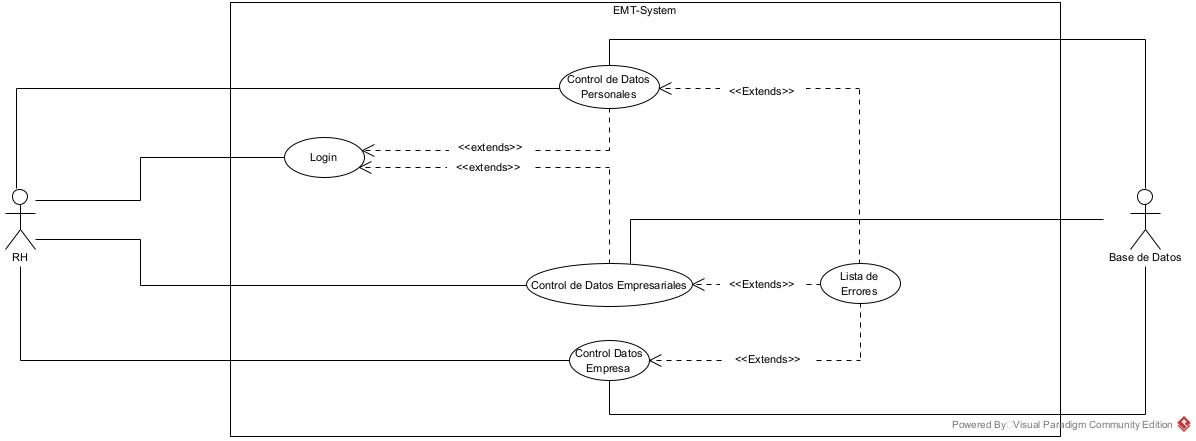
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Carrera:** | Ingeniería en Tecnologías de la Información | | | | **Cuatrimestre:** | | Enero-Abril 2024 | | | | **Fecha:** | 08/04/2024 |
| **Nombre del Estudiante:** | | | Berenice 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. | | | | | | | | | | | | |

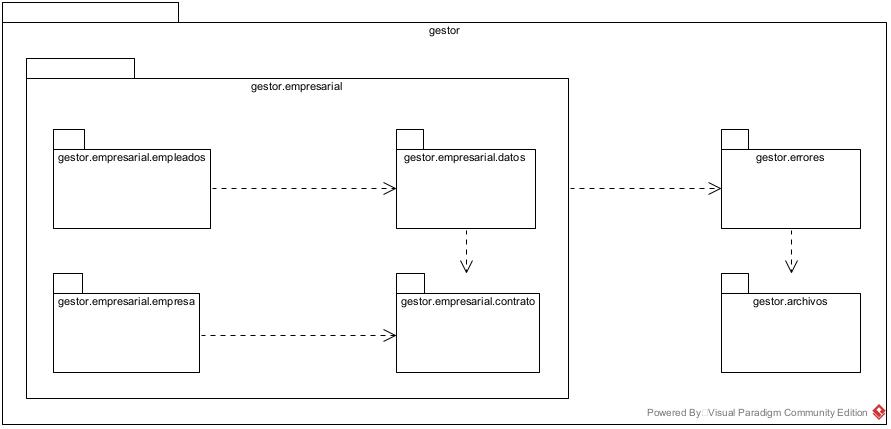
|  |  |
| --- | --- |
| televisión Special Lineal color icono | Redacción del Escenario |

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.

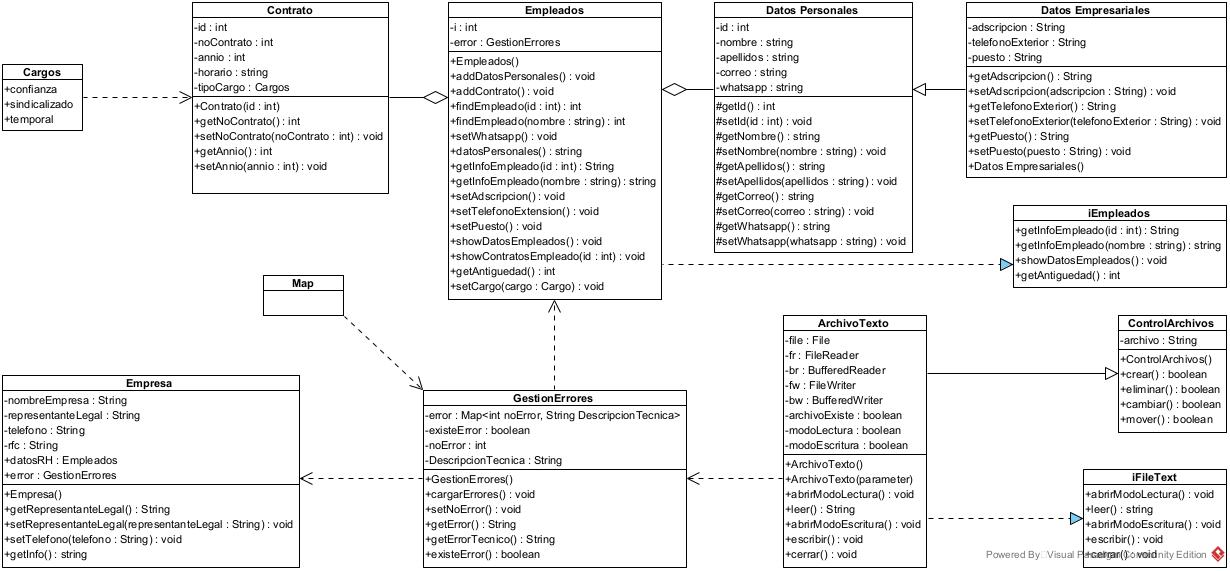
|  |  |
| --- | --- |
| https://lh4.googleusercontent.com/TMS7CCGC4H3tgb9zAL0AjAYAjBIM_bH7eKHsOUKHuHanKFE55F1CxwfylIvzYvbR3VEw7sthqhjr8my3Lbz6ULk=w1280 | Modelado: Diagramas de Casos de Uso |



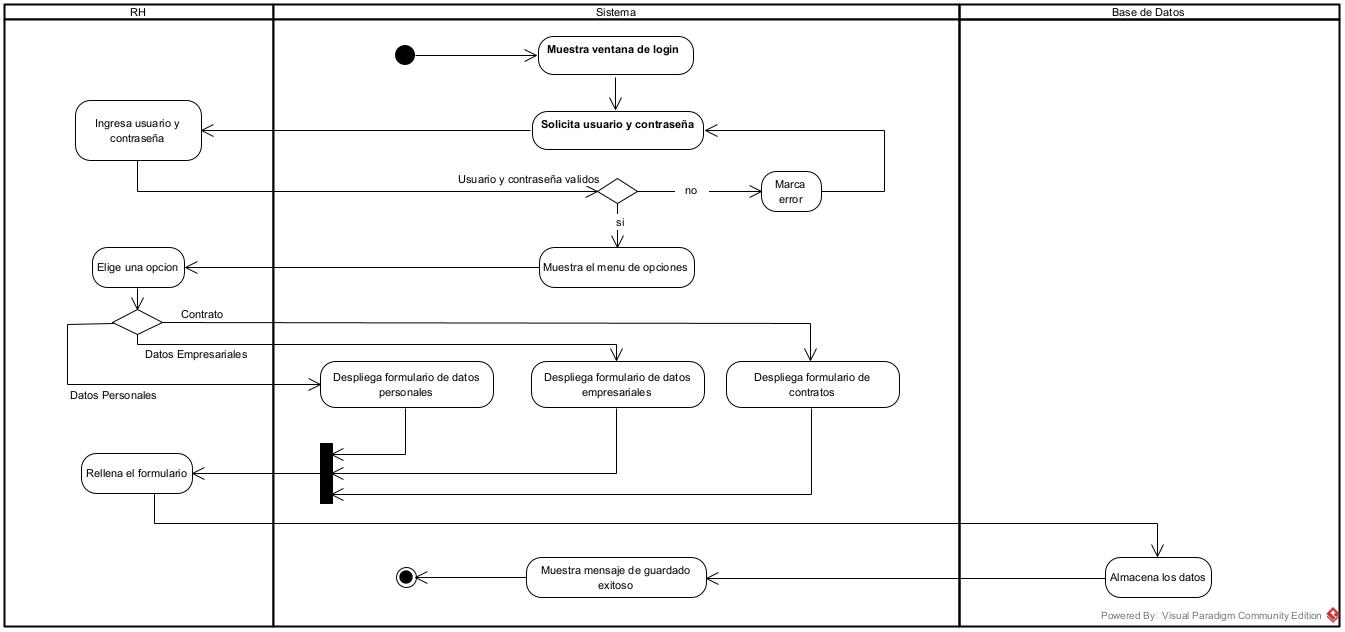
|  |  |
| --- | --- |
| distribución icono | Modelado: Diagrama: Paquetes |



|  |  |
| --- | --- |
| mapa del sitio Monochrome Blue icono | Modelado: Diagramas de Clase |



|  |  |
| --- | --- |
| Diagrama de flujo Ultimatearm Lineal Color icono | Modelado: Diagramas de Actividades |



|  |  |
| --- | --- |
| https://lh5.googleusercontent.com/nuxKncjaFtEwTkfc_e8KXfjm8MFsKmbNyAVJ-7j_iSX-VKfNBJu5JL1Q3nkAKFZmZDVR31WIYBY2CJ2jDAgNbBE=w1280 | Modelado: Diagramas de Secuencia |

|  |  |
| --- | --- |
| java Basic Rounded Flat icono | 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: 08 de abril del 2024 - 17:00 - 22:24*

package gestor.archivos;  
  
interface iFileText {  
 public void abrirModoLectura();  
 public String leer();  
 public void abrirModoEscritura();  
 public void escribir(String a);  
 public void cerrar();  
}

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()) {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.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;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public final class Contrato {  
 private static Contrato *instancia* = null;  
  
 *//Creamos listas para guardar nuestros datos* private List<Integer> id;  
 private List<String> nombre;  
 private List<Integer> noContrato;  
 private List<Integer> annio;  
 private List<String> horario;  
 private List<Cargos> tipoCargo;  
 List<String> nombresCargos = new ArrayList<>();  
  
 public Contrato(){  
 *//Inicializamos nuestras listas* id = new ArrayList<>();  
 nombre = new ArrayList<>();  
 noContrato = new ArrayList<>();  
 annio = new ArrayList<>();  
 horario = new ArrayList<>();  
 tipoCargo = new ArrayList<>();  
 }  
  
 public static Contrato getInstancia() {  
 if (*instancia* == null) {  
 *instancia* = new Contrato();  
 }  
 return *instancia*;  
 }  
  
 public void addDatos(int id, String nombre,int noContrato, int annio, String horario, Cargos tipoCargo) {  
 *//Con esta clase agregamos los datos a las listas* this.id.add(id);  
 this.nombre.add(nombre);  
 this.noContrato.add(noContrato);  
 this.annio.add(annio);  
 this.horario.add(horario);  
 this.tipoCargo.add(tipoCargo);  
 }  
  
 public void imprimirDatos() {  
 System.*out*.println("Datos guardados en Contrato:");  
 for (int i = 0; i < id.size(); i++) {  
 System.*out*.println("id: " + id.get(i) + ", Nombre: " + nombre.get(i) + ", noContrato: " + noContrato.get(i));  
 }  
 }  
  
 public List<Integer> getId() {  
 return id;  
 }  
  
 public List<String> getNombre() {  
 return nombre;  
 }  
  
 public List<Integer> getNoContrato() {  
 return noContrato;  
 }  
  
 public List<Integer> getAnnio() {  
 return annio;  
 }  
  
 public List<String> getHorario() {  
 return horario;  
 }  
  
 public List<String> getTipoCargo() {  
 for (Cargos cargo : tipoCargo) {  
 nombresCargos.add(cargo.toString());  
 }  
 return nombresCargos;  
 }  
}

package gestor.empresarial.contrato;  
  
import gestor.IntMenu;  
import gestor.empresarial.datos.DatosEmpresariales;  
  
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;  
import java.util.List;  
  
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 fieldBuscar;  
 private JComboBox comboCargo;  
 private JButton borrarButton;  
 private JButton modificarButton;  
 private JLabel labeIId;  
 private JLabel labelName;  
 private JButton cerrarButton;  
 private JLabel labelAdscripcion;  
 private JLabel labelPuesto;  
 private JTextPane contratosTextPane;  
 private JTextPane listaDeEmpleadosTextPane;  
 DefaultTableModel mt = new DefaultTableModel(); *//Creamos modelo de la tabla* private DatosEmpresariales datosEmpresariales; *//Generamos un objeto tipo DatosEmpresariales* private Contrato contrato; *//Generamos un objeto tipo Contrato* private int idEncontrado = 0;  
 private String nombreEncontrado = "";  
  
  
 public IntContratos(){  
 datosEmpresariales = DatosEmpresariales.*getInstancia*();  
 contrato = Contrato.*getInstancia*();  
  
 *//Ajustamos los parámetros de la ventana* setTitle("Menu EMT-System"); *//Establecemos el título de la ventana* this.setSize(1100,500); *//Establecemos el tamaño de la ventana* this.setLocationRelativeTo(null); *//Establecemos la posición 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 ejecución del programa al cerrar 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(); *//Código que define las funcionalidades de los botones* actualizarTablaDesdeContrato(); *//Código para obtener los datos de la tabla* }  
  
 private void initComponents() {  
 String encabezados[] = {"ID","Nombre Completo","No.Contrato", "Año","Horario","Tipo de Cargo"};  
 mt.setColumnIdentifiers(encabezados);  
 tablaC.getTableHeader().setResizingAllowed(false);  
 tablaC.getTableHeader().setReorderingAllowed(false);  
 tablaC.setModel(mt);  
 }  
  
 private void obtenerYGuardarContrato() {  
 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 DatosEmpresariales* contrato.addDatos(idEncontrado, nombreEncontrado, noContrato, annio, horario, tipocargo);  
 contrato.imprimirDatos();  
 idEncontrado = -1;  
 nombreEncontrado = "";  
 }  
  
 private void actualizarTablaDesdeContrato() {  
 contrato.imprimirDatos();  
 *//Obtenemos los datos de las listas en Contrato* List<Integer> ids = contrato.getId();  
 List<String> nombres = contrato.getNombre();  
 List<Integer> noContrato = contrato.getNoContrato();  
 List<Integer> annio = contrato.getAnnio();  
 List<String> horario = contrato.getHorario();  
 List<String> tipoCargo = contrato.getTipoCargo();  
  
 *// 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 < ids.size(); i++) {  
 mt.addRow(new Object[]{ids.get(i), nombres.get(i), noContrato.get(i), annio.get(i), horario.get(i), tipoCargo.get(i)});  
 }  
 }  
  
 public void funcionesBotones() {  
 *// 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, 3);  
 Object annio = tablaC.getValueAt(selectedRow, 2);  
 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);  
 }  
 }  
 }  
 });  
  
 fieldBuscar.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String textoBusqueda = fieldBuscar.getText();  
 if (textoBusqueda != null){  
 *// Buscando el ID ingresado en la lista de IDs en DatosPersonales* int idBuscado = Integer.*parseInt*(textoBusqueda); *// Convertir a entero* int indice = datosEmpresariales.getIds().indexOf(idBuscado);  
  
 *// Verificando si se encontró el ID* if (indice != -1) {  
 *// Obteneniendo la información relacionada al ID (nombre, adscripcion, puesto) en la lista en DatosEmpresariales* idEncontrado = idBuscado;  
 nombreEncontrado = datosEmpresariales.getNombres().get(indice);  
 String adscripcion = datosEmpresariales.getAdscripcion().get(indice);  
 String puesto = datosEmpresariales.getPuesto().get(indice);  
  
 *// Mostrando la información en la ventana* labeIId.setText("ID: "+ idEncontrado);  
 labelName.setText("Nombre: " + nombreEncontrado);  
 labelAdscripcion.setText("Adscripcion: " + adscripcion);  
 labelPuesto.setText("Puesto: " + puesto);  
 fieldBuscar.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*);  
 }  
 }  
 });  
  
 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 (idEncontrado != -1 && nombreEncontrado != ""){  
 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);  
 labeIId.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) {  
 mt.removeRow(selectedRow); *//Eliminamos la fila en la tabla  
  
 // Eliminamos los datos correspondientes en Contrato* contrato.getId().remove(selectedRow);  
 contrato.getNombre().remove(selectedRow);  
 contrato.getNoContrato().remove(selectedRow);  
 contrato.getAnnio().remove(selectedRow);  
 contrato.getHorario().remove(selectedRow);  
 contrato.getTipoCargo().remove(selectedRow);  
 } 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  
 // Obtenemos los datos modificados* String noContrato = fieldContrato.getText();  
 String annio = 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, 3);  
 tablaC.setValueAt(annio, selectedRow, 2);  
 tablaC.setValueAt(horario, selectedRow, 4);  
 tablaC.setValueAt(tipoCargo, selectedRow, 5);  
  
 *// Actualizamos los datos en Contrato* contrato.getNoContrato().set(selectedRow, Integer.*valueOf*(noContrato));  
 contrato.getAnnio().set(selectedRow, Integer.*valueOf*(annio));  
 contrato.getHorario().set(selectedRow, horario);  
 contrato.getTipoCargo().set(selectedRow, String.*valueOf*(tipoCargo));  
 }  
 }  
 });  
 }  
}

package gestor.empresarial.datos;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public final class DatosEmpresariales{  
 private static DatosEmpresariales *instancia* = null;  
  
 *//Creamos listas para guardar nuestros datos* private List<Integer> ids;  
 private List<String> nombres;  
 private List<String> extension;  
 private List<String> telefonoExterior;  
 private List<String> adscripcion;  
 private List<String> puesto;  
  
 public DatosEmpresariales(){  
 *//Inicializamos nuestras listas* ids = new ArrayList<>();  
 nombres = new ArrayList<>();  
 extension = new ArrayList<>();  
 telefonoExterior = new ArrayList<>();  
 adscripcion = new ArrayList<>();  
 puesto = new ArrayList<>();  
 }  
  
 public static DatosEmpresariales getInstancia() {  
 if (*instancia* == null) {  
 *instancia* = new DatosEmpresariales();  
 }  
 return *instancia*;  
 }  
  
 public void addDatos(int ids, String nombres,String extension, String telefonoExterior, String adscripcion, String puesto) {  
 *//Con esta clase agregamos los datos a las listas* this.ids.add(ids);  
 this.nombres.add(nombres);  
 this.extension.add(extension);  
 this.telefonoExterior.add(telefonoExterior);  
 this.adscripcion.add(adscripcion);  
 this.puesto.add(puesto);  
 }  
  
 public void imprimirDatos() {  
 System.*out*.println("Datos guardados en DatosPersonales:");  
 for (int i = 0; i < puesto.size(); i++) {  
 System.*out*.println("Telefono exterior: " + telefonoExterior.get(i) + ", Adscripcion: " + adscripcion.get(i) + ", Puesto: " + puesto.get(i));  
 }  
 }  
  
 public List<Integer> getIds() {  
 return ids;  
 }  
 public List<String> getNombres() {  
 return nombres;  
 }  
 public List<String> getExtension(){  
 return extension;  
 }  
 public List<String> getTelefonoExterior() {  
 return telefonoExterior;  
 }  
 public List<String> getAdscripcion() {  
 return adscripcion;  
 }  
 public List<String> getPuesto() {  
 return puesto;  
 }  
}

package gestor.empresarial.datos;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class DatosPersonales {  
 private static DatosPersonales *instancia* = null;  
  
 *//En las listas es posible guardar n cantidad de datos, al ser una tabla es mas conveniente* private List<Integer> id; *//Lista de ids* private List<String> nombre; *//Lista de nombres* private List<String> whatsapp; *//Lista de numeros de whatsapp* private List<String> correo; *//Lista de correos electronicos* public DatosPersonales() {  
 *// Aqui inicializamos las listas* id = new ArrayList<>();  
 nombre = new ArrayList<>();  
 whatsapp = new ArrayList<>();  
 correo = new ArrayList<>();  
 }  
  
 public static DatosPersonales getInstancia() {  
 if (*instancia* == null) {  
 *instancia* = new DatosPersonales();  
 }  
 return *instancia*;  
 }  
  
 public void addDatos(int id, String nombre, String whatsapp, String email) {  
 *//Con esta clase agregamos los datos a las listas* this.id.add(id);  
 this.nombre.add(nombre);  
 this.whatsapp.add(whatsapp);  
 this.correo.add(email);  
 }  
  
 public void imprimirDatos() {  
 System.*out*.println("Datos guardados en DatosPersonales:");  
 for (int i = 0; i < id.size(); i++) {  
 System.*out*.println("ID: " + id.get(i) + ", Nombre: " + nombre.get(i) + ", WhatsApp: " + whatsapp.get(i) + ", Email: " + correo.get(i));  
 }  
 }  
  
 *//Metodos de acceso a los valores de las listas* public List<Integer> getId() {  
 return id;  
 }  
  
 public List<String> getNombre() {  
 return nombre;  
 }  
  
 public List<String> getCorreo() {  
 return correo;  
 }  
  
 public List<String> getWhatsapp() {  
 return whatsapp;  
 }  
}

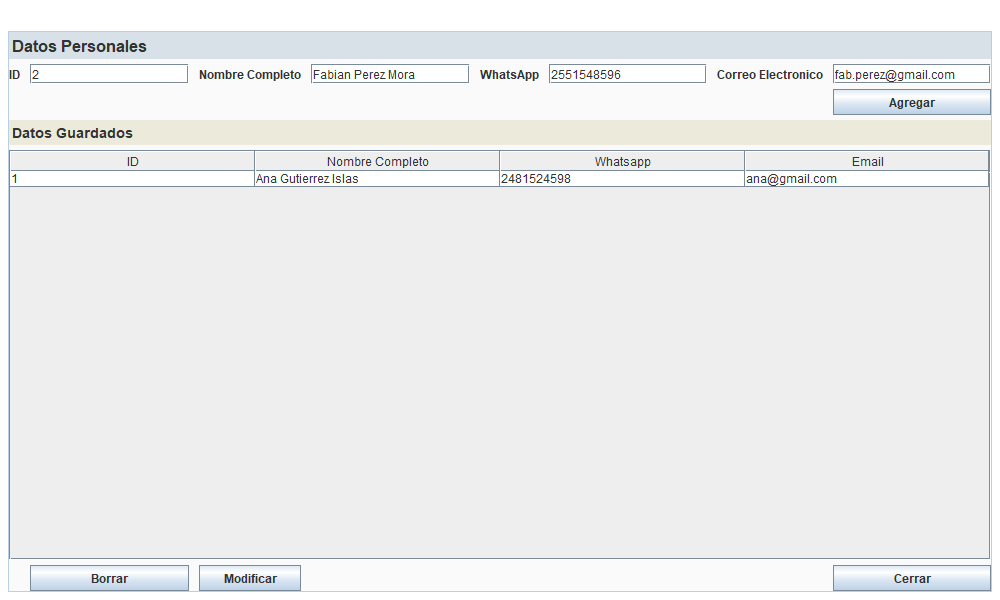
package gestor.empresarial.datos;  
  
import gestor.IntMenu;  
  
import javax.swing.\*;  
import javax.swing.event.ListSelectionEvent;  
import javax.swing.event.ListSelectionListener;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.List;  
  
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 DatosPersonales datosPersonales; *//Generamos un objeto tipo DatosPersonales* public IntMenuDos(){  
 datosPersonales = DatosPersonales.*getInstancia*();  
  
 *//Ajustamos los parametros de nuestra ventana* 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* initComponents(); *//Ajustes de la tabla* funcionesBotones(); *//Codigo de las funcionalidades de los botones* actualizarTablaDesdeDatosPersonales(); *//Codigo para obtener los de la tabla* }  
  
 private void initComponents() {  
 String ids[] = {"ID","Nombre Completo","Whatsapp","Email"};  
 mt.setColumnIdentifiers(ids);  
 tablaDP.getTableHeader().setResizingAllowed(false);  
 tablaDP.getTableHeader().setReorderingAllowed(false);  
 tablaDP.setModel(mt);  
 }  
  
 private void obtenerYGuardarDatosPersonales() {  
 int id = Integer.*parseInt*(fieldID.getText());  
 String nombre = fieldName.getText();  
 String whatsapp = fieldWhats.getText();  
 String email = fieldMail.getText();  
  
 *// Guardamos los datos en DatosPersonales* datosPersonales.addDatos(id, nombre, whatsapp, email);  
 datosPersonales.imprimirDatos();  
 }  
  
 private void actualizarTablaDesdeDatosPersonales() {  
 datosPersonales.imprimirDatos();  
 *//Obtenemos los datos de las listas en DatosPersonales* List<Integer> ids = datosPersonales.getId();  
 List<String> nombres = datosPersonales.getNombre();  
 List<String> whatsapps = datosPersonales.getWhatsapp();  
 List<String> emails = datosPersonales.getCorreo();  
  
 *// 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 < ids.size(); i++) {  
 mt.addRow(new Object[]{ids.get(i), nombres.get(i), whatsapps.get(i), emails.get(i)});  
 }  
 }  
  
 public void funcionesBotones(){  
 *// Agregar un ListSelectionListener a la JTable* tablaDP.getSelectionModel().addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 if (!e.getValueIsAdjusting()) { *// Evitar eventos de selección múltiple* int selectedRow = tablaDP.getSelectedRow();  
 if (selectedRow != -1) { *// Verificar si se seleccionó una fila  
 // Obtener 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();  
 *// Verificar que ningún campo esté vacío* if (idText.isEmpty() || nombre.isEmpty() || wh.isEmpty() || correo.isEmpty()) {  
 *// Mostrar un mensaje de error indicando al usuario qué campo olvidó rellenar* String mensaje = "Por favor, complete todos los campos:\n";  
 if (idText.isEmpty()) {  
 mensaje += "- ID\n";  
 }  
 if (nombre.isEmpty()) {  
 mensaje += "- Nombre\n";  
 }  
 if (wh.isEmpty()) {  
 mensaje += "- Whatsapp\n";  
 }  
 if (correo.isEmpty()) {  
 mensaje += "- Email\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* int id = Integer.*parseInt*(idText);  
 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) {  
 mt.removeRow(selectedRow); *// Eliminamos la fila en la tabla  
  
 // Eliminamos los datos correspondientes en DatosPersonales* datosPersonales.getId().remove(selectedRow);  
 datosPersonales.getNombre().remove(selectedRow);  
 datosPersonales.getWhatsapp().remove(selectedRow);  
 datosPersonales.getCorreo().remove(selectedRow);  
 } else {  
 JOptionPane.*showMessageDialog*(null, "No se ha seleccionado ningún candidato para borrar", "Error", 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  
 // Obtenemos los datos modificados* int id = Integer.*parseInt*(fieldID.getText());  
 String nombre = fieldName.getText();  
 String wh = fieldWhats.getText();  
 String correo = fieldMail.getText();  
  
 *// Actualizamos los datos en la fila seleccionada de la JTable* tablaDP.setValueAt(id, selectedRow, 0);  
 tablaDP.setValueAt(nombre, selectedRow, 1);  
 tablaDP.setValueAt(wh, selectedRow, 2);  
 tablaDP.setValueAt(correo, selectedRow, 3);  
  
 *// Actualizamos los datos en DatosPersonales* datosPersonales.getId().set(selectedRow, id);  
 datosPersonales.getNombre().set(selectedRow, nombre);  
 datosPersonales.getWhatsapp().set(selectedRow, wh);  
 datosPersonales.getCorreo().set(selectedRow, correo);  
 }  
 else {  
 JOptionPane.*showMessageDialog*(null, "No se ha seleccionado ningún candidato para modificar", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 });  
 }  
}

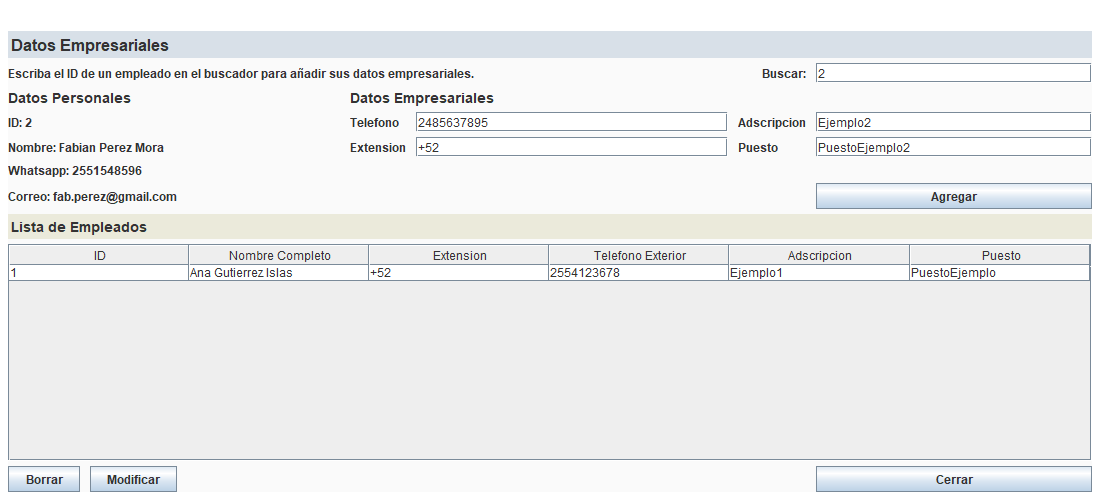
package gestor.empresarial.datos;  
  
import gestor.IntMenu;  
  
import javax.swing.\*;  
import javax.swing.event.ListSelectionEvent;  
import javax.swing.event.ListSelectionListener;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.List;  
  
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 tablaEmpleados;  
 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 DatosEmpresariales datosEmpresariales; *//Generamos un objeto tipo DatosPersonales* private DatosPersonales datosPersonales; *//Generamos un objeto tipo DatosPersonales* private int idEncontrado = 0;  
 private String nombreEncontrado = "";  
  
  
 public IntMenuTres(){  
 datosEmpresariales = DatosEmpresariales.*getInstancia*();  
 datosPersonales = DatosPersonales.*getInstancia*();  
  
 *//Ajustamos los parametros de la ventana* setTitle("Menu EMT-System"); *//Estabalecemos 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* initComponents(); *//Ajustes de la tabla* funcionesBotones(); *//Codigo que define las funcionalidades de los botones* actualizarTablaDesdeDatosEmpresariales(); *//Codigo para obtener los datos de la tabla* }  
  
 private void initComponents() {  
 String ids[] = {"ID","Nombre Completo","Extension", "Telefono Exterior","Adscripcion","Puesto"};  
 mt.setColumnIdentifiers(ids);  
 tablaEmpleados.getTableHeader().setResizingAllowed(false);  
 tablaEmpleados.getTableHeader().setReorderingAllowed(false);  
 tablaEmpleados.setModel(mt);  
 }  
  
 private void obtenerYGuardarDatosEmpresariales() {  
 String telefono = fieldTelefono.getText();  
 String extension = fieldExtension.getText();  
 String adscripcion = fieldAdscripcion.getText();  
 String puesto = fieldPuesto.getText();  
  
 *// Guardamos los datos en DatosEmpresariales* datosEmpresariales.addDatos(idEncontrado, nombreEncontrado, extension, telefono, adscripcion, puesto);  
 datosEmpresariales.imprimirDatos();  
 idEncontrado = -1;  
 nombreEncontrado = "";  
 }  
  
 private void actualizarTablaDesdeDatosEmpresariales() {  
 datosEmpresariales.imprimirDatos();  
 *//Obtenemos los datos de las listas en DatosEmpresariales* List<Integer> ids = datosEmpresariales.getIds();  
 List<String> nombres = datosEmpresariales.getNombres();  
 List<String> extensiones = datosEmpresariales.getExtension();  
 List<String> telefonos = datosEmpresariales.getTelefonoExterior();  
 List<String> adscripciones = datosEmpresariales.getAdscripcion();  
 List<String> puesto = datosEmpresariales.getPuesto();  
  
 *// 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 < telefonos.size(); i++) {  
 mt.addRow(new Object[]{ids.get(i), nombres.get(i), extensiones.get(i), telefonos.get(i), adscripciones.get(i), puesto.get(i)});  
 }  
 }  
  
 public void funcionesBotones() {  
 *// Agregar un ListSelectionListener a la JTable* tablaEmpleados.getSelectionModel().addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 if (!e.getValueIsAdjusting()) { *// Evitar eventos de selección múltiple* int selectedRow = tablaEmpleados.getSelectedRow();  
 if (selectedRow != -1) { *// Verificar si se seleccionó una fila  
 // Obtener datos de la fila seleccionada* Object telefono = tablaEmpleados.getValueAt(selectedRow, 3);  
 Object extension = tablaEmpleados.getValueAt(selectedRow, 2);  
 Object adscripcion = tablaEmpleados.getValueAt(selectedRow, 4);  
 Object puesto = tablaEmpleados.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());  
 }  
 }  
 }  
 });  
  
 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 DatosPersonales* int idBuscado = Integer.*parseInt*(textoBusqueda); *// Convertir a entero* int indice = datosPersonales.getId().indexOf(idBuscado);  
  
 *// Verificando si se encontró el ID* if (indice != -1) {  
 *// Obteneniendo la información relacionada al ID (nombre, whatsapp, correo) en la lista en DatosPersonales* idEncontrado = idBuscado;  
 nombreEncontrado = datosPersonales.getNombre().get(indice);  
 String whatsapp = datosPersonales.getWhatsapp().get(indice);  
 String correo = datosPersonales.getCorreo().get(indice);  
  
 *// Mostrando la información en la ventana* idEmpleado.setText("ID: "+ idBuscado);  
 nombreEmpleado.setText("Nombre: " + nombreEncontrado);  
 whatsEmpleado.setText("Whatsapp: " + whatsapp);  
 correoEmpleado.setText("Correo: " + correo);  
 fieldBuscador.setText("");  
 } else {  
 *// Mostrar un mensaje de error si no se encuentra el ID* JOptionPane.*showMessageDialog*(IntMenuTres.this, "ID no encontrado", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else {  
 *// Mostrar un mensaje de error si esta vacio el campo* JOptionPane.*showMessageDialog*(IntMenuTres.this, "Campo de busqueda vacio", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 });  
  
 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 (idEncontrado != -1 && nombreEncontrado != ""){  
 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 = "Por favor, complete todos los campos:\n";  
 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, procede a agregar la fila a la tabla* obtenerYGuardarDatosEmpresariales();  
 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{  
 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 = tablaEmpleados.getSelectedRow();  
 if (selectedRow != -1) {  
 mt.removeRow(selectedRow); *//Eliminamos la fila en la tabla  
  
 // Eliminamos los datos correspondientes en DatosPersonales* datosEmpresariales.getIds().remove(selectedRow);  
 datosEmpresariales.getNombres().remove(selectedRow);  
 datosEmpresariales.getExtension().remove(selectedRow);  
 datosEmpresariales.getTelefonoExterior().remove(selectedRow);  
 datosEmpresariales.getAdscripcion().remove(selectedRow);  
 datosEmpresariales.getPuesto().remove(selectedRow);  
 } 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 = tablaEmpleados.getSelectedRow();  
 if (selectedRow != -1) { *// Verificamos si se seleccionó una fila  
 // 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* tablaEmpleados.setValueAt(telefono, selectedRow, 3);  
 tablaEmpleados.setValueAt(extension, selectedRow, 2);  
 tablaEmpleados.setValueAt(adscripcion, selectedRow, 4);  
 tablaEmpleados.setValueAt(puesto, selectedRow, 5);  
  
 *// Actualizamos los datos en DatosEmpresarial* datosEmpresariales.getExtension().set(selectedRow, extension);  
 datosEmpresariales.getTelefonoExterior().set(selectedRow, telefono);  
 datosEmpresariales.getAdscripcion().set(selectedRow, adscripcion);  
 datosEmpresariales.getPuesto().set(selectedRow, puesto);  
 }  
 }  
 });  
 }  
}

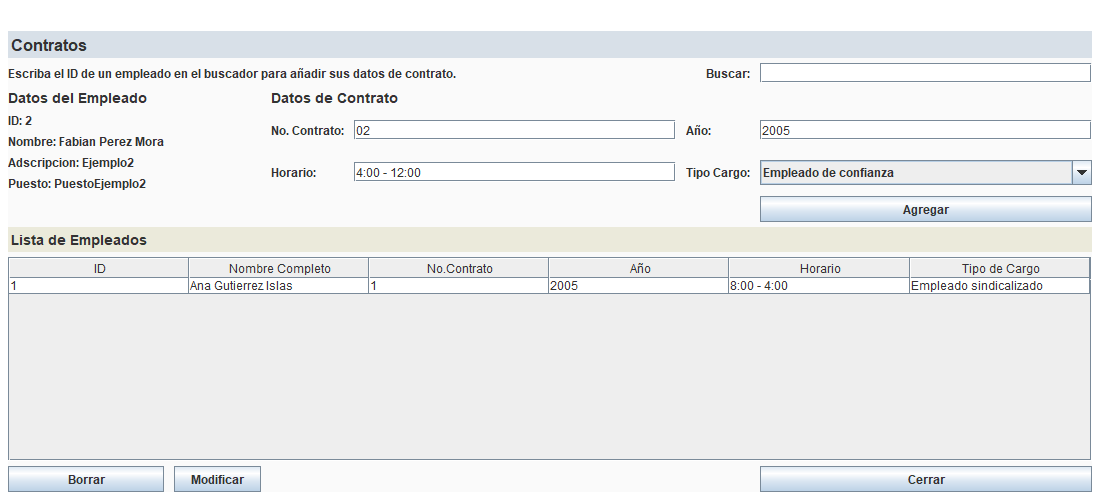
package gestor;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class IntLogin extends JFrame {  
 private JPanel panel1;  
 private JTextField usuarioTextField;  
 private JPasswordField cPass;  
 private JButton cerrarB;  
 private JButton ingresarButton;  
 private JLabel etUser;  
 private JLabel etPass;  
 private JLabel imagUser;  
 private JPasswordField pass;  
 *//boton1.setEnabled(true) para que sirva y false para que no se pueda clikear* public IntLogin(){  
 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();  
  
 boolean accesoValido = usuario.equals("admin") && contrasenia.equals("123");  
  
 if (accesoValido) {  
 IntMenu obj = new IntMenu();  
 dispose();  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Usuario o contraseña incorrectos");  
 }  
 }  
 });  
 }  
}

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.\*;  
  
public class IntMenu extends JFrame{  
 private JPanel panel1;  
 private JButton cerrarSesionButton;  
 private JButton datosPersonalesButton;  
 private JButton datosEmpresarialesButton;  
 private JButton contratosButton;  
 private DatosPersonales datosPersonales;  
 private DatosEmpresariales datosEmpresariales;  
 private Contrato contrato;  
  
 public IntMenu(){  
 datosEmpresariales = DatosEmpresariales.*getInstancia*();  
 datosPersonales = DatosPersonales.*getInstancia*();  
 setTitle("Menu EMT-System"); *//Estabalecemos 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* funcionesBotones();  
 }  
  
 private boolean datosPersonalesNoVacios() {  
 return !datosPersonales.getId().isEmpty(); *// Verificamos si ya hay informacion en DatosPersonales* }  
  
 private boolean datosEmpresarialesNoVacios() {  
 return !datosEmpresariales.getIds().isEmpty(); *// Verificamos si ya hay informacion en DatosEmpresariales* }  
  
 private boolean contratosNoVacios() {  
 *//return !contrato.getId().isEmpty(); //Verificamos si hay informacion en contratos* return 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) {  
 if (datosPersonalesNoVacios()) {  
 IntMenuTres obj = new IntMenuTres();  
 dispose();  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Aun no se han añadido datos personales", "Campos Vacíos", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 });  
  
 contratosButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (datosPersonalesNoVacios()) {  
 IntContratos obj = new IntContratos();  
 dispose();  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Aun no se han añadido datos en Datos Empresariales", "Campos Vacíos", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 });  
 }  
}import gestor.IntLogin;  
  
public class Principal {  
 public static void main(String[] args){  
 IntLogin lgin = new IntLogin();  
 }  
}

|  |  |
| --- | --- |
| dominio Generic color fill icono | Capturas de pantalla del programa en ejecución |







|  |  |
| --- | --- |
| github Pixel Perfect Flat icono | Enlace web GitHub |

https://github.com/BeMoBu20/EMT-System.git

|  |  |
| --- | --- |
| youtube Generic Flat icono | Enlace web YouTube |

|  |  |
| --- | --- |
| Código QR - Wikipedia, la enciclopedia libre | <https://youtu.be/qjbs1dW_gaE?si=Kx3LhFxV7cjLc-Je> |