



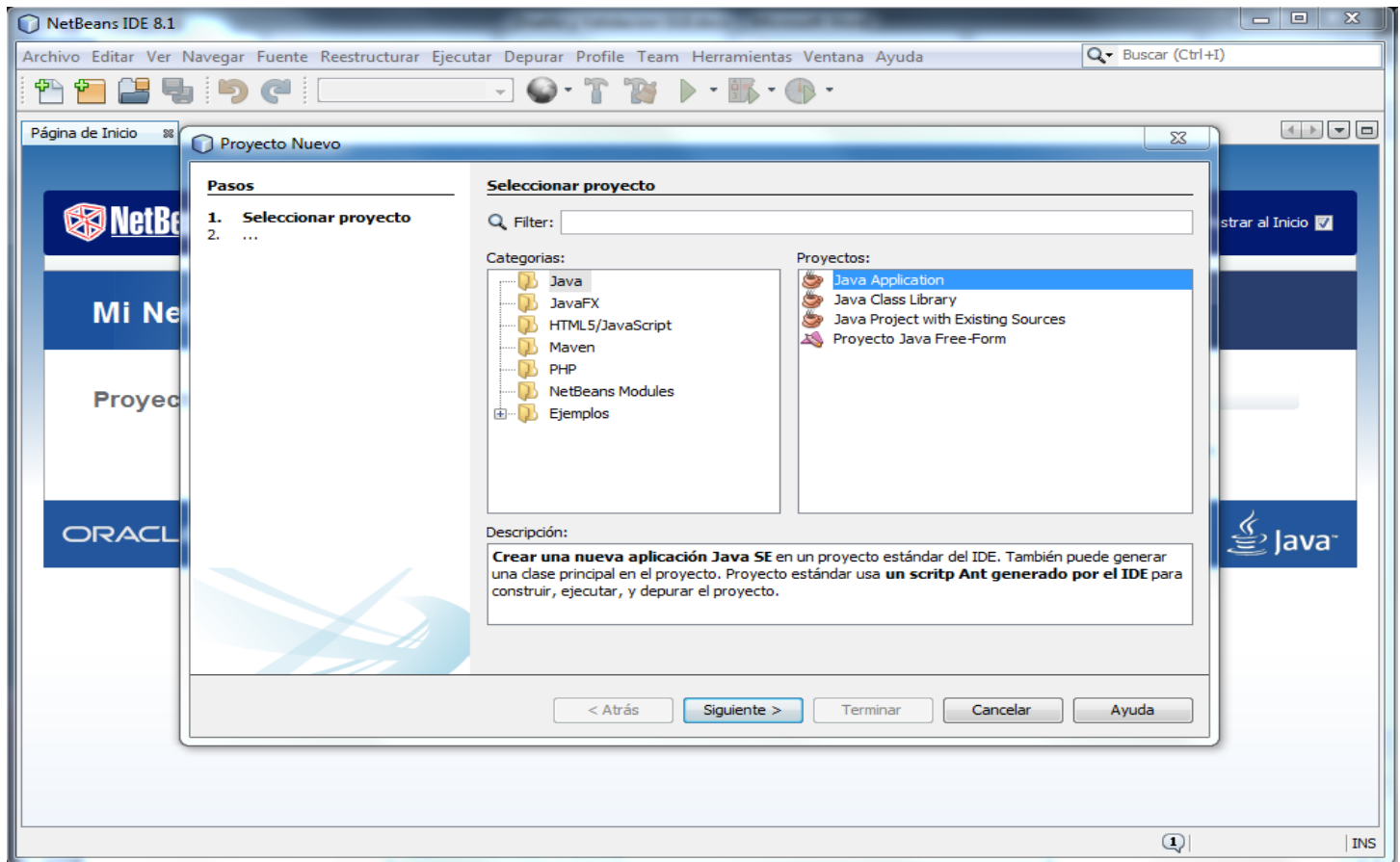
Para conectar una aplicación Java con bases de datos MySQL, se tener en cuenta lo siguiente:

1. Crear una clase java que permita manejar la conexión con la base de datos y la ejecución de las sentencias SQL, llamaremos a esta clase **ConectorBD** y estará en el proyecto de NetBeans.
2. Adicionar la Librería **Driver MySQL JDBC** al proyecto de NetBeans.
3. Crear la base de datos en MySQL, ya sea por consola o utilizando cualquier editor de sentencias SQL como el **HeidiSql** o **Workbench** de MySQL o con el mismo NetBeans.
4. Diseñar la GUI que permita interactuar con la Base de Datos

Con lo anterior creamos un proyecto nuevo llamado **PrjUniversidadBD**

## Utilizar el IDE NetBeans

- Acceder a **NetBeans**,
- al menú **Archivo**
- clic en **Proyecto Nuevo...**
- escoger en **Categorías Java** y en **Proyectos** la opción **Java Aplicación** clic en el botón **Siguiente**



Llenar los datos del proyecto:

- Nombre del proyecto: **PrjUniversidadBD**
  - Ubicación del Proyecto: Por defecto o donde queramos colocar nuestros proyectos de NetBeans
  - Crear clase principal: **pckuniversidad.Main**
- Clic en el botón **Terminar**

**Nuevo Aplicación Java**

**Pasos**

1. Seleccionar proyecto
2. **Nombre y ubicación**

**Nombre y ubicación** **PrjUniversidadBD**

Nombre proyecto: PrjUniversidadBD

Ubicación del proyecto: C:\Users\usuario\Documents\NetBeansProjects **Examinar...**

Carpeta proyecto: s\usuario\Documents\NetBeansProjects\PrjUniversidadBD

☐ Usar una carpeta dedicada para almacenar las bibliotecas

Carpeta de Bibliotecas: **Examinar...**

Usuarios y proyectos diferentes pueden compartir las mismas librerías de compilación (ver la Ayuda para más detalles).

☒ Crear clase principal pckuniversidad.Main

**pckuniversidad.Main**

< Atrás    Siguiete >    **Terminar**    Cancelar    Ayuda

1. Agregar una clase al proyecto: Código de la clase **ConectorBD.java**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ConectorBD {

    private Connection conexion;
    private Statement sentencia;

    //Datos para la conexion con la BD
    private final String servidor = "localhost";
    private final String puerto = "3306";
    private final String BD = "bd_universidad";
    private final String usuario = "root";
    private final String clave = "";
    private final String URL = "jdbc:mysql://" +
        servidor + ":" + puerto + "/" + BD;

    /**
     * Metodo constructor, que inicializa los atributos
     * internos del conector de BD
     */
    public ConectorBD() {
        this.conexion = null;
        this.sentencia = null;
    }
}
```

Continuación del código de la clase **ConectorBD.java**

```
/**
 * Metodo para crear la comunicacion con la BD
 * @return true cuando la conexion se crea correctamente,
 * y false cuando no es posible
 */
public boolean conectar() {
    boolean estado = false;
    try {
        //Levantar el Driver de Mysql
        Class.forName("com.mysql.jdbc.Driver");
        try {
            //Establecer la conexion con la BD
            conexion = DriverManager.getConnection(URL, usuario, clave);
            estado = true;
        } catch (SQLException ex) {
            System.err.println("ERROR: ConectorBD.conectar()");
            System.err.println("Al intentar la conexion con la BD");
            System.err.println(ex.getMessage());
        }
    } catch (ClassNotFoundException cex) {
        System.err.println("ERROR: ConectorBD.conectar()");
        System.err.println("No se encontro el Driver de Conexion con MySQL");
        System.err.println(cex.getMessage());
    }
    return estado;
}

/**
 * Para ejecutar sentencias SQL: SELECT
 * @param sql texto con el script sql para seleccionar registros
 * @return ResultSet con la informacion seleccionada o null en caso de no
 */
public ResultSet seleccionar(String sql) {
    ResultSet resultado = null;
    try {
        sentencia = conexion.createStatement();
        resultado = sentencia.executeQuery(sql);
    } catch (SQLException sqle) {
        System.err.println("ERROR: ConectorBD.seleccionar(sql)");
        System.err.println(sqle.getMessage());
    }
    return resultado;
}
```

Parte final de código de la clase **ConectorBD.java**

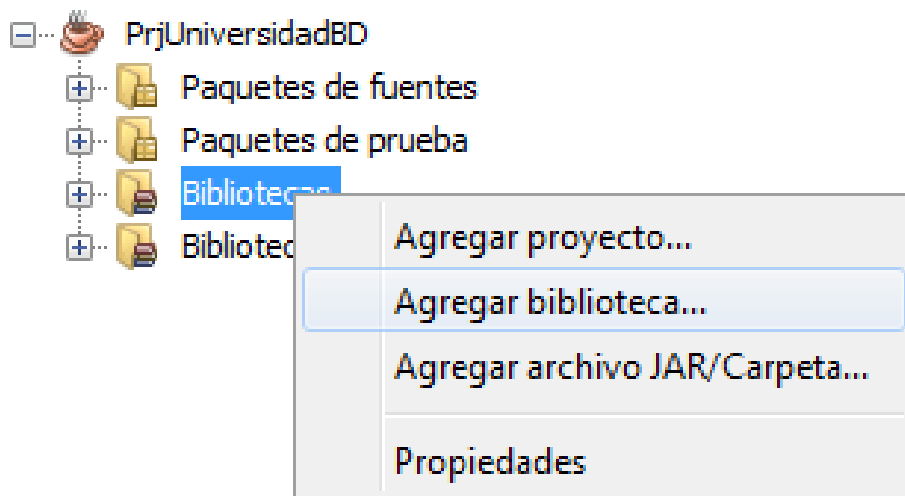
```
/**
 * Para ejecutar sentencias SQL: INSERT, UPDATE, DELETE
 * @param sql texto con el script sql para ejecutar sobre la BD
 * @return true si la sentencia tiene exito y
 * false en caso de no
 */
public boolean ejecutar(String sql){
    boolean estado = false;
    try{
        sentencia = conexion.createStatement();
        sentencia.execute(sql);
        sentencia.close();
        estado = true;
    }catch( SQLException sqle){
        System.err.println("ERROR: ConectorBD.ejecutar(sql)");
        System.err.println( sqle.getMessage() );
    }
    return estado;
}

/**
 * Para cerrar la conexion de forma correcta con la base de datos,
 * verificando que exista la conexion
 */
public void desconectar(){
    try{
        if( conexion != null ){
            conexion.close();
            conexion = null;
        }
    }catch( SQLException sqle){
        System.err.println("ERROR: ConectorBD.desconectar()");
        System.err.println( sqle.getMessage() );
    }
}

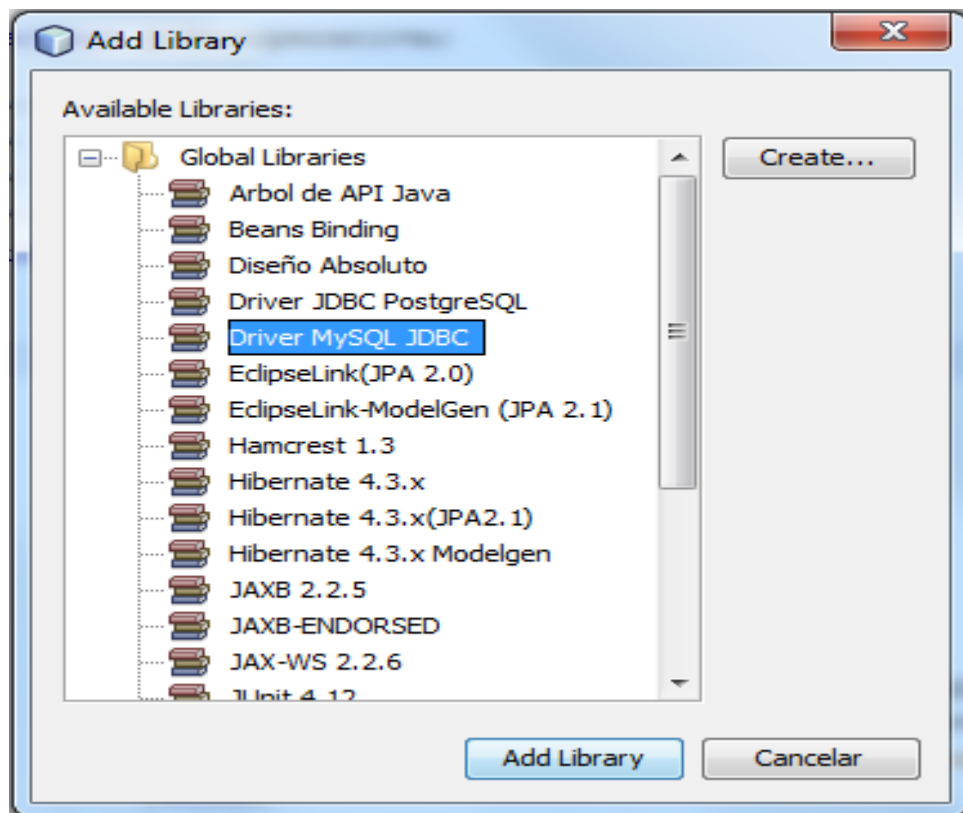
/**
 * Retorna la conexion actual que este establecida
 * @return conexion
 */
public Connection getConnection(){
    return conexion;
}
}

//Fin de la clase ConectorBD
```

2. Adicionar la librería de conexión al proyecto en NetBeans, haciendo clic derecho sobre **Bibliotecas** o **Libraries**



Después seleccionar la librería **Driver MySQL JDBC**



#### 4. Ahora debemos crear la base de datos en MySQL.

Script para crear la base de datos de la Universidad y activar su uso

```
DROP DATABASE IF EXISTS bd_universidad;  
  
CREATE DATABASE IF NOT EXISTS bd_universidad DEFAULT CHARSET=utf8;  
  
USE bd_universidad;
```

Script para crear la tabla de Estudiantes

```
DROP TABLE IF EXISTS tblestudiantes;  
  
CREATE TABLE IF NOT EXISTS tblestudiantes (  
  
    id_est BIGINT(11) NOT NULL PRIMARY KEY,  
    tipoid_est TINYINT(1) NOT NULL,  
    nombre_est VARCHAR(30) NOT NULL,  
    apellidos_est VARCHAR(30) NOT NULL,  
    genero_est VARCHAR(1) NOT NULL,  
    tiposangre_est VARCHAR(20) NOT NULL,  
    nacimiento_est DATE NOT NULL,  
    direccion_est VARCHAR(50) NOT NULL,  
    telefonos_est VARCHAR(50) NOT NULL,  
    correo_est VARCHAR(50) NOT NULL  
  
) ENGINE=InnoDB;
```

Adicionar la clase multipropósito, para facilitar ciertas tareas muy usadas, **Util.java**:

```
import java.awt.Component;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.awt.Color;  
import java.text.DecimalFormat;  
import javax.swing.JOptionPane;
```



```
/**
 * @author Ing. Emerson Emet Garay Gómez
 * Clase que proporciona métodos multipropósito
 */
public final class Util {

    public static String PATRON_FECHA_MYSQL = "YYYY-MM-dd";
    public static String PATRON_PESOS = "###,###.##";

    public static String aFechaMySQL(Date date){
        String fecha;
        SimpleDateFormat sdf = new SimpleDateFormat(PATRON_FECHA_MYSQL);
        fecha = sdf.format(date);

        return fecha;
    }

    public static String formatoPesos(double valor){
        DecimalFormat formateador = new DecimalFormat();
        String res = "$ " + formateador.format(valor);

        return res;
    }

    public static void informar(Component c, String mensaje, String titulo){
        JOptionPane.showMessageDialog(c, mensaje, titulo,
            JOptionPane.INFORMATION_MESSAGE);
    }

    public static void advertir(Component c, String mensaje, String titulo){
        JOptionPane.showMessageDialog(c, mensaje, titulo,
            JOptionPane.WARNING_MESSAGE);
    }

    public static String capturar(Component c, String mensaje, String titulo){
        String datos = JOptionPane.showInputDialog(c, mensaje, titulo,
            JOptionPane.QUESTION_MESSAGE);
        if( datos == null ){
            datos = "";
        }
        return datos;
    }
}
```

```
public static boolean confirmar(Component c, String mensaje, String titulo){
    int respuesta = JOptionPane.showConfirmDialog(c, mensaje, titulo,
        JOptionPane.YES_NO_OPTION);
    if( respuesta == JOptionPane.YES_OPTION ){
        return true;
    }else{
        return false;
    }
}

private static int hex( String color_hex ){
    return Integer.parseInt(color_hex, 16 );
}

public static Color COLOR_ENFOQUE = new Color( hex("CCFFCC") );
public static Color COLOR_SINENFOQUE = Color.WHITE;

} //Fin de la clase Util
```

REGISTRO DE ESTUDIANTES

IDENTIFICACION	TIPO IDENTIFICACION
<input type="text"/>	<input type="text"/>
NOMBRE	APELLIDOS
<input type="text"/>	<input type="text"/>
GENERO	TIPO DE SANGRE
<input type="text"/>	<input type="text"/>
FECHA NACIMIENTO	TELEFONOS
<input type="text"/>	<input type="text"/>
DIRECCION	CORREO ELECTRONICO
<input type="text"/>	<input type="text"/>






LIMPIAR GUARDAR BUSCAR ACTUALIZAR ELIMINAR

CERRAR

3. Agregar un **JFrame** al paquete **pckuniversidad**, para crear la GUI planteada:

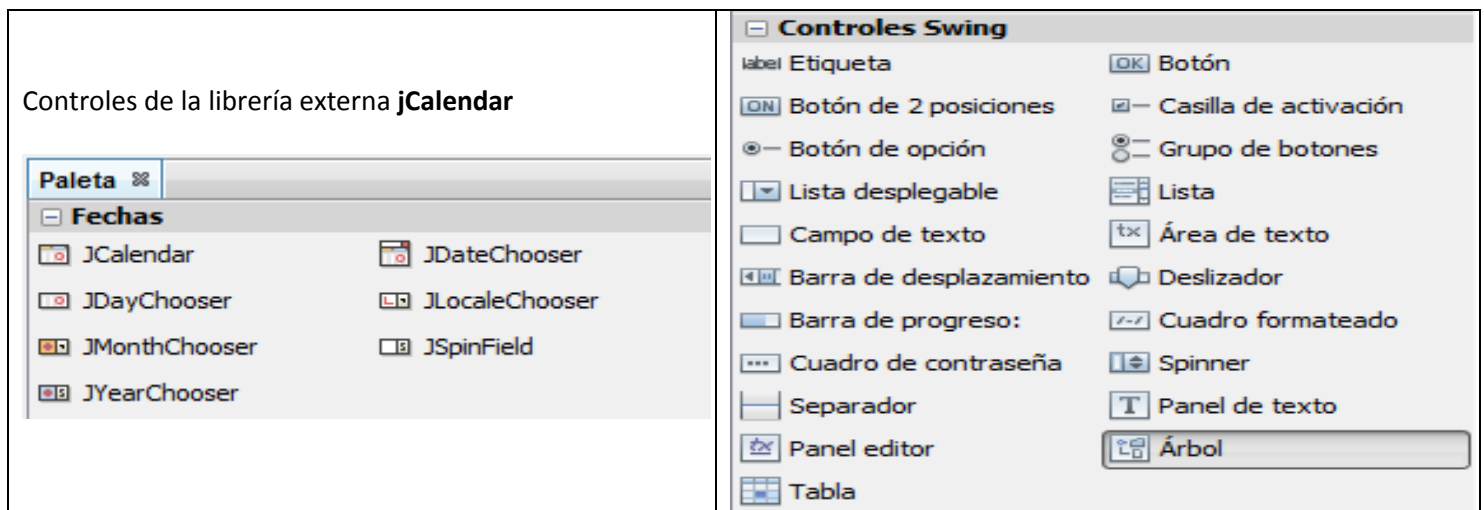
- Nombre de la clase: **FrmEstudiante**
- Paquete: **pckuniversidad**
- Clic en el botón **Terminar**

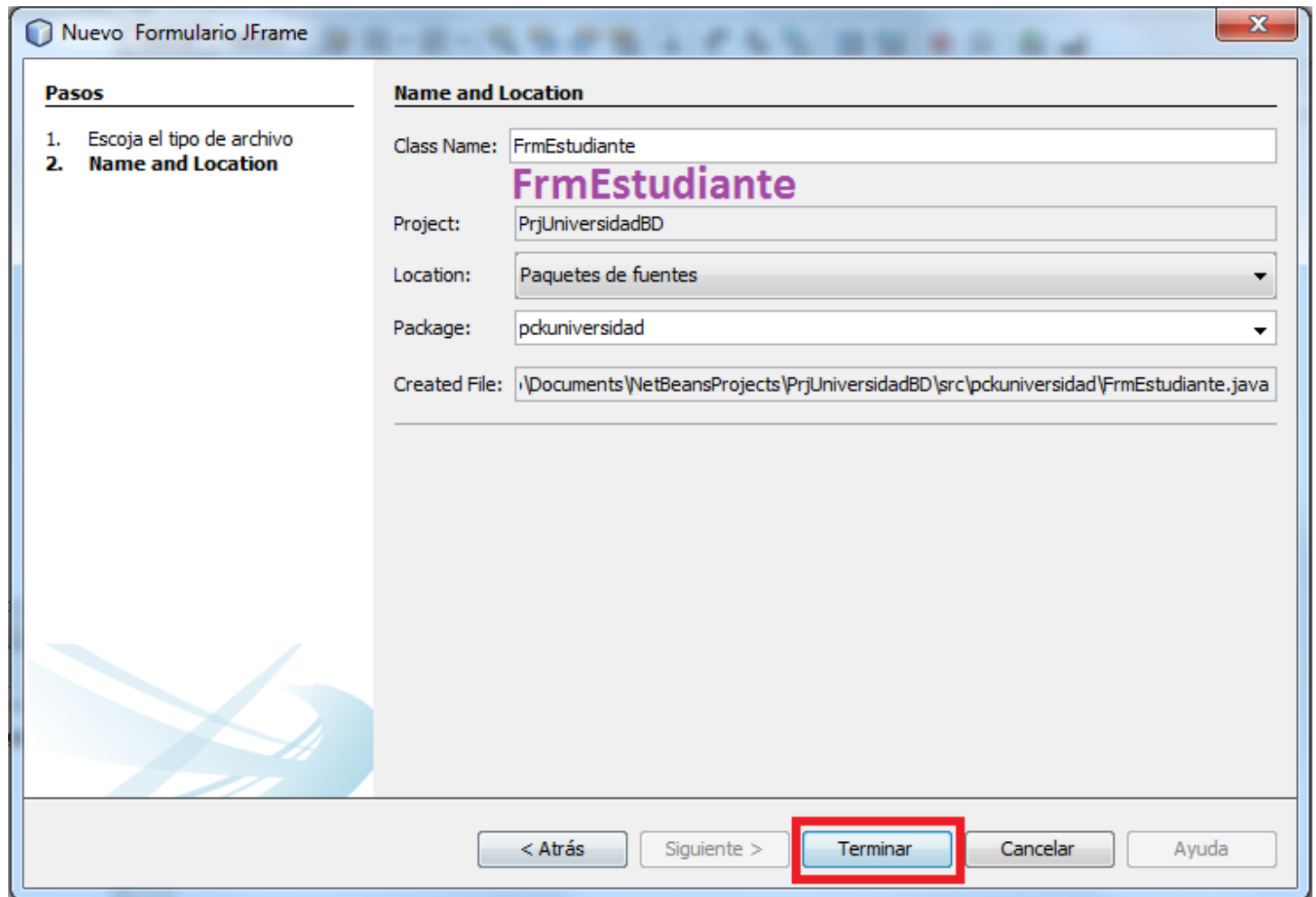
Detallado de los elementos o controles que contiene la interfaz.

1. 10 etiquetas – **JLabel**  Etiqueta
2. 6 Cajas de texto – **JTextField**  Campo de texto
3. 6 Botones de acción – **JButton**  Botón
4. 3 listas desplegables – **JComboBox**  Lista desplegable
5. Un seleccionador de fecha – **JDateChooser**  JDateChooser

Ahora vamos a crear lo anterior utilizando el **IDE NetBeans**

La construcción de la GUI es muy fácil con el IDE NetBeans, solo necesita ir seleccionado el control que quiere agregar al formulario, con un clic selecciona al elemento que está en la paleta de controles del NetBeans y se deja en el formulario en la posición que se requiera.








Adicionamos los controles con los siguientes nombres y propiedades.

(JFrame) Formulario	Propiedades			
Principal	title	resizable	layout	Código / Normas sobre el tamaño de los formularios
Formulario donde está la GUI	UNIVERSIDAD / REGISTRO DE ESTUDIANTES	false	null	Generar código para cambiar el tamaño

Componente <b>swing</b>	Propiedades		
Campo de Texto : <b>JTextField</b>  Campo de texto	Nombre del control	<b>editable</b>	<b>text</b>
1	txtId	true	
2	txtNombre	true	
3	txtApellidos	true	
4	txtTelefonos	true	
5	txtDireccion	true	
6	txtCorreo	true	

Componente <b>swing</b>	Propiedades	
Lista desplegable: <b>JComboBox</b>  Lista desplegable	Nombre del control	<b>model</b>
1	cmbTipold	*, REGISTRO CIVIL, TARJETA DE IDENTIDAD, CEDULA DE CIUDADANÍA
2	cmbGenero	*, FEMENINO, MASCULINO
3	cmbTipoSangre	*, A+, A-, B+, B-, AB+, AB-, O+, O-

Componente <b>swing</b> (Librería externa <b>jCalendar</b> )	Propiedades	
Seleccionador de Fecha : <b>JDateChooser</b>  <b>JDateChooser</b>	Nombre del control	<b>dateFormatString</b>
<b>1</b>	jdcNacimiento	d/MM/yyyy

Componente <b>swing</b>	Propiedades		
Botón de acción : <b>JButton</b>  <b>Botón</b>	Nombre del control	<b>text</b>	<b>foreground</b>
<b>1</b>	butLimpiar	LIMPIAR	[0,0,255]
<b>2</b>	butGuardar	GUARDAR	[0,0,255]
<b>3</b>	butBuscar	BUSCAR	[0,0,255]
<b>4</b>	butActualizar	ACTUALIZAR	[0,0,255]
<b>5</b>	butEliminar	ELIMINAR	[0,0,255]
<b>6</b>	butCerrar	CERRAR	[255,0,0]

**Definición de las variables globales para el formulario de Estudiantes a utilizar.**

```
private final String TABLA = "tblestudiantes";
private final ConectorBD CBD = new ConectorBD();

private String sql;
private String ID, TIPOID, NOMBRE, APELLIDOS, GENERO, TIPOSANGRE;
private String NACIMIENTO, TELEFONOS, DIRECCION, CORREO;
```

## Método limpiar GUI

```
private void limpiarGUI() {
    this.txtID.setText("");
    this.txtNombre.setText("");
    this.txtApellidos.setText("");
    this.jdcNacimiento.setDate(new Date());
    this.txtTelefonos.setText("");
    this.txtDireccion.setText("");
    this.txtCorreo.setText("");

    this.cboTipoID.setSelectedIndex(0);
    this.cboGenero.setSelectedIndex(0);
    this.cboTipoSangre.setSelectedIndex(0);

    this.txtID.setEnabled(true);
    this.btnGuardar.setEnabled(true);
    this.btnActualizar.setEnabled(false);
    this.btnEliminar.setEnabled(false);

    this.txtID.grabFocus(); //cursor en este campo
}
```

```
private void obtenerDatosGUI() {
    ID = txtID.getText();
    TIPOID = String.valueOf( cboTipoID.getSelectedIndex() );
    NOMBRE = txtNombre.getText().toUpperCase();
    APELLIDOS = txtApellidos.getText().toUpperCase();
    GENERO = (cboGenero.getSelectedItem().toString().charAt(0) + "");
    TIPOSANGRE = cboTipoSangre.getSelectedItem().toString();
    NACIMIENTO = Util.aFechaMySQL( jdcNacimiento.getDate() );
    TELEFONOS = txtTelefonos.getText();
    DIRECCION = txtDireccion.getText();
    CORREO = txtCorreo.getText();
}
```

## Crear el método para **almacenar** la información:

- Este método recibe los datos que se introdujeron en la GUI
- Se establece la conexión con la BD, si es correcta entonces se arma la sentencia SQL **INSERT** para registrar los datos recibidos.
- Ejecutar la sentencia SQL, en caso de éxito el método retornara **true** en caso contrario **false**, eso sí, antes terminamos la conexión.

```
private boolean guardar(){
    boolean estado = false;

    if( CBD.conectar() ){
        sql = "INSERT INTO " + TABLA + " VALUES ( " +
            ID + "," +
            TIPOID + "," +
            "'" + NOMBRE + "'" + "," +
            "'" + APELLIDOS + "'" + "," +
            "'" + GENERO + "'" + "," +
            "'" + TIPOSANGRE + "'" + "," +
            "'" + NACIMIENTO + "'" + "," +
            "'" + DIRECCION + "'" + "," +
            "'" + TELEFONOS + "'" + "," +
            "'" + CORREO + "'" +
            " );";

        if( CBD.ejecutar(sql) ){
            estado = true;
        }
        CBD.desconectar();
    }else{
        Util.advertir(this, "ERROR: Verifique la conexion con la BD", getTitle());
    }

    return estado;
}
```

#### Código para el evento clic del botón **guardar**:

- Obtenemos los datos desde la GUI, para su validación, en caso de ser validados con éxito se muestra un mensaje que lo indica, después se toman los datos restantes de la GUI para enviarlos hasta la función que permite **guardarlos** dentro de la BD.
- Se muestra un mensaje informado el resultado de la acción.

```
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    obtenerDatosGUI();
    if( guardar() ){
        Util.informar(this,"Datos guardados correctamente.", getTitle());
        limpiarGUI();
    }else{
        Util.advertir(this,"ERROR: No se guardaron los datos.", getTitle());
    }
}
```



**Crear el método para **actualizar** la información:**

- Este método recibe los datos que se introdujeron en la GUI, similar al de guardar.
- Se establece la conexión con la BD, si es correcta entonces se arma la sentencia SQL **UPDATE** para **actualizar** con los datos recibidos.
- Ejecutar la sentencia SQL, en caso de éxito el método retornara **true** en caso contrario **false**, eso sí, antes terminamos la conexión.

```
private boolean actualizar(){
    boolean estado = false;

    if( CBD.conectar() ){
        sql = "UPDATE " + TABLA + " SET " +
            "tipoid_est = " + TIPOID + "," +
            "nombre_est = " + "'" + NOMBRE + "'" + "," +
            "apellidos_est = " + "'" + APELLIDOS + "'" + "," +
            "genero_est = " + "'" + GENERO + "'" + "," +
            "tiposangre_est = " + "'" + TIPOSANGRE + "'" + "," +
            "nacimiento_est = " + "'" + NACIMIENTO + "'" + "," +
            "direccion_est = " + "'" + DIRECCION + "'" + "," +
            "telefonos_est = " + "'" + TELEFONOS + "'" + "," +
            "correo_est = " + "'" + CORREO + "'" +
            " WHERE id_est = " + ID + ";";

        if( CBD.ejecutar(sql) ){
            estado = true;
        }
        CBD.desconectar();
    }else{
        Util.advertir(this, "ERROR: Verifique la conexion con la BD", getTitle());
    }

    return estado;
}
```

### Código para el evento clic del botón **actualizar**:

- Obtenemos los datos desde la GUI, para su validación, en caso de ser validados con éxito se muestra un mensaje que lo indica, después se toman los datos restantes de la GUI para enviarlos hasta la función que permite **actualizarlos** dentro de la BD.
- Se muestra un mensaje informado el resultado de la acción.

```
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {  
    obtenerDatosGUI();  
    if( actualizar() ){  
        Util.informar(this, "Datos actualizados correctamente.", getTitle());  
        limpiarGUI();  
    }else{  
        Util.advertir(this, "ERROR: No se actualizaron los datos.", getTitle());  
    }  
}
```

### Crear el método para **buscar** la información:

- Este método recibe el número del documento que se haya digitado en la GUI.
- Se establece la conexión con la BD, si es correcta entonces se arma la sentencia SQL **SELECT** para **buscar** los datos según la identificación recibida.
- Ejecutar la sentencia SQL, en caso de éxito el método retornara **true** en caso contrario **false**, eso sí, antes terminamos la conexión.

```
private boolean buscar(String id){
    boolean estado = false;

    if( CBD.conectar() ){
        sql = "SELECT * FROM " + TABLA + " WHERE id_est=" + id + ";";
        ResultSet regs = CBD.seleccionar(sql);
        try {
            if( regs.next() ){
                txtID.setText( String.valueOf( regs.getInt("id_est") ) );
                cboTipoID.setSelectedIndex( regs.getInt("tipoid_est") );
                txtNombre.setText( regs.getString("nombre_est") );
                txtApellidos.setText( regs.getString("apellidos_est") );
                char c = regs.getString("genero_est").charAt(0);
                int i = ( c == '*' ? 0 : ( c == 'F' ? 1 : 2 ) );
                cboGenero.setSelectedIndex( i );
                cboTipoSangre.setSelectedItem( regs.getString("tiposangre_est") );
                jdcNacimiento.setDate( regs.getDate("nacimiento_est") );
                txtTelefonos.setText( regs.getString("telefonos_est") );
                txtDireccion.setText( regs.getString("direccion_est") );
                txtCorreo.setText( regs.getString("correo_est") );
                estado = true;
            }
        } catch (SQLException sqle) {
            System.out.println("ERROR: buscar()");
            System.err.println( sqle.getMessage() );
        }
        CBD.desconectar();
    }else{
        Util.advertir(this, "ERROR: Verifique la conexion con la BD", getTitle());
    }

    return estado;
}
```

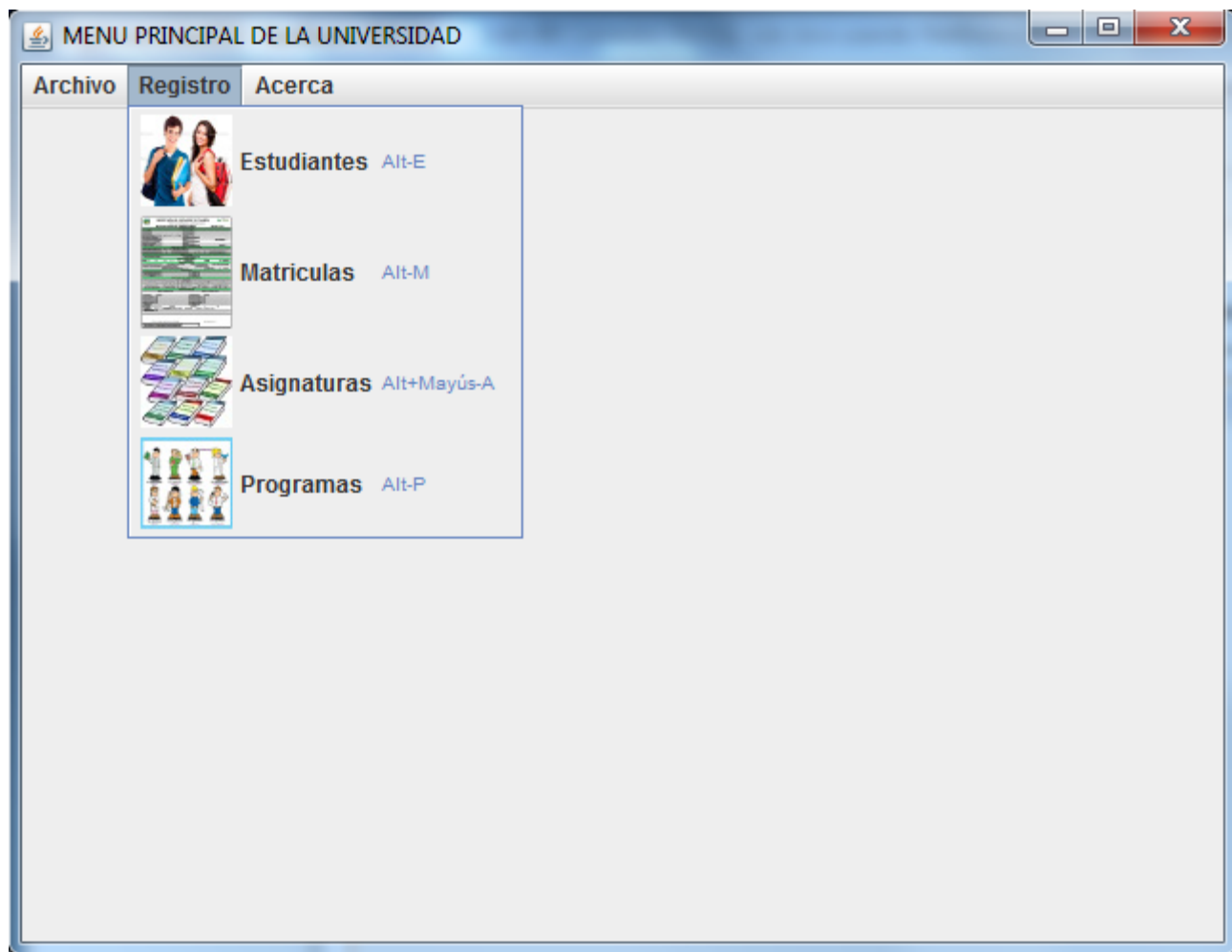
#### Código para el evento clic del botón **buscar**:

- Obtenemos la identificación desde la GUI después se envía hasta la función que permite **buscarlo** dentro de la BD.
- Se muestra un mensaje informado el resultado de la acción.

Método clic para el botón Buscar.

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {  
    String id = Util.capturar(this, "Digite la identificacion", getTitle());  
    limpiarGUI();  
    if( buscar(id) ){  
        btnGuardar.setEnabled(false);  
        btnActualizar.setEnabled(true);  
        btnEliminar.setEnabled(true);  
        txtID.setEnabled(false);  
        Util.informar(this,"Datos buscados correctamente.", getTitle());  
    }else{  
        Util.advertir(this,"No se encontraron Datos.", getTitle());  
    }  
}
```

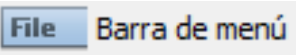
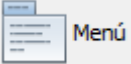
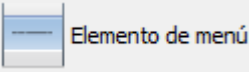
Crear el formulario **Menu.java**



4. Agregar un **JFrame** al paquete **pckuniversidad**, para crear la GUI planteada:

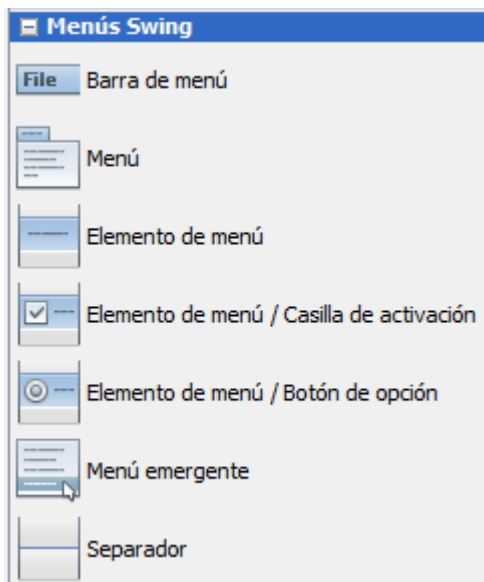
- Nombre de la clase: **FrmMenu**
- Paquete: pckuniversidad
- Clic en el botón **Terminar**

Detallado de los elementos o controles que contiene la interfaz.

6. 1 Barra de menú – **JMenuBar** 
7. 3 Menu – **JMenu** 
8. 6 elementos de menu – **JMenuItem** 

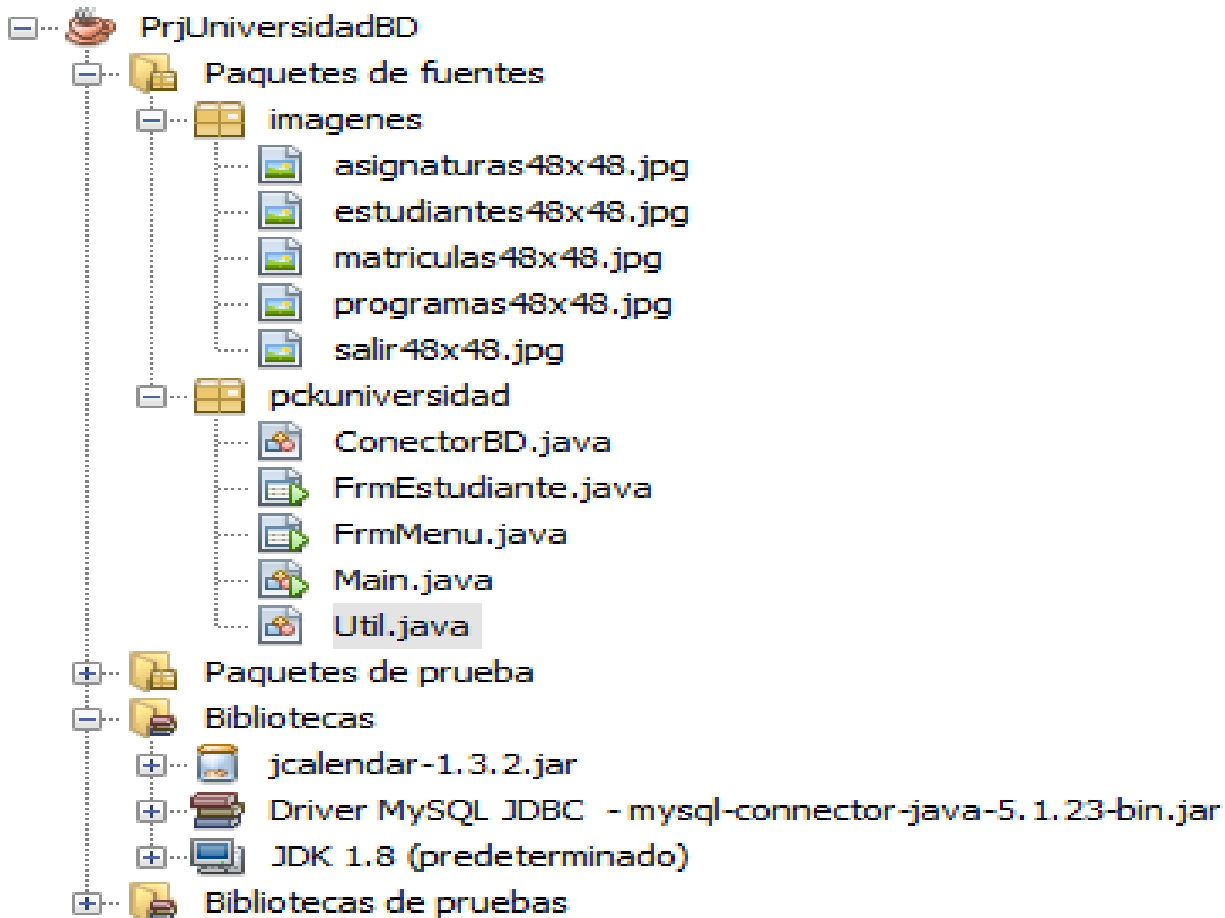
Ahora vamos a crear lo anterior utilizando el **IDE NetBeans**

La construcción de la GUI es muy fácil con el IDE NetBeans, solo necesita ir seleccionado el control que quiere agregar al formulario, con un clic selecciona al elemento que está en la paleta de controles del NetBeans y se deja en el formulario en la posición que se requiera.



Por ultimo ejecutar la aplicación.

Ahora el proyecto debe tener una estructura como la siguiente:



Notemos las partes más relevantes:

- EL paquete **pckuniversidad** contiene 5 clases
  - **ConectorBD**: Clase para manejar la conexión entre Java y **MySQL**
  - **FrmMenu**: GUI con el menú de la aplicación
  - **FrmEstudiantes**: GUI con el formulario para registrar los estudiantes
  - **Main**: Permite ejecutar una instancia del formulario de Menú
  - **Util**: Clase para manejar métodos de utilidad
- Las bibliotecas o Libraries tiene tres asociaciones
  - Librería jCalendar: para manejar la fecha con el **JDateChooser**
  - **Driver MySQL JDBC**: para la comunicación entre Java y **MySQL**
  - El kit de desarrollo de java o JDK: según la versión que tenga el pc instalada