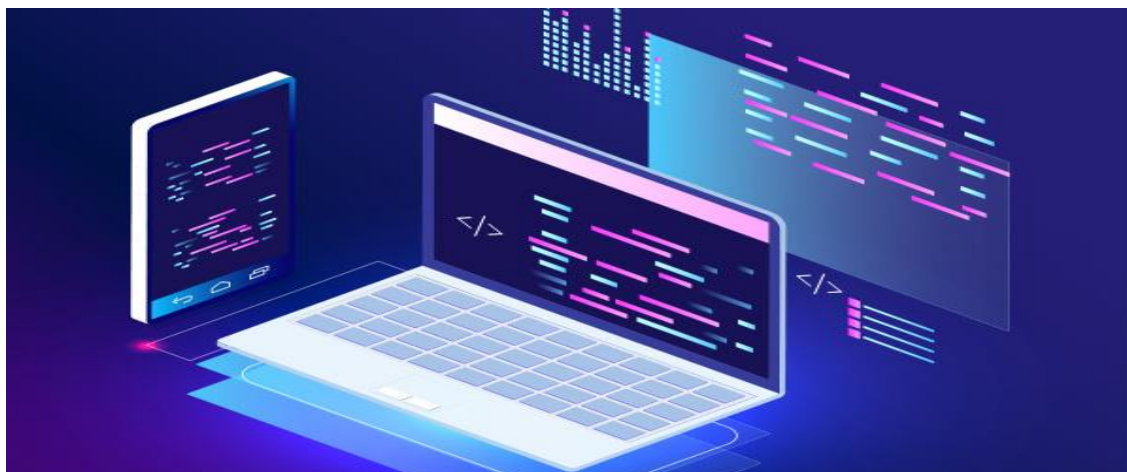




Polytechnic University of Puerto Rico
Electrical & Computer Engineering and
Computer Science Department
CECS3210 – Advanced Programming



PROYECTO #1

19 de mayo de 2023
José Rosado Gaitán #126996
Daniel A. Ovalle Cabral #103841
Alexander A. Yodice #122268



INDICE

1. Abstract del proyecto.....	4
2. Manual del usuario con fotos de la pantalla.....	5
3. Código fuente y explicación.....	5-23
4. Base de datos y almacenamiento de información.....	23

Abstracto del proyecto

El programa “Company Reception” es una aplicación de Java avanzada destinada a gestionar y registrar las interacciones de los visitantes en un entorno de empresa. Este software funciona como recepcionista automatizado, capaz de mantener registros completos de los visitantes y sus respectivas visitas.

El software incorpora varios conceptos de programación avanzada que incluyen encapsulación de datos, instanciación de clases, herencia, conceptos de interfaz gráfica de usuario (GUI), clases internas y administración de bases de datos.

En el ámbito de las funcionalidades, el software puede agregar y modificar los detalles de los visitantes, mostrar el historial de visitas de visitantes individuales y generar un informe para las visitas del día actual. También mantiene y manipula atributos como el nombre del visitante, la identificación, la empresa, la foto, el personal visitado, el número de la oficina y las marcas de tiempo para el check-in y el check-out.

Una característica única de este software es la capacidad de asociar y mostrar la foto del visitante en las funcionalidades Agregar, Modificar y Mostrar historial de visitantes, lo que mejora significativamente la recuperación de datos y la verificación de identidad. Todos los datos de los visitantes, incluidas las imágenes, se almacenan de forma segura en una base de datos y en un directorio designado, respectivamente.

Manual del Usuario con screenshots (descripción de cada screenshot).

//Main.java

```
public class Main { //The main method of the application.
    public static void main(String[] args) {
        // Create a new instance of the VisitorFrame class,
        which represents the main window of the application.
        new VisitorFrame();
    }
}
```

//Visitor.java

```
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
/**
 * The Visitor class represents a visitor to the office.
 * It includes information such as their name, the company they
 * belong to, the staff they are visiting,
 * and the time they checked in and out.
 */
public class Visitor {
    // Information about the visitor
    private String firstName;
    private String lastName;
    private String company;
    private String visitorsID;
    private String officeNo;
    private String staffVisiting;
    private Timestamp timeIn;
    private Timestamp timeOut;
    private String photo;
    /**
     * Constructor for a Visitor with only an ID. Other
     information is not known at the time of creation.
     */
}
```

```
*/

    public Visitor(String visitorsID, String firstName, String
lastName, String company,
                    String officeNo, String staffVisiting,
Timestamp timeIn, Timestamp timeOut, String photo) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.company = company;
    this.visitorsID = visitorsID;
    this.officeNo = officeNo;
    this.staffVisiting = staffVisiting;
    this.timeIn = timeIn;
    this.timeOut = timeOut;
    this.photo = photo;
}
/**
 * Constructor for a Visitor with full information provided
at the time of creation.

 * visitorsID The ID assigned to the visitor
 * firstName The visitor's first name
 * lastName The visitor's last name
 * company The company name from the visitor belongs to
 * officeNo The office number the visitor is visiting
 * staffVisiting The staff member the visitor is visiting
 * timeIn The check-in time of the visitor
 * timeOut The check-out time of the visitor
 * photo is the photo url of the visitors photo
 */

// Setters for the Visitor's information
public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public void setCompany(String company) {
    this.company = company;
}

public void setVisitorsID(String visitorsID) {
    this.visitorsID = visitorsID;
}
```

```
}

public void setOfficeNo(String officeNo) {
    this.officeNo = officeNo;
}

public void setStaffVisiting(String staffVisiting) {
    this.staffVisiting = staffVisiting;
}

public void setTimeIn(Timestamp timeIn) {
    this.timeIn = timeIn;
}

public void setTimeOut(Timestamp timeOut) {
    this.timeOut = timeOut;
}

public void setPhoto(String photo) {
    this.photo = photo;
}

// Getters for the Visitor's information
public String getFirstName() {
    return firstName;
}

public String getLastName() {
    return lastName;
}

public String getCompany() {
    return company;
}

public String getVisitorsID() {
    return visitorsID;
}

public String getOfficeNo() {
    return officeNo;
}

public String getStaffVisiting() {
    return staffVisiting;
}
```

```

    public String getTimeIn() {
        SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
        return dateFormat.format(this.timeIn);
    }

    public String getTimeOut() {
        SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
        return dateFormat.format(this.timeOut);
    }

    public String getPhoto() {
        return photo;
    }

    /**
     * Returns a string representation of the Visitor object,
     which includes all the visitor's information.
     *
     * @return A string representation of the Visitor object
     */

    @Override
    public String toString() {
        return "firstName='" + firstName + '\n' +
            ", lastName='" + lastName + '\n' +
            ", company='" + company + '\n' +
            ", visitorsID='" + visitorsID + '\n' +
            ", officeNo='" + officeNo + '\n' +
            ", staffVisiting='" + staffVisiting + '\n' +
            ", timeIn=" + timeIn +
            ", timeOut=" + timeOut + '\n' +
            ", photo='" + photo +
            '}' + "\n\n" ;
    }
    //Implementation
}

```


//VisitorFrame.java

```
import java.awt.event.*;
import java.sql.Timestamp;
import java.util.List;
import javax.swing.*;

// Define all UI components and visitor array
public class VisitorFrame extends JFrame implements
ActionListener {

    JMenuBar menuBar;
    JMenu fileMenu;
    JMenu helpMenu;

    JMenuItem addItem;
    JMenuItem modifyItem;
    JMenuItem historyVItem;
    JMenuItem todayReportItem;
    JMenuItem exitItem;
    JMenuItem helpItem;
    JMenuItem aboutItem;
    private JPanel panel1;
    private JPanel visitorPanel;
    private JPanel modifyPanel;
    private JPanel todayPanel;
    private JPanel displayPanel;
    private JPanel mainPanel;
    private JTextField toText1;
    private JTextField vidText1;
    private JTextField idText2;
    private JButton addButton;
    private JButton modifyButton;
    private JButton showInfoButton;
    private JTextField fnText1;
    private JTextField lnText1;
    private JTextField cText1;
    private JTextField onText1;
    private JTextField svText1;
    private JTextField tiText1;
    private JTextField fnText2;
```

```

private JTextField lnText2;
private JTextField cText2;
private JTextField onText2;
private JTextField svText2;
private JTextField tiText2;
private JTextField toText2;
private JTextArea vtTextArea;
private JTextArea vhtextArea;
private JTextField vIDText2;
private JTextField phText1;
private JTextField phText2;
Visitor[] visitor; // Array to store visitor objects

VisitorFrame() {
    // JFrame configurations
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setSize(610, 400);
    this.setContentPane(panel1);
    JMenuBar menuBar = new JMenuBar();
    this.setJMenuBar(menuBar);
    this.setVisible(true);

    // Initialize menu items
    fileMenu = new JMenu("File");
    helpMenu = new JMenu("Help");

    // Add action listeners to menu items
    addItem = new JMenuItem("Add Visitor");
    modifyItem = new JMenuItem("Modify Visitor");
    historyVItem = new JMenuItem("Visitor History");
    todayReportItem = new JMenuItem("Today's Report");
    exitItem = new JMenuItem("Exit");
    helpItem = new JMenuItem("Help");
    aboutItem = new JMenuItem("About");

    addItem.addActionListener(this);
    modifyItem.addActionListener(this);
    historyVItem.addActionListener(this);
    todayReportItem.addActionListener(this);
    exitItem.addActionListener(this);
    helpItem.addActionListener(this);
    aboutItem.addActionListener(this);

    // Add menu items to menus and menus to menu bar
    fileMenu.add(addItem);
    fileMenu.add(modifyItem);

```

```

fileMenu.add(historyVItem);
fileMenu.add(todayReportItem);
fileMenu.add(exitItem);
helpMenu.add(helpItem);
helpMenu.add(aboutItem);
menuBar.add(fileMenu);
menuBar.add(helpMenu);

// Set initial panel visibility
this.panell1.setVisible(true);
visitorPanel.setVisible(false);
todayPanel.setVisible(false);
displayPanel.setVisible(false);
modifyPanel.setVisible(false);
mainPanel.setVisible(true);

visitor = new Visitor[20]; // Initialize visitor array

// Add action listeners to other UI components
// These action listeners are responsible for database
operations
// (adding and modifying visitor information)
// and displaying visitor information on the screen

addButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        VisitorQueries addVisitor = new
VisitorQueries();
        String fName = fnText1.getText();
        String lName = lnText1.getText();
        String company = cText1.getText();
        String ofNum = onText1.getText();
        String sVisiting = svText1.getText();
        Timestamp tIn =
Timestamp.valueOf(tiText1.getText());
        Timestamp tOut =
Timestamp.valueOf(toText1.getText());
        String vID = vidText1.getText();
        String photo = phText1.getText();

        int result = addVisitor.addVisitor(fName, lName,
company, ofNum, sVisiting, tIn, tOut, vID, photo);
        if(result == 1){
            JOptionPane.showMessageDialog(null, "Record
Succesfully Inserted");

```

```

        } else{
            JOptionPane.showMessageDialog(null, "Record
Unsuccesfully Inserted");
        }
    }
});
showInfoButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        VisitorQueries visitorQueries = new
VisitorQueries();
        int id = Integer.parseInt(idText2.getText());

        List<Visitor> list =
visitorQueries.getVisitorByID(id);
        Visitor visitor = (Visitor) list.get(0);
        fnText2.setText(visitor.getFirstName());
        lnText2.setText(visitor.getLastName());
        cText2.setText(visitor.getCompany());
        onText2.setText(visitor.getOfficeNo());
        svText2.setText(visitor.getStaffVisiting());
        tiText2.setText(visitor.getTimeIn());
        toText2.setText(visitor.getTimeOut());
        vIDText2.setText(visitor.getVisitorsID());
        phText2.setText(visitor.getPhoto());
    }
});
idText2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        VisitorQueries visitorQueries = new
VisitorQueries();
        int id = Integer.parseInt(idText2.getText());

        List<Visitor> list =
visitorQueries.getVisitorByID(id);
        Visitor visitor = (Visitor) list.get(0);
        fnText2.setText(visitor.getFirstName());
        lnText2.setText(visitor.getLastName());
        cText2.setText(visitor.getCompany());
        onText2.setText(visitor.getOfficeNo());
        svText2.setText(visitor.getStaffVisiting());
        tiText2.setText(visitor.getTimeIn());
        toText2.setText(visitor.getTimeOut());
        vIDText2.setText(visitor.getVisitorsID());
        phText2.setText(visitor.getPhoto());
    }
});

```

```

    }
    });
    modifyButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            VisitorQueries updateVisitor = new
VisitorQueries();

            String fName = fnText2.getText();
            String lName = lnText2.getText();
            String company = cText2.getText();
            String ofNum = onText2.getText();
            String sVisiting = svText2.getText();
            Timestamp tIn =
Timestamp.valueOf(tiText2.getText());
            Timestamp tOut =
Timestamp.valueOf(toText2.getText());
            String vID = vIDText2.getText();
            String photo = phText2.getText();
            int id = Integer.parseInt(idText2.getText());

            int result = updateVisitor.updateVisitor(fName,
lName, company, ofNum, sVisiting, tIn, tOut, vID, photo, id);
            if(result == 1){
                JOptionPane.showMessageDialog(null, "Record
Successfully Updated");
            } else{
                JOptionPane.showMessageDialog(null, "Record
Unsuccessfully Updated");
            }
        }
    });
}

@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == addItem) {
        visitorPanel.setVisible(true);
        todayPanel.setVisible(false);
        displayPanel.setVisible(false);
        modifyPanel.setVisible(false);
        mainPanel.setVisible(false);
    }
    if(e.getSource() == modifyItem) {
        visitorPanel.setVisible(false);
        todayPanel.setVisible(false);

```

```

        displayPanel.setVisible(false);
        modifyPanel.setVisible(true);
        mainPanel.setVisible(false);

    }
    if(e.getSource()==historyVItem){

        VisitorQueries visitorQueries = new
VisitorQueries();
        String list =
String.valueOf(visitorQueries.getAllVisitors());
        visitorPanel.setVisible(false);
        todayPanel.setVisible(false);
        displayPanel.setVisible(true);
        modifyPanel.setVisible(false);
        mainPanel.setVisible(false);
        vhtextArea.setText(list);
    }
    if(e.getSource()==todayReportItem){
        VisitorQueries visitorQueries = new
VisitorQueries();
        String list = visitorQueries.getVisitorByDate();
        visitorPanel.setVisible(false);
        todayPanel.setVisible(true);
        displayPanel.setVisible(false);
        modifyPanel.setVisible(false);
        mainPanel.setVisible(false);
        vtTextArea.setText(list);

    }
    if(e.getSource()==exitItem){
        this.dispose();
    }
    if(e.getSource()==helpItem){
        JOptionPane.showMessageDialog(null, "Use the menu on
the top of the frame to manage the visitor data");
    }
    if(e.getSource()==aboutItem){
        JOptionPane.showMessageDialog(null, "Developers:
Jose Rosado / Daniel Ovalle / Alexander Yodice");
    }
}
}

```

//VisitorQueries.java

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * This class represents a database interface for operations
 * related to the Visitor objects.
 * It includes methods for adding, updating, and fetching
 * Visitor data from the database.
 */
public class VisitorQueries {
    private static final String URL =
"jdbc:mysql://localhost:3306/company";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "MyN3wP4ssw0rd";

    private Connection connection = null;
    private PreparedStatement selectAllVisitors = null;
    private PreparedStatement selectVisitorByID = null;
    private PreparedStatement insertNewVisitor = null;
    private PreparedStatement updateVisitor = null;
    private PreparedStatement selectVisitorByDate = null;

    /**
     * Default constructor that sets up the database connection
     and prepares SQL statements.
     */
    public VisitorQueries() {
        try {
            connection = DriverManager.getConnection(URL,
USERNAME, PASSWORD);

            selectAllVisitors =
connection.prepareStatement("SELECT * FROM company.visitors");
            selectVisitorByID =
connection.prepareStatement("SELECT * FROM company.visitors
WHERE id = ?");
            insertNewVisitor =
connection.prepareStatement("INSERT INTO company.visitors
(FirstName, LastName, Company_from, Office_visiting, " +
"Staff_visiting, date_begin, date_end,
Visitor_ID, Photo) VALUES(?,?,?,?,?,?,?,?,?)");
            updateVisitor = connection.prepareStatement("UPDATE
company.visitors SET FirstName = ?, " +
```

```

        "LastName = ?, Company_from = ?,
Office_visiting = ?, Staff_visiting = ?, date_begin = " +
        "? , date_end = ?, Visitor_ID = ?, Photo = ?
WHERE id = ?");
        selectVisitorByDate =
connection.prepareStatement("SELECT * FROM Visitors where
date(date_begin) = date(CURRENT_TIMESTAMP)");

        } catch (SQLException ex) {
            System.err.println("Connection Error with Database"
+ ex.getMessage());
            System.exit(1);
            close();
        }
    }

    /**
     * Fetches all visitors from the database.
     * A list of all visitors from the database.
     */
    public StringBuilder getAllVisitors() {
        StringBuilder text = new StringBuilder(" ");
        List<Visitor> results;
        ResultSet resultSet = null;

        try {
            resultSet = selectAllVisitors.executeQuery();
            results = new ArrayList<>();

            while (resultSet.next()) {
                results.add(new
Visitor(resultSet.getString("id"),
resultSet.getString("FirstName"),
                resultSet.getString("LastName"),
resultSet.getString("company_from"),
                resultSet.getString("Office_visiting"),
resultSet.getString("Staff_Visiting"),
                resultSet.getTimestamp("date_begin"),
resultSet.getTimestamp("date_end"),
                resultSet.getString("Photo")));
            }
            text.append(results.toString());
        } catch (SQLException ex) {
            System.err.println("Error in DB: " +
ex.getMessage());
            close();

```



```

    } finally {
        try {
            resultSet.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
            close();
        }
    }
    return text;
}

/**
 * Fetches a visitor by ID from the database.
 *
 * id The ID of the visitor to fetch.
 * A list containing the visitor with the provided ID, or
 * an empty list if no visitor was found.
 */
public List<Visitor> getVisitorByID(int id) {
    List<Visitor> results = null;
    ResultSet resultSet = null;

    try {
        selectVisitorByID.setInt(1, id);
        resultSet = selectVisitorByID.executeQuery();
        results = new ArrayList<>();

        while (resultSet.next()) {
            results.add(new
Visitor(resultSet.getString("Visitor_ID"),
resultSet.getString("FirstName"),
resultSet.getString("LastName"),
resultSet.getString("Company_from"),
resultSet.getString("Office_Visiting"),
resultSet.getString("Staff_visiting"),
resultSet.getTimestamp("date_begin"),
resultSet.getTimestamp("date_end"),
resultSet.getString("Photo")));
        }
    } catch (SQLException ex) {
        System.err.println("Error in DB: " +
ex.getMessage());
        close();
    } finally {
        try {
            resultSet.close();
        } catch (SQLException ex) {

```

```

        ex.printStackTrace();
        close();
    }
}
return results;
}

/**
 * Adds a new visitor to the database.
 *
 * fName The first name of the new visitor.
 * lName The last name of the new visitor.
 * company The company of the new visitor.
 * oNum The office number of the new visitor.
 * Visit The staff visiting of the new visitor.
 * tIn The time in of the new visitor.
 * tOut The time out of the new visitor.
 * photo The URL from the photo taken.
 * The number of rows affected in the database, or 0 if the
operation failed.
 */
public int addVisitor(String fName, String lName, String
company,
                    String oNum, String sVisit, Timestamp
tIn,
                    Timestamp tOut, String ID, String
photo) {
    int result = 0;

    try {
        insertNewVisitor.setString(1, fName);
        insertNewVisitor.setString(2, lName);
        insertNewVisitor.setString(3, company);
        insertNewVisitor.setString(4, oNum);
        insertNewVisitor.setString(5, sVisit);
        insertNewVisitor.setTimestamp(6, tIn);
        insertNewVisitor.setTimestamp(7, tOut);
        insertNewVisitor.setString(8, ID);
        insertNewVisitor.setString(9, photo);

        result = insertNewVisitor.executeUpdate();
    } catch (SQLException ex) {
        System.err.println("Error in DB: " +
ex.getMessage());
        close();
    }
}

```

```

    }
    return result;
}

/**
 * This method updates an existing visitor in the database.
The arguments include:
 * ID - The ID of the visitor to update.
 * fName - The new first name of the visitor.
 * lName - The new last name of the visitor.
 * company - The new company of the visitor.
 * oNum - The new office number of the visitor.
 * sVisit - The new staff the visitor is visiting.
 * tIn - The new timestamp representing the time the visitor
checked in.
 * tOut - The new timestamp representing the time the
visitor checked out.
 * photo - The new URL from the photo taken.
 * The method returns the number of rows affected in the
database, or 0 if the operation failed.
 */
public int updateVisitor(String fName, String lName, String
company,
                        String oNum, String sVisit,
Timestamp tIn,
                        Timestamp tOut, String vID, String
photo, int id) {
    int result = 0;

    try {
        updateVisitor.setString(1, fName);
        updateVisitor.setString(2, lName);
        updateVisitor.setString(3, company);
        updateVisitor.setString(4, oNum);
        updateVisitor.setString(5, sVisit);
        updateVisitor.setTimestamp(6, tIn);
        updateVisitor.setTimestamp(7, tOut);
        updateVisitor.setString(8, vID);
        updateVisitor.setString(9, photo );
        updateVisitor.setInt(10, id);

        result = updateVisitor.executeUpdate();
    } catch (SQLException ex) {
        System.err.println("Error in DB: " +
ex.getMessage());
    }
}

```

```

        close();
    }
    return result;

}

public String getVisitorByDate() {
    StringBuilder text = new StringBuilder(" ");
    List<Visitor> results;
    ResultSet resultSet = null;

    try {
        resultSet = selectVisitorByDate.executeQuery();
        results = new ArrayList<>();

        while (resultSet.next()) {
            results.add(new
Visitor(resultSet.getString("id"),
resultSet.getString("FirstName"),
                resultSet.getString("LastName"),
resultSet.getString("company_from"),
                resultSet.getString("Office_visiting"),
resultSet.getString("Staff_Visiting"),
                resultSet.getTimestamp("date_begin"),
resultSet.getTimestamp("date_end"),
                resultSet.getString("Photo")));
            text.append(results.toString());
        }
    } catch (SQLException ex) {
        System.err.println("Error in DB: " +
ex.getMessage());
        close();
    } finally {
        try {
            resultSet.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    return text.toString();
}

/**
 * This method is used to close the database connection.
 */
public void close(){
    try {
        connection.close();
    }

```

```
    } catch (SQLException ex) {  
        System.err.println("Error in DB: " +  
ex.getMessage());  
    }  
}  
}
```

CODIGO MYSQL DATABASE

//SQLVisitor.sql

drop database IF EXISTS company;

CREATE DATABASE company;

USE company;

CREATE TABLE Visitors

(

id int primary key auto_increment,

FirstName varchar(30) NOT NULL,

LastName varchar(30) NOT NULL,

Company_from varchar(15),

Visitor_ID varchar(10) NOT NULL,

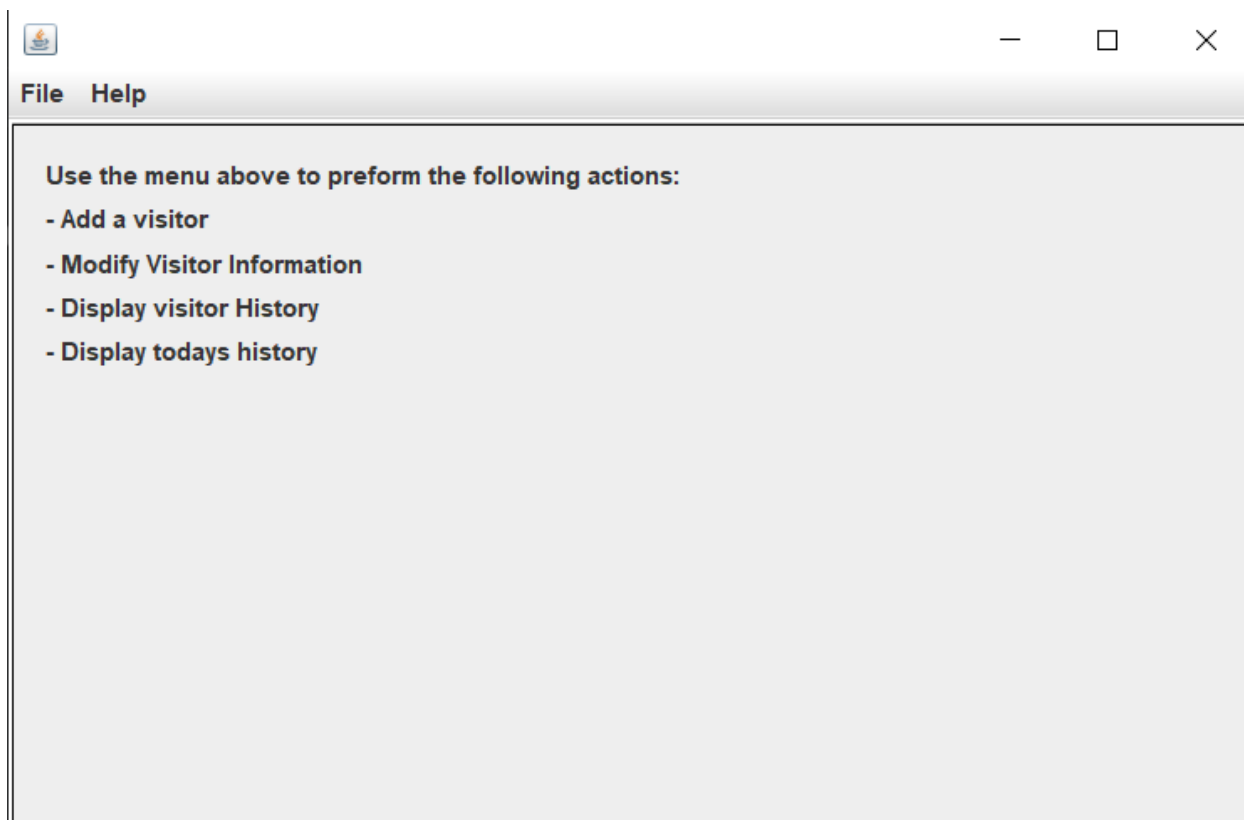


```
Photo varchar(30),
Staff_visiting varchar(60),
Office_visiting varchar(5),
date_begin timestamp NOT NULL,
date_end timestamp NULL
);

INSERT INTO Visitors
(FirstName,LastName,Company_from,Visitor_ID,Photo,Staff_visiting,Office_visiting,date
_begin,date_end)
VALUES ('Carlitos', 'Colon', 'IBM', '180578', 'js342a.jpg', 'Anibal Ramirez', 'L305',
'2018-9-26 09:12:23', '2018-9-26 10:55:36');

SELECT * FROM Visitors where date(date_begin) = date(CURRENT_TIMESTAMP);
```

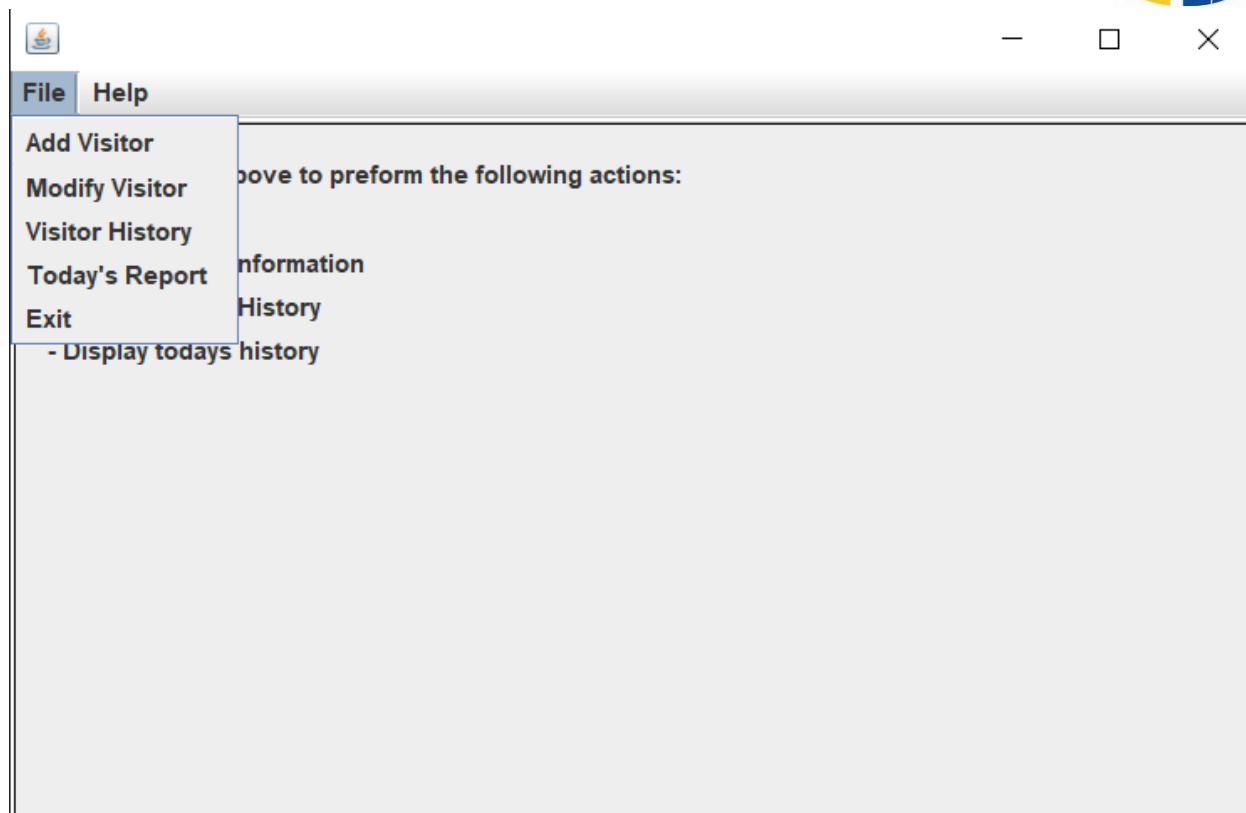
OUTPUTS Y SUS EXPLICACIONES



Este proyecto en Java presenta un menú diseñado para gestionar el registro de visitantes que entran a una empresa. El menú ofrece cuatro opciones a las que se puede acceder:

- 1. "Añadir un visitante" - Esta opción permite introducir y guardar los datos de un nuevo visitante en la base de datos. Esto incluye detalles como el nombre del visitante, la compañía de la cual procede, el ID del visitante, la foto, el personal al que visita, la oficina que visita, y la fecha y hora de inicio y fin de la visita.**
- 2. "Modificar la información de un visitante" - Esta opción permite actualizar la información de un visitante existente. Puede ser útil en caso de que haya algún error en los detalles del visitante o si el visitante cambia alguna de su información, como la empresa de la que procede.**
- 3. "Mostrar el registro de todos los visitantes" - Esta opción permite visualizar la lista completa de visitantes que se han registrado en la base de datos. Los visitantes se muestran con todos sus detalles.**
- 4. "Mostrar todos los visitantes registrados en el día de hoy" - Esta opción permite visualizar todos los visitantes que han ingresado a la empresa en la fecha actual.**

El propósito de este sistema es ayudar a la empresa a llevar un seguimiento detallado de los visitantes por razones de seguridad y administrativas.



Aquí, estamos creando un menú con las siguientes opciones:

File > Exit (Archivo > Salir): Esta opción cierra la aplicación. El programa escucha el evento de acción en este elemento del menú y, cuando se selecciona, ejecuta el código correspondiente para cerrar la aplicación.

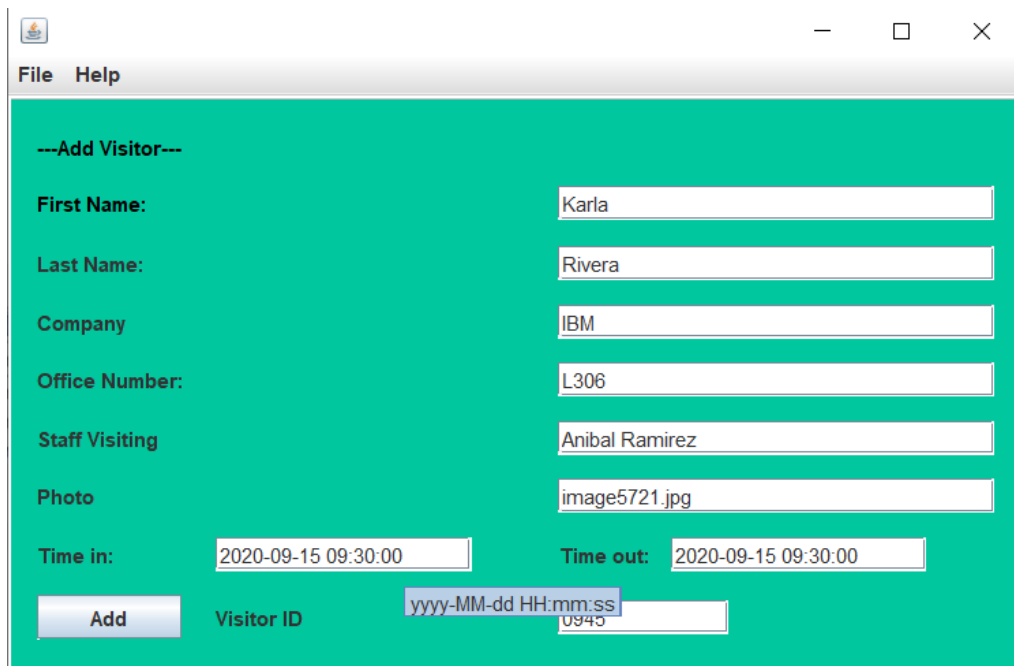
Visitor > Add Visitor (Visitante > Añadir Visitante): Esta opción abre un formulario o un diálogo para ingresar los detalles de un nuevo visitante. Cuando se completa este formulario, los detalles se envían a la base de datos para almacenarlos.

Visitor > Edit Visitor (Visitante > Editar Visitante): Esta opción permite buscar un visitante existente (generalmente a través de su ID o nombre) y editar sus detalles. Una vez editados, los cambios se guardan en la base de datos.



Visitor > Show All Visitors (Visitante > Mostrar Todos los Visitantes): Esta opción recupera todos los visitantes de la base de datos y los muestra en una lista o tabla en la interfaz de usuario.

Visitor > Show Today's Visitors (Visitante > Mostrar los Visitantes de Hoy): Esta opción recupera todos los visitantes que han visitado la empresa en la fecha actual y los muestra en la interfaz de usuario.



File Help

---Add Visitor---

First Name: Karla

Last Name: Rivera

Company: IBM

Office Number: L306

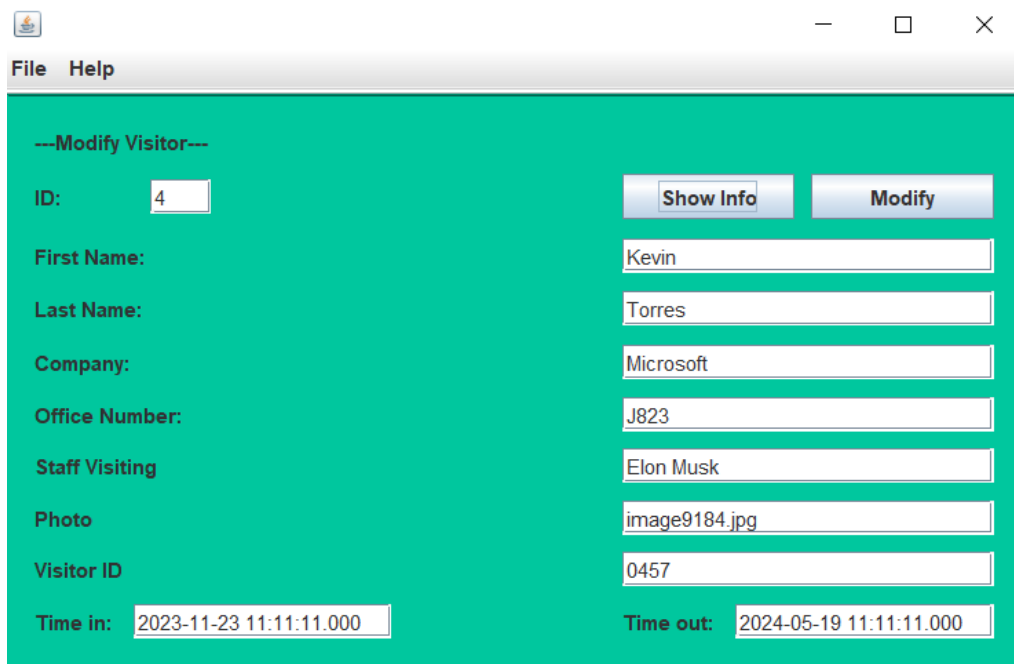
Staff Visiting: Anibal Ramirez

Photo: image5721.jpg

Time in: 2020-09-15 09:30:00 Time out: 2020-09-15 09:30:00

Add Visitor ID: yyyy-MM-dd HH:mm:ss 0945

Si seleccionamos la opción de “Add Visitor” nos presenta este output para que el visitante llene esta información. Cuando termine de añadir toda esta información, se le da click al botón de “Add” y esto almacena la información en nuestra base de datos de SQL.



File Help

---Modify Visitor---

ID: 4 Show Info Modify

First Name: Kevin

Last Name: Torres

Company: Microsoft

Office Number: J823

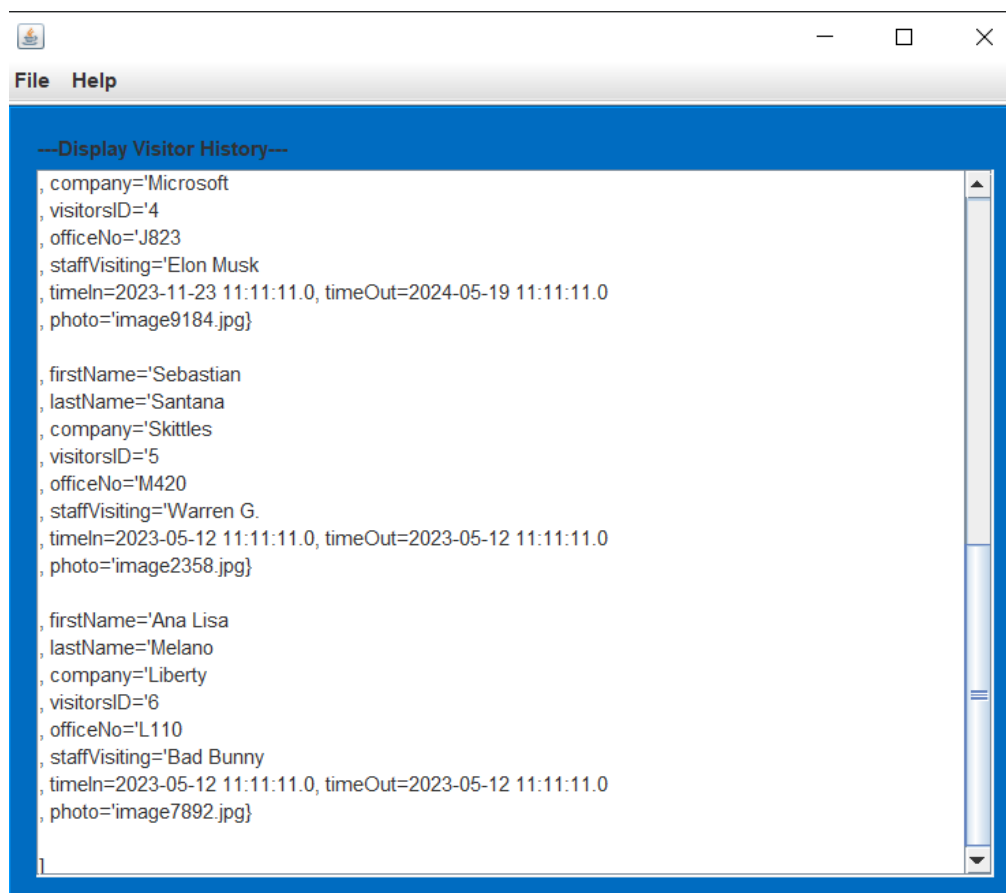
Staff Visiting: Elon Musk

Photo: image9184.jpg

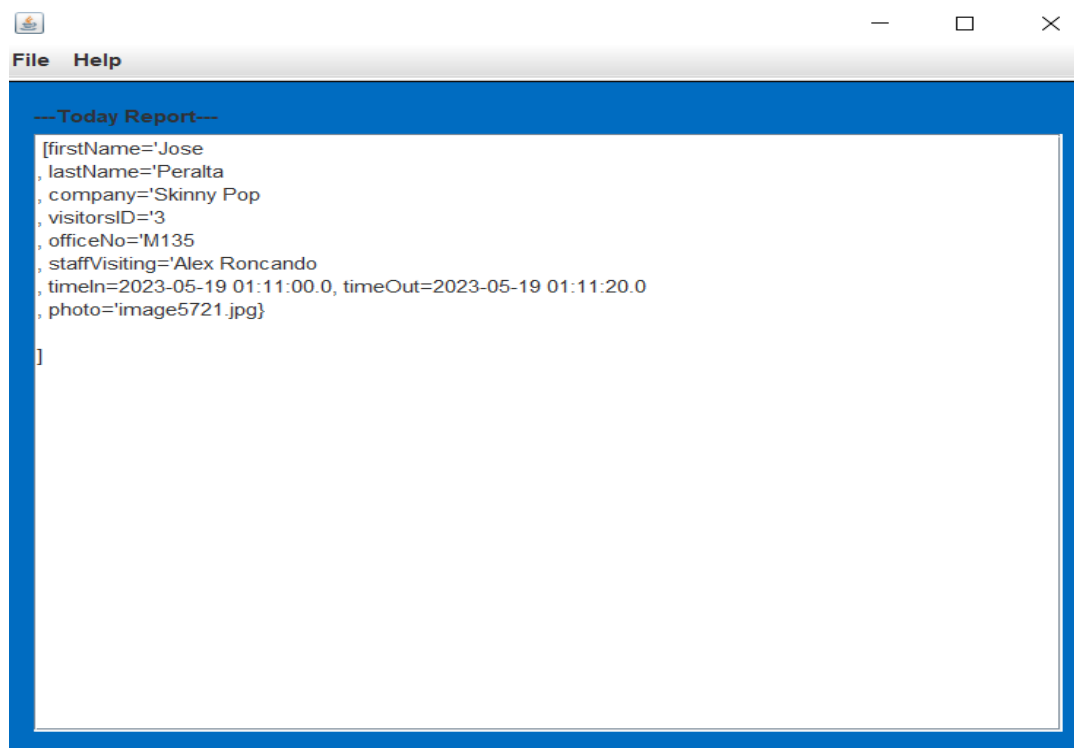
Visitor ID: 0457

Time in: 2023-11-23 11:11:11.000 Time out: 2024-05-19 11:11:11.000

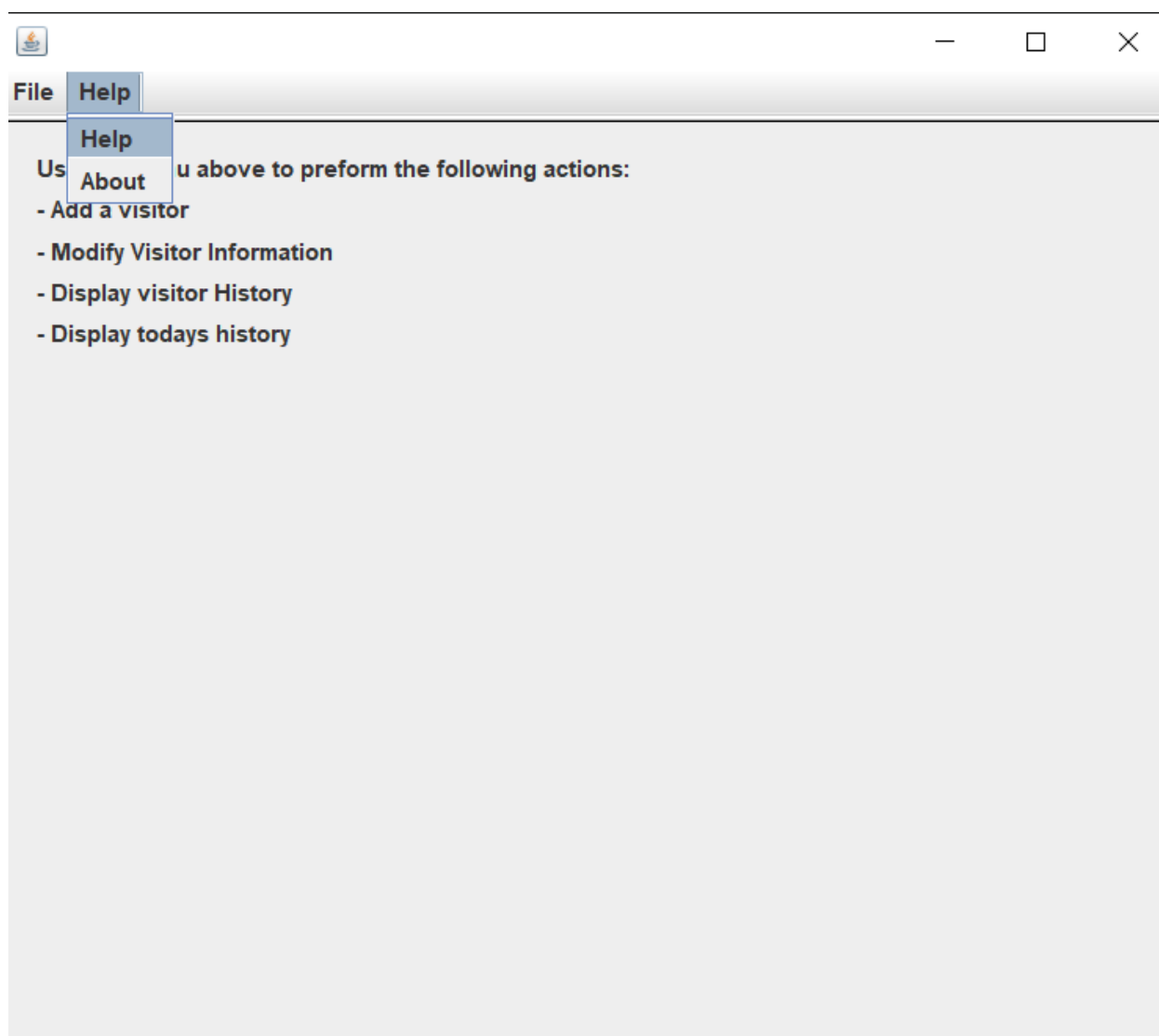
En este output te permite que el usuario modifique un registro de visita poniendo el numero de ID del mismo.



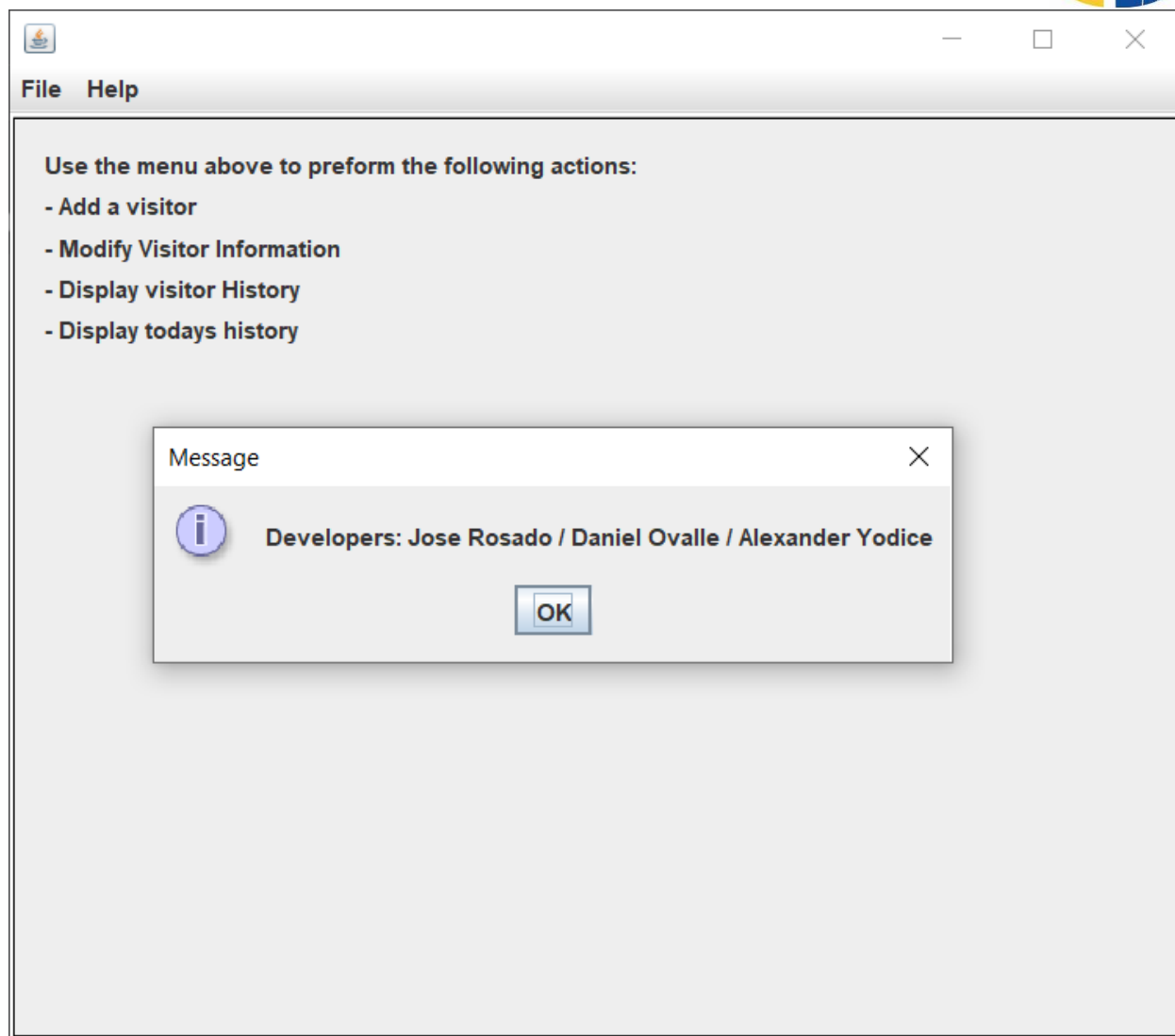
En este output se muestra lo que sucede cuando el usuario presiona el botón de “Visitor History”. Esto muestra todo el registro ingresado de los visitantes desde que fue insertada el primer visitante.



En este output es similar al anterior excepto que este es el botón de “Today’s Report”. Esto básicamente muestra todos los registros insertados en ese día (hoy).



En este output se presenta que hacen las opciones de help y about. Básicamente la opción de about menciona los creadores del programa, aquí un ejemplo de lo que pasa cuando el usuario le da click al botón de about:



Aquí un ejemplo de lo que sucede cuando presionas el botón de about.

SQL visitor x

Limit to 1000 rows

1 • SELECT * FROM Company.Visitors

Result Grid

	id	FirstName	LastName	Company_from	Visitor_ID	Photo	Staff_visiting	Office_visiting	date_begin	date_end
▶	1	Karla	Rivera	IBM	0945	image5721.jpg	Anibal Ramirez	L306	2020-09-15 09:30:00	2020-09-15 09:30:00
	2	Carlitos	Colon	IBM	0261	image5721.jpg	Anibal Ramirez	L305	2020-09-15 09:30:00	2020-09-15 09:30:00
	3	Jose	Peralta	Skinny Pop	0567	image5721.jpg	Alex Roncando	M135	2023-05-19 01:11:00	2023-05-19 01:11:20
	4	Kevin	Torres	Microsoft	0457	image9184.jpg	Elon Musk	J823	2023-11-23 11:11:11	2024-05-19 11:11:11
	5	Sebastian	Santana	Skittles	0473	image2358.jpg	Warren G.	M420	2023-05-12 11:11:11	2023-05-12 11:11:11
	6	Ana Lisa	Melano	Liberty	2508	image7892.jpg	Bad Bunny	L110	2023-05-12 11:11:11	2023-05-12 11:11:11
*		NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Form Editor

Field Types

Finalmente, aquí tenemos una captura de pantalla de la base de datos en SQL. Esencialmente, cuando un visitante introduce sus datos a través de la interfaz de la aplicación en Java, estos datos se guardan y actualizan automáticamente en la base de datos. Este almacenamiento de datos se realiza de manera estructurada, en una tabla que organiza y presenta la información de manera clara y ordenada. Cada entrada en la tabla representa un visitante y muestra su nombre, la persona a la que va a visitar, la compañía a la que pertenece el visitante, una foto del visitante, la oficina a la que se dirigirá, y el horario de entrada y salida. De este modo, la base de datos permite un seguimiento efectivo de todos los visitantes que ingresan a la empresa.

Conclusión:

En resumen, el proyecto de “Company Reception” es una aplicación avanzada en Java que permite gestionar y registrar las interacciones de los visitantes en un entorno empresarial. La aplicación utiliza conceptos de programación avanzada como encapsulación de datos, instanciación de clases, herencia, conceptos de interfaz gráfica de usuario (GUI), clases internas y administración de bases de datos.

El programa nos ofrece funcionalidades como agregar y modificar detalles de los visitantes, mostrar el historial de visitas de visitantes individuales y generar un informe para las visitas del día actual. Además, se destaca por la capacidad de asociar y mostrar la foto del visitante, lo que mejora la recuperación de datos y la verificación de identidad. El manual del usuario proporciona instrucciones detalladas sobre cómo utilizar la aplicación, incluyendo capturas de pantalla que guían al usuario a través de cada función. El código fuente muestra la implementación de las clases principales, como Main, Visitor y VisitorFrame, así como la manipulación de la base de datos utilizando consultas SQL.

Por último, se presenta una captura de pantalla de la base de datos en SQL, donde se almacenan de manera segura todos los datos de los visitantes, incluyendo sus imágenes. En conjunto, este proyecto ofrece una solución efectiva para gestionar y registrar las interacciones de los visitantes en un entorno empresarial, mejorando la seguridad y la organización en el proceso de recepción.