CONEXIÓN BLUEJ CON BD MYSQL

Vamos a realizar un CRUD, que es una palabra acrónimo del habla inglesa que quiere decir create, read, update, delete que en español significa: crear, leer, actualizar, eliminar y que se refiere a las operaciones con registros en una base de datos.

Para este ejemplo utilizaremos los patrones de diseño DAO y MVC. La utilización de patrones de diseño permite la correcta aplicación de buenas prácticas en cuanto a la solución en problemas de desarrollo de software.

Otra característica importante del uso de patrones de diseño de software es que permite dividir en partes de alguna manera independientes, ya que si deseo modificar la vista no afectaría el modelo o si llegase haber un cambio este sería mínimo.

Para desarrollo de este tema vamos a utilizar el IDE Bluej y la base de datos: MySql

1. **Scrip de Creación de la Base de Daton en Mysql**

```
CREATE DATABASE sampleoop;

USE sampleoop;

CREATE TABLE `estudiante` (
     `id` int(11) NOT NULL AUTO_INCREMENT,
     `identificacion` varchar(50) NOT NULL,
     `nombre` varchar(50) NOT NULL,
     `curso` varchar(50) NOT NULL,
     `nota1` float NOT NULL,
     `nota2` float NOT NULL,
     `nota3` float NOT NULL,
     PRIMARY KEY (`id`)
);
```

```
SET @@global.time_zone = '+00:00';

SET @@session.time_zone = '+00:00';
```

2. **En Bluej**

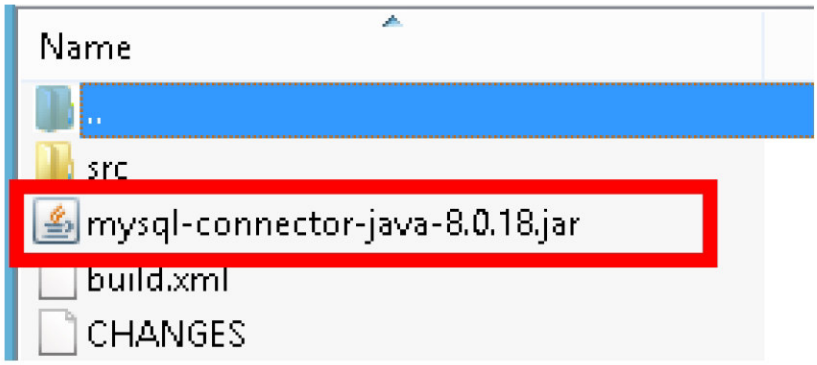
Descargamos el conector: mysql-connector-java de la siguente url:

https://dev.mysql.com/downloads/file/?id=490495

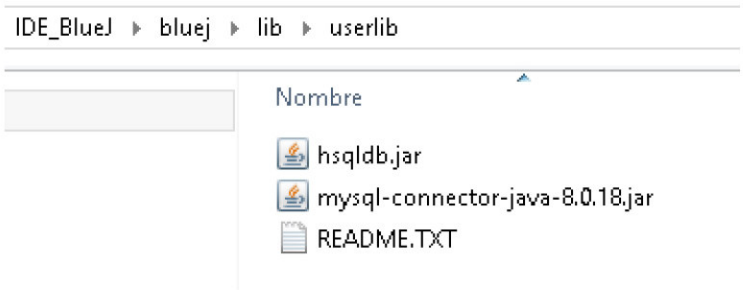
Hacemos click en:

No thanks, just start my download.

Descomprimimos el archivo zip, y ubicamos el archivo: .jar

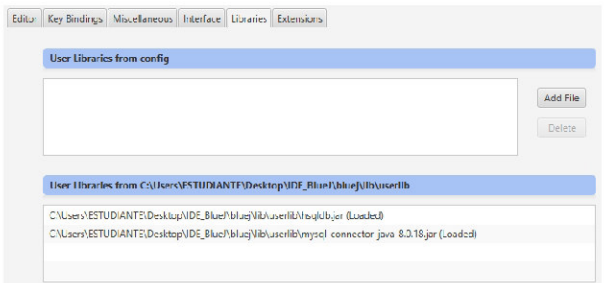


Ubicamos la carpeta: userlib dentro de BlueJ y pegamos el archivo .jar

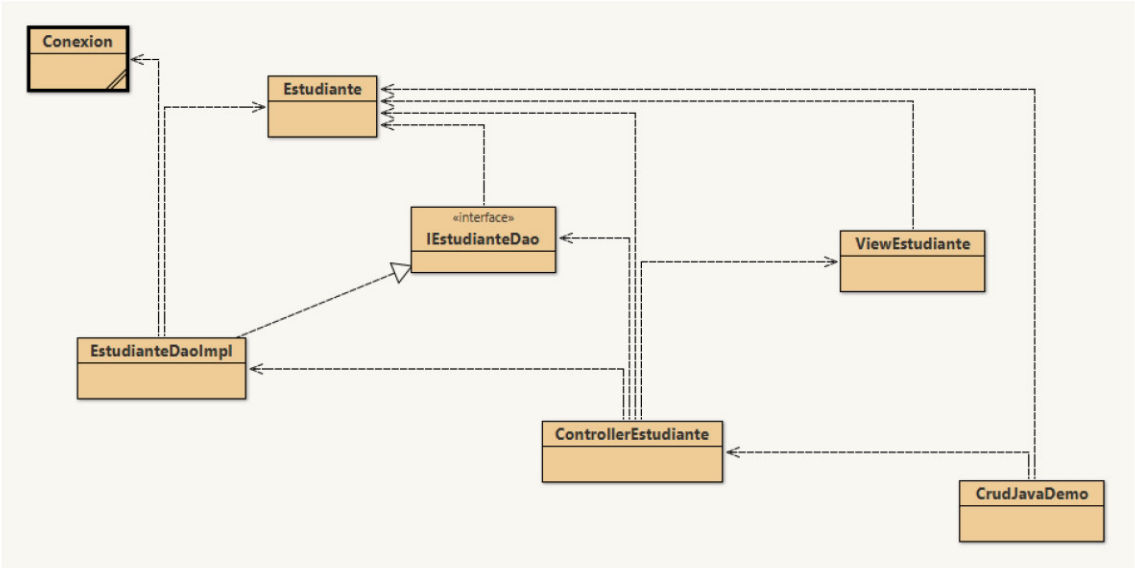


Reiniciamos el BlueJ y verificamos que haya quedado cargado el controlador, revisando la siguiente ruta:

Tools > Preferences > Libraries



Desarrollo del Proyecto – Diagrama de Clases



1. Clase: Conexión.java

   Permite la conexión de la aplicación con la Base de Datos.

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

public class Conexion {
      public static Connection conectar() {
            Connection con = null;

            String usuario = "root";
            String password = "";
            String url = "jdbc:mysql://localhost:3306/
sampleoop?user=" + usuario
                        + "&password=" + password;
            try {
                  con = DriverManager.getConnection(url);
                  if (con != null) {
                        System.out.println("Conectado");
                  }
            } catch (SQLException e) {
                  System.out.println("No se pudo conectar a la
base de datos");
                  e.printStackTrace();
            }
            return con;
      }
}
```

2. Creación de los Métodos del CRUD

   Creamos el Modelo que es la Clase Estudiante.Java

   Esta clase se encarga de mapear los atributos de la tabla estudiante en la BD: sampledb.

```
public class Estudiante{
	private int id;
	private String identificacion;
	private String nombre;
	private String curso;
	private double nota1;
	private double nota2;
	private double nota3;

	public Estudiante() {
	}

	public Estudiante(String identificacion, String nombre, String
curso, double nota1, double nota2, double nota3) {
		this.id = id;
		this.identificacion = identificacion;
		this.nombre = nombre;
		this.curso = curso;
		this.nota1 = nota1;
		this.nota2 = nota2;
		this.nota3 = nota3;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getIdentificacion() {
		return identificacion;
	}

	public void setIdentificacion(String identificacion) {
		this.identificacion = identificacion;
	}

	public String getNombre() {
		return nombre;
	}

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

	public String getCurso() {
		return curso;
	}

	public void setCurso(String curso) {
```

```
		this.curso = curso;
	}

	public double getNota1() {
		return nota1;
	}

	public void setNota1(double nota1) {
		this.nota1 = nota1;
	}

	public double getNota2() {
		return nota2;
	}

	public void setNota2(double nota2) {
		this.nota2 = nota2;
	}

	public double getNota3() {
		return nota3;
	}

	public void setNota3(double nota3) {
		this.nota3 = nota3;
	}
	@Override
	public String toString() {
		return this.id+", "+this.identificacion+", "+this.nombre+",
"+this.curso+", "+this.nota1+", "+this.nota2+", "+this.nota3;
	}
}
```

3. Creación de la Interfaz: IEstudianteDAO.java

   Esta interfaz registra los métodos del CRUD, por ser una interfaz solo contiene la definición de los métodos.

```
import java.util.List;

public interface IEstudianteDao {
      public boolean registrar(Estudiante estudiante);
      public List<Estudiante> obtener();
      public boolean actualizar(Estudiante estudiante);
      public boolean eliminar(Estudiante estudiante);
}
```

4. El siguiente paso es implementar la interfaz definida en el paso anterior, creamos la clase: EstudianteDaoImpl.java, de igual forma también utilizamos la clase Conexion.java, para la ejecución de comandos SQL.

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class EstudianteDaoImpl implements IEstudianteDao {

	@Override
	public boolean registrar(Estudiante estudiante) {
		boolean registrar = false;

		Statement stm= null;
		Connection con=null;

		String sql="INSERT INTO estudiante VALUES
(NULL,'"+estudiante.getIdentificacion()+"','"+estudiante.getNombre()+"
','"+estudiante.getCurso()+"','"+estudiante.getNota1()+"','"+estudiant
e.getNota2()+"','"+estudiante.getNota3()+"')";

		try {
			con=Conexion.conectar();
			stm= con.createStatement();
			stm.execute(sql);
			registrar=true;
			stm.close();
			con.close();
		} catch (SQLException e) {
			System.out.println("Error: Clase EstudianteDaoImpl,
metodo registrar");
			e.printStackTrace();
		}
		return registrar;
	}

	@Override
	public List<Estudiante> obtener() {
		Connection co = null;
		Statement stm = null;
		ResultSet rs = null;

		String sql="SELECT * FROM ESTUDIANTE ORDER BY ID";

		List<Estudiante> listaEstudiante= new
ArrayList<Estudiante>();

		try {
			co= Conexion.conectar();
			stm=co.createStatement();
			rs=stm.executeQuery(sql);
			while (rs.next()) {
				Estudiante c =new Estudiante();
				c.setId(rs.getInt(1));
				c.setIdentificacion(rs.getString(2));
```

```
                        c.setNombre(rs.getString(3));
                        c.setCurso(rs.getString(4));
                        c.setNota1(rs.getDouble(5));
                        c.setNota2(rs.getDouble(6));
                        c.setNota3(rs.getDouble(7));
                        listaEstudiante.add(c);
                  }
                  stm.close();
                  rs.close();
                  co.close();
            } catch (SQLException e) {
                  System.out.println("Error: Clase EstudianteDaoImpl,
metodo obtener");
                  e.printStackTrace();
            }

            return listaEstudiante;
      }

      @Override
      public boolean actualizar(Estudiante estudiante) {
            Connection connect= null;
            Statement stm= null;

            boolean actualizar=false;

            String sql="UPDATE ESTUDIANTE SET
identificacion='"+estudiante.getIdentificacion()+"',
nombre='"+estudiante.getNombre()+"',
curso='"+estudiante.getCurso()+"', nota1='"+estudiante.getNota1()+"',
nota2='"+estudiante.getNota2()+"', nota3='"+estudiante.getNota3()+"'"
+" WHERE ID="+estudiante.getId();
            try {
                  connect=Conexion.conectar();
                  stm=connect.createStatement();
                  stm.execute(sql);
                  actualizar=true;
            } catch (SQLException e) {
                  System.out.println("Error: Clase EstudianteDaoImpl,
metodo actualizar");
                  e.printStackTrace();
            }
            return actualizar;
      }

      @Override
      public boolean eliminar(Estudiante estudiante) {
            Connection connect= null;
            Statement stm= null;

            boolean eliminar=false;

            String sql="DELETE FROM ESTUDIANTE WHERE
ID="+estudiante.getId();
            try {
                  connect=Conexion.conectar();
                  stm=connect.createStatement();
                  stm.execute(sql);
                  eliminar=true;
            } catch (SQLException e) {
```

```
                    System.out.println("Error: Clase EstudianteDaoImpl,
metodo eliminar");
                    e.printStackTrace();
                }
                return eliminar;
        }

}
```

5. Aplicamos el patrón Model View Controller. Una vez creado el acceso a datos los DAO's vamos a utilizar la arquitectura MVC para poder utilizar los métodos del CRUD. El modelo ya lo habíamos creado anteriormente, así que ahora sólo vamos a crear la vista y luego el controlador. La vista: ViewEstudiante.java es el lugar donde se van mostrar los datos del modelo, hay que recordar que en la arquitectura MVC la vista no interactúa directamente ni con el modelo, ni con el acceso a los datos, solo lo hace a través del controlador.

```
import java.util.List;
public class ViewEstudiante {
	public void verEstudiante(Estudiante estudiante) {
		System.out.println("Datos del Estudiante: "+estudiante);
	}

	public void verEstudiantes(List<Estudiante> estudiantes) {
		for (Estudiante estudiante : estudiantes) {
			System.out.println("Datos del Estudiante:
"+estudiante);
		}
	}
}
```

6. Ahora creamos el controlador: ControllerEstudiante.java

   El controlador que es el encargado de enlazar el acceso a los datos con la vista.

```
import java.util.ArrayList;
import java.util.List;

public class ControllerEstudiante {

	private ViewEstudiante vista= new ViewEstudiante();

	public ControllerEstudiante() {
	}

	//llama al DAO para guardar un Estudiante
	public void registrar(Estudiante estudiante) {
		IEstudianteDao dao= new  EstudianteDaoImpl();
		dao.registrar(estudiante);
	}

	//llama al DAO para actualizar un Estudiante
	public void actualizar(Estudiante estudiante) {
		IEstudianteDao dao= new  EstudianteDaoImpl();
		dao.actualizar(estudiante);
	}

	//llama al DAO para eliminar un estudiante
	public void eliminar(Estudiante estudiante) {
		IEstudianteDao dao= new  EstudianteDaoImpl();
		dao.eliminar(estudiante);
	}

	//llama al DAO para obtener todos los estudiantes y luego los
muestra en la vista
	public void verEstudiantes(){
		List<Estudiante> estudiantes = new ArrayList<Estudiante>();
		IEstudianteDao dao= new EstudianteDaoImpl();
		estudiantes=dao.obtener();
		vista.verEstudiantes(estudiantes);
	}
}
```

7. Como parte final creamos la clase: CrudJavaDemo.java

En esta clase ingresamos los valores que nos permiten ver el funcionamiento de nuestro proyecto.

```
public class CrudJavaDemo {

   public static void main(String[] args) {

        Estudiante estudiante = new Estudiante("13579",
"Laura", "Deportes", 4, 6, 8);

        // controlador
        ControllerEstudiante controller = new
ControllerEstudiante();

        // guarda un estudiante a través del controlador
        controller.registrar(estudiante);

        // ver clientes
        controller.verEstudiantes();

        // editar un estudiante por medio del id
        estudiante.setId(2);
        estudiante.setNombre("Mariana");
        controller.actualizar(estudiante);

        // eliminar un estudiante por medio del id
        estudiante.setId(1);
        controller.eliminar(estudiante);
   }
}
```