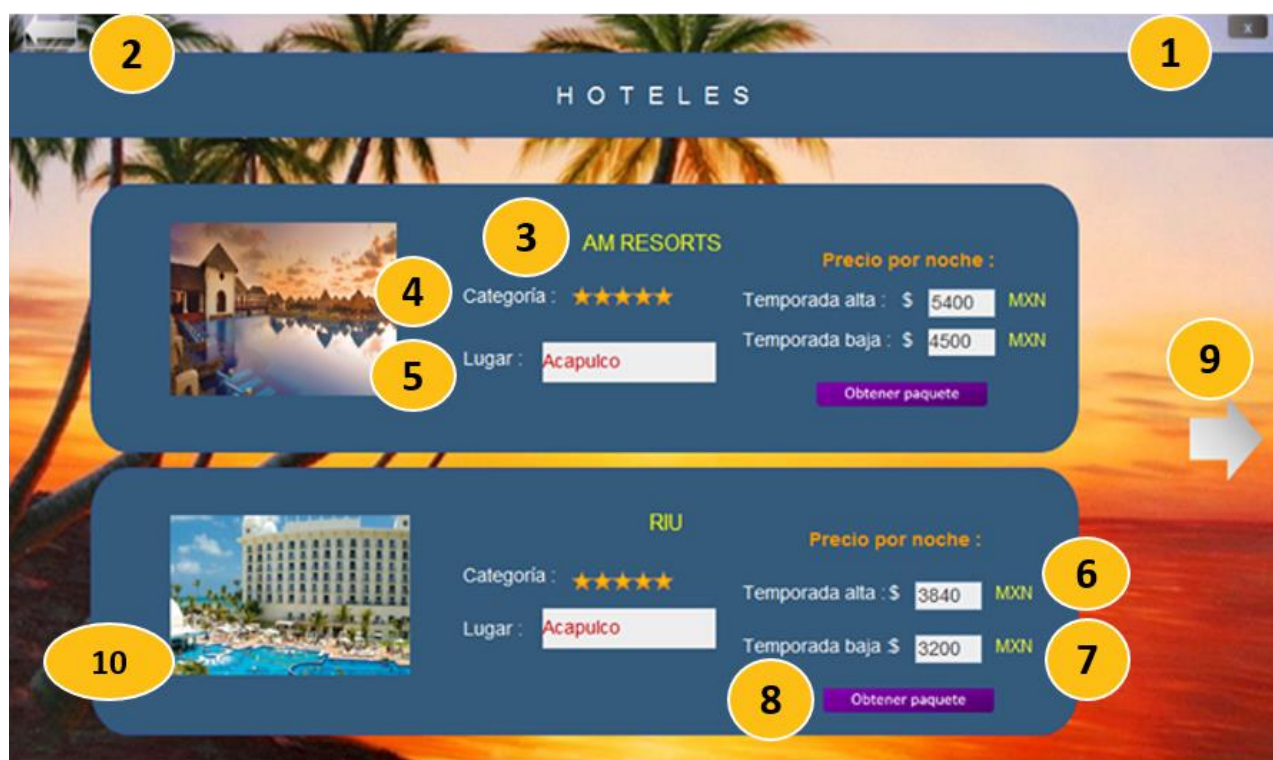
Una vez que se ha visualizado la información, el cliente tiene la opción de regresar a home o bien, de cerrar sesión.

En cada uno de las tablas se restringe su modificación directa. En futuras versiones, se puede hacer la mejora en la que el cliente podrá editar sus datos de manera instantánea. Asimismo, en versiones próximas poder realizar la búsqueda de los distintos hoteles, dependiendo de la zona en la que se encuentre.

De esta manera, generalizamos la ventana con la tabla mostrada a continuación.

Figura.	Componente.	Descripción.
1	Cajas de texto	<ul style="list-style-type: none"> <li>Se deshabilitó la opción de edición de éstas.</li> <li>Solamente sirve como modo lectura de la información del cliente.</li> </ul>
2	Etiqueta Home	<ul style="list-style-type: none"> <li>Permite regresar a la ventana principal para realizar alguna reservación.</li> </ul>
3	Etiqueta cerrar sesión.	<ul style="list-style-type: none"> <li>Finiquita la sesión iniciada.</li> </ul>

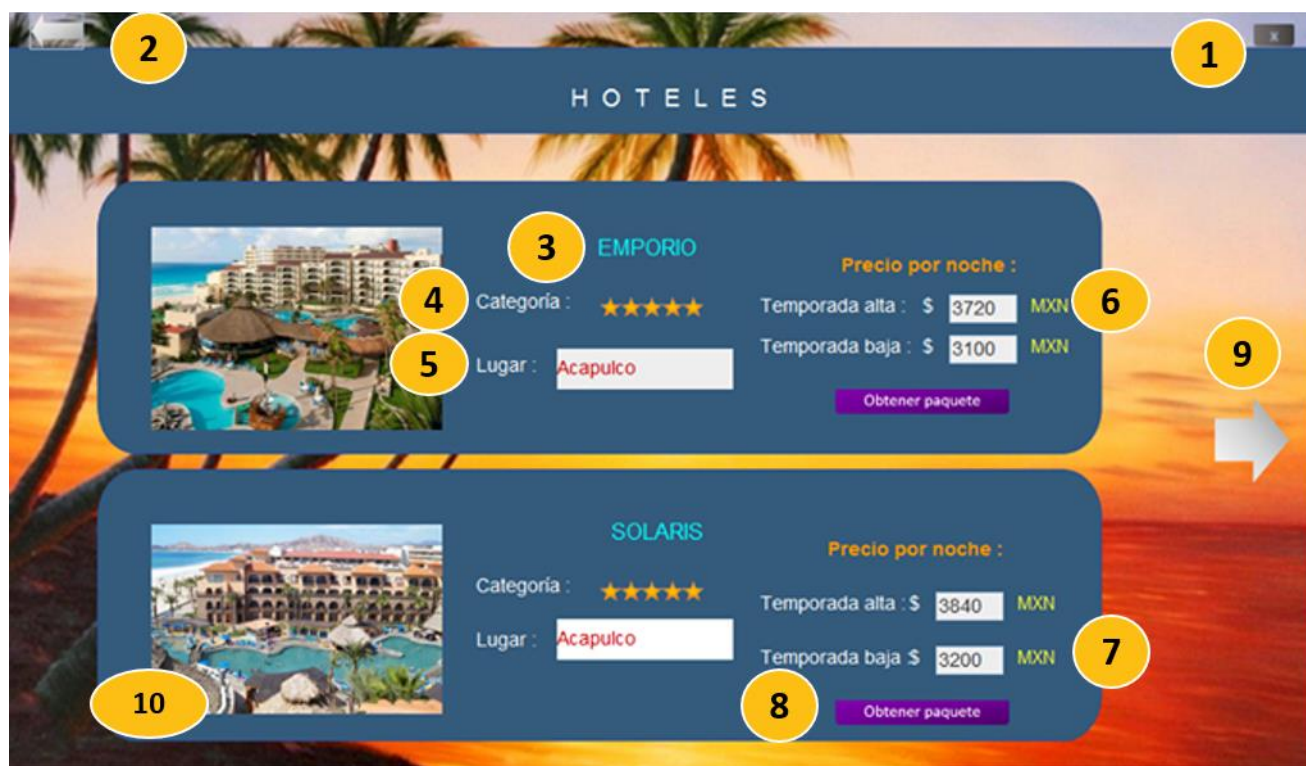
Posteriormente, una vez hecho la reservación, podremos apreciar la ventana de los hoteles disponibles.



4.1 Pantalla 1 de hoteles (AM RESORTS Y RIU).

Figura	Componentes	Descripción
1	Botón	Al dar clic sobre este botón, termina la ejecución del sistema, desaparece la ventana.
2	Botón	La flecha regresa al usuario a la pantalla de inicio (home) donde Puede elegir un nuevo destino. (Imagen 1.4)
3	Etiqueta	Nombre de la cadena hotelera, se encuentra establecido el nombre, para cada destino hay 5 cadenas hoteleras: AM RESORTS, RIU, EMPORIO, SOLARIS Y FIESTA AMERICANA. En esta ventana sólo se muestra la Información de los dos primeros.
4	Etiqueta	Esta etiqueta indica la categoría del hotel (1,2,3,4 o 5 estrellas)
5	Cuadro de texto	Muestra el nombre del destino turístico que fue elegido por el Viajero en la pantalla de inicio. (Imagen 1.4) . El campo de texto no es editable en ninguno de los recuadros.
6	Cuadro de texto	El cuadro de texto indica el monto a pagar por noche en temporada alta, el precio original almacenado en la base de datos fue aumentado en un 20%. Este elemento no puede ser modificado por el usuario.
7	Cuadro de texto	El cuadro de texto indica el monto a pagar por noche en temporada baja (en periodos no vacacionales), es el precio almacenado en la base de datos. Este elemento no puede ser modificado por el usuario.
8	Botón	"Obtener Paquete" lleva al usuario a una nueva ventana, donde podrá reservar servicios adicionales que generan un costo y que son ofrecidos por el Hotel. (Imagen 1.8)
9	Botón	Al dar clic sobre la flecha se muestra información de las dos Cadenas hoteleras siguientes: EMPORIO Y SOLARIS. (Imagen 1.6)
10	Imagen	Fotografía del hotel.



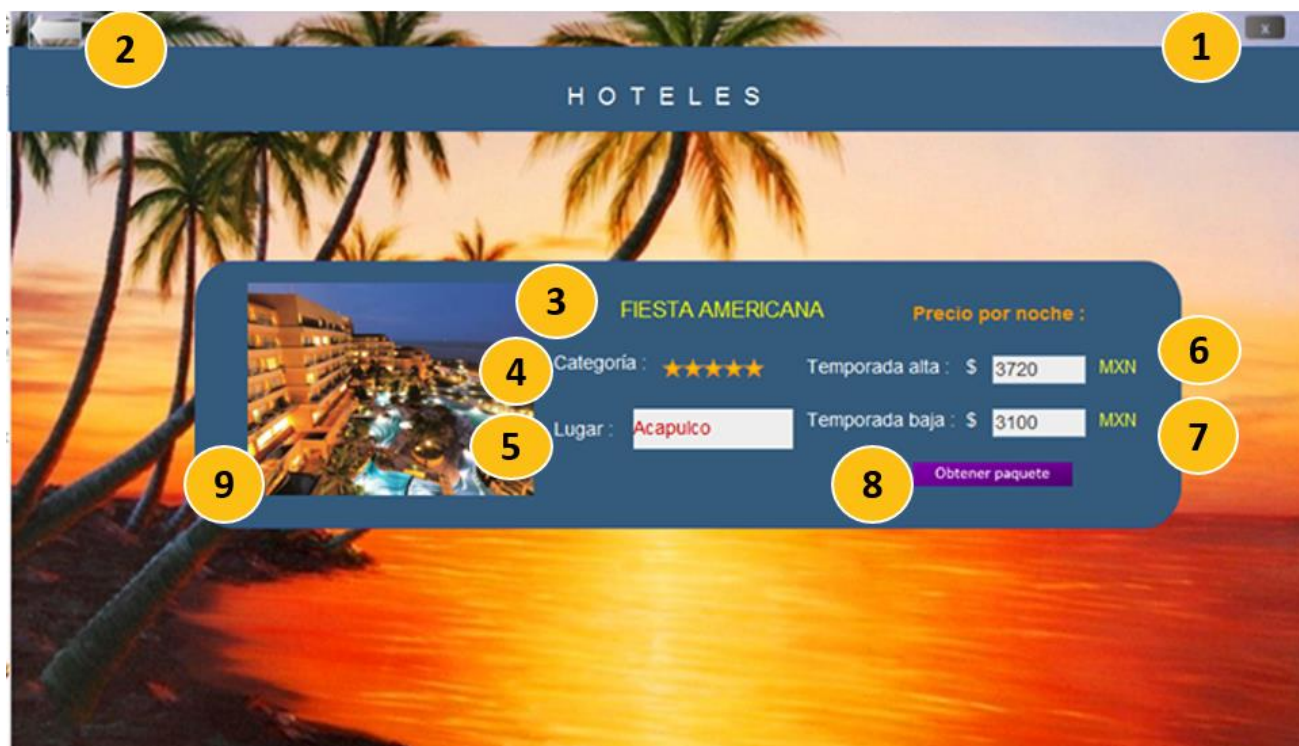


**4.2 Pantalla 2 de hoteles (EMPORIO Y SOLARIS)**

Figura	Componentes	Descripción
1	Botón	Al dar clic sobre este botón, termina la ejecución del sistema, desaparece la ventana.
2	Botón	La flecha regresa al usuario a la pantalla 1 de Hoteles donde puede ver la información de los hoteles anteriores: AM RESORTS Y RIU (Imagen 1.5)
3	Etiqueta	Nombre de la cadena hotelera, se encuentra establecido el nombre, para cada destino hay 5 cadenas hoteleras: AM RESORTS, RIU, EMPORIO, SOLARIS Y FIESTA AMERICANA. En esta ventana sólo se muestra la Información de EMPORIO Y SOLARIS.
4	Etiqueta	Esta etiqueta indica la categoría del hotel (1,2,3,4 o 5 estrellas)
5	Cuadro de texto	Muestra el nombre del destino turístico que fue elegido por el Viajero en la pantalla de inicio. (Imagen 1.4) . El campo de texto no es editable en ninguno de los recuadros.



6	Cuadro de texto	El cuadro de texto indica el monto a pagar por noche en temporada alta, el precio original almacenado en la base de datos fue aumentado en un 20%. Este elemento no puede ser modificado por el usuario.
7	Cuadro de texto	El cuadro de texto indica el monto a pagar por noche en temporada baja (en periodos no vacacionales), es el precio almacenado en la base de datos. Este elemento no puede ser modificado por el usuario.
8	Botón	"Obtener Paquete" lleva al usuario a una nueva ventana, donde podrá reservar servicios adicionales que generan un costo y que son ofrecidos por el Hotel. (Imagen 1.8)
9	Botón	Al dar clic sobre la flecha se muestra información de la Cadena hotelera siguientes FIESTA AMERICANA. (Imagen 1.7)
10	Imagen	Fotografía del hotel.



4.3 Pantalla 3 de hoteles (FIESTA AMERICANA)

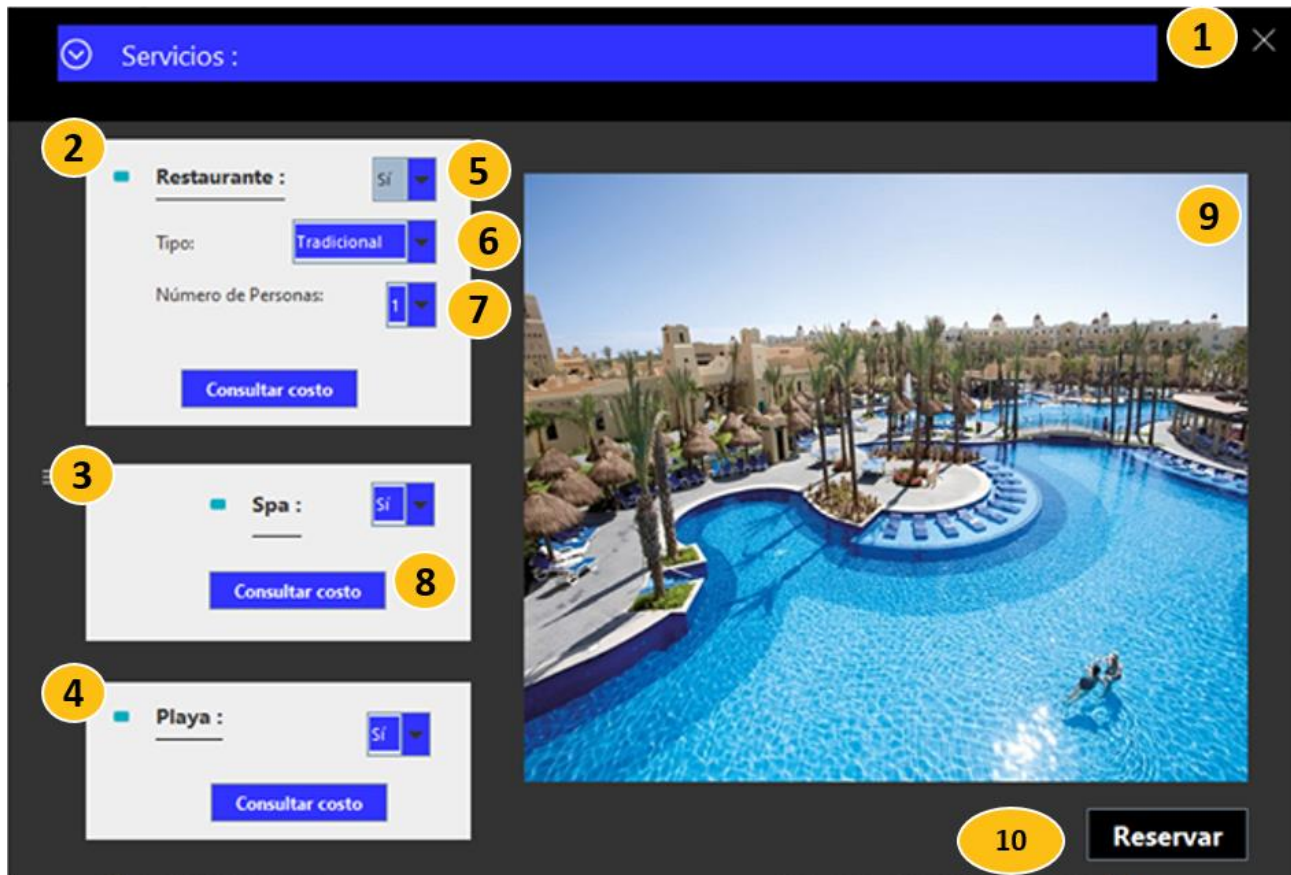
Figura	Componentes	Descripción
1	Botón	Al dar clic sobre este botón, termina la ejecución del sistema, desaparece la ventana.

<b>2</b>	Botón	La flecha regresa al usuario a la pantalla 2 de Hoteles donde puede ver la información de los hoteles anteriores: EMPORIO Y SOLARIS (Imagen 1.6)
<b>3</b>	Etiqueta	Nombre de la cadena hotelera, se encuentra establecido el nombre, para cada destino hay 5 cadenas hoteleras: AM RESORTS, RIU, EMPORIO, SOLARIS Y FIESTA AMERICANA. En esta ventana sólo se muestra la Información de FIESTA AMERICANA.
<b>4</b>	Etiqueta	Esta etiqueta indica la categoría del hotel (1,2,3,4 o 5 estrellas)
<b>5</b>	Cuadro de texto	Muestra el nombre del destino turístico que fue elegido por el Viajero en la pantalla de inicio. (Imagen 1.4) . El campo de texto no es editable en ninguno de los recuadros.
<b>6</b>	Cuadro de texto	El cuadro de texto indica el monto a pagar por noche en temporada alta, el precio original almacenado en la base de datos fue aumentado en un 20%. Este elemento no puede ser modificado por el usuario.
<b>7</b>	Cuadro de texto	El cuadro de texto indica el monto a pagar por noche en temporada baja (en periodos no vacacionales), es el precio almacenado en la base de datos. Este elemento no puede ser modificado por el usuario.
<b>8</b>	Botón	"Obtener Paquete" lleva al usuario a una nueva ventana, donde podrá reservar servicios adicionales que generan un costo y que son ofrecidos por el Hotel. (Imagen 1.8)
<b>9</b>	Imagen	Fotografía del Hotel.

Al momento de presionar el **Botón 8** de “obtener paquete” en cualquiera de las ventanas de hotel, se controlará el precio del hotel, donde se tomará en cuenta el número de plazas que estén hospedando en la reservación, además de tomar en cuenta los días en los que se está reservando y si el cliente se encuentra reservando en temporada alta o baja. Tomando como criterio el número de personas que van a hospedar el hotel, se generará un descuento del 35% si es que se encuentran 2 personas hospedadas en la habitación y el 60% si es que se trata de 3. Todo esto se verá reflejado en el **costo total**.

```
//con esto controlamos los días que se van a hospedar
String aux5 = TXT_DIA.getText();
int diaT = Integer.parseInt(aux5);

if(rpt==6 || rpt ==3 || rpt ==10 || rpt==11){
    if(plaza==1)
        H1.setCostoTempA(costoA*diaT);
        //se genera un descuento del 35%
    if(plaza==2)
        H1.setCostoTempA((int) (costoA*2*0.35*diaT));
    if(plaza==3)
        H1.setCostoTempA((int) (costoA*3*0.60*diaT));
}else{
    if(plaza==1)
        H1.setCostoTempB(costoB*diaT);
    if(plaza==2)
        H1.setCostoTempB((int) (costoB*2*0.35*diaT));
    if(plaza==3)
        H1.setCostoTempB((int) (costoB*3*0.60*diaT));
}
}
```



5.1 Pantalla de Servicios que ofrece el Hotel

Figura	Componentes	Descripción
<b>1</b>	Botón	Al dar clic sobre este botón, termina la ejecución del sistema, desaparece la ventana.
<b>2</b>	Panel	El panel contiene información clave que permite al usuario adquirir o consultar información sobre los restaurantes que tiene el Hotel elegido en las ventanas anteriores.
<b>3</b>	Panel	El panel contiene información clave que permite al usuario adquirir o consultar información sobre el servicio de Spa que tiene el Hotel elegido en las ventanas anteriores.
<b>4</b>	Panel	El panel contiene información clave que permite al viajero consultar información y reservar un lugar en alguna de las playas privadas del Hotel
<b>5</b>	Cuadro combinado	<ul style="list-style-type: none"> <li>Al dar clic sobre este cuadro se despliegan dos opciones: SI y NO.</li> <li>Si el viajero desea adquirir el servicio deberá dejar el cuadro con la opción SI seleccionada.</li> <li>Este componente es clave para saber qué servicios serán reservados, si el campo es marcado con un NO, no se aplicarán cargos y la información enviada a la base de datos será nula.</li> </ul>
<b>6</b>	Cuadro combinado	Al desplegar el cuadro, el viajero podrá seleccionar uno de los 5 tipos de restaurantes que se tienen: Tradicional, Celiacos, Vanguardista, Vegetariano o Internacional.
<b>7</b>	Cuadro combinado	Al desplegar el cuadro, el viajero podrá seleccionar el número de personas que podrán disfrutar de este servicio, el número que quede seleccionado será enviado a la base de datos si el primer Cuadro combinado quedó marcado con la opción SI
<b>8</b>	Botón	Al dar clic sobre "Consultar costo" el sistema envía un cuadro de texto con la cotización del monto a pagar en caso de que se desee adquirir el servicio. El dar clic en este botón no compromete al usuario a adquirir el servicio.
<b>9</b>	Etiqueta	<ul style="list-style-type: none"> <li>La etiqueta muestra fotografías de varios de los servicios que ofrece el Hotel seleccionado en alguna de las pantallas anteriores.</li> <li>Las imágenes cambian cada 3 segundos.</li> </ul>
<b>10</b>	Botón	Al dar clic sobre el botón "Reserva" el sistema analizará cuáles de los servicios fueron marcados con un "SI" para enviar la información a la base de datos y sumar el monto a pagar por el viajero, tomando en cuenta el costo de cada servicio seleccionado: Restaurante, Playa y Spa.

		En caso de que algún servicio sea marcado con la opción "NO", la base de datos contendrá valores nulos en esos campos y el monto total a pagar no será incrementado.
--	--	--

El botón principal de esta ventana es el botón de **Reservar**, en el cual, vamos a considerar que va a ir incrementando el precio mientras más servicios adicionales deseemos adquirir, cada uno con su respectiva consulta de precio por si el cliente desea o no adquirir el servicio. Todo esto lo controlamos con condicionales, en los cuales, podemos percatarnos de que opciones a escogido el usuario y de esta manera hacer las inserciones en la base de datos pertinentes, con respecto a la información del cliente.



**1.9 PANTALLA DE MEDIO DE TRANSPORTE**

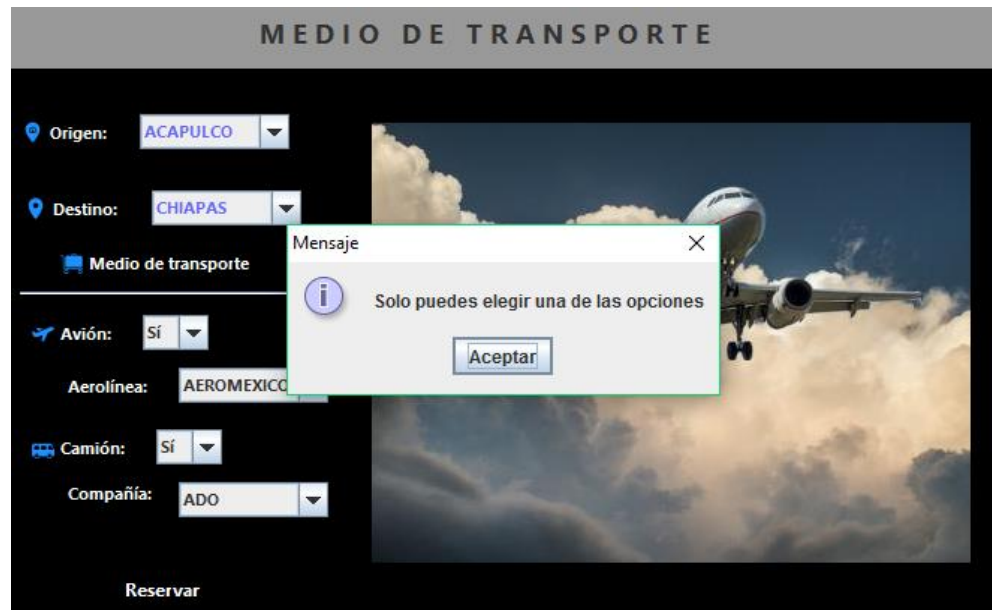
Figura	Componentes	Descripción
1	Panel	Dentro de este apartado, simplemente es un <b>Label</b> donde nos pone la palabra <b>“Origen”</b> para que el usuario sepa de qué se está hablando en dicho espacio.
2	Cuadro Combinado	En este <b>Combo box</b> simplemente se despliegan todas las opciones con respecto al <b>Origen</b>
3	Panel	Dentro de este apartado, simplemente es un <b>Label</b> donde nos pone la palabra <b>“Destino”</b> para que el usuario sepa de qué se está hablando en dicho espacio.
4	Cuadro Combinado	En este <b>Combo box</b> simplemente se despliegan todas las opciones con respecto al <b>Destino</b>
5	Panel	Dentro de este apartado, simplemente es un <b>Label</b> donde nos pone la palabra <b>“Avión”</b> para que el usuario sepa de qué se está hablando en dicho espacio.



6	Cuadro Combinado	En este <b>Combo box</b> simplemente se despliegan las opciones con respecto a la <b>afirmación o negación</b> de adquirir dicho servicio.
7	Panel	Dentro de este apartado, simplemente es un <b>Label</b> donde nos pone la palabra <b>“Aerolínea”</b> para que el usuario sepa de qué se está hablando en dicho espacio. (Va de la mano con el apartado de avión)
8	Cuadro Combinado	En este <b>Combo box</b> simplemente se despliegan todas las opciones con respecto al <b>Tipo de aerolínea que se desea.</b>
9	Panel	Dentro de este apartado, simplemente es un <b>Label</b> donde nos pone la palabra <b>“Camión”</b> para que el usuario sepa de qué se está hablando en dicho espacio.
10	Cuadro Combinado	En este <b>Combo box</b> simplemente se despliegan las opciones con respecto a la <b>afirmación o negación</b> de adquirir dicho servicio.
11	Panel	Dentro de este apartado, simplemente es un Label donde nos pone la palabra <b>“Compañía”</b> para que el usuario sepa de qué se está hablando en dicho espacio. (Va de la mano con el apartado de camión)
12	Cuadro Combinado	En este <b>Combo box</b> simplemente se despliegan todas las opciones con respecto al <b>Tipo de compañía de camion que se desea.</b>
13	Botón	Por ultimo este botón es el que nos da la confirmación de los datos capturados para que estos se manden a la base de datos.

- Algo que es de relevancia mencionar es que en la parte de los **Combo Box** (Afirmación o Negación) de avión o camión, si una se activa la otra por su parte desactiva, esto es una restricción en el código y a la hora de querer reservar pues la pantalla nos dirá que no es válido como se muestra en la siguiente imagen.

```
//Verifica que las condiciones de selección solo se determine un
//servicio, no los dos
if(sinoAvion==0 && sinoCamion==0)
{
    rpt=false;
    JOptionPane.showMessageDialog(null, "Solo puedes elegir una de las
opciones");
}
```



- Así mismo (como ya se mencionó anteriormente si pones que el Origen es igual al Destino) te manda un mensaje donde te dice que no se puede elegir el mismo origen y destino a la ves





```
//Verifica que el Destino no sea igual al Origen
if(cad1 == cad2)
{
    JOptionPane.showMessageDialog(null, "No puedes elegir el mismo origen y
destino");
    rpt=true;//Cambio el valor de la variable de control (bandera booleana)
}
```

```

//Esta es la parte en la cual se adquiere la reservacion y captura de los
datos
private void BTN_ReservarActionPerformed(java.awt.event.ActionEvent evt)
{
    Generador indice2 = new Generador();
    //Se cr//Captura lo que le llega al Combo Box de Origen y lo guarda en
cadlea un objeto de la clase Generador
//para que se utilice con la tabla de INFOCLIENTE

    int id2=indice2.generatedCode("INFOCLIENTE");
    //Se utiliza el metodo de la clase sobre INFOCLIENTE

    id2 = id2-1;
    //Esta operacion se hace para que el idTransporte valla contando desde 0

    String cad1=null,cad2=null,cad3=null,cad4=null,
cad5=null,cad6=null, //Asignacion de String

    actualiza=null,auxSuma,auxID, auxOrigen, auxDestino,s="";
    //Asignacion de String

    cad1 = CB_Origen.getSelectedItem().toString();
    //Captura lo que le llega al Combo Box de Origen y lo guarda en cad1

    cad2 = CB_Destinos.getSelectedItem().toString();
    //Captura lo que le llega al Combo Box de Destino y lo guarda en cad2

    cad3 = CB_Condicion1.getSelectedItem().toString();
    //Captura lo que le llega al Combo Box de Condicion1 y lo guarda en cad3

    cad4 = CB_Condicion2.getSelectedItem().toString();
    //Captura lo que le llega al Combo Box de Condicion2 y lo guarda en cad4

    cad5 = CB_Aerolinea.getSelectedItem().toString();
    //Captura lo que le llega al Combo Box de tipo Aerolínea y lo guarda en
cad5

    cad6 = CB_Compañia.getSelectedItem().toString();
    //Captura lo que le llega al Combo Box de Compañía y lo guarda en cad6

    int sinoAvion = CB_Condicion1.getSelectedIndex();
    //Simplemete hace el cambio de String a Int para la Condicion1

    int sinoCamion = CB_Condicion2.getSelectedIndex();
    //Simplemete hace el cambio de String a Int para la Condicion2

    boolean rpt;//Variable de tipo Boolean

    //Verifica que el Destino no sea igual al Origen
    if(cad1 == cad2)
    {
        JOptionPane.showMessageDialog(null, "No puedes elegir el mismo origen y
destino");
        rpt=true;//Cambio el valor de la variable de control (bandera booleana)
    }
}

```

```

//Verifica que las condiciones de selección solo se determine un
servicio, no los dos
if(sinoAvion==0 && sinoCamion==0)
{
    rpt=false;
    JOptionPane.showMessageDialog(null, "Solo puedes elegir una de las
opciones");
}

while(cad1 != cad2 && rpt==true)
{
    try {
        //Concatena el id2 que se utiliza en el Generador para pasarlo a
String
String auxid="" +id2;

        //Si el Origen es GUADALAJARA entonces se toma como idTransporte 1
// de igual forma se lanza una actualizacion sobre
PreparedStatement y el idReservacion con respecto al INFOCLIENTE
if(cad1=="GUADALAJARA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece
set idTransporte=1 where idReservacion="+auxid);
    pst2.executeUpdate();
}

        //Si el Origen es SAN JOSE DEL CABO entonces se toma como
idTransporte 2 de igual forma se lanza una actualizacion sobre
PreparedStatement y el idReservacion con respecto al INFOCLIENTE
if(cad1=="SAN JOSE DEL CABO")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece
set idTransporte=2 where idReservacion="+auxid);
    pst2.executeUpdate();
}

        //Si el Origen es CANCUN entonces se toma como idTransporte 3
de igual forma se lanza una actualizacion sobre PreparedStatement
y el idReservacion con respecto al INFOCLIENTE
if(cad1=="CANCUN")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece
set idTransporte=3 where idReservacion="+auxid);
    pst2.executeUpdate();
}

        //Si el Origen es ACAPULCO entonces se toma como idTransporte 4
de igual forma se lanza una actualizacion sobre PreparedStatement
y el idReservacion con respecto al INFOCLIENTE
if(cad1=="ACAPULCO")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece
set idTransporte=4 where idReservacion="+auxid);
    pst2.executeUpdate();
}
    }
}

```

```

//Si el Origen es PUERTO VALLARTA enotonces se toma como idTransporte 5
de igual forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="PUERTO VALLARTA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=5 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es CHIAPAS enotonces se toma como idTransporte 6 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="CHIAPAS")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=6 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es MAZATLAN enotonces se toma como idTransporte 7 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="MAZATLAN")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=7 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es RIVIERA MAYA enotonces se toma como idTransporte 8 de
igual forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="RIVIERA MAYA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=8 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es VERACRUZ enotonces se toma como idTransporte 9 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="VERACRUZ")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=9 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es MERIDA enotonces se toma como idTransporte 10 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="MERIDA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=10 where idReservacion="+auxid);
    pst2.executeUpdate();
}

```

```

//Si el Origen es COZUMEL enotonces se toma como idTransporte 11 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="COZUMEL")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=11 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es OAXACA enotonces se toma como idTransporte 12 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="OAXACA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=12 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es CIUDADA DE MEXICO enotonces se toma como idTransporte
13 de igual forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="CIUDAD DE MEXICO")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=13 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es VILLAHERMOSA enotonces se toma como idTransporte 14 de
igual forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="VILLAHERMOSA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=14 where idReservacion="+auxid);
    pst2.executeUpdate();
}

//Si el Origen es TIJUANA enotonces se toma como idTransporte 15 de igual
forma se lanza una actualización sobre PreparedStatement y el
idReservacion con respecto al INFOCLIENTE
if(cad1=="TIJUANA")
{
    PreparedStatement pst2 = cn.prepareStatement("update ofrece set
    idTransporte=15 where idReservacion="+auxid);
    pst2.executeUpdate();
}

```

```

//COSTO DE AVION
//Simplemente vamos a hacer una selección y vamos a proyectar los valores
//que se nos despliegan para así poder calcular el costo del Avion
String sql1 = "select a.costo from avion a, mtd m, medioTransporte mt
where mt.idTransporte = m.idTransporte "+"and m.idTD = a.idTD and
a.aerolinea ='"+cad5+"'"+" " and m.destino ='"+cad2+"'" and mt.origen
='"+cad1+"'" ;

Statement st = cn.createStatement();
//Guardamos en un Statement lo que nos despliegue la clase Connection

ResultSet rs = st.executeQuery(sql1);
//Ejecutamos una consulta y la ponemos sobre una variable ResultSet

String rpt1="";
while(rs.next())
{
    rpt1=rs.getString(1);
}
int numero1 = Integer.parseInt(rpt1);//Convertimos la cadena en un Int

//COSTO DE CAMIÓN
//Simplemente vamos a hacer una selección y vamos a proyectar los valores
//que se nos despliegan para así poder calcular el costo del Camión
String sql2 = "select a.costo from avion a, mtd m, medioTransporte mt
where mt.idTransporte = m.idTransporte "+"and m.idTD = a.idTD and
a.compañia ='"+cad6+"'"+" " and m.destino ='"+cad2+"'" and mt.origen
='"+cad1+"'" ;

Statement st2 = cn.createStatement();
//Guardamos en un Statement lo que nos despliegue la clase Connection

ResultSet rs2 = st.executeQuery(sql2);
//Ejecutamos una consulta y la ponemos sobre una variable ResultSet

String rpt2="";
while(rs2.next())
{
    rpt2=rs2.getString(1);
}
int numero2 = Integer.parseInt(rpt2);//Convertimos la cadena en un Int

//Obtenemos el costo anterior
String auxid2=""+"id2;
String sql = "select costoTotal from infocliente WHERE
idReservacion="+auxid2;
Statement stC = cn.createStatement();
String rptC="";//Almacena los datos de la consulta que vamos a realizar

ResultSet rsC = stC.executeQuery(sql);
while(rsC.next())
{
    rptC=rsC.getString(1);
}
int costoT = Integer.parseInt(rptC);

```

```

//VALIDANDO CONDICIONES DE SI O NO
int opc1=CB_Condicion1.getSelectedIndex();//Si se quiere Avión o no
int opc2=CB_Condicion2.getSelectedIndex();//Si se quiere Camión o no

String costoA = ""+(numero1+costoT);
//Se suman costos y se agregan a una sola variable

String costoC = ""+(numero2+costoT);
//Se suman costos y se agregan a una sola variable

auxID=" "+id2;
String auxCostoA =""+numero1;
String auxCostoC =""+numero2;

//SOLO CAMIÓN
//Se hacen las actualizaciones sobre la base de datos llenando sobre
//la parte de INFOCLIENTE
if (opc1==1 && opc2==0)
{
    s="update infocliente set origen='"+cad1+"', destino='"+cad2+"', "+
    "compañia='"+cad6+"', "+ "costoTotal="+costoC+" WHERE
    idReservacion="+auxID;
    PreparedStatement pst = cn.prepareStatement(s);
    pst.executeUpdate();
}

//SOLO AVIÓN
//Se hacen las actualizaciones sobre la base de datos llenando sobre
//la parte de INFOCLIENTE
if (opc1==0 && opc2==1)
{
    s="update infocliente set origen='"+cad1+"', destino='"+cad2+"', "+
    "aerolinea='"+cad5+"', "+ "costoTotal="+costoA+" WHERE
    idReservacion="+auxID;
    PreparedStatement pst = cn.prepareStatement(s);
    pst.executeUpdate();
}

} catch (SQLException ex)
{
    Logger.getLogger(MedioTransporte.class.getName()).log(Level.SEVERE,
    null, ex);
}
rpt=false;

reserva.VentanaReserva prueba = new reserva.VentanaReserva();
setVisible(false);
prueba.setVisible(true);

} //Fin de programa para el Medio de Transporte

```

## ANEXOS.

---

### Clase Cliente (Código)

```
package Relaciones;
/*@author Jesús Kaimorts Díaz Medina*/

public class Cliente
{
    private String ID;
    private String nombre;
    private String A_Paterno;
    private String A_Materno;
    private String Password;
    private String confirmPassword;
    private String email;

    public Cliente() {}

    public Cliente(String ID, String nombre, String A_Paterno, String
A_Materno, String Password, String confirmPassword, String email) {
        this.ID = ID;
        this.nombre = nombre;
        this.A_Paterno = A_Paterno;
        this.A_Materno = A_Materno;
        this.Password = Password;
        this.confirmPassword = confirmPassword;
        this.email = email;
    }
    //Setters & Getters of every data
}
```

### Clase Tarjeta (Código)

```
/*@author Jesús Kaimorts Díaz Medina*/
public class Tarjeta {

    private String noTarjeta;
    private int CVV;
    private String tipo;
    private String fecha; // Date fecha;

    public Tarjeta() {
    }

    public Tarjeta(String noTarjeta, int CVV, String tipo, String fecha)
{
        this.noTarjeta = noTarjeta;
        this.CVV = CVV;
        this.tipo = tipo;
        this.fecha = fecha;
    }

    ...
    <<getters>>
    <<setters>>

}
```



Clase Administrador.

Método cargarBD

```
public void cargarBD() {
    try {
        connectionMySQL con = new connectionMySQL();
        try (Connection conexion = con.getConnection()) {
            Statement st = conexion.createStatement();
            ResultSet rs = st.executeQuery("SHOW DATABASES;");
            CB_BD.removeAllItems();
            while (rs.next()) {
                CB_BD.addItem(rs.getString(1));
            }
            rs.close();
            conexion.close();
        }
    } catch (SQLException e) {
        e.getMessage();
    }
}
```

Método cargarTablas

```
public void cargarTablas() {
    try {
        connectionMySQL con = new connectionMySQL();
        try (Connection conexion = con.getConnection()) {
            Statement st = conexion.createStatement();
            ResultSet rs = st.executeQuery("SHOW TABLES;");
            CB_Relacion.removeAllItems();
            while (rs.next()) {
                CB_Relacion.addItem(rs.getString(1));
            }
            rs.close();
            conexion.close();
        }
    } catch (SQLException e) {
        e.getMessage();
    }
}
```

JFrame VentanaReserva (Home de Inicio)

BTN\_BUSCARHOTELActionPerformed (Código)  
"Botón Buscar"

```
//insertamos la fecha en ofrece
//ALMACENAR FECHAS
String formato = CALENDARIO_LLEGADA.getDateFormatString();
Date date = CALENDARIO_LLEGADA.getDate();
SimpleDateFormat sdf = new SimpleDateFormat(formato);
//FECHA DE LLEGADA
String fechaLlegada = String.valueOf(sdf.format(date));
String formato2 = CALENDARIO_SALIDA.getDateFormatString();
Date date2 = CALENDARIO_SALIDA.getDate();

SimpleDateFormat sdf2 = new SimpleDateFormat(formato2);
//FECHA DE SALIDA
String fechaSalida = String.valueOf(sdf2.format(date2));

//COMPARAMOS LAS FECHAS
Calendar cal1 = Calendar.getInstance();
Calendar cal2 = Calendar.getInstance();
cal1.setTime(date); //date = fecha de llegada
cal2.setTime(date2); //date2 = fecha de salida

//-----
//dias
int fLdia = date.getDate();
int fIdia = date2.getDate();

//meses
int fLmes = date.getMonth();
int fImes = date2.getMonth();

//con esto podemos controlar los días
int diaT = fIdia - fLdia;

//obtenemos un true si la fecha de ida es despues que la de
llegada
boolean fechaDespues = cal2.after(cal1);

//verificamos que la fecha de llegada no sea antes que la de ida
if (fechaDespues == false)
    JOptionPane.showMessageDialog(null, "Ingresa una fecha
    valida");
}

//verificamos que la reservación se haga en el mes
seleccionado(que no pase de un mes)
boolean auxMes = true;
if (fLmes != fImes) {
    JOptionPane.showMessageDialog(null, "Sólo se permite un
    máximo de un mes para dar de alta tu reservacion");
    auxMes = false;
}
```

```

String plaza = TXF_NUMPERSONAS.getText();
int rpt = Integer.parseInt(plaza);
boolean auxP = true;
if (rpt > 3) {
    JOptionPane.showMessageDialog(null, "Sólo se
    permite un máximo de 3 personas por reservacion");
    auxP = false;
}

while (fechaDespues == true && auxMes == true && auxP == true) {

    //insertamos el registro
    try {

        //insertamos un valor para la llave primaria de la r
        relacion ofrece
        PreparedStatement pstid = cn.prepareStatement("INSERT
        INTO ofrece(idReservacion,idCliente) VALUES (?,?)");
        pstid.setInt(1, id);
        pstid.setInt(2, id);
        pstid.executeUpdate();

        String auxid = "" + id;
        PreparedStatement pst =
        cn.prepareStatement("update ofrece set fecha='" +
        fechaLlegada + "' "
        + "WHERE idReservacion=" + auxid);

        pst.executeUpdate();
    } catch (SQLException ex) {

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

    //insertamos el segurodeViajero, la llave primaria
    try {
        PreparedStatement pstcomp = cn.prepareStatement("INSERT INTO
        seguroViajero(idCliente) VALUES (?)");
        pstcomp.setInt(1, id);
        pstcomp.executeUpdate();

        String aux = TXF_NUMPERSONAS.getText();
        int numPersonas = Integer.parseInt(aux);
        String tipo = "";
        int costo = 0;
        if (numPersonas == 1) {
            tipo = "Individual";
            costo = 150;
        }

        if (numPersonas == 2) {
            tipo = "Pareja";
            costo = 250;
        }
    }
}

```

```

if (numPersonas == 3) {
    tipo = "Familiar";
    costo = 400;
}

String auxCosto = "" + costo;

String auxid = "" + id;
PreparedStatement pst =
cn.prepareStatement("update seguroViajero set tipo='" + tipo
+ "'"
+ ",costo=" + auxCosto + " WHERE idCliente=" +
auxid);

pst.executeUpdate();
} catch (SQLException ex) {

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

//llenamos el atributo de plaza
try {

String plaza2 = TXF_NUMPERSONAS.getText();
//int numero = Integer.parseInt(plaza);
String use = "";
String sql = "update ofrece set plaza=" +
plaza2 + " where "
+ "idReservacion=" + id;
PreparedStatement psta = cn.prepareStatement(sql);
psta.executeUpdate();

} catch (SQLException ex) {

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

//Siguiendo formulario
Hoteles hol = new Hoteles();
hol.setVisible(true);

this.dispose();
Connection cn;
Conectar con = new Conectar();
cn = con.conectar();

String auxfLmes = "" + fLmes;
if (cn != null) {

Hoteles.TXF_LUGAR_AMRESORTS.setText(COMBO_DESTINOS.getSelected
dItem().toString());

```

```

private void LBL_CERRSESIONMouseClicked(java.awt.event.MouseEvent
evt) {
    // TODO add your handling code here:
    setVisible(false);
    newSession ns = new newSession();
    ns.setVisible(true);
}

private void TXT_NombreUserActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
}

private void LBL_PERFILMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    System.out.println(email+" ESTO TIENE");
    System.out.println(pss+" ESTO TIENE-");
    System.out.println(TXT_email.getText() +" ESTO TIENE-");
    System.out.println(TXT_pss.getText() +" ESTO TIENE-");
    if (!(TXT_email.getText().equals("jTextField1") &&
TXT_pss.getText().equals("jTextField1"))) {
        System.out.println("NO ESTÁN VACIOS");
        Cliente user = new Cliente();
        user.setEmail(TXT_email.getText());
        user.setPassword(TXT_pss.getText());
        User_Profile prof = new User_Profile();
        prof.insertData(user, cn);
        prof.asignar();
        setVisible(false);
        prof.setVisible(true);
    }else
    {
        System.out.println("SI ESTÁN VACIOS");
        Cliente user = new Cliente();
        user.setEmail(email);
        user.setPassword(pss);
        User_Profile prof = new User_Profile();
        prof.insertData(user, cn);
        prof.asignar();
        setVisible(false);
        prof.setVisible(true);
    }
}

```

JFrame Home19 (Servicios de Hotel)

jButton4ActionPerformed (Código)  
"Botón Reservar"

```
try {
    //creamos una variable del generador que nos va a ayudar a
    controlar la llave primaria
    Generador indice = new Generador();
    int id=indice.generatedCode("INFOCLIENTE");

    //insertamos un valor para la llave primaria
    PreparedStatement pstid = cn.prepareStatement("INSERT INTO
    infoCliente(idReservacion) VALUES (?)");
    pstid.setInt(1, id);
    pstid.executeUpdate();

    //costoSpa
    String TH=nomHotel;
    String sql1 = "select s.costo from spa s, hotel h where
    h.idhotel=s.idhotel and "
        + "h.nombre='"+TH+"'";
    Statement st = cn.createStatement();
    String rpt1="";
    ResultSet rs = st.executeQuery(sql1);
    while(rs.next())
    {
        rpt1=rs.getString(1);
    }
    int costoSpa = Integer.parseInt(rpt1);

    //costoPlaya
    String sql2 = "select p.costo from playa p, hotel h where
    h.idhotel=p.idhotel and "
        + "h.nombre='"+TH+"'";
    Statement st2 = cn.createStatement();
    String rpt2="";
    //almacena los datos de la consulta que vamos a realizar
    ResultSet rs2 = st2.executeQuery(sql2);
    while(rs2.next())
    {
        rpt2=rs2.getString(1);
    }
    int costoPlaya = Integer.parseInt(rpt2);

    //tipoRestaurante, numPersonas
    int i=Box_tipoR.getSelectedIndex();
    String TR="";
    if(i==0)
        TR="TRADICIONAL";
    if(i==1)
        TR="CELÍACOS";
    if(i==2)
        TR="VANGUARDISTA";
    if(i==3)
```

```

int aux=Box_numR.getSelectedIndex();
int rpt=0;
if(aux==0)
    rpt=1;
if(aux==1)
    rpt=2;
if(aux==2)
    rpt=3;

//precioRestaurante
String auxrpt="" + rpt;
String auxT = "select r.precio from restaurante r, hotel h
where h.idhotel=r.idhotel "
        + "and h.nombre='" + TH + "'" + " and r.numPersonas=" + auxrpt + "
and r.tipo= "
        + " '" + TR + "'";
Statement st3 = cn.createStatement();
String rpt3="";

//almacena los datos de la consulta que vamos a realizar
ResultSet rs3 = st3.executeQuery(auxT);
while(rs3.next())
{
    rpt3=rs3.getString(1);
}
int costoRestaurante = Integer.parseInt(rpt3);

//obtenemos todos los atributos que vamos a utilizar
int opc1=Box_opcR.getSelectedIndex();
int opc2=Box_costoS.getSelectedIndex();
int opc3=Box_opcP.getSelectedIndex();

String auxSpa = "" + costoSpa;
String auxPlaya= "" + costoPlaya;
String auxnumPersonas="" + rpt;
String auxPR="" + costoRestaurante;
String auxid="" + id;

String auxSV = "select costo from seguroViajero r, where
idCliente=" + auxid;
Statement stSV = cn.createStatement();
String rptSV="";

//almacena los datos de la consulta que vamos a realizar
ResultSet rsSV = stSV.executeQuery(auxT);
while(rsSV.next())
{
    rptSV=rsSV.getString(1);
}
int costoSV = Integer.parseInt(rptSV);
indice.UpdateCode("OFRECE");

//con esto controlo el costo de temp baja/alta
int costoTemp = costoTempA + costoTempB + costoSV;

```

```

//si desea todos los servicios
//hacemos un update en la relacion infocliente para agregar
los servicios adquiridos
if(opc1==0&&opc2==0&&opc3==0){
    String s="update infocliente set
costoSpa="+auxSpa+", costoPlaya="+auxPlaya+", "
    + "restaurante='"+TR+"",
numPersonas="+auxnumPersonas+", "
+ "precioRestaurante="+auxPR+" WHERE idReservacion="+auxid;
PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();

    //hacemos un update en infocliente para sumar los
costos de los servicios adquiridos
int
costoTotal=costoSpa+costoRestaurante+costoPlaya+costoTemp;
String auxCT=""+costoTotal;
    String s2="update infocliente set
costoTotal="+auxCT+" WHERE idReservacion="+auxid;
PreparedStatement pst2 = cn.prepareStatement(s2);
pst2.executeUpdate();
}

//si no desea adquirir el spa
if(opc1==0&&opc2==1&&opc3==0){
String s="update infocliente set costoPlaya="+auxPlaya+", "
    + "restaurante='"+TR+"",
numPersonas="+auxnumPersonas+", "
    + "precioRestaurante="+auxPR+" WHERE
idReservacion="+auxid;

PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();

//hacemos un update en infocliente para sumar los costos de
los servicios adquiridos
int costoTotal=costoPlaya+costoRestaurante+costoTemp;
String auxCT=""+costoTotal;

String s2="update infocliente set costoTotal="+auxCT+" WHERE
idReservacion="+auxid;
PreparedStatement pst2 = cn.prepareStatement(s2);
pst2.executeUpdate();
}

//si no desea adquirir la playa
if(opc1==0&&opc2==0&&opc3==1){
String s="update infocliente set costoSpa="+auxSpa+", "
+ "restaurante='"+TR+"", numPersonas="+auxnumPersonas+", "
+ "precioRestaurante="+auxPR+" WHERE idReservacion="+auxid;

PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();
}

```



```

//si no desea adquirir el restaurante
if(opc1==1&&opc2==0&&opc3==0){
    String s="update infocliente set
costoSpa="+auxSpa+", costoPlaya="+auxPlaya+", "
    + "WHERE idReservacion="+auxid;

PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();

    //hacemos un update en infocliente para sumar los
    costos de los servicios adquiridos
    int costoTotal=costoPlaya+costoSpa+costoTemp;
    String auxCT="" + costoTotal;
    String s2="update infocliente set
costoTotal="+auxCT+" WHERE idReservacion="+auxid;
PreparedStatement pst2 = cn.prepareStatement(s2);
pst2.executeUpdate();
}

//si desea el restaurante pero no spa, ni playa
if(opc1==0&&opc2==1&&opc3==1){
String s="update infocliente set "
    + "restaurante='"+TR+"',
numPersonas="+auxnumPersonas+", "
    + "precioRestaurante="+auxPR+" WHERE
idReservacion="+auxid;

PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();

    //hacemos un update en infocliente para sumar los
    costos de los servicios adquiridos
    int costoTotal=costoRestaurante+costoTemp;
    String auxCT="" + costoTotal;
    String s2="update infocliente set
costoTotal="+auxCT+" WHERE idReservacion="+auxid;
PreparedStatement pst2 = cn.prepareStatement(s2);
pst2.executeUpdate();
}

//si desea spa, pero ni restaurante ni playa
if(opc1==1&&opc2==0&&opc3==1){
    String s="update infocliente set
costoSpa="+auxSpa+" WHERE idReservacion="+auxid;

PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();

    //hacemos un update en infocliente para sumar los
    costos de los servicios adquiridos
    int costoTotal=costoSpa+costoTemp;
    String auxCT="" + costoTotal;
    String s2="update infocliente set
costoTotal="+auxCT+" WHERE idReservacion="+auxid;
PreparedStatement pst2 = cn.prepareStatement(s2);
pst2.executeUpdate();
}

```

```

//si desea playa, pero ni restaurante ni spa
if(opc1==1&&opc2==1&&opc3==0){
    String s="update infocliente set
costoPlaya="+auxPlaya+" WHERE idReservacion="+auxid;

PreparedStatement pst = cn.prepareStatement(s);
pst.executeUpdate();

    //hacemos un update en infocliente para sumar los
    costos de los servicios adquiridos
    int costoTotal=costoPlaya+costoTemp;
    String auxCT=""+costoTotal;
    String s2="update infocliente set
costoTotal="+auxCT+" WHERE idReservacion="+auxid;
    PreparedStatement pst2 = cn.prepareStatement(s2);
    pst2.executeUpdate();
}

    //hacemos un update en costoTotal de infocliente
    por si no desea adquirir ninguno de
    //los servicios adicionales
    if(opc1==1&&opc2==1&&opc3==1){
    int costoTotal=costoTemp;
    String auxCT=""+costoTotal;
    String s="update infocliente set
costoTotal="+auxCT+" WHERE idReservacion="+auxid;
    PreparedStatement pst = cn.prepareStatement(s);
    pst.executeUpdate();
}

//AGREGAMOS EL ATRIBUTO DE nombreHotelRegion
    String s2="update infocliente set
nombreHotelRegion='"+nomHotel+" "+nomRegion+" ' "
    + "WHERE idReservacion="+auxid;

PreparedStatement pste = cn.prepareStatement(s2);
pste.executeUpdate();

//actualizamos el índice
indice.UpdateCode("INFOCLIENTE");

//System.exit(0);
    Servicios.MedioTransporte go = new
Servicios.MedioTransporte();
    setVisible(false);
    go.setVisible(true);

} catch (SQLException ex) {

    Logger.getLogger(Homel9.class.getName()).log(Level.SEVERE,
null, ex);
}

```