UPPue

Universidad Politécnica de Puebla

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Asignatura
Fundamentos de Programación Orientada a Objetos

Cuatrimestre
Enero-Abril 2024

Título de la Actividad
Sistema para le Gestión de Datos Empresariales EMT System
Desarrollado en Java

Nombre Completo del/la Estudiante
Berenice Morales Bustamante

Grupo
5° ”A”

Docente
M.S.C. Tomás González Alvarado

Carrera:	Ingeniería en Tecnologías de la Información	**Cuatrimestre:**	Enero-Abril 2024	**Fecha:**	19/04/2024
Nombre del Estudiante:	Berenice Morales Bustamante			**Grupo:**	5A
Tarea [1]:	Proyecto Final	**Lenguaje de Programación:**	Java17	**IDE de Desarrollo:**	IntelliJ Idea

Describe la funcionalidad del algoritmo:

Diagrama de clases y programa en Java que permite la gestión de los datos de empleados de la empresa Itera México, incluyendo el registro y control de datos personales y empresariales de los empleados, así como sus respectivos contratos.

Redacción del Escenario

La empresa internacional mexicana Itera S.A. de C.V., especializada en negocios y proyectos desarrollados en la nube, tiene planes de expandirse en
los paises de Brasil, Colombia y Canadá.

Itera México, necesita tener el control de sus empleados, ya
que su enfoque se basa en la productividad, comunicación y capacitación de dichos empleados. Por lo anterior, ha decidido firmar un contrato por $33,500,000.00 para el desarrollo de un sistema de software para la gestión de sus empleados y contratos, considerando los cargos que tendrá cada uno, y así vincularlo, posteriormente a otro sistema para la gestión de capacitaciones especializadas.

El sistema en cuestión deberá funcionar de la siguiente
manera:

El sistema denominado EMT-SYSTEM, estopor el acrónimo
Employee Control System, deberá incluir de manera inicial con un control de datos personales de cada uno de los aspirantes a los puestos y una vez contratados les solicitarán un poco más información para considerarlos como empleados:

Datos personales: id, nombre, apellido, correo, WhastApp

Datos Empresariales: Adscripción, teléfono y Extensión y

finalmente puesto.

EMT-SYSTEM, deberá almacenar y controlar la información de múltiples empleados, ya que ellos estarán en un proceso de capacitación inicial para que conozcan la misión, visión, valores y objetivos que la empresa Itera México quiere alcanzar. En este sentido, los contratos son asignados dos meses después, entonces no es necesario asignarles un contrato al momento de registrar sus datos personales o empresariales.

Es importante considerar que en el interior del contrato solo existirán 3 tipos de cargos:

-de confianza
-Sindicalizado
-Contrato temporal

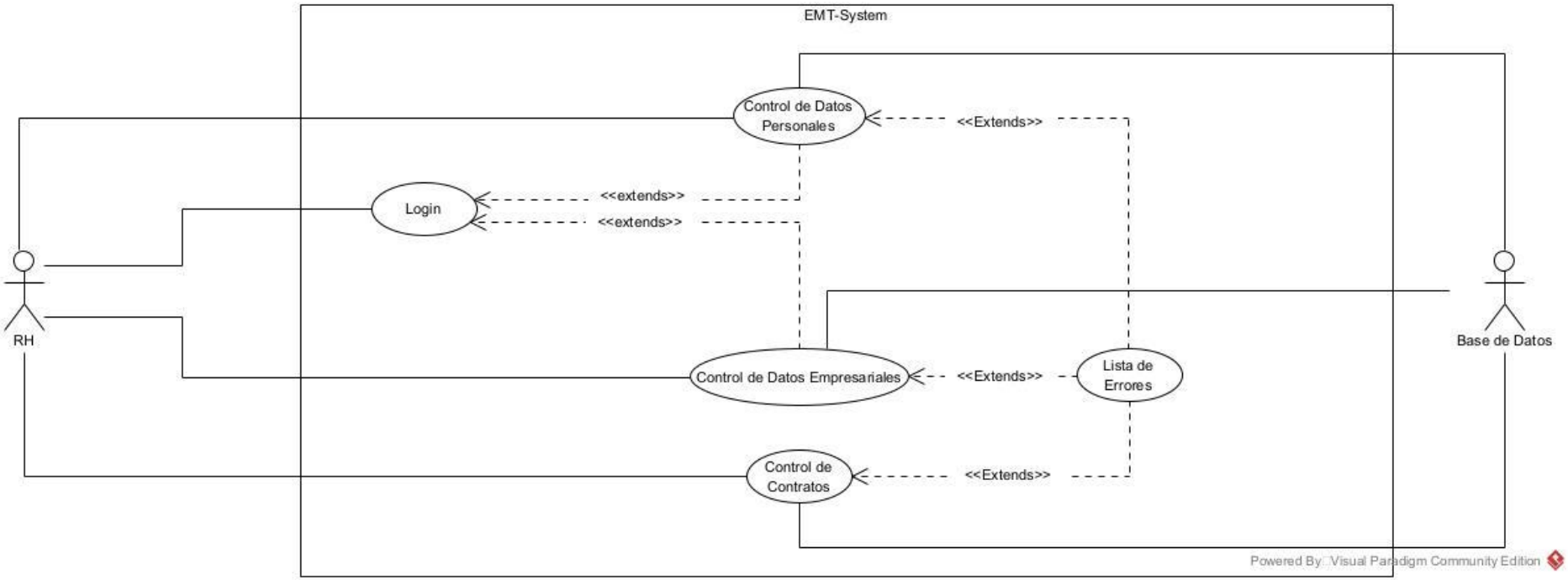
Lo anterior para ubicar el tipo de prestaciones que dispondrá cada empleado.

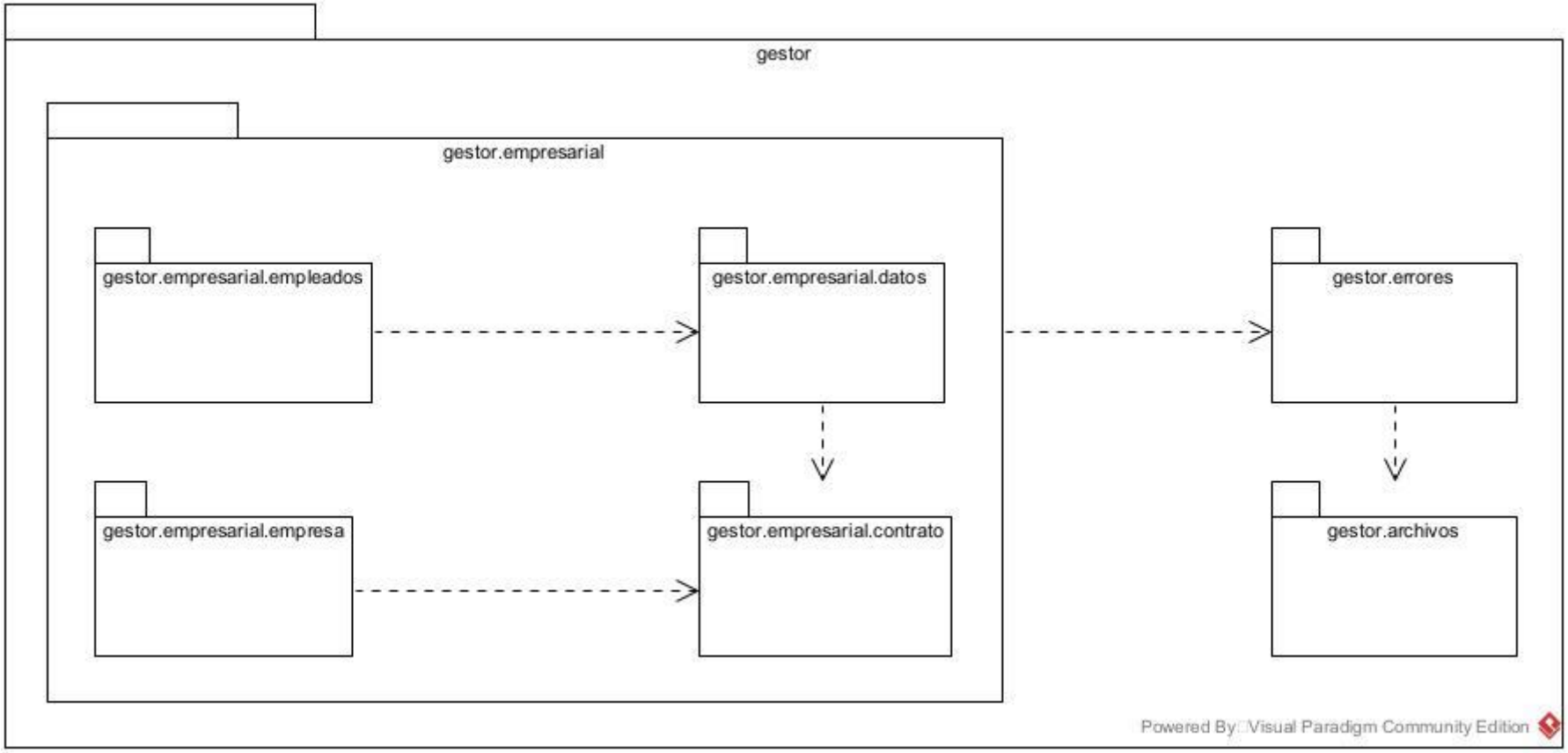
Importante: No será necesario controlar el tipo de prestaciones para cada empleado, basta con identificar el tipo de cargo que se le asignará a cada empleado.

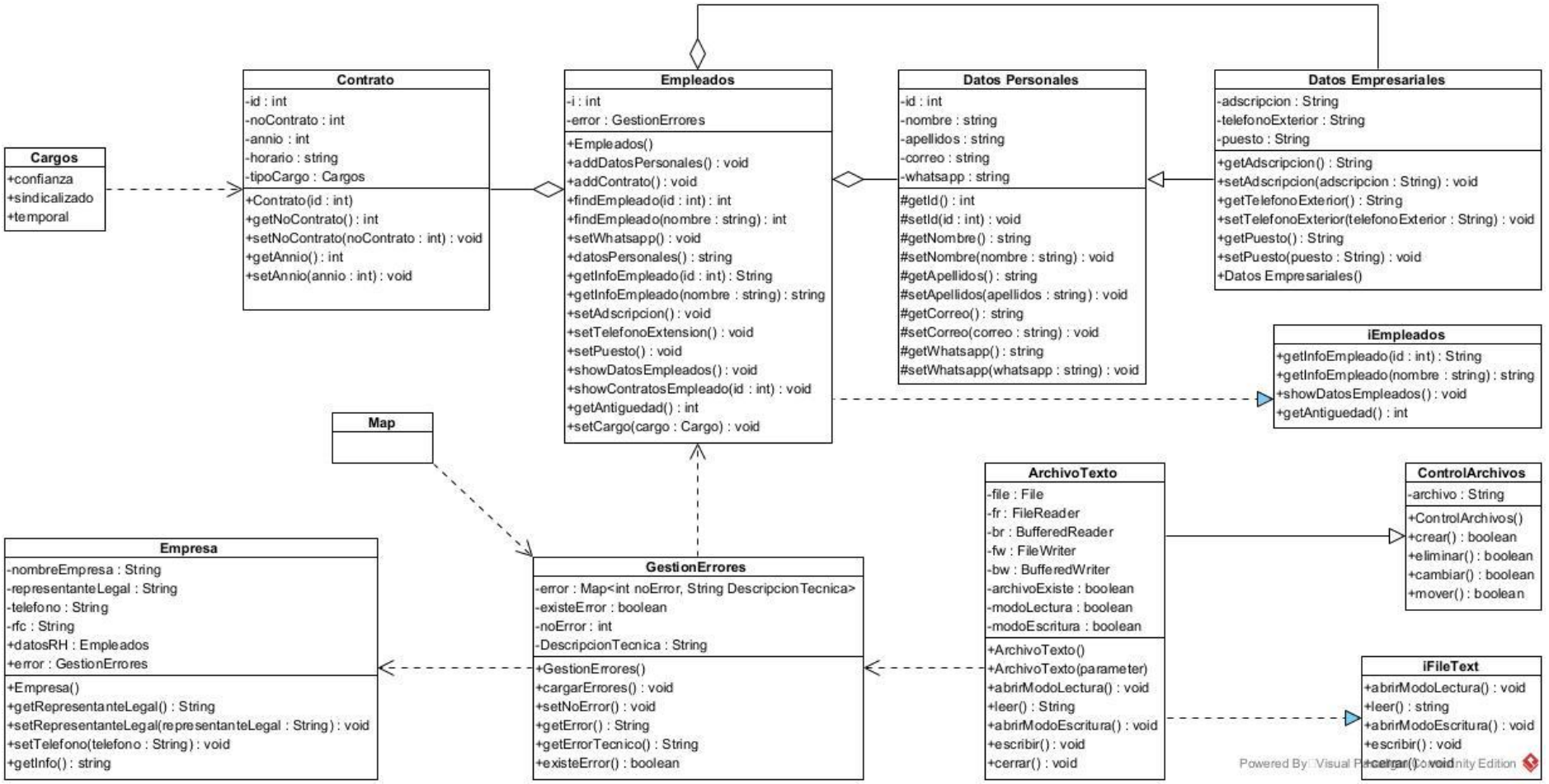
Por otro lado, EMT-SYSTEM deberá resguardar perfectamente los datos esenciales de la empresa, ya que en un futuro, se usará esta información para generar los contratos de manera automática. Los datos a resguardar serán los siguientes:

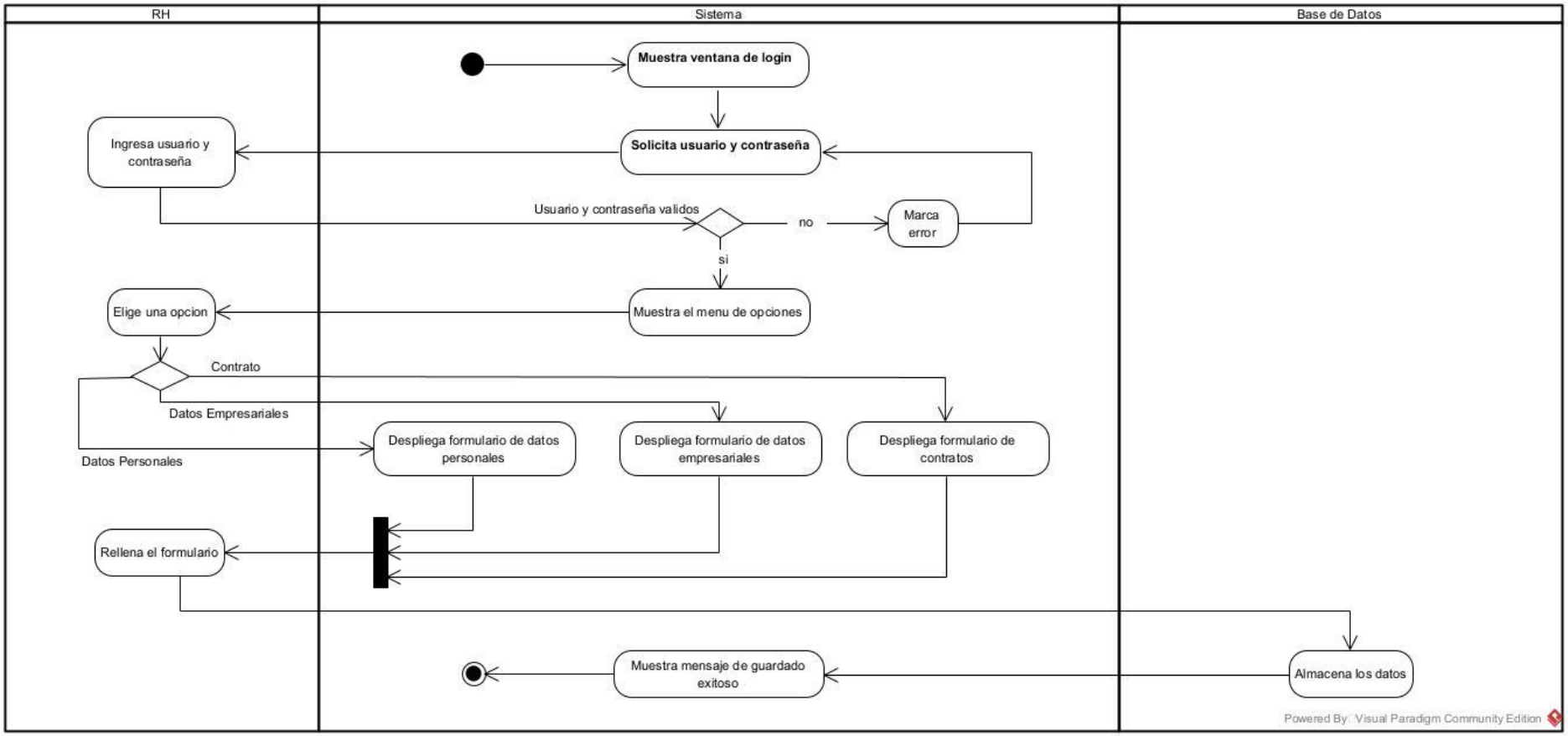
-Nombre de la empresa
-Teléfono
-RFC

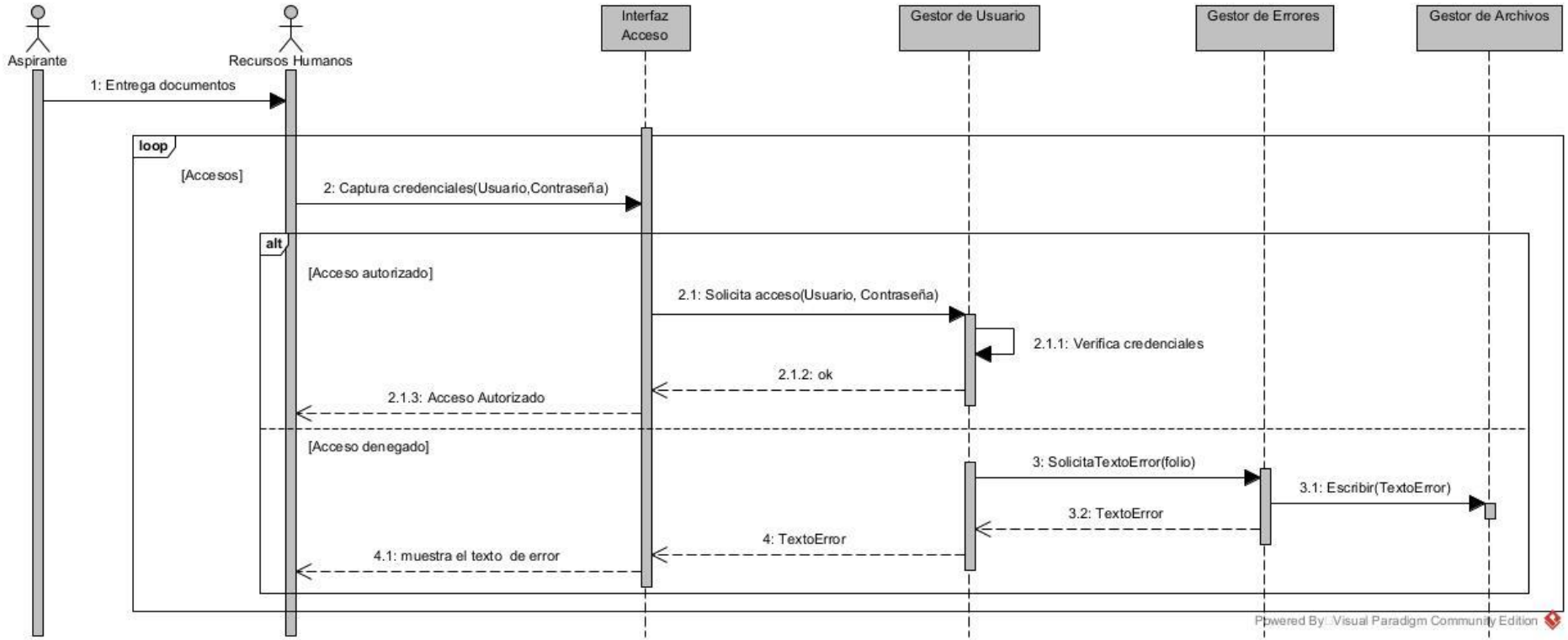
Entonces el uso de EMT-SYSTEM como el sistema oficial de la empresa Itera México en los países mencionados, resguardará la información personal y empresarial de cada empleado.

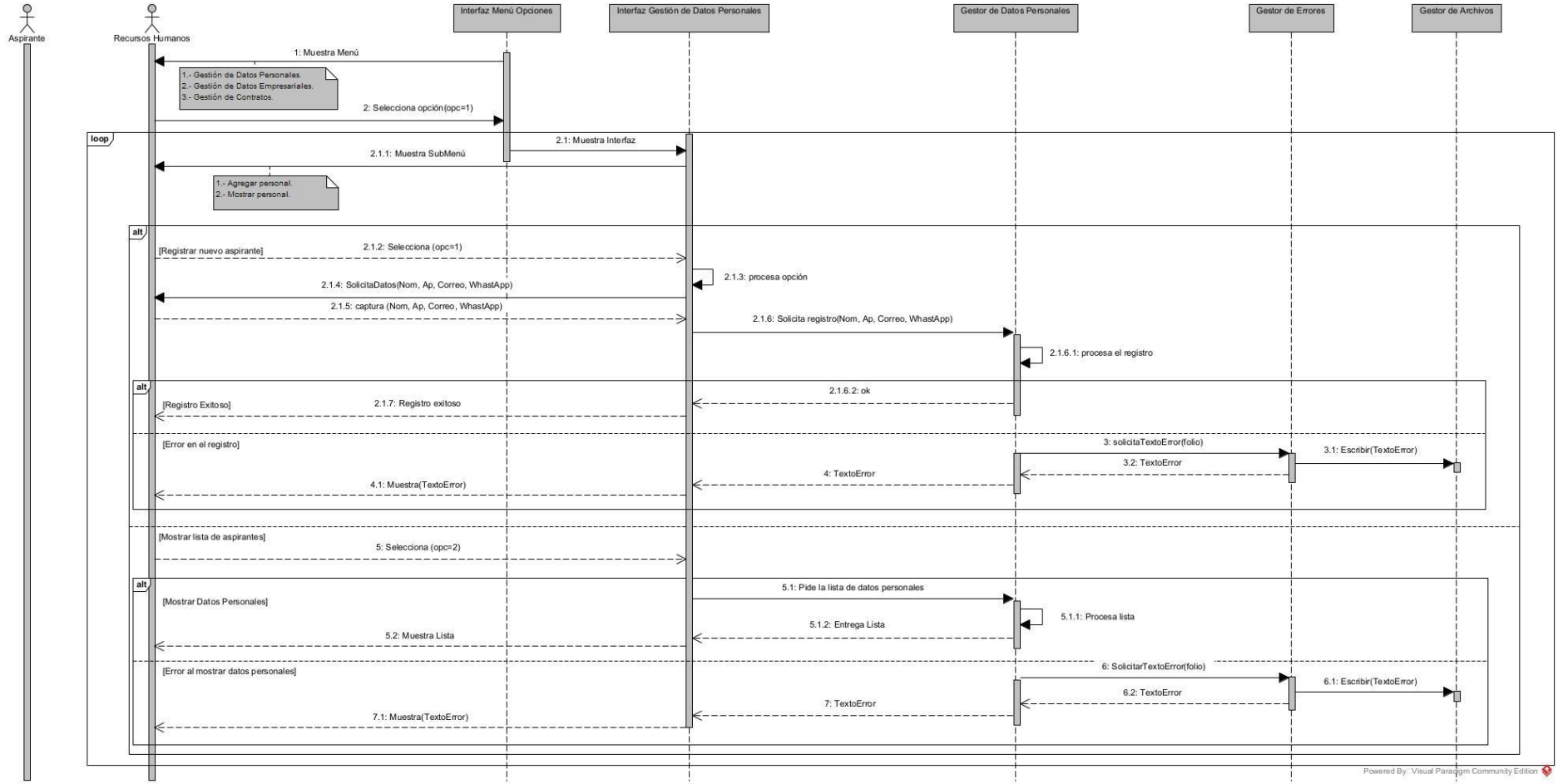

EMT-System
Control de Datos Personales
<<Extends>>
Login
<<extends>>
<<extends>>
RH
Base de Datos
Control de Datos Empresariales
<<Extends>>
Lista de Errores
Control de Contratos
<<Extends>>
Powered By: Visual Paradigm Community Edition

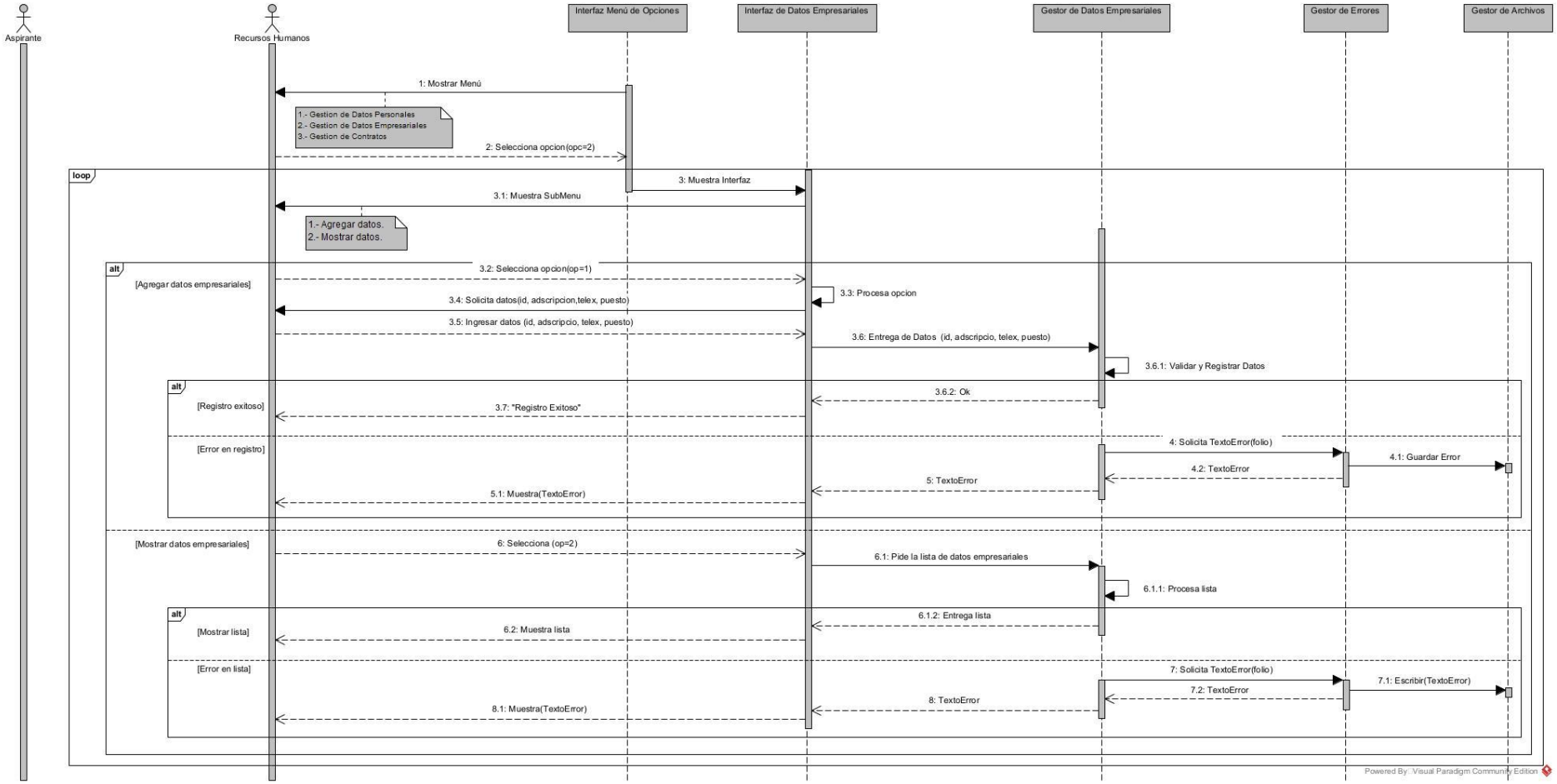

gestor
gestor.empresarial
gestor.empresarial.empleados
gestor.empresarial.datos
gestor.errores
gestor.empresarial.empresa
gestor.empresarial.contrato
gestor.archivos
Powered By: Visual Paradigm Community Edition

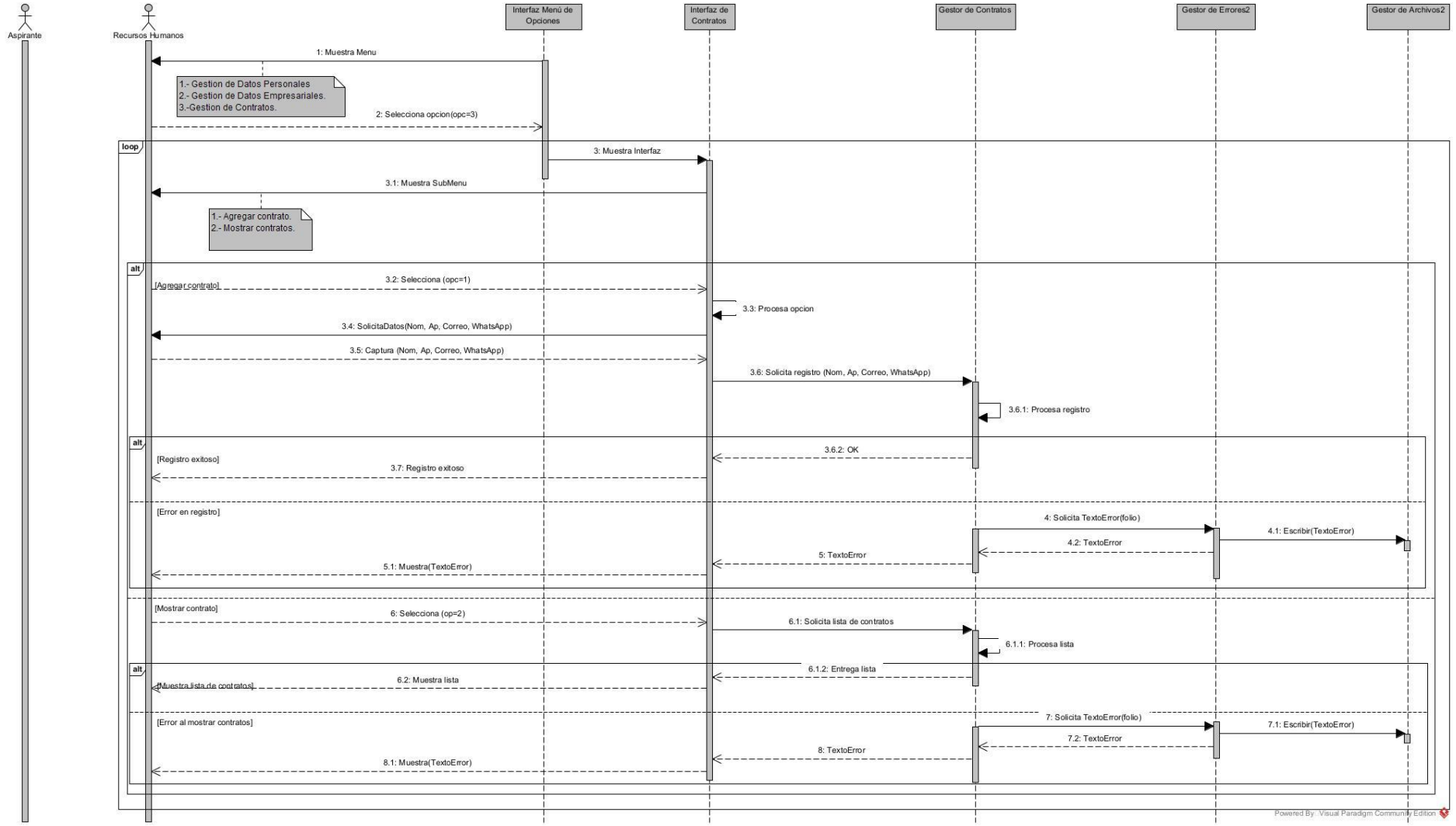

Contrato
-id : int
-noContrato : int
-annio : int
-horario : string
-tipoCargo : Cargos
+Contrato(id : int)
+getNoContrato() : int
+setNoContrato(noContrato : int) : void
+getAnnio() : int
+setAnnio(annio : int) : void
Empleados
-i : int
-error : GestionErrores
+Empleados()
+addDatosPersonales() : void
+addContrato() : void
+findEmpleado(id : int) : int
+findEmpleado(nombre : string) : int
+setWhatsapp() : void
+datosPersonales() : string
+getInfoEmpleado(id : int) : String
+getInfoEmpleado(nombre : string) : string
+setAdscripcion() : void
+setTelefonoExtension() : void
+setPuesto() : void
+showDatosEmpleados() : void
+showContratosEmpleado(id : int) : void
+getAntiguedad() : int
+setCargo(cargo : Cargo) : void
Datos Personales
-id : int
-nombre : string
-apellidos : string
-correo : string
-whatsapp : string
#getId() : int
#setId(id : int) : void
#getNombre() : string
#setNombre(nombre : string) : void
#getApellidos() : string
#setApellidos(apellidos : string) : void
#getCorreo() : string
#setCorreo(correo : string) : void
#getWhatsapp() : string
#setWhatsapp(whatsapp : string) : void
Datos Empresariales
-adscripcion : String
-telefonoExterior : String
-puesto : String
+getAdscripcion() : String
+setAdscripcion(adscripcion : String) : void
+getTelefonoExterior() : String
+setTelefonoExterior(telefonoExterior : String) : void
+getPuesto() : String
+setPuesto(puesto : String) : void
+Datos Empresariales()
Cargos
+confianza
+sindicalizado
+temporal
iEmpleados
+getInfoEmpleado(id : int) : String
+getInfoEmpleado(nombre : string) : string
+showDatosEmpleados() : void
+getAntiguedad() : int
Map
Empresa
-nombreEmpresa : String
-representanteLegal : String
-telefono : String
-rfc : String
+datosRH : Empleados
+error : GestionErrores
+Empresa()
+getRepresentanteLegal() : String
+setRepresentanteLegal(representanteLegal : String) : void
+setTelefono(telefono : String) : void
+getInfo() : string
GestionErrores
-error : Map<int noError, String DescripcionTecnica>
-existeError : boolean
-noError : int
-DescripcionTecnica : String
+GestionErrores()
+cargarErrores() : void
+setNoError() : void
+getError() : String
+getErrorTecnico() : String
+existeError() : boolean
ArchivoTexto
-file : File
-fr : FileReader
-br : BufferedReader
-fw : FileWriter
-bw : BufferedWriter
-archivoExiste : boolean
-modoLectura : boolean
-modoEscritura : boolean
+ArchivoTexto()
+ArchivoTexto(parameter)
+abrirModoLectura() : void
+leer() : String
+abrirModoEscritura() : void
+escribir() : void
+cerrar() : void
ControlArchivos
-archivo : String
+ControlArchivos()
+crear() : boolean
+eliminar() : boolean
+cambiar() : boolean
+mover() : boolean
iFileText
+abrirModoLectura() : void
+leer() : string
+abrirModoEscritura() : void
+escribir() : void
+cerrar() : void
Powered By Visual Paradigm Community Edition


RH
Sistema
Base de Datos
Muestra ventana de login
Solicita usuario y contraseña
Ingresa usuario y contraseña
Usuario y contraseña validos
no
Marca error
si
Muestra el menu de opciones
Elige una opcion
Contrato
Datos Empresariales
Datos Personales
Despliega formulario de datos personales
Despliega formulario de datos empresariales
Despliega formulario de contratos
Rellena el formulario
Muestra mensaje de guardado exitoso
Almacena los datos
Powered By Visual Paradigm Community Edition


Aspirante
Recursos Humanos
Interfaz Acceso
Gestor de Usuario
Gestor de Errores
Gestor de Archivos
1: Entrega documentos
loop
[Accesos]
2: Captura credenciales(Usuario,Contraseña)
alt
[Acceso autorizado]
2.1: Solicita acceso(Usuario, Contraseña)
2.1.1: Verifica credenciales
2.1.2: ok
2.1.3: Acceso Autorizado
[Acceso denegado]
3: SolicitaTextoError(folio)
3.1: Escribir(TextoError)
3.2: TextoError
4: TextoError
4.1: muestra el texto de error
Powered By Visual Paradigm Community Edition


Aspirante
Recursos Humanos
Interfaz Menú Opciones
Interfaz Gestión de Datos Personales
Gestor de Datos Personales
Gestor de Errores
Gestor de Archivos
1: Muestra Menú
1.- Gestión de Datos Personales.
2.- Gestión de Datos Empresariales.
3.- Gestión de Contratos.
2: Selecciona opción(opc=1)
loop
2.1: Muestra Interfaz
2.1.1: Muestra SubMenú
1.- Agregar personal.
2.- Mostrar personal.
alt
[Registrar nuevo aspirante]
2.1.2: Selecciona (opc=1)
2.1.3: procesa opción
2.1.4: SolicitaDatos(Nom, Ap, Correo, WhastApp)
2.1.5: captura (Nom, Ap, Correo, WhastApp)
2.1.6: Solicita registro(Nom, Ap, Correo, WhastApp)
2.1.6.1: procesa el registro
alt
[Registro Exitoso]
2.1.6.2: ok
2.1.7: Registro exitoso
[Error en el registro]
3: solicitaTextoError(folio)
3.1: Escribir(TextoError)
3.2: TextoError
4: TextoError
4.1: Muestra(TextoError)
[Mostrar lista de aspirantes]
5: Selecciona (opc=2)
alt
[Mostrar Datos Personales]
5.1: Pide la lista de datos personales
5.1.1: Procesa lista
5.1.2: Entrega Lista
5.2: Muestra Lista
[Error al mostrar datos personales]
6: SolicitarTextoError(folio)
6.1: Escribir(TextoError)
6.2: TextoError
7: TextoError
7.1: Muestra(TextoError)
Powered By Visual Paradigm Community Edition


Aspirante
Recursos Humanos
Interfaz Menú de Opciones
Interfaz de Datos Empresariales
Gestor de Datos Empresariales
Gestor de Errores
Gestor de Archivos
1: Mostrar Menú
1.- Gestion de Datos Personales
2.- Gestion de Datos Empresariales
3.- Gestion de Contratos
2: Selecciona opcion(opc=2)
loop
3: Muestra Interfaz
3.1: Muestra SubMenu
1.- Agregar datos.
2.- Mostrar datos.
alt
3.2: Selecciona opcion(op=1)
[Agregar datos empresariales]
3.3: Procesa opcion
3.4: Solicita datos(id, adscripcion,telex, puesto)
3.5: Ingresar datos (id, adscripcio, telex, puesto)
3.6: Entrega de Datos (id, adscripcio, telex, puesto)
3.6.1: Validar y Registrar Datos
alt
[Registro exitoso]
3.6.2: Ok
3.7: "Registro Exitoso"
[Error en registro]
4: Solicita TextoError(folio)
4.1: Guardar Error
4.2: TextoError
5: TextoError
5.1: Muestra(TextoError)
[Mostrar datos empresariales]
6: Selecciona (op=2)
6.1: Pide la lista de datos empresariales
6.1.1: Procesa lista
alt
[Mostrar lista]
6.1.2: Entrega lista
6.2: Muestra lista
[Error en lista]
7: Solicita TextoError(folio)
7.1: Escribir(TextoError)
7.2: TextoError
8: TextoError
8.1: Muestra(TextoError)
Powered By: Visual Paradigm Community Edition


Aspirante
Recursos Humanos
Interfaz Menú de Opciones
Interfaz de Contratos
Gestor de Contratos
Gestor de Errores2
Gestor de Archivos2
1: Muestra Menu
1.- Gestion de Datos Personales
2.- Gestion de Datos Empresariales.
3.-Gestion de Contratos.
2: Selecciona opcion(opc=3)
loop
3: Muestra Interfaz
3.1: Muestra SubMenu
1.- Agregar contrato.
2.- Mostrar contratos.
alt
[Agregar contrato]
3.2: Selecciona (opc=1)
3.3: Procesa opcion
3.4: SolicitaDatos(Nom, Ap, Correo, WhatsApp)
3.5: Captura (Nom, Ap, Correo, WhatsApp)
3.6: Solicita registro (Nom, Ap, Correo, WhatsApp)
3.6.1: Procesa registro
alt
[Registro exitoso]
3.6.2: OK
3.7: Registro exitoso
[Error en registro]
4: Solicita TextoError(folio)
4.1: Escribir(TextoError)
4.2: TextoError
5: TextoError
5.1: Muestra(TextoError)
[Mostrar contrato]
6: Selecciona (op=2)
6.1: Solicita lista de contratos
6.1.1: Procesa lista
alt
6.1.2: Entrega lista
[Muestra lista de contratos]
6.2: Muestra lista
[Error al mostrar contratos]
7: Solicita TextoError(folio)
7.1: Escribir(TextoError)
7.2: TextoError
8: TextoError
8.1: Muestra(TextoError)
Powered By Visual Paradigm Community Edition

Código Fuente

```
//Título: Proyecto EMT System
//Autor: Berenice Morales Bustamante
//Grupo: 5A
//Fecha de creación: 15 de febrero del 2024 - 08:00 - 10:00
//Última actualización: 15 de abril del 2024 - 00:00 - 01:20

import gestor.IntLogin;

public class Principal {
    public static void main(String[] args){
        IntLogin lgin = new IntLogin();
    }
}
```

```
package gestor.empresarial.contrato;

public enum Cargos {
    confianza("Empleado de confianza"),
    sindicalizado("Empleado sindicalizado"),
    temporal("Empleado temporal");

    private final String nombre;

    Cargos(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return nombre;
    }
}
```

```
package gestor.empresarial.contrato;

public enum Cargos {
    confianza("Empleado de confianza"),
```

```
    sindicalizado("Empleado sindicalizado"),
    temporal("Empleado temporal");
    //Metodos necesarios para obtener el nombre del cargo en String
    private final String nombre;

    Cargos(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return nombre;
    }
}
```

```
package gestor.empresarial.contrato;

import java.util.ArrayList;
import java.util.List;

public final class Contrato {
    private int noContrato;
    private int annio;
    private String horario;
    private Cargos tipoCargo;

    public Contrato(int noContrato, int annio, String horario, Cargos tipoCargo){
        this.noContrato = noContrato;
        this.annio = annio;
        this.horario = horario;
        this.tipoCargo = tipoCargo;
    }

    public int getNoContrato() {
        return noContrato;
    }

    public int getAnnio() {
        return annio;
    }

    public String getHorario() {
```

```
        return horario;
    }

    public Cargos getTipoCargo() {
        return tipoCargo;
    }

    public void setTipoCargo(Cargos tipoCargo) {
        this.tipoCargo = tipoCargo;
    }
}
```

```
package gestor.empresarial.contrato;

import gestor.IntMenu;
import gestor.empresarial.datos.DatosEmpresariales;
import gestor.empresarial.datos.DatosPersonales;
import gestor.empresarial.empleados.Empleados;

import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class IntContratos extends JFrame{

    private JPanel panel1;
    private JTextField fieldContrato;
    private JTextField fieldAnnio;
    private JTextField fieldHorario;
    private JButton agregarButton;
    private JScrollPane scrollForTable;
    private JTable tablaC;
    private JTextField fieldBuscador;
    private JComboBox comboCargo;
    private JButton borrarButton;
    private JButton modificarButton;
    private JLabel labelId;
    private JLabel labelName;
    private JButton cerrarButton;
```

```
    private JLabel labelAdscripcion;
    private JLabel labelPuesto;
    private JTextPane contratosTextPane;
    private JTextPane listaDeEmpleadosTextPane;
    private JTextPane empleadoTextPane;
    private JTextPane empresaTextPane;
    DefaultTableModel mt = new DefaultTableModel(); //Creamos modelo de la tabla
    private Empleados empleados;
    private int indice = -1;


    public IntContratos(){
        empleados = empleados.getInstancia();

        ajustesVentana(); //Ajustamos los parametros de la ventana

        comboCargo.setModel(new DefaultComboBoxModel<>(Cargos.values()));// Establecemos el modelo del JComboBox
utilizando los valores de la clase enum Cargos

        initComponents(); //Ajustes de la tabla
        funcionesBotones(); //Codigo que define las funcionalidades de los botones
    }

    private void ajustesVentana(){
        setTitle("Menu EMT-System"); //Establecemos el titulo de la ventana
        this.setSize(1100,500); //Establecemos el tamaño de la ventana
        this.setLocationRelativeTo(null); //Establecemos la posicion inicial de la ventana en el centro
        this.getContentPane().add(panel1);
        this.setVisible(true); //Volvemos nuestra ventana visible
        setDefaultCloseOperation(EXIT_ON_CLOSE); //Indicamos que termine la ejecucion del programa al cerrar la
ventana
    }

    private void initComponents() {
        String encabezados[] = {"ID","Nombre Completo","No.Contrato", "Año","Horario","Tipo de Cargo"};
        mt.setColumnIdentifiers(encabezados); //Definimos los titulos de las columnas
        tablaC.getTableHeader().setResizingAllowed(false); //No permitimos que se cambie el tamaño de la ventana
        tablaC.getTableHeader().setReorderingAllowed(false); //No permitimos que el usuario reordene las columnas de
la tabla
        tablaC.setModel(mt); //Establecemos el diseño de la tabla
        if (empleados.datosContratoVacios() == false) {
            actualizarTablaDesdeContrato();// Si los arreglos no están vacíos, muestra los datos en la tabla
```

```
    }
}

private void obtenerYGuardarContrato() {
    //Obtenemos los datos de los JTextField
    int noContrato = Integer.parseInt(fieldContrato.getText());
    int annio = Integer.parseInt(fieldAnnio.getText());
    String horario = fieldHorario.getText();
    Cargos tipocargo = (Cargos) comboCargo.getSelectedItem();

    //Guardamos los datos en Contrato
    Contrato obj = new Contrato(noContrato, annio, horario, tipocargo);

    //Guardamos nuestro obj en Empleados
    empleados.addContrato(indice,obj);
    empleados.imprimirDatos();
}

private void actualizarTablaDesdeContrato() {
    // Limpiamos la tabla antes de agregar los nuevos datos para evitar duplicados
    mt.setRowCount(0);

    // Agregamos los datos a la tabla
    for (int i = 0; i < 100; i++) {
        Contrato obj = empleados.getInfoContrato(i);
        DatosPersonales obj2 = empleados.getInfoPersonal(i);
        if(obj != null){
            int id = empleados.getID(i);
            String nombre = obj2.getNombre();
            int noContrato = obj.getNoContrato();
            int annio = obj.getAnnio();
            String horario = obj.getHorario();
            Cargos cargo = obj.getTipoCargo();
            mt.addRow(new Object[]{id,nombre,noContrato,annio,horario,cargo});
        }
    }
}

public void funcionesBotones() {
    fieldBuscador.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
```

```
                String textoBusqueda = fieldBuscador.getText();
                if (textoBusqueda != null){
                    // Buscando el ID ingresado en la lista de IDs en Empleados
                    int idBuscado = Integer.parseInt(textoBusqueda); // Convertir a entero
                    indice = empleados.findEmpleado(idBuscado);

                    // Verificando si se encontró el ID
                    if (indice != -1) {
                        DatosPersonales datosPersonales = empleados.getInfoPersonal(indice);
                        DatosEmpresariales datosEmpresariales = empleados.getInfoEmpresarial(indice);

                        // Obteneniendo la información relacionada al ID (nombre, adscripcion, puesto) en la lista
en DatosEmpresariales
                        String nombre= datosPersonales.getNombre();
                        String adscripcion = datosEmpresariales.getAdscripcion();
                        String puesto = datosEmpresariales.getPuesto();

                        // Mostrando la información en la ventana
                        labelId.setText("ID: "+ idBuscado);
                        labelName.setText("Nombre: " + nombre);
                        labelAdscripcion.setText("Adscripcion: " + adscripcion);
                        labelPuesto.setText("Puesto: " + puesto);
                        fieldBuscador.setText("");
                    } else {
                        // Mostrar un mensaje de error si no se encuentra el ID
                        JOptionPane.showMessageDialog(IntContratos.this, "ID no encontrado", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                else {
                    // Mostrar un mensaje de error si esta vacio el campo
                    JOptionPane.showMessageDialog(IntContratos.this, "Campo de busqueda vacio", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        // Agregar un ListSelectionListener a la JTable
        tablaC.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                if (!e.getValueIsAdjusting()) { // Evitar eventos de selección múltiple
```

```
            int selectedRow = tablaC.getSelectedRow();
            if (selectedRow != -1) { // Verificar si se seleccionó una fila
                // Obtener datos de la fila seleccionada
                Object noContrato = tablaC.getValueAt(selectedRow, 2);
                Object annio = tablaC.getValueAt(selectedRow, 3);
                Object horario = tablaC.getValueAt(selectedRow, 4);
                Object tipoCargo = tablaC.getValueAt(selectedRow, 5);

                // Mostrar los datos en los JTextField
                fieldContrato.setText(noContrato.toString());
                fieldAnnio.setText(annio.toString());
                fieldHorario.setText(horario.toString());
                comboCargo.setSelectedItem(tipoCargo);
            }
        }
    }
});



cerrarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        IntMenu obj = new IntMenu();
        dispose();
    }
});

agregarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (indice != -1){
            String noContrato = fieldContrato.getText();
            String annio = fieldAnnio.getText();
            String horario = fieldHorario.getText();
            Cargos tipoCargo = (Cargos) comboCargo.getSelectedItem();
            // Verificar que ningún campo esté vacío
            if (noContrato.isEmpty() || annio.isEmpty() || horario.isEmpty() || tipoCargo == null) {
                // Mostrar un mensaje de error indicando al usuario qué campo olvidó rellenar
                String mensaje = "Por favor, complete todos los campos:\n";
                if (noContrato.isEmpty()) {
                    mensaje += "- No.Contrato\n";
```

```
                    }
                    if (annio.isEmpty()) {
                        mensaje += "- Año\n";
                    }
                    if (horario.isEmpty()) {
                        mensaje += "- Horario\n";
                    }
                    if (tipoCargo == null) {
                        mensaje += "- Tipo de Cargo\n";
                    }
                    JOptionPane.showMessageDialog(null, mensaje, "Campos Vacíos", JOptionPane.ERROR_MESSAGE);
                } else {
                    // Si todos los campos están llenos, procede a agregar la fila a la tabla
                    obtenerYGuardarContrato();
                    actualizarTablaDesdeContrato();
                    // Limpiamos los JTextField después de agregar la fila
                    fieldContrato.setText("");
                    fieldAnnio.setText("");
                    fieldHorario.setText("");
                    comboCargo.setSelectedItem(null);
                    labelIId.setText("");
                    labelName.setText("");
                    labelAdscripcion.setText("");
                    labelPuesto.setText("");
                }
            }
            else{
                JOptionPane.showMessageDialog(null,"No se ha seleccionado a ningun empleado para añadir los
datos", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    borrarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedRow = tablaC.getSelectedRow();
            if (selectedRow != -1) {
                int id = (int) tablaC.getValueAt(selectedRow, 0); //Obtenemos el id en la fila seleccionada de
la tabla
                int indice = empleados.findEmpleado(id); //Buscamos al empleado en la lista de Empleados
                mt.removeRow(selectedRow); // Eliminamos la fila en la tabla
```

```
                    empleados.borrarEmpleado(indice); // Eliminamos los datos correspondientes en Empleados
                } else {
                    JOptionPane.showMessageDialog(null,"No se ha seleccionado a ningun empleado para borrar",
"Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        modificarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int selectedRow = tablaC.getSelectedRow();
                if (selectedRow != -1) { // Verificamos si se seleccionó una fila
                    int id = (int) tablaC.getValueAt(selectedRow,0);
                    int indice = empleados.findEmpleado(id); //Buscamos al empleado en la lista de Empleados

                    // Obtenemos los datos modificados
                    int noContrato = Integer.parseInt(fieldContrato.getText());
                    int annio = Integer.parseInt(fieldAnnio.getText());
                    String horario = fieldHorario.getText();
                    Cargos tipoCargo = (Cargos) comboCargo.getSelectedItem();


                    // Actualizamos los datos en la fila seleccionada de la JTable
                    tablaC.setValueAt(noContrato, selectedRow, 2);
                    tablaC.setValueAt(annio, selectedRow, 3);
                    tablaC.setValueAt(horario, selectedRow, 4);
                    tablaC.setValueAt(tipoCargo, selectedRow, 5);

                    //Creamos un objeto con los datos actualizados
                    Contrato obj = new Contrato(noContrato,annio,horario,tipoCargo);

                    //Guardamos nuestro obj en Empleados
                    empleados.addContrato(indice,obj);
                }
            }
        });
    }
}

package gestor.empresarial.datos;
```

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

public final class DatosEmpresariales{
    private String telefonoExterior;
    private String extension;
    private String adscripcion;
    private String puesto;

    public DatosEmpresariales(String telefonoExterior, String extension, String adscripcion, String puesto){
        this.telefonoExterior = telefonoExterior;
        this.extension = extension;
        this.adscripcion = adscripcion;
        this.puesto = puesto;
    }

    public String getTelefonoExterior() {
        return telefonoExterior;
    }

    public String getExtension() {
        return extension;
    }

    public String getAdscripcion() {
        return adscripcion;
    }

    public String getPuesto() {
        return puesto;
    }
}
```

```java
package gestor.empresarial.datos;

import java.util.ArrayList;
import java.util.List;

public class DatosPersonales {
    private String nombre;
    private String whatsapp;
    private String correo;
```

```
    public DatosPersonales(String nombre, String whatsapp, String correo) {
        this.nombre = nombre;
        this.whatsapp = whatsapp;
        this.correo = correo;
    }

    public String getNombre() {
        return nombre;
    }

    public String getWhatsapp() {
        return whatsapp;
    }


    public String getCorreo() {
        return correo;
    }
}
```

```
package gestor.empresarial.datos;

import gestor.IntMenu;
import gestor.empresarial.empleados.Empleados;
import gestor.errores.GestionErrores;

import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class IntMenuDos extends JFrame{
    private JPanel panel1;
    private JTable tablaDP;
    private JTextField fieldID;
    private JTextField fieldName;
    private JTextField fieldMail;
    private JButton cerrarButton;
    private JButton agregarButton;
    private JTextField fieldWhats;
```

```
    private JButton borrarButton;
    private JButton modificarButton;
    private JPanel panelbase;
    private JScrollPane scrollForTable;
    private JTextPane datosGuardadosTextPane;
    private JTextPane datosPersonalesTextPane;
    DefaultTableModel mt = new DefaultTableModel(); //Creamos modelo de la tabla
    private Empleados empleados;
    private GestionErrores gestionErrores;

    public IntMenuDos(){
        empleados = empleados.getInstancia();
        gestionErrores = new GestionErrores();

        ajustesVentana(); //Ajustamos los parametros de nuestra ventana

        initComponents(); //Ajustes de la tabla
        funcionesBotones(); //Codigo de las funcionalidades de los botones
    }

    public void ajustesVentana(){
        setTitle("Menu EMT-System"); //Establecemos el titulo de la ventana
        this.setSize(1000,600); //Establecemos el tamaño de la ventana
        this.setLocationRelativeTo(null); //Establecemos la posicion inicial de la ventana en el centro
        this.getContentPane().add(panel1); //Obtenemos el contenido del panel
        this.setVisible(true); //Volvemos nuestra ventana visible
        setDefaultCloseOperation(EXIT_ON_CLOSE); //Indicamos que termine la ejecucion del programa al cerrar la
ventana
    }

    private void initComponents() {
        String ids[] = {"ID","Nombre Completo","Whatsapp","Email"};
        mt.setColumnIdentifiers(ids);
        tablaDP.getTableHeader().setResizingAllowed(false);
        tablaDP.getTableHeader().setReorderingAllowed(false);
        tablaDP.setModel(mt);
        if (empleados.datosPerVacios() == false) {
            actualizarTablaDesdeDatosPersonales();// Si los arreglos no están vacíos, muestra los datos en la tabla
        }
    }

    private void obtenerYGuardarDatosPersonales() {
```

```
        //Obtenemos los datos de los JTextField
        String nombre = fieldName.getText();
        String whatsapp = fieldWhats.getText();
        String email = fieldMail.getText();

        // Guardamos los datos en DatosPersonales
        DatosPersonales obj = new DatosPersonales(nombre,whatsapp,email);

        //Guardamos nuestro obj en Empleados
        empleados.addDatosPersonales(obj);
        empleados.imprimirDatos();
    }

    private void actualizarTablaDesdeDatosPersonales() {
        // Limpiamos la tabla antes de agregar los nuevos datos para evitar duplicados
        mt.setRowCount(0);

        // Agregamos los datos a la tabla
        for (int i = 0; i < 100; i++) {
            DatosPersonales obj = empleados.getInfoPersonal(i);
            if(obj != null){
                int id = empleados.getID(i);
                String nombre = obj.getNombre();
                String whatsapp = obj.getWhatsapp();
                String correo = obj.getCorreo();
                mt.addRow(new Object[]{id,nombre,whatsapp,correo});
            }
        }
    }

    public boolean verificarCampos(){
        boolean camposCorrectos = true;
        String titulo;
        if (fieldID.getText().isEmpty() || fieldName.getText().isEmpty() || fieldWhats.getText().isEmpty() ||
fieldMail.getText().isEmpty()) {
            // Mostramos un mensaje de error indicando al usuario qué campos olvidó rellenar
            titulo = gestionErrores.getDescripcionTecnica(1);
            String mensaje = "Por favor, complete todos los campos:\n";
            if (fieldID.getText().isEmpty()) {
                mensaje += "- ID\n";
            }
            if (fieldName.getText().isEmpty()) {
```

```
                mensaje += "- Nombre\n";
            }
            if (fieldWhats.getText().isEmpty()) {
                mensaje += "- Whatsapp\n";
            }
            if (fieldMail.getText().isEmpty()) {
                mensaje += "- Email\n";
            }
            JOptionPane.showMessageDialog(null, mensaje, titulo, JOptionPane.ERROR_MESSAGE);
            camposCorrectos = false;
        } else {
            boolean hayDuplicados =
empleados.buscarDuplicadosP(Integer.parseInt(fieldID.getText()),fieldName.getText(),fieldWhats.getText(),fieldMail.g
etText());
            if (hayDuplicados) {
                titulo = gestionErrores.getDescripcionTecnica(6);
                JOptionPane.showMessageDialog(null, "Ya se ha guardado un empleado con esos mismos datos, favor de
verificar la información", titulo, JOptionPane.ERROR_MESSAGE);
                camposCorrectos = false;
            }
        }
        return camposCorrectos;
    }

    public void funcionesBotones(){
        // Agregamos un ListSelectionListener a la JTable
        tablaDP.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                if (!e.getValueIsAdjusting()) { // Con esto evitamos eventos de selección múltiple
                    int selectedRow = tablaDP.getSelectedRow();
                    if (selectedRow != -1) { // Verificamos si se seleccionó una fila
                        // Obtenemos datos de la fila seleccionada
                        Object id = tablaDP.getValueAt(selectedRow, 0);
                        Object nombre = tablaDP.getValueAt(selectedRow, 1);
                        Object whatsapp = tablaDP.getValueAt(selectedRow, 2);
                        Object email = tablaDP.getValueAt(selectedRow, 3);

                        // Mostrar los datos en los JTextField
                        fieldID.setText(id.toString());
                        fieldName.setText(nombre.toString());
                        fieldWhats.setText(whatsapp.toString());
```

```
                fieldMail.setText(email.toString());
            }
        }
    }
});

cerrarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        IntMenu obj = new IntMenu();
        dispose();
    }
});

agregarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String idText = fieldID.getText();
        String nombre = fieldName.getText();
        String wh = fieldWhats.getText();
        String correo = fieldMail.getText();
        boolean camposCorrectos = verificarCampos();// Verificar que ningún campo esté vacío o duplicado
        if (camposCorrectos == true) {
            // Si todos los campos están correctos, procedemos a agregar la fila a la tabla
            obtenerYGuardarDatosPersonales();
            actualizarTablaDesdeDatosPersonales();
            // Limpiamos los JTextField después de agregar la fila
            fieldID.setText("");
            fieldName.setText("");
            fieldWhats.setText("");
            fieldMail.setText("");
        }
    }
});

borrarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int selectedRow = tablaDP.getSelectedRow();
        if (selectedRow != -1) {
            int id = (int) tablaDP.getValueAt(selectedRow, 0);
            int indice = empleados.findEmpleado(id); //Buscamos al empleado en la lista de Empleados
```

```
                mt.removeRow(selectedRow); // Eliminamos la fila en la tabla
                empleados.borrarEmpleado(indice); // Eliminamos los datos correspondientes en Empleados
            } else {
                String titulo = gestionErrores.getDescripcionTecnica(5);
                JOptionPane.showMessageDialog(null, "No se ha seleccionado ningún candidato para borrar",
titulo, JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    modificarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedRow = tablaDP.getSelectedRow();
            if (selectedRow != -1) { // Verificamos si se seleccionó una fila
                int id = (int) tablaDP.getValueAt(selectedRow, 0);
                int indice = empleados.findEmpleado(id); //Buscamos al empleado en la lista de Empleados

                // Obtenemos los datos modificados
                String nombre = fieldName.getText();
                String whats = fieldWhats.getText();
                String correo = fieldMail.getText();

                // Actualizamos los datos en la fila seleccionada de la JTable
                tablaDP.setValueAt(nombre, selectedRow, 1);
                tablaDP.setValueAt(whats, selectedRow, 2);
                tablaDP.setValueAt(correo, selectedRow, 3);

                // Creamos un objeto con los datos actualizados
                DatosPersonales obj = new DatosPersonales(nombre,whats,correo);

                //Guardamos nuestro obj en Empleados
                empleados.modificarEmpleado(indice,obj);
            }
            else {
                String titulo = gestionErrores.getDescripcionTecnica(5);
                JOptionPane.showMessageDialog(null, "No se ha seleccionado ningún candidato para modificar",
titulo, JOptionPane.ERROR_MESSAGE);
            }
        }
    });
```

```
    }
}

package gestor.empresarial.datos;

import gestor.IntMenu;
import gestor.empresarial.empleados.Empleados;
import gestor.errores.GestionErrores;

import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class IntMenuTres extends JFrame{
    private JPanel panel1;
    private JTextField fieldTelefono;
    private JTextField fieldAdscripcion;
    private JTextField fieldExtension;
    private JTextField fieldPuesto;
    private JButton agregarButton;
    private JButton cerrarButton;
    private JTable tablaDE;
    private JScrollPane scrollForTable;
    private JPanel panel2;
    private JButton borrarButton;
    private JButton modificarButton;
    private JLabel idEmpleado;
    private JLabel nombreEmpleado;
    private JLabel whatsEmpleado;
    private JLabel correoEmpleado;
    private JTextField fieldBuscador;
    private JTextPane datosEmpresarialesTextPane;
    private JTextPane listaDeEmpleadosTextPane;
    DefaultTableModel mt = new DefaultTableModel(); //Creamos modelo de la tabla
    private Empleados empleados;
    private int indice = -1;
    private GestionErrores gestionErrores;
```

```
    public IntMenuTres(){
        empleados = empleados.getInstancia();
        gestionErrores = new GestionErrores();

        ajustesVentana(); //Ajustamos los parametros de la ventana

        initComponents(); //Ajustes de la tabla
        funcionesBotones(); //Codigo que define las funcionalidades de los botones
    }

    public void ajustesVentana(){
        setTitle("Menu EMT-System"); //Establecemos el titulo de la ventana
        this.setSize(1100,500); //Establecemos el tamaño de la ventana
        this.setLocationRelativeTo(null); //Establecemos la posicion inicial de la ventana en el centro
        this.getContentPane().add(panel1);
        this.setVisible(true); //Volvemos nuestra ventana visible
        setDefaultCloseOperation(EXIT_ON_CLOSE); //Indicamos que termine la ejecucion del programa al cerrar la
ventana
    }

    private void initComponents() {
        String encabezados[] = {"ID","Nombre Completo","Extension", "Telefono Exterior","Adscripcion","Puesto"};
        mt.setColumnIdentifiers(encabezados); //Definimos los titulos de las columnas
        tablaDE.getTableHeader().setResizingAllowed(false); //No permitimos que se cambie el tamaño de la ventana
        tablaDE.getTableHeader().setReorderingAllowed(false); //No permitimos que el usuario reordene las columnas
de la tabla
        tablaDE.setModel(mt); //Establecemos el diseño de la tabla
        if (empleados.datosEmpVacios() == false) {
            actualizarTablaDesdeDatosEmpresariales();// Si los arreglos no están vacíos, muestra los datos en la
tabla
        }
    }

    private void obtenerYGuardarDatosEmpresariales(int indice) {
        //Obtenemos los datos de los JTextField
        String telefono = fieldTelefono.getText();
        String extension = fieldExtension.getText();
        String adscripcion = fieldAdscripcion.getText();
        String puesto = fieldPuesto.getText();

        // Guardamos los datos en DatosEmpresariales
        DatosEmpresariales obj = new DatosEmpresariales(telefono, extension, adscripcion, puesto);
```

```
        //Guardamos nuestro obj en Empleados
        empleados.addDatosEmpresariales(indice,obj);
        this.indice = -1;
        empleados.imprimirDatos();
    }

    private void actualizarTablaDesdeDatosEmpresariales() {
        // Limpiamos la tabla antes de agregar los nuevos datos para evitar duplicados
        mt.setRowCount(0);

        // Agregamos los datos a la tabla
        for (int i = 0; i < 100; i++) {
            DatosEmpresariales obj = empleados.getInfoEmpresarial(i);
            DatosPersonales obj2 = empleados.getInfoPersonal(i);
            if(obj != null){
                int id = empleados.getID(i);
                String nombre = obj2.getNombre();
                String telefono = obj.getTelefonoExterior();
                String extension = obj.getExtension();
                String adscripcion = obj.getAdscripcion();
                String puesto = obj.getPuesto();
                mt.addRow(new Object[]{id,nombre,extension,telefono,adscripcion,puesto});
            }
        }
    }

    public void funcionesBotones() {
        fieldBuscador.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String textoBusqueda = fieldBuscador.getText();
                if (textoBusqueda != null){
                    // Buscando el ID ingresado en la lista de IDs en Empleados
                    int idBuscado = Integer.parseInt(textoBusqueda); // Convertir a entero el id
                    indice = empleados.findEmpleado(idBuscado);

                    // Verificando si se encontró el ID
                    if (indice != -1) {
                        DatosPersonales datosPersonales = empleados.getInfoPersonal(indice);

                        //Obteneniendo la información relacionada al ID (nombre, whatsapp, correo) en el arreglo de
```

```
DatosPersonales
                        String nombre = datosPersonales.getNombre();
                        String whatsapp = datosPersonales.getWhatsapp();
                        String correo = datosPersonales.getCorreo();

                        // Mostrando la información en la ventana
                        idEmpleado.setText("ID: "+ idBuscado);
                        nombreEmpleado.setText("Nombre: " + nombre);
                        whatsEmpleado.setText("Whatsapp: " + whatsapp);
                        correoEmpleado.setText("Correo: " + correo);
                        fieldBuscador.setText("");
                    } else {
                        String mensaje = gestionErrores.getDescripcionTecnica(4);
                        JOptionPane.showMessageDialog(IntMenuTres.this, mensaje, "ID no valido",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                else {
                    String titulo = gestionErrores.getDescripcionTecnica(1);
                    JOptionPane.showMessageDialog(IntMenuTres.this, "Campo de busqueda vacio", titulo,
JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        tablaDE.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                if (!e.getValueIsAdjusting()) { // Evitar eventos de selección múltiple
                    int selectedRow = tablaDE.getSelectedRow();
                    if (selectedRow != -1) { // Verificar si se seleccionó una fila
                        // Obtener datos de la fila seleccionada
                        Object telefono = tablaDE.getValueAt(selectedRow, 3);
                        Object extension = tablaDE.getValueAt(selectedRow, 2);
                        Object adscripcion = tablaDE.getValueAt(selectedRow, 4);
                        Object puesto = tablaDE.getValueAt(selectedRow, 5);

                        // Mostrar los datos en los JTextField
                        fieldTelefono.setText(telefono.toString());
                        fieldExtension.setText(extension.toString());
                        fieldAdscripcion.setText(adscripcion.toString());
                        fieldPuesto.setText(puesto.toString());
```

```
            }
        }
    }
});


cerrarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        IntMenu obj = new IntMenu();
        dispose();
    }
});

agregarButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (indice != -1){
            String telefonos = fieldTelefono.getText();
            String extension = fieldExtension.getText();
            String adscripcion = fieldAdscripcion.getText();
            String puesto = fieldPuesto.getText();
            // Verificar que ningún campo esté vacío
            if (telefonos.isEmpty() || extension.isEmpty() || adscripcion.isEmpty() || puesto.isEmpty()) {
                // Mostrar un mensaje de error indicando al usuario qué campo olvidó rellenar
                String mensaje = gestionErrores.getDescripcionTecnica(1);
                if (telefonos.isEmpty()) {
                    mensaje += "- Telefono\n";
                }
                if (extension.isEmpty()) {
                    mensaje += "- Extension\n";
                }
                if (adscripcion.isEmpty()) {
                    mensaje += "- Adscripcion\n";
                }
                if (puesto.isEmpty()) {
                    mensaje += "- Puesto\n";
                }
                JOptionPane.showMessageDialog(null, mensaje, "Campos Vacíos", JOptionPane.ERROR_MESSAGE);
            } else {
                // Si todos los campos están llenos, procedemos a agregar la fila a la tabla
```

```
                    obtenerYGuardarDatosEmpresariales(indice);
                    actualizarTablaDesdeDatosEmpresariales();
                    // Limpiamos los JTextField después de agregar la fila
                    fieldTelefono.setText("");
                    fieldExtension.setText("");
                    fieldAdscripcion.setText("");
                    fieldPuesto.setText("");
                    idEmpleado.setText("");
                    nombreEmpleado.setText("");
                    whatsEmpleado.setText("");
                    correoEmpleado.setText("");
                }
            }
            else{
                String mensaje = gestionErrores.getDescripcionTecnica(5);
                JOptionPane.showMessageDialog(null,mensaje, "Datos no agregados", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    borrarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedRow = tablaDE.getSelectedRow();
            if (selectedRow != -1) {
                int id = (int) tablaDE.getValueAt(selectedRow, 0); //Obtenemos el id en la fila seleccionada de
la tabla
                int indice = empleados.findEmpleado(id); //Buscamos al empleado en la lista de Empleados
                mt.removeRow(selectedRow); // Eliminamos la fila en la tabla
                empleados.borrarEmpleado(indice); // Eliminamos los datos correspondientes en Empleados
            } else {
                String mensaje = gestionErrores.getDescripcionTecnica(5);
                JOptionPane.showMessageDialog(null,mensaje, "No se puede borrar", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    modificarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedRow = tablaDE.getSelectedRow();
            if (selectedRow != -1) { // Verificamos si se seleccionó una fila
```

```
                int id = (int) tablaDE.getValueAt(selectedRow, 0);
                int indice = empleados.findEmpleado(id); //Buscamos al empleado en la lista de Empleados

                // Obtenemos los datos modificados
                String telefono = fieldTelefono.getText();
                String extension = fieldExtension.getText();
                String adscripcion = fieldAdscripcion.getText();
                String puesto = fieldPuesto.getText();

                // Actualizamos los datos en la fila seleccionada de la JTable
                tablaDE.setValueAt(telefono, selectedRow, 3);
                tablaDE.setValueAt(extension, selectedRow, 2);
                tablaDE.setValueAt(adscripcion, selectedRow, 4);
                tablaDE.setValueAt(puesto, selectedRow, 5);

                // Creamos un objeto con los datos actualizados
                DatosEmpresariales obj = new DatosEmpresariales(telefono,extension,adscripcion,puesto);

                //Guardamos nuestro obj en Empleados
                empleados.addDatosEmpresariales(indice,obj);
            } else {
                String mensaje = gestionErrores.getDescripcionTecnica(5);
                JOptionPane.showMessageDialog(null,mensaje, "No se puede modificar", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
  }
}
```

```
package gestor.empresarial.empleados;
import gestor.empresarial.datos.*;
import gestor.errores.GestionErrores;
import gestor.empresarial.contrato.*;
import java.time.Year;

public final class Empleados implements iEmpleados{
    private int i=0;
    public GestionErrores error;
    private static Empleados instancia = null; // Creamos una istancia de la propia clase para aplicar el patrón de
diseño Singleton
    private int[] ids = new int[100];
```

```
    private DatosPersonales[] datosP = new DatosPersonales[100];
    private DatosEmpresariales[] datosE = new DatosEmpresariales[100];
    private Contrato[] datosC = new Contrato[100];

    public Empleados() { //Constructor

    }
    public static Empleados getInstancia() { //Instancia unica para mantener los datos guardados toda la ejecucion
del programa
        if (instancia == null) {
            instancia = new Empleados();
        }
        return instancia;
    }

    public void imprimirDatos(){
        if (datosPerVacios() == false){
            for(int j=0; j<100; j++){
                if (datosP[j] != null){
                    String nombre = datosP[j].getNombre();
                    String correo = datosP[j].getCorreo();
                    String whatsapp = datosP[j].getWhatsapp();
                    System.out.println("ID:" + ids[j] + "\tNombre:" + nombre + "\tCorreo:" + correo + "\tWhats:" +
whatsapp);
                }
            }
        }

        if (datosEmpVacios() == false){
            for(int j=0; j<100; j++){
                if (datosE[j] != null){
                    String telefono = datosE[j].getTelefonoExterior();
                    String extencion = datosE[j].getExtension();
                    String adscripcion = datosE[j].getAdscripcion();
                    String puesto = datosE[j].getPuesto();
                    System.out.println("ID:" + ids[j] + "\tTelefono:" + telefono + "\tExtension:" + extencion +
"\tAdscripcion:" + adscripcion + "\tPuesto:" + puesto);
                }
            }
        }

        if (datosContratoVacios() == false){
```

```
            for(int j=0; j<100; j++){
                if (datosC[j] != null){
                    int noContrato = datosC[j].getNoContrato();
                    int annio = datosC[j].getAnnio();
                    String horario = datosC[j].getHorario();
                    Cargos tipoCargo = datosC[j].getTipoCargo();
                    System.out.println("ID:" + ids[j] + "\tNoConctato:" + noContrato + "\tAnnio:" + annio +
"\tHorario:" + horario + "\tCargo:" + tipoCargo);
                }
            }
        }
    }

    //METODOS PARA AGREGAR LOS DATOS
    public void addDatosPersonales(DatosPersonales datosPersonales){
        this.datosP[this.i] = datosPersonales;
        ids[i]=i+1;
        this.i++;
    }

    public void addDatosEmpresariales(int indice, DatosEmpresariales datosEmpresariales){
        this.datosE[indice] = datosEmpresariales;
    }

    public void addContrato(int indice, Contrato contrato){
        this.datosC[indice] = contrato;
    }

    //METODOS PARA BUSCAR A UN EMPLEADO

    public int findEmpleado(int id){
        int indice = -1;
        for (int j = 0; j < 100; j++){
            if (ids[j] == id){
                indice = j; // Si se encuentra el ID se actualizamos el indice
                break;
            }
        }
        return indice;
    }
    public int findEmpleado(String nombre){
        int indice = -1;
```

```
        for (int j = 0; j < 100; j++){
            if (datosP[j] != null && datosP[j].getNombre().equals(nombre)){
                indice = j; // Si se encuentra el ID actualizamos el indice
                break;
            }
        }
        return indice;
    }

    //METODOS PARA RETORNAR LA INFORMACION DE UN EMPLEADO
    public int getID(int indice){
        return ids[indice];
    }
    public DatosPersonales getInfoPersonal(int indice){
        DatosPersonales datosPersonales = this.datosP[indice];
        return datosPersonales;
    }

    public DatosEmpresariales getInfoEmpresarial(int indice){
        DatosEmpresariales datosEmpresariales = this.datosE[indice];
        return datosEmpresariales;
    }

    public Contrato getInfoContrato(int indice){
        Contrato contrato = this.datosC[indice];
        return contrato;
    }

    @Override
    public Object getInfoEmpleado(int a) {
        return null;
    }

    @Override
    public String getInfoEmpleado(String b) {
        return null;
    }

    //METODOS PARA COMPROBAR SI LOS ARREGLOS ESTAN VACIOS

    public boolean datosPerVacios(){
        int suma=0;
```

```
    boolean vacio;
    for(int j=0; j<100; j++){
        if(datosP[j] != null){
            suma += 1;
        }
    }
    if (suma>0){
        vacio = false;
    }else{
        vacio = true;
    }
    return vacio;
}

public boolean datosEmpVacios(){
    int suma=0;
    boolean vacio;
    for(int j=0; j<100; j++){
        if(datosE[j] != null){
            suma += 1;
        }
    }
    if (suma>0){
        vacio = false;
    }else{
        vacio = true;
    }
    return vacio;
}

public boolean datosContratoVacios(){
    int suma=0;
    boolean vacio;
    for(int j=0; j<100; j++){
        if(datosC[j] != null){
            suma += 1;
        }
    }
    if (suma>0){
        vacio = false;
    }else{
        vacio = true;
```

```
        }
        return vacio;
    }

    public void showDatosEmpleado() {

    }
    public void showContratosEmpleado(int i){

    }
    public int getAntiguedad(int annio) {
        int antiguedad = Year.now().getValue() - annio;
        return 0;
    }

    //METODO PARA BORRAR A UN EMPLEADO
    public void borrarEmpleado(int indice){
        ids[indice] = -1;
        datosP[indice] = null;
        datosE[indice] = null;
        datosC[indice] = null;
    }

    //METODO PARA MODIFICAR UN EMPLEADO
    public void modificarEmpleado(int indice, DatosPersonales datosPersonales){
        this.datosP[indice] = datosPersonales;
    }

    //METODO PARA BUSCAR DATOS DUPLICADOS
    public boolean buscarDuplicadosP(int id, String nombre, String whatsapp, String correo) {
        boolean hayDuplicados = false;
        for (int j = 0; j < 100; j++) {
            DatosPersonales obj = getInfoPersonal(j);
            if (obj != null) {
                int idP = ids[j];
                String nombreP = obj.getNombre();
                String whatsP = obj.getWhatsapp();
                String correoP = obj.getCorreo();
                if (idP == id || nombreP.equals(nombre) || whatsP.equals(whatsapp) || correoP.equals(correo)) {
                    hayDuplicados = true;
                    break;
                }
```

```
            }
        }
        return hayDuplicados;
    }
}
```

```
package gestor.empresarial.empleados;

interface iEmpleados {
    public Object getInfoEmpleado(int a);
    public String getInfoEmpleado(String b);
    public void showDatosEmpleado();
    public int getAntiguedad(int a);
}
```

```
package gestor.empresarial.empresa;
import gestor.empresarial.empleados.*;
import gestor.errores.GestionErrores;

public final class Empresa {
    private String nombreEmpresa = "Itera Process";
    private String representanteLegal = "Ariel Súcari";
    private String telefono = "(+52) (55) 3300 0650";
    private String rfc = "IPM181023H34";

    public Empresa(){

    }

    public String getNombreEmpresa() {
        return nombreEmpresa;
    }

    public String getRepresentanteLegal() {
        return representanteLegal;
    }

    public String getTelefono() {
        return telefono;
    }

    public String getRfc() {
```

```
        return rfc;
    }
}
package gestor.errores;

import gestor.archivos.ArchivoTexto;

public class GestionErrores {
    private String descripcionTecnica;
    private MapaErrores mapaErrores;
    private ArchivoTexto archivoTexto;

    public GestionErrores() {
        mapaErrores = new MapaErrores();
        archivoTexto = new ArchivoTexto("ListadoErrores");
    }

    public String getDescripcionTecnica(int noError) {
        // Obtenemos el mensaje del error que corresponde a ese numero de error
        descripcionTecnica = mapaErrores.obtenerMensajeError(noError);

        // Escribimos el código y el mensaje de error en el archivo de texto
        escribirEnArchivo(noError, descripcionTecnica);
        return descripcionTecnica;
    }

    private void escribirEnArchivo(int noError, String descripcionTecnica) {
        archivoTexto.abrirModoEscritura();
        archivoTexto.escribir("Código de error: " + noError + "\t");
        archivoTexto.escribir("Mensaje de error: " + descripcionTecnica + "\n");
        archivoTexto.cerrar();
    }


}

package gestor.errores;

import java.util.HashMap;
import java.util.Map;

public class MapaErrores {
```

```
    private Map<Integer, String> errores;

    public MapaErrores() {
        errores = new HashMap<>();
        inicializarErrores();
    }

    private void inicializarErrores() {
        errores.put(1, "Campos de texto vacios, favor de rellenar todos los campos:");
        errores.put(2, "El usuario o contraseña son incorrectos, favor de ingresar datos validos");
        errores.put(3, "Campos de texto vacios, por favor ingrese su usuario y contraseña");
        errores.put(4, "ID no valido, este ID no existe");
        errores.put(5, "No se ha seleccionado algun empleado para modificar");
        errores.put(6, "Datos duplicados");
    }

    public String obtenerMensajeError(int codigoError) {
        return errores.get(codigoError);
    }
}
```

```
package gestor;

import gestor.errores.GestionErrores;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class IntLogin extends JFrame {
    private JPanel panel1;
    private JTextField usuarioTextField;
    private JPasswordField cPass;
    private JButton cerrarB;
    private JButton ingresarButton;
    private JLabel etUser;
    private JLabel etPass;
    private JLabel imagUser;
    private JPasswordField pass;
    private GestionErrores gestionErrores;

    public IntLogin(){
```

```
        gestionErrores = new GestionErrores();
        setTitle("EMT-System"); //Estabalecemos el titulo de la ventana
        this.setSize(300,300); //Establecemos el tamaño de la ventana
        this.setResizable(false);
        this.setLocationRelativeTo(null); //Establecemos la posicion inicial de la ventana en el centro
        this.getContentPane().add(panel1);
        this.setVisible(true); //Volvemos nuestra ventana visible
        setDefaultCloseOperation(EXIT_ON_CLOSE); //Indicamos que termine la ejecucion del programa al cerrar la
ventana

        cerrarB.addActionListener(e -> System.exit(0));

        ingresarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String usuario = usuarioTextField.getText();
                String contrasenia = pass.getText();
                if (!usuario.isEmpty() && !contrasenia.isEmpty()) {
                    if (usuario.equals("Tomas") && contrasenia.equals("talento")) {
                        IntMenu obj = new IntMenu();
                        dispose();
                    } else {
                        String mensaje = gestionErrores.getDescripcionTecnica(2);
                        JOptionPane.showMessageDialog(null, mensaje, "Datos no válidos", JOptionPane.ERROR_MESSAGE);
                    }
                } else {
                    String mensaje = gestionErrores.getDescripcionTecnica(3);
                    JOptionPane.showMessageDialog(null, mensaje, "Campos vacíos", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
    }
}

package gestor;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import gestor.IntLogin;
import gestor.empresarial.datos.*;
```

```
import gestor.empresarial.contrato.*;
import gestor.empresarial.empleados.Empleados;

public class IntMenu extends JFrame{
    private JPanel panel1;
    private JButton cerrarSesionButton;
    private JButton datosPersonalesButton;
    private JButton datosEmpresarialesButton;
    private JButton contratosButton;
    Empleados empleados;

    public IntMenu(){
        empleados = empleados.getInstancia();
        setTitle("Menu EMT-System"); //Establecemos el titulo de la ventana
        this.setSize(800,400); //Establecemos el tamaño de la ventana
        this.setResizable(false);
        this.setLocationRelativeTo(null); //Establecemos la posicion inicial de la ventana en el centro
        this.getContentPane().add(panel1);
        this.setVisible(true); //Volvemos nuestra ventana visible
        setDefaultCloseOperation(EXIT_ON_CLOSE); //Indicamos que termine la ejecucion del programa al cerrar la
ventana
        habilitarBotones();
        funcionesBotones();
    }

    public void habilitarBotones(){
        if(empleados.datosPerVacios() == true){
            datosEmpresarialesButton.setEnabled(false);
            contratosButton.setEnabled(false);
        } else if(empleados.datosEmpVacios()){
            contratosButton.setEnabled(false);
        }
    }

    public void funcionesBotones(){

        cerrarSesionButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                IntLogin obj = new IntLogin();
                dispose();
            }
```

```
        });

        datosPersonalesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                IntMenuDos obj = new IntMenuDos();
                dispose();
            }
        });

        datosEmpresarialesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                IntMenuTres obj = new IntMenuTres();
                dispose();
            }
        });

        contratosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                IntContratos obj = new IntContratos();
                dispose();
            }
        });
    }
}

package gestor.archivos;
import java.io.*;

public final class ArchivoTexto extends ControlArchivos implements iFileText{
    private File file;
    private FileReader fr;
    private BufferedReader br;
    private FileWriter fw;
    private BufferedWriter bw;
    private boolean archivoExiste;
    private boolean modoLectura;
    private boolean modoEscritura;
    public ArchivoTexto(String tituloArchivo){
        super();
```

```
        try{
            file = new File(tituloArchivo);
            if(!file.exists())
                file.createNewFile();
            this.archivoExiste=true;
            this.modoLectura=false;
            this.modoEscritura=false;
        }
        catch (Exception e){
            System.out.println("Error al intentar buscar el archivo");
            this.archivoExiste=false;
        }
    }

    public void abrirModoLectura(){
        if(archivoExiste==true){
            try{
                fr = new FileReader(this.file.getAbsoluteFile());
                br = new BufferedReader(this.fr);
                this.modoLectura=true;
                System.out.println("Archivo abierto en modo lectura");
            } catch (Exception e){
                System.out.println("Error: El archivo no se puede abrir en modo lectura");
            }
        }
    }

    public String leer(){
        if(archivoExiste==true) {
            try {
                return this.br.readLine();
            } catch (Exception e) {

            }
        }
        return null;
    }

    public void abrirModoEscritura(){
        if(archivoExiste==true){
            try{
                fw = new FileWriter(this.file.getAbsoluteFile(),true);
```

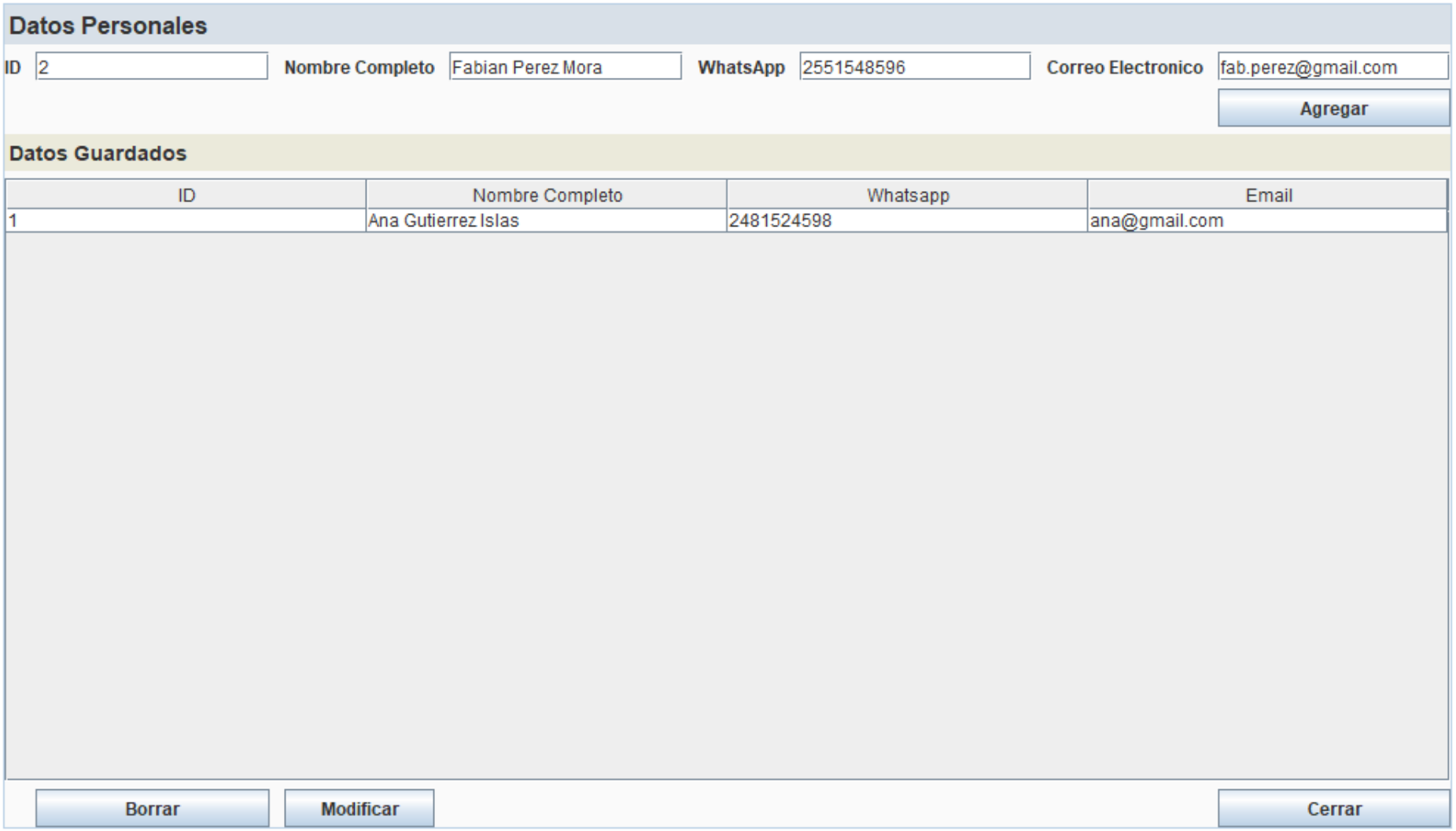
```
                bw = new BufferedWriter(this.fw);
                modoEscritura=true;
                System.out.println("Archivo abierto en modo escritura");
            } catch (Exception e){
                System.out.println("Error: El archivo no se puede abrir en modo escritura");
            }
        }
    }

    public void escribir(String texto){
        if (archivoExiste==true){
            try{
                this.bw.write(texto +"\n");
            } catch (Exception e){
                System.out.println("Error no se puede escribir informacion en el archivo");
            }
        }
    }

    public void cerrar(){
        if(modoEscritura==true){
            try{
                this.bw.close();
                this.fw.close();
            } catch (Exception e){ }
        }
        else if (modoLectura==true) {
            try {
                this.br.close();
                this.fr.close();
            } catch (Exception e) {
            }
        }
    }
}

package gestor.archivos;
import java.io.*;

public abstract class ControlArchivos {
    private String archivo;
```

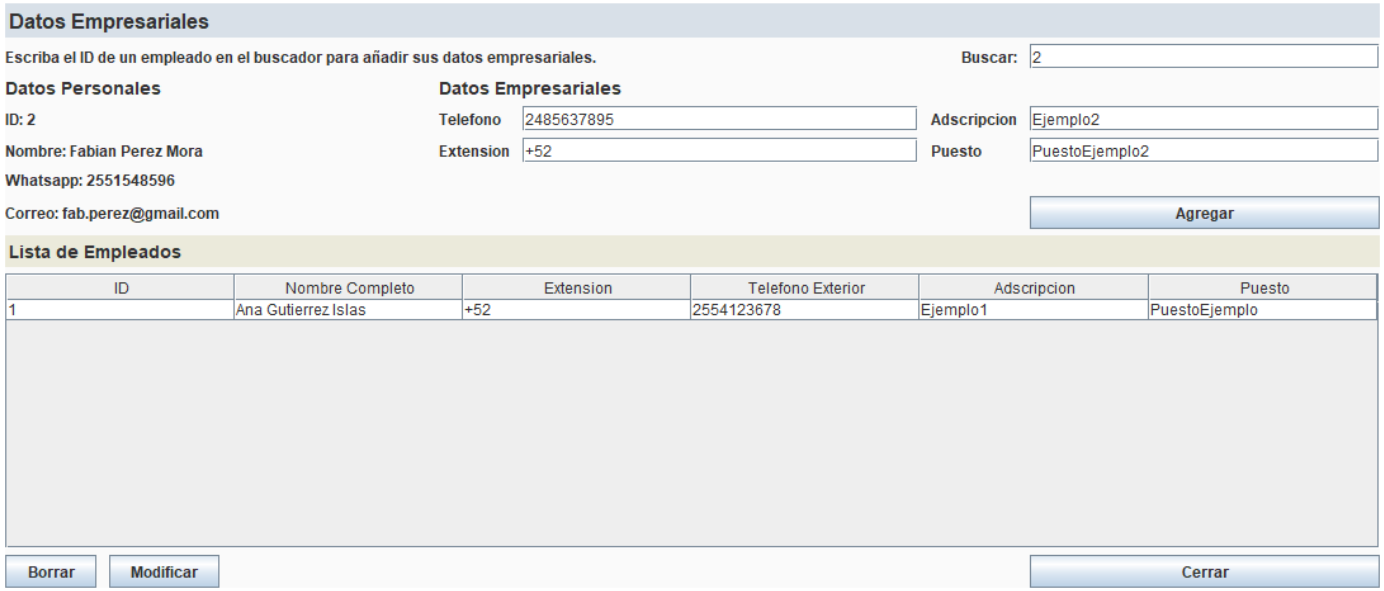
```
public ControlArchivos(){
    this.archivo = archivo;
}
public boolean crear(){
    File file = new File(archivo);
    try {
        if (file.exists()) {
            System.out.println("Error: El archivo ya existe.");
            return false;
        } else {
            if (file.createNewFile()) {
                System.out.println("Archivo creado exitosamente.");
                return true;
            } else {
                System.out.println("Error: No se pudo crear el archivo.");
                return false;
            }
        }
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
        return false;
    }
}
public boolean eliminar(){
    File file = new File(archivo);
    try {
        if (file.exists()) {
            // Confirmación de eliminación
            // Cerrar y eliminar si es posible
            if (file.delete()) {
                System.out.println("Archivo eliminado exitosamente.");
                return true;
            } else {
                System.out.println("Error: No se pudo eliminar el archivo.");
                return false;
            }
        } else {
            System.out.println("Error: El archivo no existe.");
            return false;
        }
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
```

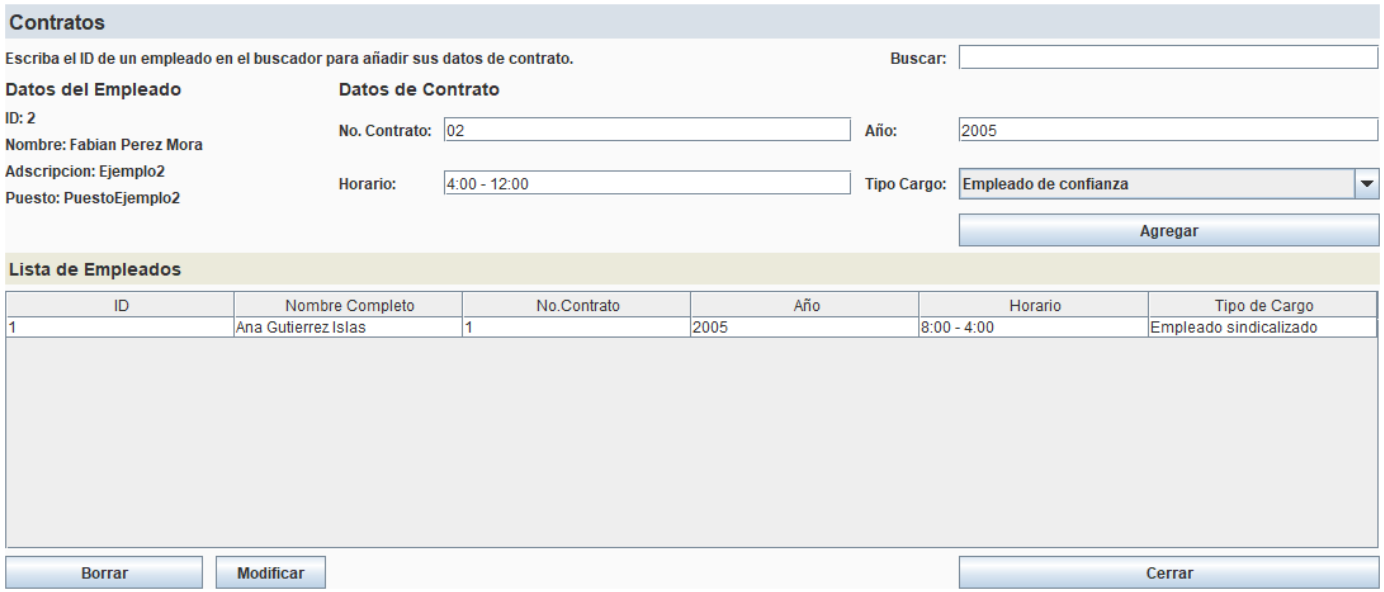
```
            return false;
        }
    }
    public boolean cambiar(String archivo){
        ArchivoTexto obj = new ArchivoTexto(archivo);
        obj.abrirModoEscritura();
        return true;
    }
    public boolean mover(String archivo){
        File file = new File(archivo);
        try {
            String nuevaUbicacion = null;
            File nuevaUbicacionFile = new File(nuevaUbicacion);
            if (file.exists()) {
                if (file.renameTo(nuevaUbicacionFile)) {
                    System.out.println("Archivo movido exitosamente.");
                    return true;
                } else {
                    System.out.println("Error: No se pudo mover el archivo.");
                    return false;
                }
            } else {
                System.out.println("Error: El archivo no existe.");
                return false;
            }
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            return false;
        }
    }
}

package gestor.archivos;

interface iFileText {
    public void abrirModoLectura();
    public String leer();
    public void abrirModoEscritura();
    public void escribir(String a);
    public void cerrar();
}
```


Usuario
admin
Contraseña
Cerrar
Ingresar


Cerrar Sesion
Datos Personales
Datos Empresariales
Contratos
itera
it & business process


Datos Personales
ID
2
Nombre Completo
Fabian Perez Mora
WhatsApp
2551548596
Correo Electronico
fab.perez@gmail.com
Agregar
Datos Guardados
ID
Nombre Completo
Whatsapp
Email
1
Ana Gutierrez Islas
2481524598
ana@gmail.com
Borrar
Modificar
Cerrar


Datos Empresariales
Escriba el ID de un empleado en el buscador para añadir sus datos empresariales.
Buscar: 2
Datos Personales
ID: 2
Nombre: Fabian Perez Mora
Whatsapp: 2551548596
Correo: fab.perez@gmail.com
Datos Empresariales
Telefono 2485637895
Extension +52
Adscripcion Ejemplo2
Puesto PuestoEjemplo2
Agregar
Lista de Empleados
ID | Nombre Completo | Extension | Telefono Exterior | Adscripcion | Puesto
1 | Ana Gutierrez Islas | +52 | 2554123678 | Ejemplo1 | PuestoEjemplo
Borrar
Modificar
Cerrar


Contratos
Escriba el ID de un empleado en el buscador para añadir sus datos de contrato.
Buscar:
Datos del Empleado
ID: 2
Nombre: Fabian Perez Mora
Adscripcion: Ejemplo2
Puesto: PuestoEjemplo2
Datos de Contrato
No. Contrato: 02
Año: 2005
Horario: 4:00 - 12:00
Tipo Cargo: Empleado de confianza
Agregar
Lista de Empleados
ID | Nombre Completo | No.Contrato | Año | Horario | Tipo de Cargo
1 | Ana Gutierrez Islas | 1 | 2005 | 8:00 - 4:00 | Empleado sindicalizado
Borrar
Modificar
Cerrar

Enlace web GitHub

https://github.com/BeMoBu20/EMTSystem.git

Enlace web YouTube

https://youtu.be/e0dhvGRd-q4