



Tecnología y educación permanente

MATERIA:

PROGRAMACION ORIENTADA OBJETOS/POO

Profesor:

Rolando Antonio Del Rosario Mariñez

ID:

10153998

ESTUDIANTE:

Esmerly Rafael Fernando Roman perez

Santiago de los caballeros, República Dominicana

Tabla de contenido

- ❖ Requerimientos
- ❖ Crear base de datos
- ❖ Crear proyecto
- ❖ Diseño de la vista
- ❖ Conexión a MySQL
 - Método para conexión
- ❖ Métodos CRUD
 - Método del botón para guardar
 - Método del botón para buscar
 - Método del botón para editar
 - Método del botón para eliminar
 - Método del botón limpiar

JAVA APLICACIÓN PROCEDIMIENTO

Durante EL desarrollo de aplicacion en Java y MySQL aprendemos a crear un CRUD, que es el acronimo de Create, Read, Update and Delete (Crear, Leer, Actualizar y Borrar), que se usa para referirse a las transacciones básicas en bases de datos.

Primero aprenderemos a realizar la aplicación con programación estructurada para familiarizarnos con el lenguaje de programación Java y la integración de datos de MySQL. Más adelante haremos una aplicación similar pero implementado la arquitectura MVC.

Instalación y Configuración

Requisitos Previos

Java Development Kit (JDK): Debe tener el JDK instalado. Este proyecto requiere JDK 8.0 o superior. Puedes descargarlo desde Oracle JDK o cualquier otra distribución de Java que prefieras.

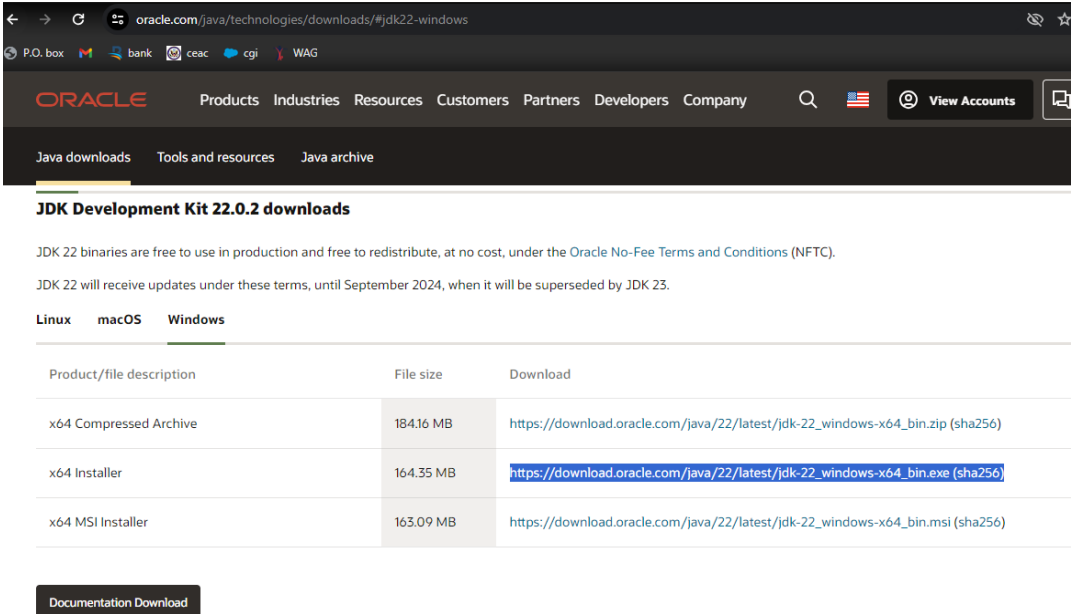
PASOS PARA INTALAR:

Para descargar el JDK.

<https://www.oracle.com/java/technologies/downloads/#jdk22-windows>

Seleccionar sistema operativo y descargar el paquete que desea.

El que recomiendo es el marcado en azul.



The screenshot shows the Oracle JDK 22.0.2 downloads page for Windows. The page has a dark header with the Oracle logo and navigation links. Below the header, there's a section titled "JDK Development Kit 22.0.2 downloads". It includes a note about the Oracle No-Fee Terms and Conditions (NFTC) and a statement that JDK 22 will receive updates until September 2024. There are tabs for Linux, macOS, and Windows, with Windows selected. A table lists the download options for Windows x64:

Product/file description	File size	Download
x64 Compressed Archive	184.16 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.zip (sha256)
x64 Installer	164.35 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.09 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.msi (sha256)

At the bottom of the table, there is a button labeled "Documentation Download".

Luego el procedimiento es el instalación.

Crear base de datos

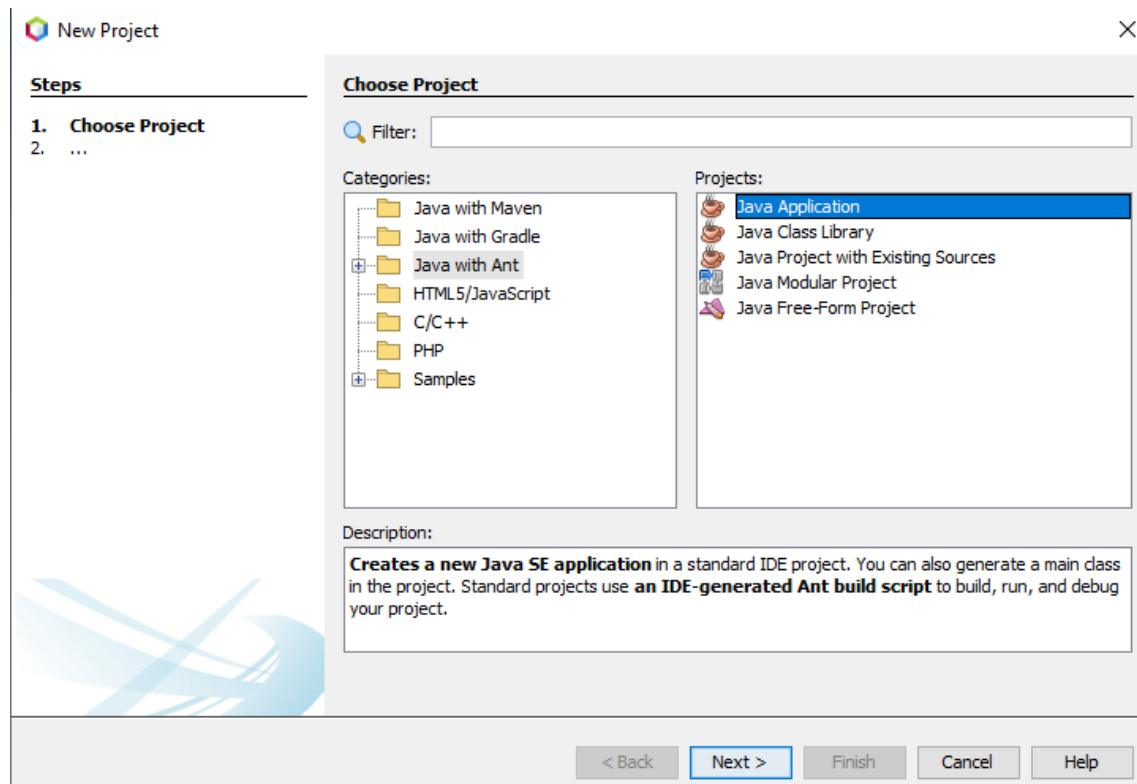
```
CREATE DATABASE IF NOT EXISTS `escuela` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;

USE `escuela`;

CREATE TABLE `persona` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `clave` varchar(10) COLLATE utf8mb4_unicode_ci NOT NULL,
  `nombre` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `domicilio` varchar(200) COLLATE utf8mb4_unicode_ci NOT NULL,
  `telefono` varchar(15) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `correo_electronico` varchar(45) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `fecha_nacimiento` date DEFAULT NULL,
  `genero` varchar(10) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Crear proyecto

El proyecto lo hemos desarrollado en el IDE **Netbeans** actualmente **Apache Netbeans**. Creamos un proyecto de tipo **Java Application** y desmarcamos la casilla Create Main Class:



Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class

< Back

Next >

Finish

Cancel

Help

Esto creará un proyecto vacío por lo cual agregaremos un paquete dan clic derecho sobre el nombre del proyecto > New > Java Package y lo nombramos como app.

Dentro de este paquete agregamos un JFrame dando clic derecho sobre el nombre del paquete > New > JFrame Form y lo nombramos como persona.

La estructura del proyecto se será así:



Diseño de la vista

Para agregar el diseño de nuestro formulario abrimos el archivo **persona.java** dando doble clic, agregamos el siguiente diseño:

The screenshot shows the 'Design' view of a Java Swing window. The window contains a form with the following elements:

- Clave:** A text field followed by a small square button.
- Nombre:** A text field.
- Domicilio:** A text field.
- Telefono:** A text field.
- Email:** A text field.
- Fecha Nacimiento:** A text field.
- Genero:** A dropdown menu with the text 'Selecciona' and a downward arrow.
- Buttons:** Four buttons at the bottom: 'Guarda', 'Modifica', 'Elimina', and 'Limpiar'.
- Search Button:** A button labeled 'Buscar' located next to the 'Clave' field.

Descripción de los elementos

Elemento	Texto	Variable
JLabel	Clave	
JLabel	Nombre	
JLabel	Domicilio	
JLabel	Telefono	
JLabel	Email	
JLabel	Fecha de nacimiento	
JLabel	Genero	
TextField		txtClave
TextField		txtId
TextField		txtNombre
TextField		txtDomicilio
TextField		txtTelefono
TextField		txtEmail
TextField		txtFecha
ComboBox		cbxGenero
Button	Buscar	btnBuscar
Button	Guarda	btnGuarda
Button	Modifica	btnModifica
Button	Elimina	btnElimina
Button	Limpiar	btnLimpiar

Para el elemento JComboBox ve a la propiedades y en Model agrega las opciones:

- ❖ Selecciona
- ❖ Masculino
- ❖ Femenino

Conexión a MySQL

Para conectar Java con MySQL se necesita un controlador JDBC compatible entre la versión JDK y MySQL. Descargar

<https://dev.mysql.com/downloads/connector/jj/>

Después de descargar el controlador vamos al proyecto y damos clic derecho en la carpeta **Libraries** y seleccionamos **Add JAR/Folder**, buscamos el controlador **.jar** y lo agregamos. Teniendo el controlador agregado ya podemos conectarnos a la base de datos de MySQL para esto regresamos al JFrame pero ahora seleccionamos la pestaña source para ver el código fuente del formulario.

Método para conexión

Dentro del código fuente de la clase persona.java trabajaremos después del constructor que es el método que se llama igual que la clase.

```
public static Connection getConnection() {  
    Connection con = null;  
    String base = "escuela"; //Nombre de la base de datos  
    String url = "jdbc:mysql://localhost:3306/" + base; //Direccion, puerto y nombre de la Base de Datos  
    String user = "root"; //Usuario de Acceso a MySQL  
    String password = "password"; //Password del usuario  
  
    try {  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        con = DriverManager.getConnection(url, user, password);  
    } catch (ClassNotFoundException | SQLException e) {  
        System.err.println(e);  
    }  
    return con;  
}
```

Métodos CRUD

Después de realizar la conexión a MySQL crearemos los métodos para las transacciones básicas; Registrar, Leer o Buscar, Modificar y Eliminar. Para estos se necesita implementar eventos de ActionListener la cual se usa para detectar y manejar eventos de acción, como el clic en los botones.

Podemos activar este eventos de forma automática regresando a la vista diseño y dando doble clic en los botones o dando clic derecho sobre el botón > Events > Action > actionPerformed.

Antes de agregar el código a los botón crearemos un método que elimine el contenido de las cajas de texto (JTextField).

```
private void limpiarCajas() {  
    txtClave.setText(null);  
    txtNombre.setText(null);  
    txtDomicilio.setText(null);  
    txtTelefono.setText(null);  
    txtEmail.setText(null);  
    txtFecha.setText(null);  
    cbxGenero.setSelectedIndex(0);  
}
```

Método del botón para guardar

```
private void btnGuardaActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Connection con;  
  
    try {  
        con = getConection();  
        ps = con.prepareStatement("INSERT INTO persona (clave, nombre, domicilio, telefono,  
        correo_electronico, fecha_nacimiento, genero) VALUES(?,?,?,?,?,?,?) ");  
        ps.setString(1, txtClave.getText());  
        ps.setString(2, txtNombre.getText());  
        ps.setString(3, txtDomicilio.getText());  
        ps.setString(4, txtTelefono.getText());  
        ps.setString(5, txtEmail.getText());  
        ps.setDate(6, Date.valueOf(txtFecha.getText()));  
        ps.setString(7, cbxGenero.getSelectedItem().toString());  
  
        int res = ps.executeUpdate();  
  
        if (res > 0) {  
            JOptionPane.showMessageDialog(null, "Persona Guardada");  
        } else {  
            JOptionPane.showMessageDialog(null, "Error al Guardar persona");  
        }  
  
        limpiarCajas();  
        con.close();  
  
    } catch (HeadlessException | SQLException e) {  
        System.err.println(e);  
    }  
}
```


Método del botón para buscar

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Connection con;  
  
    try {  
        con = getConexion();  
        ps = con.prepareStatement("SELECT * FROM persona WHERE clave = ?");  
        ps.setString(1, txtClave.getText());  
  
        rs = ps.executeQuery();  
  
        if (rs.next()) {  
            txtId.setText(rs.getString("id"));  
            txtNombre.setText(rs.getString("nombre"));  
            txtDomicilio.setText(rs.getString("domicilio"));  
            txtTelefono.setText(rs.getString("telefono"));  
            txtEmail.setText(rs.getString("correo_electronico"));  
            txtFecha.setText(rs.getString("fecha_nacimiento"));  
            cbxGenero.setSelectedItem(rs.getString("genero"));  
        } else {  
            JOptionPane.showMessageDialog(null, "No existe una persona con la clave");  
            limpiarCajas();  
        }  
  
    } catch (HeadlessException | SQLException e) {  
        System.err.println(e);  
    }  
}
```

Método del botón para editar

```
private void btnModificaActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Connection con;  
  
    try {  
        con = getConexion();  
        ps = con.prepareStatement("UPDATE persona SET clave=?, nombre=?, domicilio=?, telefono=?,  
correo_electronico=?, fecha_nacimiento=?, genero=? WHERE id=?");  
        ps.setString(1, txtClave.getText());  
        ps.setString(2, txtNombre.getText());  
        ps.setString(3, txtDomicilio.getText());  
        ps.setString(4, txtTelefono.getText());  
        ps.setString(5, txtEmail.getText());  
        ps.setDate(6, Date.valueOf(txtFecha.getText()));  
        ps.setString(7, cbxGenero.getSelectedItem().toString());  
        ps.setString(8, txtId.getText());  
  
        int res = ps.executeUpdate();  
  
        if (res > 0) {  
            JOptionPane.showMessageDialog(null, "Persona Modificada");  
        } else {  
            JOptionPane.showMessageDialog(null, "Error al Modificar persona");  
        }  
        limpiarCajas();  
        con.close();  
  
    } catch (HeadlessException | SQLException e) {  
        System.err.println(e);  
    }  
}
```

Método del botón para eliminar

```
private void btnEliminaActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Connection con;  
  
    try {  
        con = getConexion();  
        ps = con.prepareStatement("DELETE FROM persona WHERE id=?");  
        ps.setInt(1, Integer.parseInt(txtId.getText()));  
  
        int res = ps.executeUpdate();  
  
        if (res > 0) {  
            JOptionPane.showMessageDialog(null, "Persona Eliminada");  
        } else {  
            JOptionPane.showMessageDialog(null, "Error al eliminar persona");  
        }  
        limpiarCajas();  
        con.close();  
    } catch (HeadlessException | NumberFormatException | SQLException e) {  
        System.err.println(e);  
    }  
}
```

Método del botón limpiar

```
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {  
    limpiarCajas();  
}
```

Como adicional podemos hacer que el campo de texto para guardar temporalmente el ID del registro no sea visible. Para esto agregamos la propiedad **setVisible(false)** a la variable **txtId**, esto será en el método constructor después de iniciar los componentes

```
public persona() {  
    initComponents();  
    txtId.setVisible(false);  
}
```