



Instituto Tecnológico de Oaxaca

“Departamento de sistemas y computación”

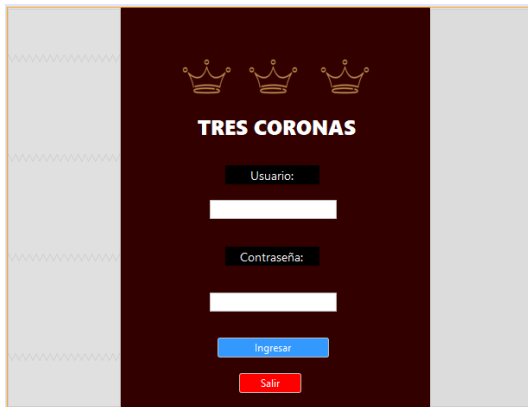
PROYECTO FINAL
ALQUILER DE AUTOS
TOPICOS AVANZADOS DE PROGRAMACION
EQUIPO 22
CANSECO YAHIR ALEJANDRO
MARTINEZ NIETO ADELINA
8 – 9 AM

13 de diciembre del 2024

Mi proyecto consiste en un alquiler de autos, este tiene como objetivo administrar vehículos para que el usuario pueda alquilar alguna de su gusto para sus necesidades, este proyecto tiene un apartado de usuario estándar y otro de administrador para que se pueda hacer una correcta gestión de los vehículos disponibles y sus complementos.

Empezamos por la clase loginInvitado:

Este se estructura de la siguiente manera



```
import javax.swing.JOptionPane;
```

```
import java.sql.Connection;
```

Hago el uso de esas librerías para obtener la conexión a mi base de datos y para hacer el uso de los textos y botones de tipo JOptionPane.

```
private void IngresarActionPerformed(java.awt.event.ActionEvent evt) {  
    String usuario = Usuario.getText();  
    char[] pass = Contraseña.getPassword();  
    String contraseña = new String(pass);  
  
    if (usuario.isEmpty() || contraseña.isEmpty()) {  
        JOptionPane.showMessageDialog(this, "Por favor, completa todos los campos.", "Advertencia", JOptionPane.WARNING_MESSAGE);  
        return;  
    }  
    try {  
        // Conectar a la base de datos  
        Conexion conexion = new Conexion();  
        Connection con = conexion.conectar();  
  
        if (con != null) {  
            // Consulta para verificar si existe el usuario y la contraseña en la base de datos  
            String query = "SELECT * FROM usuarios WHERE usuario = ? AND contraseña = ?";  
            java.sql.PreparedStatement stmt = con.prepareStatement(query);  
            stmt.setString(1, usuario); // Reemplaza el primer ? con el valor de 'usuario'  
            stmt.setString(2, contraseña); // Reemplaza el segundo ? con el valor de 'contraseña'  
  
            // Ejecutar la consulta  
            java.sql.ResultSet rs = stmt.executeQuery();  
  
            // Si existe un registro que coincida con el usuario y la contraseña  
            if (rs.next()) {  
                // Si el usuario y la contraseña son correctos, mostramos un mensaje de éxito  
                vistaR nl = new vistaR();  
                nl.setVisible(true);  
                this.dispose();  
            }  
        }  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(this, "Error al conectar a la base de datos.", "Error", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

Este código es parte de un sistema de inicio de sesión en una aplicación. Su función principal es tomar un usuario y contraseña ingresados en la interfaz gráfica de la aplicación, luego intentar verificar si esos datos coinciden con los almacenados en una base de datos.

Captura los datos ingresados: Primero, el código toma el valor del campo de texto donde el usuario escribe su nombre de usuario (usuario) y la contraseña (contraseña), que es tomada como un arreglo de caracteres y convertida a una cadena de texto.

Verificación de campos vacíos: Luego verifica si los campos de usuario o contraseña están vacíos. Si es así, muestra una advertencia para que el usuario complete ambos campos.

Conexión a la base de datos: Si los campos no están vacíos, intenta conectar a la base de datos usando una clase llamada Conexion. Si se conecta con éxito, procede con la consulta.

Consulta SQL: El código prepara una consulta SQL para verificar si el nombre de usuario y la contraseña coinciden con algún registro en la base de datos. Para esto, se utiliza un PreparedStatement que protege la consulta de posibles ataques de inyección SQL.

Verificación de usuario y contraseña: Si la consulta devuelve un resultado (rs.next()), significa que encontró un usuario con ese nombre y contraseña, por lo que abre una nueva ventana (vistaR) y cierra la ventana actual.

Manejo de errores: Si no hay coincidencias, muestra un mensaje de error. Si hay algún problema con la conexión a la base de datos o con la consulta SQL, también muestra un mensaje de error.

```
private void agregarValidacionesUsuarioYContraseña() {
    // Validación para Usuario: Solo letras, números y guion bajo (_)
    Usuario.addKeyListener(new java.awt.event.KeyAdapter() {
        @Override
        public void keyTyped(java.awt.event.KeyEvent evt) {
            char c = evt.getKeyChar();
            if (!Character.isLetterOrDigit(c) && c != '_') {
                evt.consume(); // Evita que se escriba el carácter
            }
        }
    });

    // Validación para Contraseña: Evitar espacios al inicio y al final
    Contraseña.addKeyListener(new java.awt.event.KeyAdapter() {
        @Override
        public void keyTyped(java.awt.event.KeyEvent evt) {
            char c = evt.getKeyChar();
            if (Character.isWhitespace(c)) {
                evt.consume(); // Evita que se escriba espacios
            }
        }
    });
}
```

Este bloque de código define una validación para el campo "Usuario". Usa un KeyListener, que se activa cada vez que el usuario presiona una tecla.

Cuando el usuario escribe una tecla, el código verifica si el carácter ingresado es una letra (Character.isLetterOrDigit(c)) o un número, o si es el guion bajo (_).

Si el carácter no es ninguno de esos, el código usa evt.consume(), que detiene la escritura de ese carácter, evitando que se ingrese un símbolo no permitido.

Ahora pasamos a la ventana VistaR

TRES CORONAS

Recolección

¿En que estado comienza el viaje?

Fecha: AAAA-MM-DD

2024-12-16

Horario de recolección

12:00:00

Devolución

¿Donde termina el viaje?

Fecha: AAAA-MM-DD

2024-12-17

Horario de devolución

01:00:00

Limpiar

Continuar

Activar Windows
Ve a Configuración para activar

Hago el uso de estas librerías:

```
import java.sql.Connection;  
import javax.swing.*;  
import javax.swing.event.ChangeEvent;  
import javax.swing.event.ChangeListener;  
import java.util.Date;  
import java.sql.SQLException;  
import java.sql.PreparedStatement;
```

```
public class vistaR extends javax.swing.JFrame {  
    Conexion con1 = new Conexion();  
    Connection conet;
```

Con este bloque mando a llamar a mi clase Conexión que es la que se encargara de comunicarse por así decirlo con la base de datos.

```

private void continuarRecoleccionActionPerformed(java.awt.event.ActionEvent evt) {
    // Recuperar datos de los campos
    String lugarRec = lugar_recoleccion.getSelectedItem().toString();
    String lugarDev = lugar_devolucion.getSelectedItem().toString();
    String fechaInicio = fecha_inicio_recoleccion.getText();
    String fechaFin = fecha_fin_devolucion.getText();
    String horaInicio = hora_inicio_recoleccion.getText();
    String horaFin = hora_fin_devolucion.getText();

    // Validaciones
    if (lugarRec.equals("¿En qué estado comienza el viaje?") || lugarDev.equals("¿Dónde termina el viaje?")) {
        JOptionPane.showMessageDialog(this, "Por favor selecciona un lugar Válido.");
        return;
    }

    if (fechaInicio.isEmpty() || fechaFin.isEmpty() || horaInicio.isEmpty() || horaFin.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Por favor llena todos los campos.");
        return;
    }

    try {
        // Validar formato de fecha
        java.sql.Date.valueOf(fechaInicio); // Formato esperado: YYYY-MM-DD
        java.sql.Date.valueOf(fechaFin);
    } catch (IllegalArgumentException e) {
        JOptionPane.showMessageDialog(this, "Por favor ingresa las fechas en el formato YYYY-MM-DD.");
        return;
    }

    try {
        // Validar formato de hora
        java.sql.Time.valueOf(horaInicio); // Formato esperado: HH:MM:SS
    } catch (IllegalArgumentException e) {
        JOptionPane.showMessageDialog(this, "Por favor ingresa las horas en el formato HH:MM:SS.");
        return;
    }

    // Crear el SQL de inserción
    String sql = "INSERT INTO alquileres (lugar_recoleccion, lugar_devolucion, fecha_inicio_recoleccion, fecha_fin_devolucion, id_alquiler) VALUES (?, ?, ?, ?, ?)";

    try {
        conet = con1.conectar();
        PreparedStatement pstmt = conet.prepareStatement(sql, java.sql.Statement.RETURN_GENERATED_KEYS);

        pstmt.setString(1, lugarRec); // lugar de recolección
        pstmt.setString(2, lugarDev); // lugar de devolución
        pstmt.setDate(3, java.sql.Date.valueOf(fechaInicio)); // fecha de inicio
        pstmt.setDate(4, java.sql.Date.valueOf(fechaFin)); // fecha de fin
        pstmt.setTime(5, java.sql.Time.valueOf(horaInicio)); // hora de inicio
        pstmt.setTime(6, java.sql.Time.valueOf(horaFin)); // hora de fin

        // Ejecutar la inserción
        pstmt.executeUpdate();

        // Recuperar el id generado automáticamente
        java.sql.ResultSet rs = pstmt.getGeneratedKeys();
        if (rs.next()) {
            id_alquiler = rs.getInt(1);
            System.out.println("ID generado: " + id_alquiler); // Verifica si se muestra el ID generado
        }
    }
}

```

1. Recuperar datos de los campos:

Se obtienen los valores de los campos de la interfaz de usuario: lugares de recolección y devolución, fechas y horas de inicio y fin. Estos datos se guardan en variables.

2. Validaciones:

Validación de los lugares:

Se comprueba si el lugar de recolección o el lugar de devolución contienen los valores predeterminados (por ejemplo, "¿En qué estado comienza el viaje?"). Si es así, muestra un mensaje pidiendo al usuario que seleccione un lugar válido.

Validación de los campos de fecha y hora:

Se verifica que los campos de fecha (inicio y fin) y hora (inicio y fin) no estén vacíos. Si alguno está vacío, se muestra un mensaje de advertencia.

Validación del formato de fecha:

Se valida que las fechas estén en el formato YYYY-MM-DD. Si no se cumple este formato, se lanza una excepción y se muestra un mensaje pidiendo al usuario que ingrese las fechas correctamente.

Validación del formato de hora:

De manera similar, se valida que las horas estén en el formato HH:MM:SS. Si no lo están, se muestra un mensaje de error.

3. Inserción de datos en la base de datos:

Si todas las validaciones pasan correctamente, el código construye una consulta SQL (INSERT INTO alquileres...) para guardar los datos en la tabla de alquileres de la base de datos.

Se utiliza un PreparedStatement para insertar los valores de los campos en la base de datos de manera segura, previniendo inyecciones SQL.

Luego, la inserción se ejecuta con executeUpdate().

4. Recuperación del ID generado:

Después de insertar el nuevo alquiler en la base de datos, se recupera el ID generado automáticamente por la base de datos (usualmente un valor autoincremental) y se muestra en la consola. Este ID puede ser utilizado para otras operaciones o para asociarlo con otros datos.

5. Mensajes y navegación:

Si la inserción es exitosa, muestra un mensaje de confirmación.

Luego, abre una nueva ventana (probablemente una ventana para registrar los pasajeros asociados con el alquiler) pasando el ID del alquiler recién creado.

Finalmente, cierra la ventana actual (this.dispose()).

6. Manejo de errores:

Si ocurre un error durante la inserción (por ejemplo, problemas con la conexión a la base de datos), se captura la excepción SQLException y se muestra un mensaje de error con detalles.

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

private void fecha_inicio_recoleccionMouseClicked(java.awt.event.MouseEvent evt) {
    fecha_inicio_recoleccion.setText("");
}

private void fecha_fin_devolucionMouseClicked(java.awt.event.MouseEvent evt) {
    fecha_fin_devolucion.setText("");
    // TODO add your handling code here:
}

private void hora_inicio_recoleccionMouseClicked(java.awt.event.MouseEvent evt) {
    hora_inicio_recoleccion.setText(""); // TODO add your handling code here:
}

private void hora_fin_devolucionMouseClicked(java.awt.event.MouseEvent evt) {
    hora_fin_devolucion.setText(""); // TODO add your handling code here:
}

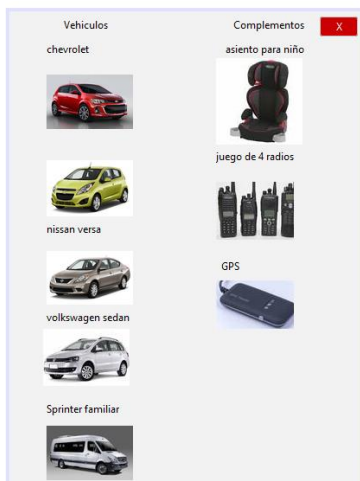
private void limpiarMouseClicked(java.awt.event.MouseEvent evt) {
    hora_fin_devolucion.setText("01:00:00");
    hora_inicio_recoleccion.setText("12:00:00");
    fecha_inicio_recoleccion.setText("2024-12-16");
    fecha_fin_devolucion.setText("2024-12-17");
    // TODO add your handling code here:
}
```

En esta clase lo que hago es editar mis entradas de texto para que esto muestre un poco mejor el diseño del programa, lo que hice fue editar las funciones del mouse sobre cada entrada de texto esto para que cuando corra el programa por default aparezca la fecha y hora en los campos y que cuando le de click encima se borren en automatico.

Posteriormente pasamos a la clase “pasajeros”

The screenshot shows a web application interface for 'TRES CORONAS'. At the top, there is a dark red header with a crown icon and the text 'TRES CORONAS'. Below the header, there is a section titled 'Cantidad de pasajeros' with two dropdown menus for 'Niños' and 'Adultos', both set to '1'. Below this, there are two main sections: 'Seleccionar vehiculo' and 'Complementos'. The 'Seleccionar vehiculo' section contains a table with columns: Marca, Modelo, Año, Tipo, Color, Matricula, Precio, and Disponi... The 'Complementos' section contains a table with columns: Nombre, Descripcion, Precio, and Disponible. At the bottom of the 'Seleccionar vehiculo' section, there is a button labeled 'ver vehiculos'. In the bottom right corner, there is a Windows watermark that says 'Activar Windows' and 'Ve a Configuración para activar Windows'.

Aquí el usuario puede elegir su vehiculo y su complemento si es que desea agregar alguno, además esta clase se enlaza con el administrador ya que él puede modificar los vehículos y complementos, abajo esta un botón llamado “ver vehículos” el cual al darle click nos lleva a esta ventana



En esta ventana podemos visualizar los vehículos y los complementos nada mas, regresando a nuestra clase “pasajeros”

```

*/
public void cargarVehiculos() {
    String query = "SELECT marca, modelo, año, tipo, color, matricula, precio_dia, disponibilidad FROM vehiculos";
    try (Connection con = con1.conectar(); Statement stmt = con.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
        // Crear el modelo para la tabla de vehiculos
        DefaultTableModel modeloVehiculos = (DefaultTableModel) tabla_models.getModel();
        modeloVehiculos.setRowCount(0); // Limpiar cualquier dato anterior

        while (rs.next()) {
            // Obtener los datos de cada vehiculo
            String marca = rs.getString("marca");
            String modelo = rs.getString("modelo");
            String año = rs.getString("año");
            String tipo = rs.getString("tipo");
            String color = rs.getString("color");
            String matricula = rs.getString("matricula");
            double precioDia = rs.getDouble("precio_dia");
            double disponible = rs.getDouble("disponibilidad");

            // Añadir una fila a la tabla (con la URL de la imagen)
            Object[] row = {marca, modelo, año, tipo, color, matricula, precioDia, disponible};
            modeloVehiculos.addRow(row);
        }

    } catch (SQLException e) {
        System.out.println("Error al cargar vehiculos: " + e.getMessage());
    }
}

```

Esta recupera los detalles de los vehículos almacenados en la tabla vehiculos de la base de datos. La información solicitada incluye: marca, modelo, año, tipo, color, matrícula, precio por día y disponibilidad.

Ademas, el código utiliza un Connection (con1.conectar()) para conectarse a la base de datos.


Luego, un Statement se usa para ejecutar la consulta (stmt.executeQuery(query)).

El ResultSet (rs) almacena los resultados obtenidos de la consulta.

Básicamente lo que hace es que carga los datos de los vehículos almacenados en una base de datos y los muestra en la tabla. Primero, se hace una consulta a la base de datos, luego los resultados se recorren y se agregan a la tabla para que el usuario pueda verlos de forma estructurada. Si ocurre algún error en el proceso, se muestra un mensaje de error en la consola.

La siguiente ventana es esta:

Regresar

 **TRES CORONAS**

Nombre (s)

Apellidos

Edad

Numero telefonico

Correo Electronico


```

private void nombre_clienteActionPerformed(java.awt.event.ActionEvent evt) {
    // Obtener el texto del JTextField
    String nombre = nombre_cliente.getText();

    // Validar que el campo no esté vacío
    if (nombre.isEmpty()) {
        javax.swing.JOptionPane.showMessageDialog(this, "El campo 'Nombre' no puede estar vacío.");
        return;
    }

    // Conexión a la base de datos
    String sql = "INSERT INTO alquileres (nombre_cliente) VALUES (?)";
    try {
        conet = con1.conectar(); // Asegúrate de que Conexión tenga un método getConnection()
        java.sql.PreparedStatement pstmt = conet.prepareStatement(sql);
        pstmt.setString(1, nombre); // Establecer el nombre en la consulta

        int resultado = pstmt.executeUpdate(); // Ejecutar la consulta
        if (resultado > 0) {
            javax.swing.JOptionPane.showMessageDialog(this, "Nombre guardado exitosamente!");
            nombre_cliente.setText(""); // Limpiar el JTextField
        } else {
            javax.swing.JOptionPane.showMessageDialog(this, "Error al guardar el nombre.");
        }
    } catch (java.sql.SQLException e) {
        javax.swing.JOptionPane.showMessageDialog(this, "Error al conectar con la base de datos: " + e.getMessage());
    }
}

```

Activar Windows
Ve a Configuración para activar Windows.

El cual se encarga de insertar el nombre del cliente en la base de datos y si mandas el campo vacío este te da una alerta, además hace el uso de if y else para avisarte cuando el nombre es guardado de manera exitosa o algún error al guardar el nombre.

También tenemos el método para validar el teléfono del cliente

```

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

private void ineActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Selecciona el archivo del INE");
    fileChooser.setFileFilter(new javax.swing.filechooser.FileNameExtensionFilter("Archivos JPG", "jpg"));

    int returnValue = fileChooser.showOpenDialog(this);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        rutaINE = fileChooser.getSelectedFile().getAbsolutePath(); // Asignar la ruta seleccionada
        JOptionPane.showMessageDialog(this, "Archivo INE seleccionado: " + rutaINE);
    } else {
        JOptionPane.showMessageDialog(this, "No se seleccionó ningún archivo.");
    }
}

```

Y para que pida lo que es la licencia e ine, este impedirá el hecho de omitir este paso ya que es un requisito necesario para seguir usando el programa.

```

private void licenciaActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Selecciona el archivo de la Licencia de Conducir");
    fileChooser.setFileFilter(new javax.swing.filechooser.FileNameExtensionFilter("Archivos JPG", "jpg"));

    int returnValue = fileChooser.showOpenDialog(this);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        rutaLicencia = fileChooser.getSelectedFile().getAbsolutePath(); // Asignar la ruta seleccionada
        JOptionPane.showMessageDialog(this, "Archivo Licencia seleccionado: " + rutaLicencia);
    } else {
        JOptionPane.showMessageDialog(this, "No se seleccionó ningún archivo.");
    }
}


private void regresareservActionPerformed(java.awt.event.ActionEvent evt) {
    tarjeta d = new tarjeta();
    d.setVisible(true);
    this.dispose(); // TODO add your handling code here:
}

// Método para validar todos los campos
private boolean validarCampos() {
    // Validar que los campos no estén vacíos
    if (nombre_cliente.getText().trim().isEmpty() ||
        apellidos_cliente.getText().trim().isEmpty() ||
        edad_cliente.getText().trim().isEmpty() ||

```

Activar Windows
Ve a Configuración para activar Windows.

De la reserva me manda a solicitar los datos de la tarjeta del usuario, esto en caso de cualquier tipo de golpe o daño físico que reciba el vehículo


TRES CORONAS

Nombre de la tarjeta

Numero de la tarjeta

Fecha de expiracion

Codigo de seguridad

```

private void btnContinuarActionPerformed(java.awt.event.ActionEvent evt) {
    // Validar campos
    if (!validarCampos()) {
        return; // Detener si hay errores en la validación
    }

    // Obtener los datos después de validar
    String nombreTarjeta = tarjeta.getText();
    String numeroTarjeta = numtarjeta.getText();
    Date fechaExpiracion = (Date) exp.getValue();
    String codigoSeguridad = codseg.getText();

    // Formatear fecha para MySQL
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String fechaExp = sdf.format(fechaExpiracion);

    // Insertar datos en la base de datos
    String sql = "INSERT INTO tarjeta (Nombretarjeta, Ntarjeta, Fexpiracion, Nseguridad) VALUES (?, ?, ?, ?)";

    try {
        conet = con1.conectar();
        PreparedStatement pstmt = conet.prepareStatement(sql);

        pstmt.setString(1, nombreTarjeta);
        pstmt.setString(2, numeroTarjeta);
        pstmt.setString(3, fechaExp);
        pstmt.setString(4, codigoSeguridad);

        pstmt.executeUpdate();

        javax.swing.JOptionPane.showMessageDialog(this, "Tarjeta guardada exitosamente.");
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Error al guardar la tarjeta: " + ex.getMessage());
    }
}

```

Este método se encarga de validar que todos los campos relacionados con los datos de una tarjeta de crédito estén correctamente formateados y sean válidos. Si alguno de los campos no es válido (por ejemplo, si el número de tarjeta no tiene 16 dígitos o si la fecha de expiración está en el pasado), se muestra un mensaje de error y se devuelve false. Si todo es correcto, se devuelve true.

Al darle click en continuar nos vamos al apartado de la confirmación del alquiler.



Aquí nos aparecen los datos que ingresamos y este al darle clic en aceptar lo que hará es generar un pdf con la ayuda de la librería “itext5.jar” además que al mismo tiempo establecerá la conexión con la base de datos extrayendo los datos y los enviara al correo electrónico previamente ingresado.

```

- import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import com.itextpdf.text.Document;
import com.itextpdf.text.Font;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;
import java.io.FileOutputStream;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.activation.*;
import java.io.File;
import com.itextpdf.text.pdf.PdfReader;
import java.io.IOException;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Confirmacion extends javax.swing.JFrame {
    private Connection connection;

    /**
     * Creates new form Confirmacion
     */
    public Confirmacion() {
        initComponents();
        conectarBaseDatos();
        cargarDatosEnAreaDeTexto();
        setLocationRelativeTo(null);
    }

    private void conectarBaseDatos() {
        String url="jdbc:mysql://localhost:3306/alquilerautos";
        String usuario="root";
        String contraseña="yahir";
        try{
            connection=DriverManager.getConnection(url,usuario,contraseña);
        }catch(SQLException e){
            e.printStackTrace();
            JOptionPane.showMessageDialog(this,"Error al conectar con la base de datos");
        }
    }

    private void cargarDatosEnAreaDeTexto(){
        String consulta = "SELECT nombre_cliente, apellidos_cliente, edad_cliente, telefono_cliente, " +
            "correo_cliente, lugar_recoleccion, lugar_devolucion, cantidad_ninos, cantidad_adultos, " +
            "vehiculo_seleccionado, complementos, fecha_inicio_recoleccion, hora_inicio_recoleccion, " +
            "fecha_fin_devolucion, hora_fin_devolucion " +
            "FROM alquileres " +
            "ORDER BY id_alquiler DESC LIMIT 1"; // Seleccionamos solo el último registro
    }
}

```

```

StringBuilders datos = new StringBuilders();

if (rs.next()) {
    datos.append("Nombre: ").append(rs.getString("nombre_cliente")).append(" ");
    datos.append(rs.getString("apellidos_cliente")).append("\n");
    datos.append("Edad: ").append(rs.getInt("edad_cliente")).append("\n");
    datos.append("Teléfono: ").append(rs.getString("telefono_cliente")).append("\n");
    datos.append("Correo: ").append(rs.getString("correo_cliente")).append("\n");
    datos.append("Lugar de Recolección: ").append(rs.getString("lugar_recoleccion")).append("\n");
    datos.append("Lugar de Devolución: ").append(rs.getString("lugar_devolucion")).append("\n");
    datos.append("Cantidad de Niños: ").append(rs.getInt("cantidad_ninos")).append("\n");
    datos.append("Cantidad de Adultos: ").append(rs.getInt("cantidad_adultos")).append("\n");
    datos.append("Vehículo Seleccionado: ").append(rs.getString("vehiculo_seleccionado")).append("\n");
    datos.append("Complementos: ").append(rs.getString("complementos")).append("\n");
    datos.append("Fecha de Inicio de Recolección: ").append(rs.getString("fecha_inicio_recoleccion")).append("\n");
    datos.append("Hora de Inicio de Recolección: ").append(rs.getString("hora_inicio_recoleccion")).append("\n");
    datos.append("Fecha de Fin de Devolución: ").append(rs.getString("fecha_fin_devolucion")).append("\n");
    datos.append("Hora de Fin de Devolución: ").append(rs.getString("hora_fin_devolucion")).append("\n");
}

AreaDeTexto.setText(datos.toString());

} catch (SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(this, "Error al recuperar los datos");
}

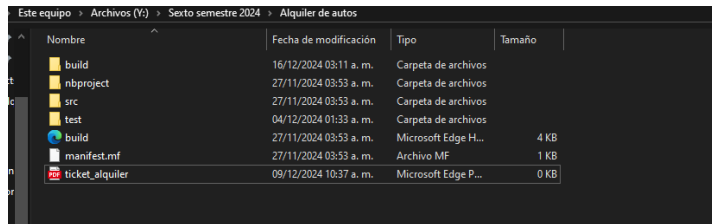
private void borrarDatosDeLaBaseDeDatos() {
    String consulta = "DELETE FROM alquileres";

    try (Statement stat = connection.createStatement()) {

```

El correo demora unos segundos en llegar pero es funcional!!!

También se puede visualizar el pdf en la raíz de nuestro programa



Esta es mi clase que establece la conexión con la base de datos

```

public class Conexion {
    String bd = "alquilerautos";
    String url = "jdbc:mysql://localhost:3306/";
    String user = "root";
    String password = "yahir";
    String driver = "com.mysql.cj.jdbc.Driver";
    Connection cx;

    public Conexion() {
    }

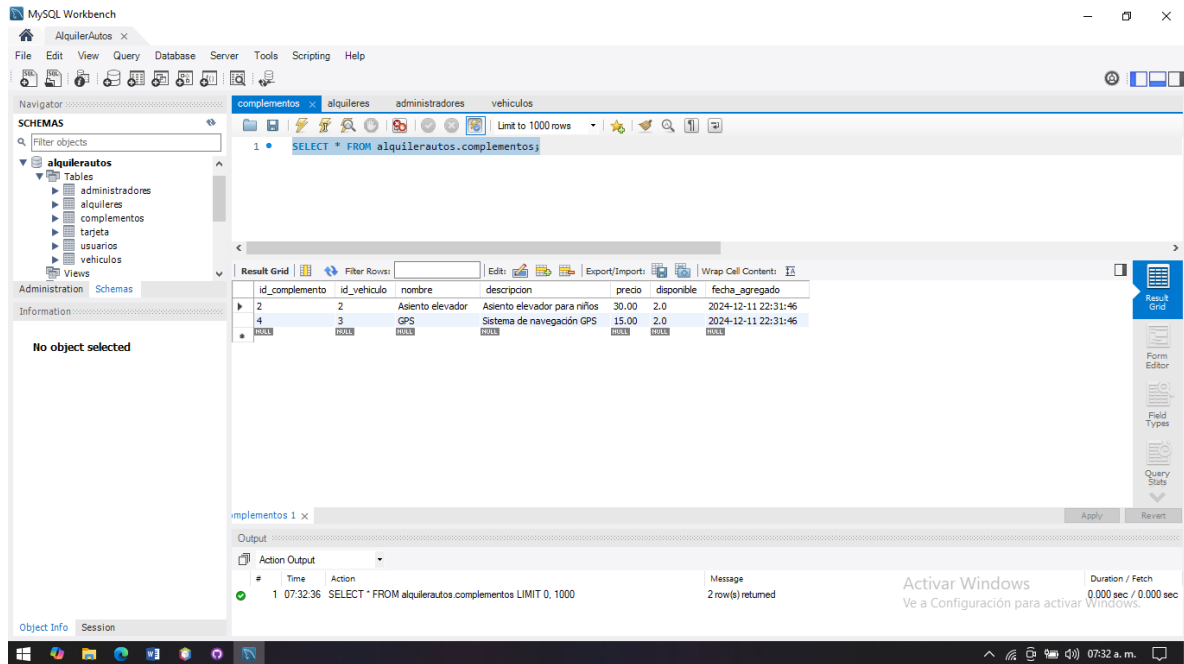
    public Connection conectar() {
        try {
            Class.forName(driver);
            cx = DriverManager.getConnection(url+bd, user, password);
            System.out.println("Se conecto a bd "+bd);
        } catch (ClassNotFoundException | SQLException ex) {
            System.out.println("No se conecto a bd "+bd);
            Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, null, ex);
        }
        return cx;
    }

    public void desconectar() {
        try {
            cx.close();
        } catch (SQLException ex) {
            Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

Me pide contraseña de mi base de datos, mi usuario que por defecto es root, el puerto y la ruta de acceso.

Anexo capturas de mi base de datos elaborada en Workbench



The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'Schemas' tree with the 'alquilerautos' database selected. The main editor window shows a SQL query: `SELECT * FROM alquilerautos.complementos;`. The 'Result Grid' tab is active, showing the following data:

	id_complemento	id_vehiculo	nombre	descripcion	precio	disponible	fecha_agregado
2	2		Asiento elevador	Asiento elevador para niños	30.00	2.0	2024-12-11 22:31:46
4	3		GPS	Sistema de navegación GPS	15.00	2.0	2024-12-11 22:31:46
5							

The bottom status bar shows the query execution details: `SELECT * FROM alquilerautos.complementos LIMIT 0, 1000` returned 2 rows in 0.000 seconds.