

	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN/INGENIERÍA DE SISTEMAS		ASIGNATURA: PROGRAMACIÓN APLICADA	
NRO. PROYECTO:	1.1	TÍTULO PROYECTO: Prueba Practica 1 Desarrollo e implementación de un sistema de gestión de matrimonios de la ciudad de Cuenca	
OBJETIVO: Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (Java 8, Programación Genérica, Reflexión y Patrones de Diseño) en un contexto real.			
INSTRUCCIONES:		1. Revisar el contenido teórico y práctico del tema	
		2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje Java y la documentación disponible en fuentes académicas en línea.	
		3. Deberá desarrollar un sistema informático para la gestión de matrimonios, almacenar en archivos y una interfaz gráfica.	
		4. Deberá generar un informe de la practica en formato PDF y en conjunto con el código se debe subir al GitHub personal.	
		5. Fecha de entrega: El sistema debe ser subido al git hasta 27 de noviembre del 2020 – 23:55.	
ACTIVIDADES POR DESARROLLAR			

	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

1. Enunciado:

Realizar el diagrama de clase y el programa para gestionar los matrimonios de la ciudad de Cuenca empleando las diferentes técnicas de programación revisadas en clase.

Problema: De cada matrimonio se almacena la fecha, el lugar de la celebración y los datos personales (nombre, apellido, cédula, dirección, género y fecha de nacimiento) de los contrayentes. Es importante validar la equidad de género.

Igualmente se guardan los datos personales de los dos testigos y de la autoridad civil (juez o autoridad) que formalizan el acto. Además de gestionar la seguridad a través de un sistema de Usuarios y Autenticación.

Calificación:

- Diagrama de Clase 20%
- MVC: 20%
- Patrón de Diseño aplicado: 30%
- Técnicas de Programación aplicadas (Java 8, Reflexión y Programación Genérica): 20%
- Informe: 10%

2. Informe de Actividades:

- Planteamiento y descripción del problema.
- Diagramas de Clases.
- Patrón de diseño aplicado
- Descripción de la solución y pasos seguidos.
- Conclusiones y recomendaciones.
- Resultados.

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar la persistencia en archivos.

RECOMENDACIONES:


- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.**


BIBLIOGRAFIA:

[1]: <https://www.ups.edu.ec/evento?calendarBookingId=98892>

Docente / Técnico Docente: Ing. Diego Quisi Peralta Msc.

Firma: _____

	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

			FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES		
CARRERA: Computación			ASIGNATURA: Programación Aplicada		
NRO. PRÁCTICA:	1.1	TÍTULO PRÁCTICA: Prueba Practica 1 Desarrollo e implementación de un sistema de gestión de matrimonios de la ciudad de Cuenca			
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (Java 8, Programación Genérica, Reflexión y Patrones de Diseño) en un contexto real. Construir una app en la que se puedan ingresar datos de personas para realizar un matrimonio en el que se especifiquen la mayoría de datos de las personas a conocer. 					
ACTIVIDADES DESARROLLADAS					
1. Planteamiento y descripción del problema.					
2. Diagrama de clase.					
3. Patrón de diseño aplicado					
4. Descripción de la solución y pasos seguidos.					
APP PARA GESTION DE MATRIMONIOS Paquetes y clases que se emplearon para realizar la app. Lo primero que se debe hacer es implementar los paquetes que se van a necesitar cada clase. <ul style="list-style-type: none"> - Ec.edu.ups.modelo - Ec.edu.ups.controlador - Ec.edu.ups.vista Lo siguiente es asignar cada clase a su respectivo paquete y también la creación de ventanas. <ul style="list-style-type: none"> - Ec.edu.ups.modelo 					
Persona <pre> package ec.edu.ups.modelo; import java.util.Date; import java.util.Objects; /** * * @author paul_ */ public class Persona { private String cedula; private String nombres; private String apellidos; private String direccion; private String genero; private Date fechaDeNacimiento; private String estadoCivil; private String rol; public Persona() { } public Persona(String cedula, String nombres, String apellidos, String direccion, String genero, Date fechaDeNacimiento, String estadoCivil, String rol) { this.cedula = cedula; this.nombres = nombres; this.apellidos = apellidos; </pre>					

```
this.direccion = direccion;
this.genero = genero;
this.fechaDeNacimiento = fechaDeNacimiento;
this.estadoCivil = estadoCivil;
this.rol = rol;
}

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}

public String getNombres() {
    return nombres;
}

public void setNombres(String nombres) {
    this.nombres = nombres;
}

public String getApellidos() {
    return apellidos;
}

public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getGenero() {
    return genero;
}

public void setGenero(String genero) {
    this.genero = genero;
}

public Date getFechaDeNacimiento() {
    return fechaDeNacimiento;
}

public void setFechaDeNacimiento(Date fechaDeNacimiento) {
    this.fechaDeNacimiento = fechaDeNacimiento;
}

public String getEstadoCivil() {
    return estadoCivil;
}

public void setEstadoCivil(String estadoCivil) {
    this.estadoCivil = estadoCivil;
}
```

```
}

public String getRol() {
    return rol;
}

public void setRol(String rol) {
    this.rol = rol;
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 41 * hash + Objects.hashCode(this.cedula);
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Persona other = (Persona) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Persona{cedula=").append(cedula);
    sb.append(", nombres=").append(nombres);
    sb.append(", apellidos=").append(apellidos);
    sb.append(", direccion=").append(direccion);
    sb.append(", genero=").append(genero);
    sb.append(", fechaDeNacimiento=").append(fechaDeNacimiento);
    sb.append(", estadoCivil=").append(estadoCivil);
    sb.append(", rol=").append(rol);
    sb.append('}');
    return sb.toString();
}
```

Matrimonio

```
package ec.edu.ups.modelo;

import java.util.Date;

/**
 *
 * @author paul_
 */
public class Matrimonio {
```

```
private int numMatrimonio;
private String lugar;
private Date fecha;
private Persona contrayente1;
private Persona contrayente2;
private Persona testigo1;
private Persona testigo2;
private Persona autoridadCivil;

public Matrimonio() {
}

public Matrimonio(int numMatrimonio, String lugar, Date fecha, Persona contrayente1, Persona contrayente2, Persona testigo1, Persona testigo2, Persona autoridadCivil) {
    this.numMatrimonio = numMatrimonio;
    this.lugar = lugar;
    this.fecha = fecha;
    this.contrayente1 = contrayente1;
    this.contrayente2 = contrayente2;
    this.testigo1 = testigo1;
    this.testigo2 = testigo2;
    this.autoridadCivil = autoridadCivil;
}

public int getNumMatrimonio() {
    return numMatrimonio;
}

public void setNumMatrimonio(int numMatrimonio) {
    this.numMatrimonio = numMatrimonio;
}

public String getLugar() {
    return lugar;
}

public void setLugar(String lugar) {
    this.lugar = lugar;
}

public Date getFecha() {
    return fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

public Persona getContrayente1() {
    return contrayente1;
}

public void setContrayente1(Persona contrayente1) {
    this.contrayente1 = contrayente1;
}

public Persona getContrayente2() {
    return contrayente2;
}
```

```
public void setContrayente2(Persona contrayente2) {
    this.contrayente2 = contrayente2;
}

public Persona getTestigo1() {
    return testigo1;
}

public void setTestigo1(Persona testigo1) {
    this.testigo1 = testigo1;
}

public Persona getTestigo2() {
    return testigo2;
}

public void setTestigo2(Persona testigo2) {
    this.testigo2 = testigo2;
}

public Persona getAutoridadCivil() {
    return autoridadCivil;
}

public void setAutoridadCivil(Persona autoridadCivil) {
    this.autoridadCivil = autoridadCivil;
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 97 * hash + this.numMatrimonio;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Matrimonio other = (Matrimonio) obj;
    if (this.numMatrimonio != other.numMatrimonio) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Matrimonio{numMatrimonio=").append(numMatrimonio);
    sb.append(", lugar=").append(lugar);
    sb.append(", fecha=").append(fecha);
    sb.append(", contrayente1=").append(contrayente1);
```

```
sb.append(", contrayente2=").append(contrayente2);
sb.append(", testigo1=").append(testigo1);
sb.append(", testigo2=").append(testigo2);
sb.append(", autoridad=").append(autoridadCivil);
sb.append('}');
return sb.toString();
}

}

Autoridad Civil
package ec.edu.upes.modelo;

import java.util.Date;
import java.util.Objects;

/**
 *
 * @author paul_
 */
public class AutoridadCivil extends Persona {

    private String cargo;
    private String correo;
    private String pass;

    public AutoridadCivil() {

    }

    public AutoridadCivil(String cargo, String correo, String pass, String cedula,
        String nombres, String apellidos,
        String direccion, String genero, Date fechaDeNacimiento, String es-
        tadoCivil, String rol) {
        super(cedula, nombres, apellidos, direccion, genero, fechaDeNacimiento, es-
        tadoCivil, rol);
        this.cargo = cargo;
        this.correo = correo;
        this.pass = pass;
    }

    public AutoridadCivil(String trim, String trim0, String trim1, String trim2,
        String trim3, Date fechaN, String trim4, String autoridad, String trim5, String
        trim6, String trim7) {
        throw new UnsupportedOperationException("Not supported yet."); //To change
        body of generated methods, choose Tools | Templates.
    }

    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }
}
```



```
public String getPass() {
    return pass;
}

public void setPass(String contrasenia) {
    this.pass = contrasenia;
}
}

- Ec.edu.ups.controlador
Abstract Controlador
package ec.edu.ups.controlador;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

/**
 *
 * @author paul_
 */
public abstract class AbstractControlador<E> {

    private List<E> lista;
    public String ruta;

    public AbstractControlador(String ruta) {
        lista = new ArrayList();
        this.ruta=ruta;
        cargarDatos();
    }

    public List<E> getLista() {
        return lista;
    }

    public void setLista(List<E> lista) {
        this.lista = lista;
    }

    public void cargarDatos() {

        try{
            FileInputStream archivo = new FileInputStream(ruta);
            ObjectInputStream datos = new ObjectInputStream (archivo);
            lista =(List<E>) datos.readObject();
        }catch(ClassNotFoundException e) {
            e.printStackTrace();
        }catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
public void guardarDatos(String ruta) throws FileNotFoundException, IOException
{
    FileOutputStream archivo = new FileOutputStream(ruta);
    ObjectOutputStream datos = new ObjectOutputStream(archivo);
    datos.writeObject(lista);
}

public boolean crear(E objeto) {

    if (validar(objeto) == true) {
        return lista.add(objeto);
    }
    return false;

}

public Optional<E> buscar(E comparar) {
    return lista.stream().filter(objeto -> objeto.equals(comparar)).findFirst();
}

public boolean eliminar(E objeto) {
    Optional<E> buscar = buscar(objeto);
    E objetoE = buscar.get();
    if (objetoE != null) {
        System.out.println("Verdadero");
        return lista.remove(objetoE);
    }
    System.out.println("Falso");
    return false;
}

public boolean actualizar(E objetoA) {
    int pos = posicion(objetoA);
    if (pos >= 0) {
        lista.set(pos, objetoA);
        System.out.println("TRUE");
        return true;
    }
    System.out.println("FALSE");
    return false;
}

public int posicion(E objetoC) {
    for (int i = 0; i < lista.size(); i++) {
        E objetoL = lista.get(i);
        if (objetoL.equals(objetoC)) {
            return i;
        }
    }
    return -1;
}

public abstract boolean validar(E objeto);

public abstract int generarId();
```

Controlador Persona

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Persona;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorPersona extends AbstractControlador<Persona> {

    public ControladorPersona(String ruta) {
        super(ruta);
    }

    @Override
    public boolean validar(Persona objeto) {
        int suma = 0;
        String x = objeto.getCedula();
        if (x.length() == 9) {
            return false;
        } else {
            int a[] = new int[x.length() / 2];
            int b[] = new int[(x.length() / 2)];
            int c = 0;
            int d = 1;
            for (int i = 0; i < x.length() / 2; i++) {
                a[i] = Integer.parseInt(String.valueOf(x.charAt(c)));
                c = c + 2;
                if (i < (x.length() / 2) - 1) {
                    b[i] = Integer.parseInt(String.valueOf(x.charAt(d)));
                    d = d + 2;
                }
            }

            for (int i = 0; i < a.length; i++) {
                a[i] = a[i] * 2;
                if (a[i] > 9) {
                    a[i] = a[i] - 9;
                }
                suma = suma + a[i] + b[i];
            }
            int aux = suma / 10;
            int dec = (aux + 1) * 10;
            if ((dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length()
- 1)))) {
                return true;
            } else if (suma % 10 == 0 && x.charAt(x.length() - 1) == '0') {
                return true;
            } else {
                return false;
            }
        }
    }

    @Override
    public int generarId() {
```

```
        return 0;

    }

    public List<Persona> personas () {

        List<Persona> listaP = new ArrayList ();
        Persona persona;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            persona = (Persona) i.next();
            listaP.add(persona);
        }
        return listaP;
    }
}
```

Controlador Autoridad

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.AutoridadCivil;

/**
 *
 * @author paul_
 */
public class ControladorAutoridadCivil extends AbstractControlador<AutoridadCivil>{

    private AutoridadCivil autoridad;

    public ControladorAutoridadCivil(String ruta) {
        super(ruta);
    }

    public AutoridadCivil getAutoridad() {
        return autoridad;
    }

    public void setAutoridad(AutoridadCivil autoridad) {
        this.autoridad = autoridad;
    }

    @Override
    public boolean validar(AutoridadCivil objeto) {
        if (objeto.getRol().equals("Autoridad")) {
            int suma = 0;
            String x = objeto.getCedula();
            if (x.length() == 9) {
                return false;
            } else {
                int a[] = new int[x.length() / 2];
                int b[] = new int[(x.length() / 2)];
                int c = 0;
                int d = 1;
                for (int i = 0; i < x.length() / 2; i++) {
                    a[i] = Integer.parseInt(String.valueOf(x.charAt(c)));
                    c = c + 2;
                    if (i < (x.length() / 2) - 1) {
                        b[i] = Integer.parseInt(String.valueOf(x.charAt(d)));
                        d = d + 2;
                    }
                }
            }
        }
    }
}
```

```

    }
}

for (int i = 0; i < a.length; i++) {
    a[i] = a[i] * 2;
    if (a[i] > 9) {
        a[i] = a[i] - 9;
    }
    suma = suma + a[i] + b[i];
}
int aux = suma / 10;
int dec = (aux + 1) * 10;
if ((dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length()
- 1)))) {
    return true;
} else if (suma % 10 == 0 && x.charAt(x.length() - 1) == '0') {
    return true;
} else {
    return false;
}
}
} else {
    return false;
}
}

@Override
public int generarId() {
    return 0;
}

public boolean iniciarSesion(String correo, String pass) {

    for (AutoridadCivil usu : super.getList()) {
        AutoridadCivil u = (AutoridadCivil) usu;
        if (u.getCorreo().equals(correo) && u.getPass().equals(pass)) {
            this.autoridad = u;
            return true;
        }
    }
    return false;
}
}

```

Controlador Matrimonio

```

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Matrimonio;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorMatrimonio extends AbstractControlador<Matrimonio> {

    public ControladorMatrimonio(String ruta) {
        super(ruta);
    }
}

```

```

    }

    @Override
    public boolean validar(Matrimonio objeto) {
        if(objeto.getContrayente1().getRol().equals("Contrayente") && objeto.getCon-
        trayente2().getRol().equals("Contrayente")){
            if(objeto.getContrayente1().getEstadoCivil() != "Casado" && objeto.getCon-
            trayente2().getEstadoCivil() != "Casado"){
                return true;
            }
        }

        return false;
    }

    @Override
    public int generarId() {
        List<Matrimonio> temp = new ArrayList();
        for (Matrimonio matrimonio : super.getList()) {
            Matrimonio m = (Matrimonio) matrimonio;
            temp.add(m);
        }

        if (temp.size() > 0 && temp != null) {
            return temp.get(temp.size() - 1).getNumMatrimonio() + 1;
        } else {
            return 1;
        }
    }

    public List<Matrimonio> registros() {

        List<Matrimonio> listaM = new ArrayList();
        Matrimonio matrimonio;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            matrimonio = (Matrimonio) i.next();
            listaM.add(matrimonio);
        }
        return listaM;
    }
}

```

Lo siguiente es implementar los métodos asignados en cada clase a las respectivas ventanas.

- **Ec.edu.ups.vista**


Ventana Principal

```

package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorAutoridadCivil;
import ec.edu.ups.controlador.ControladorMatrimonio;
import ec.edu.ups.controlador.ControladorPersona;
import javax.swing.JDesktopPane;
import javax.swing.JMenu;

```

	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

```

import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

/**
 *
 * @author user
 */
public class VentanaPrincipal extends javax.swing.JFrame {

    private VentanaIniciarSesion ventanaIniciarSesion;
    private VentanaGestionPersona registrarPersona;
    private VentanaRegistroMatrimonio registrarMatrimonio;
    private VentanaRegistroAutoridad registrarUsuario;
    private ControladorPersona controladorPersona;
    private ControladorAutoridadCivil controladorAutoridad;
    private ControladorMatrimonio controladorMatrimonio;

    public VentanaPrincipal() {
        initComponents();
        GestionMenu.setVisible(false);
        btnCerrarSesionMenu.setVisible(false);
        controladorPersona = new ControladorPersona("datos/Persona.obj");
        controladorAutoridad = new ControladorAutoridadCivil("datos/Autoridad.obj");
        controladorMatrimonio = new ControladorMatrimonio("datos/Matrimonio.obj");
        ventanaIniciarSesion = new VentanaIniciarSesion(this, controladorAutoridad);
        registrarPersona = new VentanaGestionPersona(controladorPersona);
        registrarMatrimonio = new VentanaRegistroMatrimonio(controladorPersona, controladorAutoridad, controladorMatrimonio);
        registrarUsuario = new VentanaRegistroAutoridad(controladorAutoridad);
    }

    private void btnSalirMenuActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void btnRegistrarUsuarioMenuActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(registrarUsuario);
        registrarUsuario.setVisible(true);
    }

    private void btnMenuIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(ventanaIniciarSesion);
        ventanaIniciarSesion.setVisible(true);
    }

    private void btnCerrarSesionMenuActionPerformed(java.awt.event.ActionEvent evt) {
        btnMenuIniciarSesion.setVisible(true);
        btnRegistrarUsuarioMenu.setVisible(true);
        GestionMenu.setVisible(false);
        btnSalirMenu.setVisible(true);
    }

    private void btnRegistroMatrimonioActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(registrarMatrimonio);
        registrarMatrimonio.setVisible(true);
    }

```

}

```
private void btnGestionPersonaActionPerformed(java.awt.event.ActionEvent evt) {  
    desktopPane.add(registrarPersona);  
    registrarPersona.setVisible(true);  
}
```

Ventana Iniciar Sesión

```
package ec.edu.ups.vista;
```

```
import ec.edu.ups.controlador.ControladorAutoridadCivil;  
import javax.swing.JOptionPane;
```

```
/**  
 *  
 * @author Juanc  
 */
```

```
public class VentanaIniciarSesion extends javax.swing.JInternalFrame {
```

```
    private VentanaPrincipal ventanaPrincipal;  
    private ControladorAutoridadCivil controladorUsuario;
```

```
/**  
 * Creates new form VentanaIniciarSesion  
 *  
 * @param ventanaPrincipal  
 * @param controladorUsuario  
 */
```

```
public VentanaIniciarSesion(VentanaPrincipal ventanaPrincipal, ControladorAuto-  
ridadCivil controladorUsuario) {  
    initComponents();  
    this.ventanaPrincipal = ventanaPrincipal;  
    this.controladorUsuario = controladorUsuario;  
}
```

```
private void btnIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {  
    controladorUsuario.cargarDatos();  
    String usuario = txtCorreo.getText();  
    String pass = "";  
    char[] pass1 = txtPass.getPassword();  
    for (int i = 0; i < pass1.length; i++) {  
        pass = pass + pass1[i];  
    }
```

```
    if (controladorUsuario.iniciarSesion(usuario, pass)) {  
        ventanaPrincipal.getBtnMenuIniciarSesion().setVisible(false);  
        ventanaPrincipal.getBtnCerrarSesionMenu().setVisible(true);  
        ventanaPrincipal.getGestionMenu().setVisible(true);
```

```
        ventanaPrincipal.getBtnSalirMenu().setVisible(false);  
        ventanaPrincipal.getBtnRegistrarPersonaMenu().setVisible(false);  
        ventanaPrincipal.getGestionMenu().setVisible(true);
```

```
        this.dispose();  
        JOptionPane.showMessageDialog(this, "Inicio de sesion exitoso");  
    } else {
```

```
        JOptionPane.showMessageDialog(this, "Usuario o contrasena incorrecta ");  
        Limpiar();  
    }
```

}


```
public void Limpiar() {  
    txtCorreo.setText("");  
    txtPass.setText("");  
}  
  
// Variables declaration - do not modify  
private javax.swing.JButton btnIniciarSesion;  
private javax.swing.JLabel jLabel2;  
private javax.swing.JLabel lblContra;  
private javax.swing.JLabel lblCorreo;  
private javax.swing.JTextField txtCorreo;  
private javax.swing.JPasswordField txtPass;  
// End of variables declaration  
}
```

Ventana Registrar Usuario

```
package ec.edu.ups.vista;  
  
import ec.edu.ups.controlador.ControladorAutoridadCivil;  
import ec.edu.ups.modelo.AutoridadCivil;  
import java.io.IOException;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.swing.JOptionPane;  
  
/**  
 *  
 * @author paul_  
 */  
public class VentanaRegistroAutoridad extends javax.swing.JInternalFrame {  
  
    private ControladorAutoridadCivil controlador;  
  
    public VentanaRegistroAutoridad(ControladorAutoridadCivil controlador) {  
        initComponents();  
        this.controlador = controlador;  
        controlador.cargarDatos();  
    }  
  
    public void Limpiar() {  
        txtGcedula.setText("");  
        txtGnombre.setText("");  
        txtGapellido.setText("");  
        txtCargo.setText("");  
        txtEstadoCivil.setText("");  
        txtDireccion.setText("");  
        txtCorreo.setText("");  
        jGpr.setText("");  
    }  
  
    private void txtGcedulaActionPerformed(java.awt.event.ActionEvent evt) {  
        // TODO add your handling code here:  
    }  
  
    private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {  
        this.dispose();  
    }  
  
    private void cbxGeneroActionPerformed(java.awt.event.ActionEvent evt) {
```

```

}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    String pass = "";
    char[] pass1 = jGpr.getPassword();
    for (int i = 0; i < pass1.length; i++) {
        pass = pass + pass1[i];
    }
    SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
    Date fechaN = new Date();
    try {
        fechaN = formato.parse(txtFechaN.getText().trim());
    } catch (ParseException ex) {
        System.out.println(ex);
    }

    //txtCargo.getText().trim(),txtCor-
    reo.getText().trim(),txtGcedula.getText().trim(),txtGnom-
    bre.getText().trim(),txtGapellido.getText().trim(),txtDirec-
    cion.getText().trim(),cbxGenero.getSelectedI-
    tem().toString().trim(),fechaN,txtEstadoCivil.getText().trim(),"Ciudadano"
    //String cargo, String correo, String pass, String cedula, String nombres,
    String apellidos, String direccion, String genero, Date fechaNacimiento, String es-
    tadoCivil, String rol
    AutoridadCivil usuario = new AutoridadCivil(txtGcedula.getText().trim(),
    txtGnombre.getText().trim(), txtGapellido.getText().trim(),
    txtDireccion.getText().trim(), cbxGenero.getSelectedI-
    tem().toString().trim(),
    fechaN, txtEstadoCivil.getText().trim(), "Autoridad", txt-
    Cargo.getText().trim(),
    txtCorreo.getText().trim(), pass.trim());

    System.out.println(usuario);
    if (controlador.validar(usuario)) {
        controlador.crear(usuario);
        JOptionPane.showMessageDialog(this, "Usuario creado correctamente");
        Limpiar();
        this.dispose();
    } else {
        JOptionPane.showMessageDialog(this, "La cedula ingresada no es valida");
        Limpiar();
    }
    ;
    try {
        controlador.guardarDatos("datos/Autoridad.obj");
    } catch (IOException ex) {
        Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE, null, ex);
    }
}

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

// Variables declaration - do not modify
private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnGuardar;

```

```
private javax.swing.JComboBox<String> cbxGenero;
private javax.swing.JPasswordField jGpr;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JTextField txtCargo;
private javax.swing.JTextField txtCorreo;
private javax.swing.JTextField txtDireccion;
private javax.swing.JTextField txtEstadoCivil;
private javax.swing.JFormattedTextField txtFechaN;
private javax.swing.JTextField txtGapellido;
private javax.swing.JTextField txtGcedula;
private javax.swing.JTextField txtGnombre;
// End of variables declaration
}

Ventana Registro Matrimonio
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorAutoridadCivil;
import ec.edu.ups.controlador.ControladorMatrimonio;
import ec.edu.ups.controlador.ControladorPersona;
import ec.edu.ups.modelo.AutoridadCivil;
import ec.edu.ups.modelo.Matrimonio;
import ec.edu.ups.modelo.Persona;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author paul_
 */
public class VentanaRegistroMatrimonio extends javax.swing.JInternalFrame {

    private ControladorPersona controladorPersona;
    private ControladorAutoridadCivil controladorAutoridad;
    private ControladorMatrimonio controladorMatrimonio;

    public VentanaRegistroMatrimonio(ControladorPersona controladorPersona,ControladorAutoridadCivil controladorAutoridad,ControladorMatrimonio controladorMatrimonio )
    {
        initComponents();
        this.controladorAutoridad=controladorAutoridad;
        this.controladorPersona=controladorPersona;
        this.controladorMatrimonio= controladorMatrimonio;
    }

    public void cargarSiguienteCodigo() {
```

```

lblCodigoR.setText (String.valueOf (controladorAutoridad.generarId ()));

}

private void btnBuscarContrayenteActionPerformed(java.awt.event.ActionEvent evt) {
    Persona comparar = new Persona ();
    comparar.setCedula (txtCedulaC1.getText ().trim ());
    Optional<Persona> p= controladorPersona.buscar (comparar);
    Persona persona =p.get ();
    System.out.println (" "+persona);
    if (persona!=null){
        txtNombrel.setText (persona.getNombres ());
        txtApellido1.setText (persona.getApellidos ());
        txtDireccion.setText (persona.getDireccion ());
        txtFN1.setText (persona.getFechaDeNacimiento ().toString ());
        txtGenero1.setText (persona.getGenero ());
        persona.setRol ("Contrayente");
        if (controladorPersona.actualizar (persona)) {
            JOptionPane.showMessageDialog (this, "Datos Contrayente1");
        }
        try {
            controladorMatrimonio.guardarDatos ("datos/Persona.obj");
        } catch (IOException ex) {
            Logger.getLogger (VentanaRegistroAutoridad.class.getName ()) .log (Level.SEVERE, null, ex);
        }
    }

}

private void btnBuscarTestigo2ActionPerformed(java.awt.event.ActionEvent evt) {
    Persona comparar = new Persona ();
    comparar.setCedula (txtTestigoCed.getText ().trim ());
    Optional<Persona> p= controladorPersona.buscar (comparar);
    Persona persona =p.get ();
    System.out.println (" "+persona);

    persona.setRol ("Testigo");
    if (persona!=null){
        lbl7.setText (persona.getNombres ()+" "+persona.getApellidos ()+" "+persona.getCedula ());

        if (controladorPersona.actualizar (persona)) {
            JOptionPane.showMessageDialog (this, "Datos Testigo 1");
        }

        try {
            controladorMatrimonio.guardarDatos ("datos/Persona.obj");
        } catch (IOException ex) {
            Logger.getLogger (VentanaRegistroAutoridad.class.getName ()) .log (Level.SEVERE, null, ex);
        }
    }

}

private void btnBuscarContrayente1ActionPerformed(java.awt.event.ActionEvent
evt) {
    Persona comparar = new Persona ();
    comparar.setCedula (txtCedulaC1.getText ().trim ());

```

```
Optional<Persona> p= controladorPersona.buscar(comparar);
Persona persona =p.get();
System.out.println(""+persona);
if (persona!=null){
    txt3.setText(persona.getNombres());
    txt5.setText(persona.getApellidos());
    txt.setText(persona.getDireccion());
    txt4.setText(persona.getFechaDeNacimiento().toString());
    txtGenero.setText(persona.getGenero());
    persona.setRol("Contrayente");
    if(controladorPersona.actualizar(persona)){
        JOptionPane.showMessageDialog(this, "Datos Contrayente1");
    }
    try {
        controladorMatrimonio.guardarDatos("datos/Persona.obj");
    } catch (IOException ex) {
        Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void btnBuscarTestigo1ActionPerformed(java.awt.event.ActionEvent evt) {
    Persona comparar = new Persona ();
    comparar.setCedula(txtTestigoCed1.getText().trim());
    Optional<Persona> p= controladorPersona.buscar(comparar);
    Persona persona =p.get();
    System.out.println(""+persona);

    persona.setRol("Testigo");
    if (persona!=null){
        lbl7.setText(persona.getNombres()+" "+persona.getApellidos()+" "+persona.getCedula());

        if(controladorPersona.actualizar(persona)){
            JOptionPane.showMessageDialog(this, "Datos Testigo 1");
        }
        try {
            controladorMatrimonio.guardarDatos("datos/Persona.obj");
        } catch (IOException ex) {
            Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent evt) {
    cargarSiguienteCodigo();
    lblAutoridad1.setText(controladorAutoridad.getAutoridad().getNombres()
        +" "+controladorAutoridad.getAutoridad().getApellidos()+" "+controladorAutoridad.getAutoridad().getCargo());
    Date fecha = new Date();
    lblFechaRegistro.setText(fecha.toString());
    controladorPersona.cargarDatos();
}

private void btnRegistrarMActionPerformed(java.awt.event.ActionEvent evt) {
    Persona t1 = new Persona();
```

```

Persona t2 = new Persona();

Persona Com= new Persona();
Persona Com2 = new Persona();

Com.setCedula(txtCedulaC1.getText().trim());
Com2.setCedula(txtCedulaC3.getText().trim());

Optional<Persona> p= controladorPersona.buscar(Com);

Optional<Persona> p2= controladorPersona.buscar(Com2);
t1=p.get();
t2=p2.get();

AutoridadCivil a= controladorAutoridad.getAutoridad();

SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
Date fN=null;
try {
    fN = formato.parse(txtFN1.getText().trim());
} catch (ParseException ex) {
    Logger.getLogger(VentanaRegistroMatrimonio.class.getName()).log(Level.SEVERE, null, ex);
}

Date f=null;
try {
    f = formato.parse(txt4.getText().trim());
} catch (ParseException ex) {
    Logger.getLogger(VentanaRegistroMatrimonio.class.getName()).log(Level.SEVERE, null, ex);
}

Date fecha = new Date();
Persona c1= new Persona(txtCedulaC1.getText().trim(),txtNombrel1.getText().trim(),txtApellidol1.getText().trim(),txtDireccion.getText().trim(),txtGenerol1.getText().trim(),fN,"Casado "+txt3.getText().trim()+" "+txtApellidol1,"Ciudadano");
Persona c2= new Persona(txtCedulaC3.getText().trim(),txt3.getText().trim(),txt5.getText().trim(),txt6.getText().trim(),txtGenero.getText().trim(),f,"Casado "+txtNombrel1.getText().trim()+" "+txt5,"Ciudadano");
//String cedula, String nombres, String apellidos, String direccion, String
genero, Date fechaNacimiento, String estadoCivil, String rol
//int codigoM, String lugar, Date fecha, Persona contrayente1, Persona
contrayente2, Persona testigo1, Persona testigo2, Persona autoridad
Matrimonio m = new Matrimonio(Integer.parseInt(lblCodigoR.getText().trim()),txtLugar.getText().trim(),fecha,c1,c2,t1,t2,a);

controladorMatrimonio.crear(m);

controladorPersona.actualizar(c1);
controladorPersona.actualizar(c2);

JOptionPane.showMessageDialog(this, "MATRIMONIO REGISTRADO CON EXITO : FELICITACIONES A LOS NOVIOS");
try {
    controladorMatrimonio.guardarDatos("datos/Persona.obj");
} catch (IOException ex) {
    Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE, null, ex);
}

```

```

    }

    try {
        controladorMatrimonio.guardarDatos("datos/Matrimonio.obj");
    } catch (IOException ex) {
        Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE, null, ex);
    }

}

private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    lblFechaRegistro.setText("");
    txtLugar.setText("");
    cargarSiguienteCodigo();
    txtCedulaC1.setText("");
    txtNombre1.setText("");
    txtApellido1.setText("");
    txtDireccion.setText("");
    txtFN1.setText("");
    txtGenero1.setText("");
    lbl7.setText("");
    txtCedulaC3.setText("");
    txt3.setText("");
    txt5.setText("");
    txt4.setText("");
    txtGenero.setText("");
    txtTestigoCed1.setText("");
    lbl9.setText("");
}

```

Ventana Gestion Personas

```

package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorPersona;
import ec.edu.ups.modelo.Persona;
import ec.edu.ups.modelo.AutoridadCivil;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Iterator;
import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public class VentanaGestionPersona extends javax.swing.JInternalFrame {

    private ControladorPersona controlador;

    /**
     * Creates new form GestionUsuario
     */
    public VentanaGestionPersona(ControladorPersona controlador) {
        initComponents();
    }
}

```

```

        this.controlador = controlador;
    }

    public void limpiarTabla() {
        DefaultTableModel modelo = (DefaultTableModel) tblPersonas.getModel();
        modelo.setRowCount(0);
    }

    public void cargarPersonasTbl() {
        DefaultTableModel modelo = (DefaultTableModel) tblPersonas.getModel();
        modelo.setRowCount(0);
        if(controlador.personas()!=null){
            for (Iterator it = controlador.personas().iterator(); it.hasNext(); ) {
                Persona persona = (Persona) it.next();
                Object[] rowData = {persona.getCedula(), persona.getNombres(), persona.getApellidos(), persona.getDireccion(), persona.getFechaDeNacimiento().toString(), persona.getGenero(), persona.getEstadoCivil()};
                modelo.addRow(rowData);
                tblPersonas.setModel(modelo);
            }
        }else{
            System.out.println("LISTA VACIA");
        }
    }

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

    private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
    }

    private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
        Date fechaN = new Date();
        try {
            fechaN = formato.parse(txtFechaN.getText().trim());
        } catch (ParseException ex) {
            System.out.println(ex);
        }
        Persona persona = new Persona(txtGcedula.getText().trim(), txtGnombre.getText().trim(), txtGapellido.getText().trim(), txtGDireccion.getText().trim(), cbxGenero.getSelectedIndex().toString().trim(), fechaN, txtEstadoCivil.getText().trim(), "Ciudadano");
        if(controlador.crear(persona)){
            JOptionPane.showMessageDialog(this, "Persona creada exitosamente");
            Limpiar();
            cargarPersonasTbl();
            try {
                controlador.guardarDatos("datos/Persona.obj");
            } catch (IOException ex) {
                Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE, null, ex);
            }
        }else{
            JOptionPane.showMessageDialog(this, "No se pudo crear a la Persona :ERROR");
        }
    }

```



```
private void cbxGenerolActionPerformed(java.awt.event.ActionEvent evt) {
}

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
    Date fechaN = new Date();
    try {
        fechaN = formato.parse(txtFechaN.getText().trim());
    } catch (ParseException ex) {
        System.out.println(ex);
    }
    Persona persona = new Persona(txtGcedula.getText().trim(), txtGnom-
bre.getText().trim(), txtGapellido.getText().trim(), txtGDireccion.getText().trim(),
        cbxGenerol.getSelectedItem().toString().trim(), fechaN, txtEstadoCi-
vil.getText().trim(), "Ciudadano");

    if(controlador.actualizar(persona)){
        JOptionPane.showMessageDialog(this, "Persona actualizada exitosamente");
        Limpiar();
        cargarPersonasTbl();
    }else{
        JOptionPane.showMessageDialog(this, "No se pudo actualizar la persona
:ERROR");
    }
    try {
        controlador.guardarDatos("datos/Persona.obj");
    } catch (IOException ex) {
        Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE,
        null, ex);
    }
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int d = JOptionPane.showConfirmDialog(this, "Esta seguro de eliminar la per-
sona con cedula" + txtGcedula.getText().trim());
    if (d == JOptionPane.YES_OPTION) {
        Persona comparar = new Persona();
        comparar.setCedula(txtGcedula.getText().trim());
        Optional<Persona> p= controlador.buscar(comparar);
        Persona persona=p.get();
        System.out.println(""+persona);

        if (controlador.eliminar(persona)) {
            JOptionPane.showMessageDialog(this, "Persona eliminada exitosa-
mente");
            cargarPersonasTbl();
        } else if (d == JOptionPane.NO_OPTION) {
            cargarPersonasTbl();
        }
    }
    try {
        controlador.guardarDatos("datos/Persona.obj");
    } catch (IOException ex) {
        Logger.getLogger(VentanaRegistroAutoridad.class.getName()).log(Level.SEVERE,
        null, ex);
    }
}

private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
```

```
if(controlador.personas() != null) {
    cargarPersonasTbl();

    controlador.cargarDatos();

}

}

private void tblPersonasMouseClicked(java.awt.event.MouseEvent evt) {
    int index = tblPersonas.getSelectedRow();

    String cedula = "" + tblPersonas.getValueAt(index, 0);
    String nombre = "" + tblPersonas.getValueAt(index, 1);
    String apellido = "" + tblPersonas.getValueAt(index, 2);
    String Direccion = "" + tblPersonas.getValueAt(index, 3);
    String fechaN = "" + tblPersonas.getValueAt(index, 4);
    String genero = "" + tblPersonas.getValueAt(index, 5);
    String eCivil = "" + tblPersonas.getValueAt(index, 6);

    txtGcedula.setText(cedula);
    txtGnombre.setText(nombre);
    txtGapellido.setText(apellido);
    txtGDireccion.setText(Direccion);
    txtFechaN.setText(fechaN);
    cbxGenero.setSelectedItem(genero.trim());
    txtEstadoCivil.setText(eCivil);

}

public void Limpiar() {
    txtGcedula.setText("");
    txtGnombre.setText("");
    txtGapellido.setText("");
    txtGDireccion.setText("");
    txtFechaN.setText("");
    cbxGenero.setSelectedIndex(0);
    txtEstadoCivil.setText("Soltero");
    // jGpr.setText("");
}

}
```

Ventana Lista Registro Matrimonios

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorMatrimonio;
import ec.edu.ups.modelo.Matrimonio;
import ec.edu.ups.modelo.Persona;
import java.util.Iterator;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public class VentanaListaRegistrosDeMatrimonio extends javax.swing.JFrame {

    private ControladorMatrimonio controladorMatrimonio;
```

```

public VentanaListaRegistrosDeMatrimonio (ControladorMatrimonio controladorMatri-
monio) {
    initComponents();
    this.controladorMatrimonio= controladorMatrimonio;
}
private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent evt) {
    controladorMatrimonio.cargarDatos();
    DefaultTableModel modelo = (DefaultTableModel) tblRegistros.getModel();
    modelo.setRowCount(0);
    if(controladorMatrimonio.registros() != null){
        for (Iterator it = controladorMatrimonio.registros().iterator(); it.has-
Next();) {
            Matrimonio matrimonio = (Matrimonio) it.next();
            Object[] rowData = {matrimonio.getNumMatrimonio(),matrimonio.getLu-
gar(),matrimonio.getFecha().toString(),matrimonio.getContrayente1().getNombres()
            +" "+matrimonio.getContrayente1().getApellidos(),matrimonio.get-
Contrayente2().getNombres()+" "+matrimonio.getContrayente2().getApellidos()
            , matrimonio.getTestigo1().getNombres().concat(matrimonio.get-
Testigo1().getApellidos()),matrimonio.getTestigo2().getNombres()+" "+matrimonio.get-
Testigo2().getApellidos(),matrimonio.getAutoridadCivil().getNombres()
            +" "+matrimonio.getAutoridadCivil().getApellidos()};
            modelo.addRow(rowData);
            tblRegistros.setModel(modelo);
        }
    }else{
        System.out.println("LISTA VACIA");
    }
}

// Variables declaration - do not modify
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable tblRegistros;
// End of variables declaration
}

```

Finalmente, la ejecución del programa.

Ventana Inicio



Ventana Iniciar Sesión



Inicio

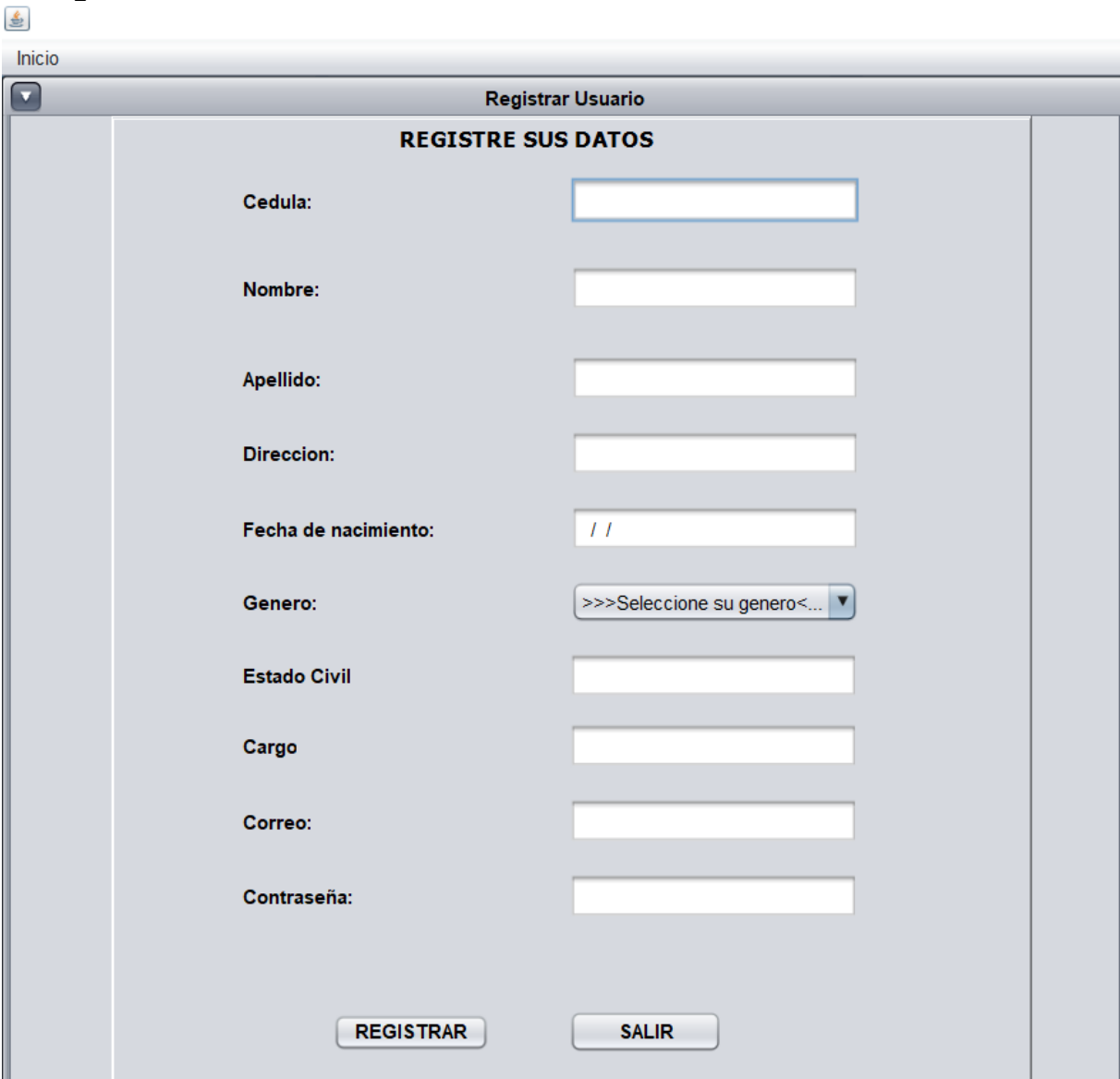
INICIAR SESION

Correo:

Contraseña:

Iniciar Sesion

Ventana Registrar Usuario



Inicio

Registrar Usuario

REGISTRE SUS DATOS

Cedula:

Nombre:

Apellido:

Direccion:

Fecha de nacimiento:

Genero: >>>Seleccione su genero<...>

Estado Civil

Cargo

Correo:

Contraseña:

REGISTRAR SALIR

Ventana Registrar Matrimonio

CODIGO DE REGISTRO: 0

FECHA: **LUGAR:**

Cedula contrayente:

Nombre:

Apellido:

Dirección:

Estado Civil

Fecha de nacimiento:

Genero:

Cedula Testigo

Nombres del testigo :

Cedula contrayente:

Nombre:

Apellido:

Dirección:

Estado Civil

Fecha de nacimiento:

Genero:

Cedula Testigo

Nombres del testigo :

Ventana Gestión Personas

REGISTRE SUS DATOS

Cedula:

Nombre:

Apellido:

Dirección:

Fecha de nacimiento:

Genero:

Estado Civil

Cedula	Nombre	Apellido	Dirección	Fecha de Nacimiento	Genero	Estado Civil

Ventana Lista Personas

LISTA DE PERSONAS

Código	Lugar	Fecha	Contrayente 1	Contrayente 2	Testigo 1	Testigo 2	Autoridad

De esta manera observamos que cada interfaz tiene su función dentro de la app que es muy eficaz.

5. Fecha de entrega: El sistema debe ser subido al git hasta **27 de noviembre del 2020 – 23:55**.

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar la persistencia en archivos.

RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.

Nombre de estudiante:

PAUL ALEXANDER GUAPUCAL CARDENAS

Firma de estudiante:

