
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.


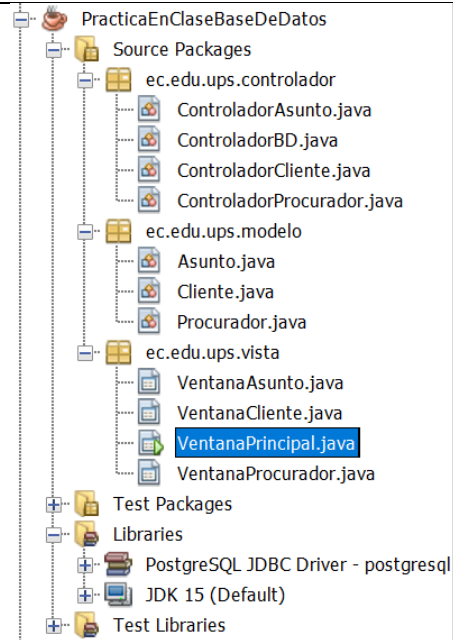
Alcance


El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos

- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:		TÍTULO PRÁCTICA: Base de Datos	
OBJETIVO: Identificar las sentencias SQL Diseñar e Implementar codigos DDL, DML Entender cada una de las características del uso de una Base de Datos			
ACTIVIDADES DESARROLLADAS			
 <p>Como se puede visualizar en la imagen tenemos el patron de diseño MVC que esta implementado en este programa.</p> <p>Paquete Modelo Clase Cliente package ec.edu.ups.modelo; import java.util.Objects; /** * * @author NANCY */ public class Cliente { private String cedula; private String nombre; private String apellido; </p>			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

private String direccion;

public Cliente() {
}

public Cliente(String cedula, String nombre, String apellido, String direccion) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.direccion = direccion;
}

public String getCedula() {
    return cedula;
}

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

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}


public String getDireccion() {
    return direccion;
}

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

@Override
public int hashCode() {
    int hash = 3;
    hash = 17 * hash + Objects.hashCode(this.cedula);
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Cliente other = (Cliente) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Cliente{" + "cedula=" + cedula + ", nombre=" + nombre + ", apellido=" + apellido + ", direccion=" + direccion + "}";
}
}

```

En el código de la clase cliente podemos ver los atributos que hacen al objeto cliente y todos sus geter and seters en si cada atributo creado me sirve para después ser implementado o pasado a la base de datos como un dato de dicho objeto

Clase Asunto

```

package ec.edu.ups.modelo;

import java.util.Date;


/**
 *
 * @author NANCY
 */
public class Asunto {
    private int codigo;
    private Date fechalnicio;
    private Date fechaFinalizacion;
    private String estado;
    private String cedulaClientefk;

    public Asunto() {
    }

    public Asunto(int codigo, Date fechalnicio, Date fechaFinalizacion, String estado, String cedulaClientefk) {
        this.codigo = codigo;
        this.fechalnicio = fechalnicio;
        this.fechaFinalizacion = fechaFinalizacion;
        this.estado = estado;
        this.cedulaClientefk = cedulaClientefk;
    }

    public int getCodigo() {
        return codigo;
    }

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

}

public void setCodigo(int codigo) {
    this.codigo = codigo;
}

public Date getFechaInicio() {
    return fechaInicio;
}

public void setFechaInicio(Date fechaInicio) {
    this.fechaInicio = fechaInicio;
}

public Date getFechaFinalizacion() {
    return fechaFinalizacion;
}

public void setFechaFinalizacion(Date fechaFinalizacion) {
    this.fechaFinalizacion = fechaFinalizacion;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}


public String getCedulaClientefk() {
    return cedulaClientefk;
}

public void setCedulaClientefk(String cedulaClientefk) {
    this.cedulaClientefk = cedulaClientefk;
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 29 * hash + this.codigo;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

if (getClass() != obj.getClass()) {
    return false;
}
final Asunto other = (Asunto) obj;
if (this.codigo != other.codigo) {
    return false;
}
return true;
}

```

```

@Override
public String toString() {
    return "Asunto{" + "codigo=" + codigo + ", fechaInicio=" + fechaInicio + ", fechaFinalizacion=" + fechaFinalizacion + ", estado=" + estado + ", cedulaClientefk=" + cedulaClientefk + '}';
}
}

```

En la clase asunto tenemos los atributos que hacen a un asunto ente ellos tenemos la cedula del cliente la cual va a hacer el fk de relación ente las dos tablas ya que un asunto siempre va a contener a un cliente de la misma manera todos los atributos tienen sus geter and seters para obtener los datos o también se pueden pasar a través de un constructor.

Clase Procurador.

```

package ec.edu.ups.modelo;

import java.util.Objects;


/**
 *
 * @author NANCY
 */
public class Procurador {
    private String cedula;
    private String nombre;
    private String apellido;
    private int codigofk;

    public Procurador() {
    }

    public Procurador(String cedula, String nombre, String apellido, int codigofk) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.codigofk = codigofk;
    }

    public String getCedula() {
        return cedula;
    }
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

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

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}


public int getCodigofk() {
    return codigofk;
}

public void setCodigofk(int codigofk) {
    this.codigofk = codigofk;
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 17 * 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 Procurador other = (Procurador) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    return true;
}

```


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```
@Override
public String toString() {
    return "Procurador{" + "cedula=" + cedula + ", nombre=" + nombre + ", apellido=" + apellido + ", codigofk=" + codigofk +
};
}
```

La clase procurador contiene los atributos q lo hacen al procurador, pero aparte contiene un código de uno de los asuntos que va a ser el de la relación con la tabla de procuradores y asuntos ya que varios procuradores pueden tener el mismo asunto y por ende el mismo cliente para el caso.

Paquete Controlador Controlador BD

```
package ec.edu.ups.controlador;
import java.sql.Connection;
import java.sql.DriverManager;
```


```
/**
 *
 * @author NANCY
 */
public class ControladorBD {

    public static Connection conectar(){
        Connection con = null;
        try {
            Class.forName("org.postgresql.Driver");
            con=DriverManager.getConnection("jdbc:postgresql://localhost:5432/PracticaClase", "postgres", "Raaf2001");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return con;
    }

    public static void Desconectar(Connection con){
        try{
            con.close();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

En esta clase se encuentran los métodos conectar y desconectar de la base de datos estos métodos son muy importantes para el programa ya que a través de estos vamos a acceder a la base de datos y a sus respectivas tablas para la manipulación de sus datos.

Controlador Cliente

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

package ec.edu.ups.controlador;
import ec.edu.ups.modelo.Cliente;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author NANCY
 */
public class ControladorCliente {
    private List<Cliente>listadoCliente;

    public ControladorCliente() {
        listadoCliente= new ArrayList<>();
        this.cargardatos();
    }

    public boolean insertarAsunto(Cliente cliente){
        Connection con = ControladorBD.conectar();
        try{
            String sql="INSERT INTO CLIENTE(cedula, nombre, apellido, direccion) VALUES(?, ?, ?, ?)";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, cliente.getCedula());
            ps.setString(2, cliente.getNombre());
            ps.setString(3, cliente.getApellido());
            ps.setString(4, cliente.getDireccion());
            ps.executeUpdate();
        }catch(Exception e){
            e.printStackTrace();
            return false;
        }finally{
            ControladorBD.Desconectar(con);
        }
        return true;
    }

    public boolean actualizar(Cliente cliente){
        Connection con = ControladorBD.conectar();
        try{
            String sql="UPDATE CLIENTE SET nombre = ?, apellido = ?, direccion= ? WHERE cedula = ?";
            PreparedStatement ps = con.prepareStatement(sql);

            ps.setString(1, cliente.getNombre());
            ps.setString(2, cliente.getApellido());
            ps.setString(3, cliente.getDireccion());
            ps.setString(4, cliente.getCedula());
            ps.executeUpdate();
            return true;
        }catch(Exception e){
            e.printStackTrace();
            return false;
        }finally{

```

```
ControladorBD.Desconectar(con);
    }
}

public void cargardatos(){


    listadoCliente.clear();
    Connection con = ControladorBD.conectar();
    try{
        String sql="SELECT * FROM CLIENTE";
        PreparedStatement ps = con.prepareStatement(sql);
        ResultSet resultado= ps.executeQuery();
        while(resultado.next()){
            Cliente c= new Cliente();
            c.setCedula(resultado.getString(1));
            c.setNombre(resultado.getString(2));
            c.setApellido(resultado.getString(3));
            c.setDireccion(resultado.getString(4));
            listadoCliente.add(c);
        }
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        ControladorBD.Desconectar(con);
    }
}

public List<Cliente> getListadoCliente() {
    return listadoCliente;
}

public boolean eliminar(String cedula){
    Connection con = ControladorBD.conectar();
    try{
        String sql="DELETE FROM CLIENTE WHERE CEDULA = ?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, cedula);
        ps.executeUpdate();
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }finally{
        ControladorBD.Desconectar(con);
    }
}
}
```

En el controlador cliente tenemos los métodos del CRUD mediante sentencias SQL y dentro de cada método vamos a conectar y desconectar de la base de datos para obtener los datos y hacer las acciones respectivas
Controlador Asunto

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

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

import ec.edu.ups.modelo.Asunto;
import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author NANCY
 */
public class ControladorAsunto {
    private List<Asunto>listadoAsunto;


    public ControladorAsunto() {
        listadoAsunto= new ArrayList<>();
        this.cargardatos();
    }

    public boolean insertarAsunto(Asunto asunto){
        Connection con = ControladorBD.conectar();
        try{
            String sql="INSERT INTO ASUNTO(codigo, fecha_inicio, fecha_Finalizacion, estado, cedula_clientefk) VALUES(?,
?, ?, ?, ?)";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, asunto.getCodigo());
            ps.setDate(2, new java.sql.Date(asunto.getFechaInicio().getTime()));
            ps.setDate(3, new java.sql.Date(asunto.getFechaFinalizacion().getTime()));
            ps.setString(4, asunto.getEstado());
            ps.setString(5, asunto.getCedulaCliente());
            ps.executeUpdate();
        }catch(Exception e){
            e.printStackTrace();
            return false;
        }finally{
            ControladorBD.Desconectar(con);
        }
        return true;
    }

    public void actualizar(){
        Connection con = ControladorBD.conectar();
        try{
            String sql="DELETE FROM ASUNTO WHERE CODIGO = ?";
            PreparedStatement ps = con.prepareStatement(sql);
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            ControladorBD.Desconectar(con);
        }
    }

    public void cargardatos(){

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

listadoAsunto.clear();
Connection con = ControladorBD.conectar();
try{
    String sql="SELECT * FROM ASUNTO";
    PreparedStatement ps = con.prepareStatement(sql);
    ResultSet resultado= ps.executeQuery();
    while(resultado.next()){
        Asunto a= new Asunto();
        a.setCodigo(resultado.getInt(1));
        a.setFechaInicio(new Date(resultado.getDate(2).getTime()));
        a.setFechaFinalizacion(new Date(resultado.getDate(3).getTime()));
        a.setEstado(resultado.getString(4));
        a.setCedulaClientefk(resultado.getString(5));
        listadoAsunto.add(a);
    }
}catch(Exception e){
    e.printStackTrace();
}finally{
    ControladorBD.Desconectar(con);
}
}

public List<Asunto> getListadoAsunto() {
    return listadoAsunto;
}

public boolean eliminar(int codigo){
    Connection con = ControladorBD.conectar();
    try{
        String sql="DELETE FROM ASUNTO WHERE CODIGO = ?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, codigo);
        ps.executeUpdate();
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }finally{
        ControladorBD.Desconectar(con);
    }
}
}
}

```


El controlador asunto al igual que el anterior también posee los métodos del CRUD para poder manipular los datos de nuestra base de datos para así poder gestionarlos en el programa para esto también usamos sentencias SQL

Controlador Procurador.

```

package ec.edu.ups.controlador;
import ec.edu.ups.modelo.Procurador;

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author NANCY
 */
public class ControladorProcurador {
    private List<Procurador>listadoProcurador;

    public ControladorProcurador() {
        listadoProcurador= new ArrayList<>();
        this.cargardatos();
    }


    public boolean insertarAsunto(Procurador procurador){
        Connection con = ControladorBD.conectar();
        try{
            String sql="INSERT INTO PROCURADOR(cedula, nombre, apellido, codigofk) VALUES(?, ?, ?, ?)";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, procurador.getCedula());
            ps.setString(2, procurador.getNombre());
            ps.setString(3, procurador.getApellido());
            ps.setInt(4, procurador.getCodigofk());
            ps.executeUpdate();
        }catch(Exception e){
            e.printStackTrace();
            return false;
        }finally{
            ControladorBD.Desconectar(con);
        }
        return true;
    }

    public void actualizar(){
        Connection con = ControladorBD.conectar();
        try{
            String sql="DELETE FROM ASUNTO WHERE CODIGO = ?";
            PreparedStatement ps = con.prepareStatement(sql);
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            ControladorBD.Desconectar(con);
        }
    }

    public void cargardatos(){

        listadoProcurador.clear();
        Connection con = ControladorBD.conectar();
        try{
            String sql="SELECT * FROM PROCURADOR";

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

PreparedStatement ps = con.prepareStatement(sql);
ResultSet resultado= ps.executeQuery();
while(resultado.next()){
    Procurador p= new Procurador();
    p.setCedula(resultado.getString(1));
    p.setNombre(resultado.getString(2));
    p.setApellido(resultado.getString(3));
    p.setCodigofk(resultado.getInt(4));
    listadoProcurador.add(p);
}
}catch(Exception e){
    e.printStackTrace();
}finally{
    ControladorBD.Desconectar(con);
}
}

public List<Procurador> getListadoProcurador() {
    return listadoProcurador;
}

public boolean eliminar(String cedula){
    Connection con = ControladorBD.conectar();
    try{
        String sql="DELETE FROM Procurador WHERE CEDULA = ?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, cedula);
        ps.executeUpdate();
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }finally{
        ControladorBD.Desconectar(con);
    }
}
}
}

```

En el controlador procurador es muy similar a los otros controladores con la misma diferencia que cada uno accede a la tabla que le corresponde para hacer las debidas modificaciones con los métodos del CRUD que utilizan sentencias SQL.

Paquete Vista

Ventana Principal

```

package ec.edu.ups.vista;


import ec.edu.ups.controlador.ControladorAsunto;
import ec.edu.ups.controlador.ControladorCliente;
import ec.edu.ups.controlador.ControladorProcurador;

```

```

/**
 *

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

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

    private VentanaCliente ventanaCliente;
    private VentanaProcurador ventanaProcurador;
    private VentanaAsunto ventanaAsunto;
    private ControladorCliente controladorCliente;
    private ControladorProcurador controladorProcurador;
    private ControladorAsunto controladorAsunto;
    /**
     * Creates new form VentanaPrincipal
     */
    public VentanaPrincipal() {
        initComponents();

        controladorCliente = new ControladorCliente();
        controladorProcurador = new ControladorProcurador();
        controladorAsunto = new ControladorAsunto();

        ventanaCliente = new VentanaCliente(controladorCliente);
        ventanaProcurador = new VentanaProcurador(controladorProcurador);
        ventanaAsunto = new VentanaAsunto(controladorAsunto);
    }
    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }


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

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

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

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {

```


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
    if ("Nimbus".equals(info.getName())) {
        javax.swing.UIManager.setLookAndFeel(info.getClassName());
        break;
    }
}
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new VentanaPrincipal().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JDesktopPane desktopPane;
private javax.swing.JMenuItem exitMenuItem;
private javax.swing.JMenu fileMenu;
private javax.swing.JMenuBar menuBar;
private javax.swing.JMenuItem openMenuItem;
private javax.swing.JMenuItem saveAsMenuItem;
private javax.swing.JMenuItem saveMenuItem;
// End of variables declaration
}

```

En esta ventana se adjuntan al panel las ventanas internas que vas a ser parte del programa qui seleccionamos la ventana que deseamos abrir.

Ventana Cliente


```

package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorCliente;
import ec.edu.ups.modelo.Cliente;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author NANCY
 */
public class VentanaCliente extends javax.swing.JInternalFrame {

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

private ControladorCliente controladorCliente;
/**
 * Creates new form VentanaCliente
 * @param controladorCliente
 */
public VentanaCliente(ControladorCliente controladorCliente) {
    initComponents();
    this.controladorCliente=controladorCliente;
    cargarDatos();
}

public void cargarDatos() {
    DefaultTableModel modeloTabla = (DefaultTableModel) tblDatos.getModel();

    modeloTabla.setRowCount(0);
    for (Cliente cliente : controladorCliente.getListadoCliente()) {
        Object[] rowData = {cliente.getCedula(), cliente.getNombre(), cliente.getApellido(), cliente.getDireccion()};
        modeloTabla.addRow(rowData);
    }
    tblDatos.setModel(modeloTabla);
}

public void limpiar(){
    txtCedula.setText("");
    txtNombre.setText("");
    txtApellido.setText("");
    txtDireccion.setText("");
}

```

En esta ventana gestionamos a los clientes actualizándolos o eliminándolos a la vez que creamos nuevos clientes en la base de datos.

Ventana Asunto

```

package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorAsunto;
import ec.edu.ups.modelo.Asunto;
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;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author NANCY
 */
public class VentanaAsunto extends javax.swing.JInternalFrame {

    private ControladorAsunto controladorAsunto;

```

```
/**
 * Creates new form VentanaAsunto
 * @param controladorAsunto
 */
public VentanaAsunto(ControladorAsunto controladorAsunto) {
    initComponents();
    this.controladorAsunto= controladorAsunto;
}

public void cargarDatos() {
    DefaultTableModel modeloTabla = (DefaultTableModel) tblDatos.getModel();

    modeloTabla.setRowCount(0);
    for (Asunto asunto: controladorAsunto.getListadoAsunto()) {
        Object[] rowData =
{asunto.getCodigo(),asunto.getFechaInicio(),asunto.getFechaFinalizacion(),asunto.getEstado(),asunto.getCedulaClientefk()};
        modeloTabla.addRow(rowData);
    }
    tblDatos.setModel(modeloTabla);
}


public void limpiar(){
    txtCedula.setText("");
    txtEstado.setText("");
    txtCodigo.setText("");
    txtFechaFinalizacion.setText("");
    txtFechaInicio.setText("");
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int codigo= Integer.parseInt(txtCodigo.getText());
    String estado= txtEstado.getText();
    String cedulafk = txtCedula.getText();

    boolean resultado = false;

    if(txtFechaFinalizacion.getText().equals("")||txtFechaFinalizacion.getText().equals("")||estado.isEmpty()||cedulafk.isEmpty()){
        JOptionPane.showMessageDialog(this, "Llene todos los campos solicitados");
    }else{
        int eliminar = JOptionPane.showConfirmDialog(this, "¿Seguro quieres eliminar este Alumno?");
        if (eliminar == JOptionPane.YES_OPTION) {
            resultado= controladorAsunto.eliminar(codigo);
        }
        JOptionPane.showMessageDialog(this, "Operación : " + resultado);
        cargarDatos();
        limpiar();
        dispose();
    }
}

private void tblDatosMouseClicked(java.awt.event.MouseEvent evt) {
    int fila = tblDatos.getSelectedRow();
    int codigo = (int) tblDatos.getValueAt(fila, 0);
    Date fechaInicio = (Date) tblDatos.getValueAt(fila, 1);
    Date fechaFinalizacion = (Date) tblDatos.getValueAt(fila, 2);
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

String estado = (String) tblDatos.getValueAt(fila, 3);
String cedula = (String)tblDatos.getValueAt(fila, 4);

txtCodigo.setText(String.valueOf(codigo));
txtFechalnicio.setText(fechalnicio.toString());
txtFechaFinalizacion.setText(fechaFinalizacion.toString());
txtCedula.setText(cedula);
txtEstado.setText(estado);

}

private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");

    int codigo= Integer.parseInt(txtCodigo.getText());
    Date fechalnicio=null;
    Date fechaFinalizacion =null;
    try {
        fechalnicio = formato.parse(txtFechalnicio.getText());
        fechaFinalizacion = formato.parse(txtFechaFinalizacion.getText());
    } catch (ParseException ex) {
        Logger.getLogger(VentanaAsunto.class.getName()).log(Level.SEVERE, null, ex);
    }

    String estado= txtEstado.getText();
    String cedulafk = txtCedula.getText();


    boolean resultado = false;

    if(txtFechaFinalizacion.getText().equals("")||txtFechaFinalizacion.getText().equals("")||estado.isEmpty()||cedulafk.isEmpty()){
        JOptionPane.showMessageDialog(this, "Llene todos los campos solicitados");
    }else{
        Asunto asunto= new Asunto(codigo, fechalnicio, fechaFinalizacion, estado, cedulafk);
        resultado= controladorAsunto.insertarAsunto(asunto);
        JOptionPane.showMessageDialog(this, "Operación : " + resultado);
        cargarDatos();
        limpiar();
        dispose();
    }
}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent evt) {
    controladorAsunto.cargardatos();
    cargarDatos();
}

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

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```
private javax.swing.JButton btnEliminar;
private javax.swing.JLabel jLabel1;
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.JScrollPane jScrollPane1;
private javax.swing.JTable tblDatos;
private javax.swing.JTextField txtCedula;
private javax.swing.JTextField txtCodigo;
private javax.swing.JTextField txtEstado;
private javax.swing.JTextField txtFechaFinalizacion;
private javax.swing.JTextField txtFechaInicio;
// End of variables declaration
}
```

En esta ventana gestionamos el asunto requerido por el cliente.

Ventana Procurador.

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

import ec.edu.ups.controlador.ControladorProcurador;
import ec.edu.ups.modelo.Procurador;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author NANCY
 */
public class VentanaProcurador extends javax.swing.JFrame {

    private ControladorProcurador controladorProcurador;
    /**
     * Creates new form VentanaProcurador
     */
    public VentanaProcurador(ControladorProcurador controladorProcurador) {
        initComponents();
        this.controladorProcurador= controladorProcurador;
    }

    public void cargarDatos() {
        DefaultTableModel modeloTabla = (DefaultTableModel) tblDatos.getModel();

        modeloTabla.setRowCount(0);
        for (Procurador procurador : controladorProcurador.getListadoProcurador()) {
            Object[] rowData =
{procurador.getCedula(),procurador.getNombre(),procurador.getApellido(),procurador.getCodigofk()};
            modeloTabla.addRow(rowData);
        }
        tblDatos.setModel(modeloTabla);
    }
}
```


```
public void limpiar(){
    txtCedula.setText("");
    txtNombre.setText("");
    txtApellido.setText("");
    txtCodigo.setText("");
}
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    txtCedula = new javax.swing.JTextField();
    txtNombre = new javax.swing.JTextField();
    jScrollPane1 = new javax.swing.JScrollPane();
    tblDatos = new javax.swing.JTable();
    txtApellido = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();
    btnAgregar = new javax.swing.JButton();
    btnActualizar = new javax.swing.JButton();
    btnEliminar = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    btnCancelar = new javax.swing.JButton();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    txtCodigo = new javax.swing.JTextField();

    setClosable(true);
    addInternalFrameListener(new javax.swing.event.InternalFrameListener() {
        public void internalFrameActivated(javax.swing.event.InternalFrameEvent evt) {
            formInternalFrameActivated(evt);
        }
        public void internalFrameClosed(javax.swing.event.InternalFrameEvent evt) {
        }
        public void internalFrameClosing(javax.swing.event.InternalFrameEvent evt) {
        }
        public void internalFrameDeactivated(javax.swing.event.InternalFrameEvent evt) {
        }
        public void internalFrameDeiconified(javax.swing.event.InternalFrameEvent evt) {
        }
        public void internalFrameIconified(javax.swing.event.InternalFrameEvent evt) {
        }
        public void internalFrameOpened(javax.swing.event.InternalFrameEvent evt) {
        }
    });

    tblDatos.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null}
    },
    new String [] {
        "Cedula", "Nombre", "Apellido", "Codigo asunto"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, false, false, false
    };

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});
tblDatos.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        tblDatosMouseClicked(evt);
    }
});
jScrollPane1.setViewportViewView(tblDatos);

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
jLabel5.setText("Procurador");

btnAgregar.setText("Agregar");
btnAgregar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnAgregarActionPerformed(evt);
    }
});

btnActualizar.setText("Actualizar");

btnEliminar.setText("Eliminar");
btnEliminar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnEliminarActionPerformed(evt);
    }
});

jLabel1.setText("Cedula");

btnCancelar.setText("Cancelar");


jLabel2.setText("Nombre");

jLabel3.setText("Apellido");

jLabel4.setText("Codigo Asunto");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);


```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(25, 25, 25)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(31, 31, 31)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(btnAgregar)
                        .addGap(26, 26, 26)
                        .addComponent(btnActualizar)
                        .addGap(18, 18, 18)
                        .addComponent(btnEliminar)
                        .addGap(18, 18, 18)
                        .addComponent(btnCancelar))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2)
                            .addComponent(jLabel3)
                            .addComponent(jLabel4))
                        .addGap(107, 107, 107)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(txtCedula, javax.swing.GroupLayout.DEFAULT_SIZE, 140, Short.MAX_VALUE)
                            .addComponent(txtNombre)
                            .addComponent(txtApellido)
                            .addComponent(txtCodigo))))))
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequentialGroup()
                .addGap(195, 195, 195)
                .addComponent(jLabel5)))
        .addContainerGap(35, Short.MAX_VALUE))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel5)
        .addGap(27, 27, 27)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel1)
            .addComponent(txtCedula, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(txtNombre, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel3)

```


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        .addComponent(txtApellido, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel4)
        .addComponent(txtCodigo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(29, 29, 29)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(btnAgregar)
        .addComponent(btnActualizar)
        .addComponent(btnEliminar)
        .addComponent(btnCancelar))
        .addGap(18, 18, 18)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(32, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>


private void tblDatosMouseClicked(java.awt.event.MouseEvent evt) {
    int fila = tblDatos.getSelectedRow();
    String cedula = (String)tblDatos.getValueAt(fila, 0);
    String nombre = (String)tblDatos.getValueAt(fila, 1);
    String Apellido = (String)tblDatos.getValueAt(fila, 2);
    int codigofk = (int)tblDatos.getValueAt(fila, 3);

    txtCedula.setText(cedula);
    txtNombre.setText(nombre);
    txtApellido.setText(Apellido);
    txtCodigo.setText(String.valueOf(codigofk));
}

private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    String cedula = txtCedula.getText();
    String nombre= txtNombre.getText();
    String apellido= txtApellido.getText();
    int codigofk= Integer.parseInt(txtCodigo.getText());
    boolean resultado = false;

    if(cedula.isEmpty()||nombre.isEmpty()||apellido.isEmpty()){
        JOptionPane.showMessageDialog(this, "Llene todos los campos solicitados");
    }else{
        Procurador procurador= new Procurador(cedula, nombre, apellido,codigofk);
        resultado= controladorProcurador.insertarAsunto(procurador);
        JOptionPane.showMessageDialog(this, "Operación : " + resultado);
        cargarDatos();
        limpiar();
        dispose();
    }
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    String cedula = txtCedula.getText();
    String nombre = txtNombre.getText();
    String apellido = txtApellido.getText();

    boolean resultado = false;
    if (cedula.isEmpty() || nombre.isEmpty() || apellido.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Llene todos los campos solicitados");
    } else {
        int eliminar = JOptionPane.showConfirmDialog(this, "¿Seguro quieres eliminar este Alumno?");
        if (eliminar == JOptionPane.YES_OPTION) {
            resultado= controladorProcurador.eliminar(cedula);
        }
        JOptionPane.showMessageDialog(this, "Operación : " + resultado);
        cargarDatos();
        limpiar();
        dispose();
    }
}


private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent evt) {
    controladorProcurador.cargardatos();
    cargarDatos();
}

// Variables declaration - do not modify
private javax.swing.JButton btnActualizar;
private javax.swing.JButton btnAgregar;
private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnEliminar;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable tblDatos;
private javax.swing.JTextField txtApellido;
private javax.swing.JTextField txtCedula;
private javax.swing.JTextField txtCodigo;
private javax.swing.JTextField txtNombre;
// End of variables declaration
}

```

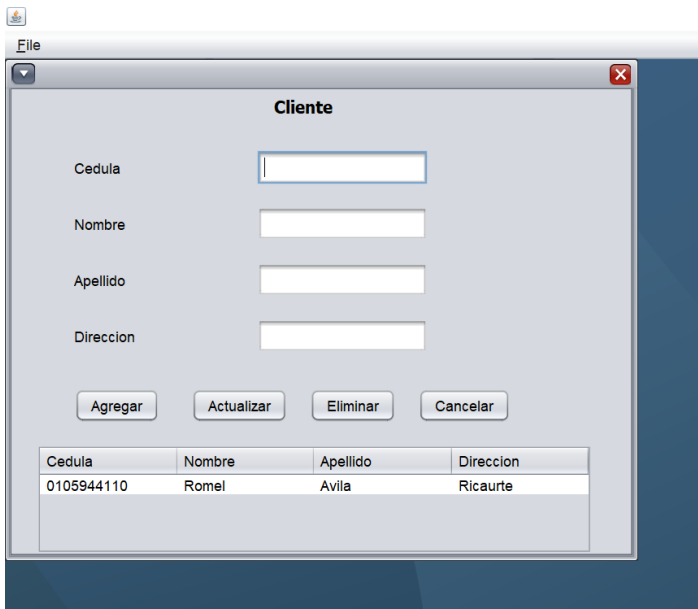
En esta ventana creamos al procurador para asignarle el caso o asunto requerido

Funcionalidad.

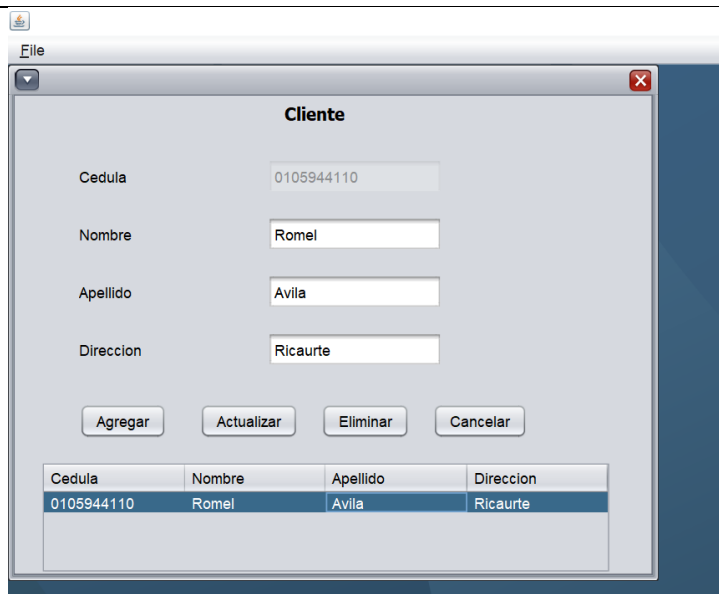
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



Al iniciar el programa nos aparece la siguiente ventana.

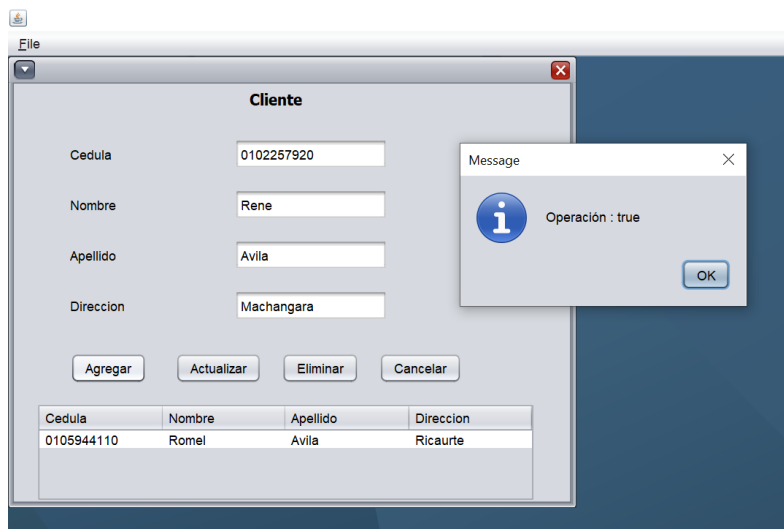


Si presionamos en cliente nos aparece la ventana para gestionar al cliente




Cedula	Nombre	Apellido	Direccion
0105944110	Romel	Avila	Ricaurte

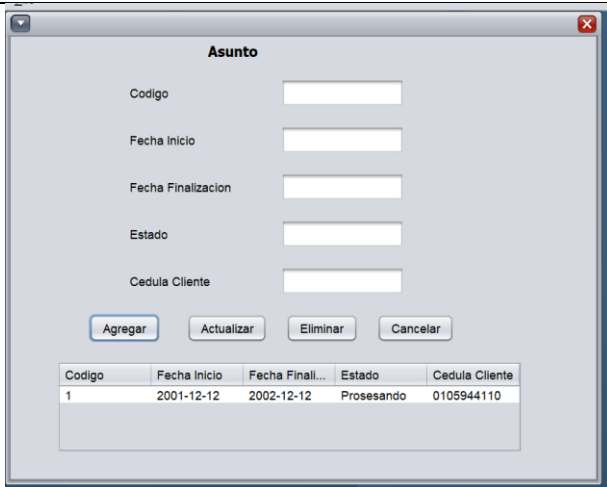
Si seleccionamos el cliente de la tabla nos deja editarlo



Cedula	Nombre	Apellido	Direccion
0105944110	Romel	Avila	Ricaurte

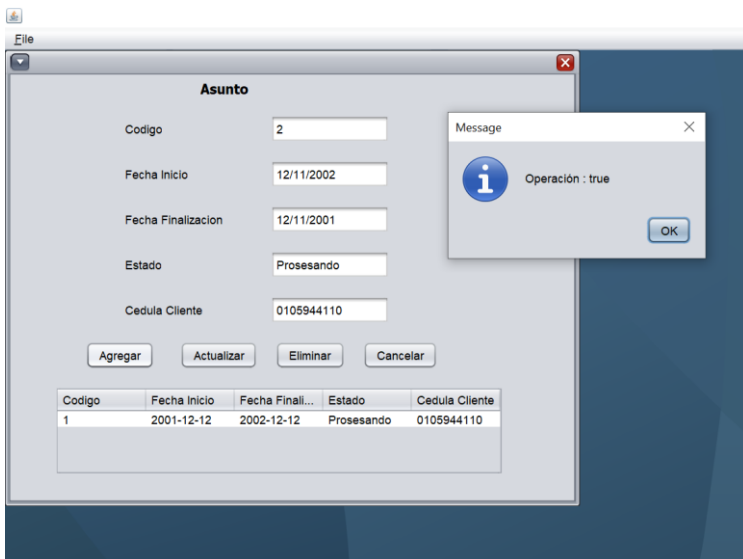
Creamos un cliente que se agregara en la tabla

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



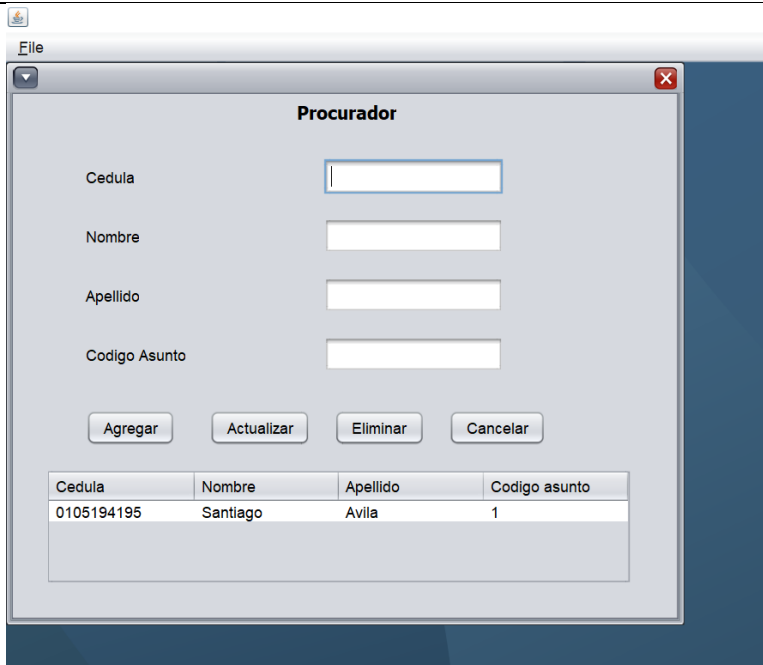
Codigo	Fecha Inicio	Fecha Finalizacion	Estado	Cedula Cliente
1	2001-12-12	2002-12-12	Prosesando	0105944110

Al presionar en asunto nos abre la siguiente ventana.



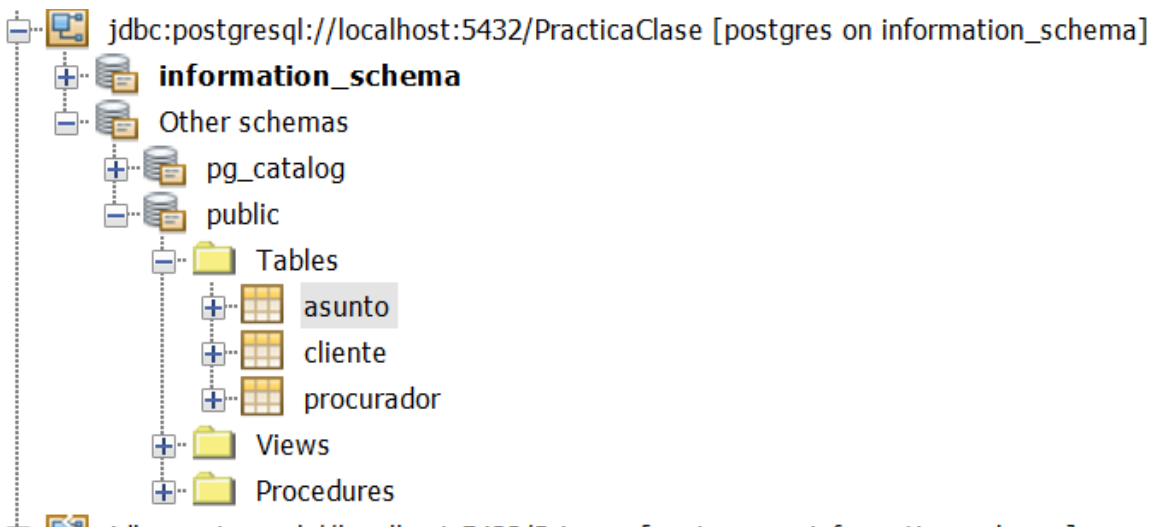
Codigo	Fecha Inicio	Fecha Finalizacion	Estado	Cedula Cliente
1	2001-12-12	2002-12-12	Prosesando	0105944110
2	12/11/2002	12/11/2001	Prosesando	0105944110

Al llenar los datos y poder guardar la información



Finalmente tenemos la ventana del procurador en la cual tenemos que poner el código del asunto que se le asignara al procurador.


Base de datos



Como se puede observar en la imagen tenemos creada ya la base de datos para verificar el uso de la misma con sentencias SQL.

RESULTADO(S) OBTENIDO(S):

Como resultados podemos encontrar el manejo de algunos comandos para poder manejar las bases de datos tanto con los datos guardados y así ser aplicados a un programa funcional sustituyendo los archivos de texto para el manejo de la información.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

CONCLUSIONES:

En conclusión, Podemos decir que aprendimos un poco más sobre el manejo de base de datos y ya tenemos un conocimiento introductorio al mismo gracias a esta práctica.

RECOMENDACIONES:

No hay recomendaciones

Estudiantes: Romel Ávila

Firma:

