


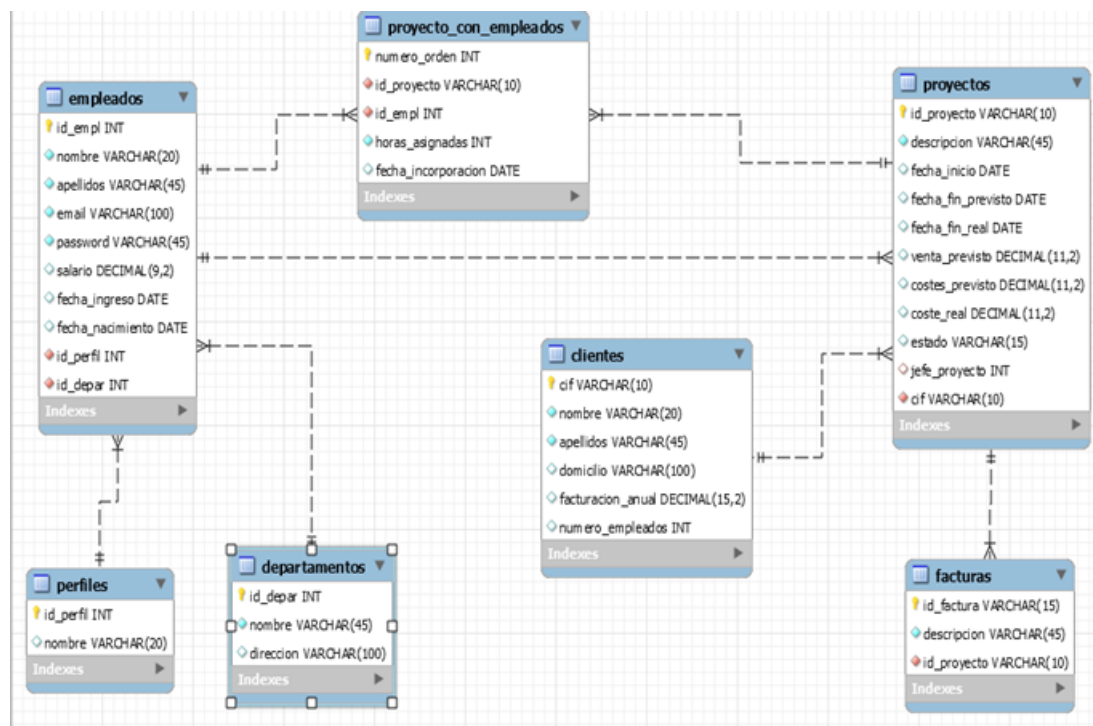
PROGRAMACIÓN

AD-7. Tarea en equipo. Acceso a Bases de Datos

Enunciado

Realizar una Aplicación de gestión de proyectos, de acuerdo con la siguiente base de Datos:

 modelo_clientes_proyectos_empleados_2023_OK.sql



Código de modelo_clientes_proyectos_empleados_2023_OK.sql

```
1 • drop database if exists clientes_proyectos_empleados_2023;
2 • create database if not exists clientes_proyectos_empleados_2023;
3 • use clientes_proyectos_empleados_2023;
4
5 • create table clientes (
6     cif varchar(10) not null primary key,
7     nombre varchar(20) not null,
8     apellidos varchar(45) not null,
9     domicilio varchar(100),
10    facturacion_anual dec (15,2),
11    numero_empleados int
12 );
13 • create table departamentos (
14     id_depar int primary key,
15     nombre varchar(45) not null,
16     direccion varchar(100)
17 );
18 • create table perfiles (
19     id_perfil int not null auto_increment primary key,
20     nombre varchar(20)
21 );
22 • create table empleados (
23     id_empl int not null auto_increment primary key,
24     nombre varchar(20) not null,
25     apellidos varchar(45) not null,
26     sexo char(1) not null,
27     email varchar(100) not null unique,
28     password varchar(45) not null,
29     salario dec (9,2),
30     fecha_ingreso date,
31     fecha_nacimiento date,
32     id_perfil int not null,
33     id_depar int not null,
34     check(sexo in('H','h','M','m')),
35     foreign key(id_depar) references departamentos(id_depar),
36     foreign key(id_perfil) references perfiles (id_perfil)
37 );
38 • create table proyectos(
39     id_proyecto varchar(10) not null primary key,
40     descripcion varchar(45) not null,
41     fecha_inicio date,
42     fecha_fin_previsto date,
43     fecha_fin_real date,
44     venta_previsto dec(11,2),
45     costes_previsto dec(11,2),
46     coste_real dec (11,2),
47     estado varchar(15),
48     jefe_proyecto int,
49     cif varchar(10) not null,
50     foreign key(cif) references clientes(cif),
51     foreign key(jefe_proyecto) references empleados(id_empl)
52 );
```

```

53 • create table proyecto_con_empleados(
54     numero_orden int auto_increment primary key,
55     id_proyecto varchar(10) not null,
56     id_empl int not null,
57     horas_asignadas int not null,
58     fecha_incorporacion date,
59     foreign key(id_proyecto) references proyectos(id_proyecto),
60     foreign key(id_empl) references empleados(id_empl)
61 );
62 • create table facturas(
63     id_factura varchar(15) not null primary key,
64     descripcion varchar(45) not null,
65     id_proyecto varchar(10) not null,
66     foreign key(id_proyecto) references proyectos(id_proyecto)
67 );
68
69 • INSERT INTO clientes (cif, nombre, apellidos, domicilio, facturacion_anual, numero_empleados) values
70     ('A22222222', 'Carlos', 'March', 'Madrid', '12000000', '1500'),
71     ('B33333333', 'Sara', 'Varas', 'Sevilla', '1500000', '345');
72 • insert into departamentos values
73     (10, 'Gestion Personas', 'Madrid'),
74     (20, 'Software', 'Madrid'),
75     (30, 'Hardware', 'Madrid'),
76     (40, 'Financiero', 'Sevilla');
77 • insert into perfiles (nombre) values
78     ('Control de Gestion'), ('Jefe de Proyecto'), ('Operativo'), ('Recursos Humanos');
79 • INSERT INTO empleados VALUES
80     ('100', 'esteban', 'Diaz', 'H', 'ediaz@tt.com', 'esteban', '90000', '1990-01-12', '1977-02-12', 4, 10),
81     ('101', 'Sara', 'Hernandez', 'M', 'shernandez@tt.com', 'sara', '45000', '2005-07-07', '1987-05-15', 4, 10),
82     ('114', 'Rafael', 'Raphaelly', 'H', 'rraphaelly@tt.com', 'rafael', '78000', '2005-09-07', '1977-02-18', 2, 20),
83     ('115', 'Carlos', 'Koo', 'H', 'ckoo@tt.com', 'carlos', '33000', '2015-09-07', '1983-02-04', 3, 20),
84     ('116', 'Carmen', 'Baida', 'M', 'cbaida@tt.com', 'carmen', '32000', '2015-09-08', '1983-12-04', 3, 20),
85     ('117', 'Alejandro', 'Himuro', 'H', 'ahimuro@tt.com', 'alejandro', '25000', '2015-09-09', '1984-12-04', 3, 20),
86     ('118', 'Eva', 'Colmenares', 'M', 'ecolmenares@tt.com', 'eva', '25000', '2015-09-09', '1984-12-04', 3, 20),
87     ('119', 'Eva', 'Tobias', 'M', 'etobias@tt.com', 'eva', '25000', '2015-09-09', '1984-12-04', 3, 20),
88     ('120', 'Raquel', 'Oliva', 'M', 'roliva@tt.com', 'raquel', '38000', '2015-09-09', '1982-04-21', 1, 40);
89 • INSERT INTO proyectos (id_proyecto, descripcion, fecha_inicio, fecha_fin_previsto, fecha_fin_real, venta_previsto, costes_previsto, coste_real, estado, jefe_proyecto, cif) values
90     ('FOR2020001', 'Formacion de habilidades directivas', '2020-01-15', '2020-07-31', '2020-07-31', '50000', '30000', '315000', 'TERMINADO', '114', 'A22222222'),
91     ('FOR2021001', 'Formacion de jefes de proyecto', '2021-09-15', '2021-12-31', '2021-12-10', '50000', '30000', null, 'ACTIVO', '114', 'A22222222'),
92     ('FOR2021002', 'Formacion de jefes de proyecto', '2021-09-15', '2021-12-31', '2022-01-08', '50000', '30000', null, 'ACTIVO', '114', 'B33333333'),
93     ('FOR2021003', 'Formacion de habilidades directivas', '2022-01-01', '2022-08-31', '2022-06-30', '70000', '40000', '50000', 'TERMINADO', '115', 'B33333333');
94 • INSERT INTO proyecto_con_empleados ('numero_orden', 'id_proyecto', 'id_empl', 'horas_asignadas', 'fecha_incorporacion') VALUES
95     ('1', 'FOR2020001', '115', '45', '2020-01-16'),
96     ('2', 'FOR2020001', '116', '30', '2020-01-17'),
97     ('3', 'FOR2021001', '117', '54', '2021-10-11'),
98     ('4', 'FOR2021001', '118', '100', '2021-10-14');
99 • INSERT INTO facturas ('id_factura', 'descripcion', 'id_proyecto') VALUES
100     ('F2020001', 'Formacion a cliente 1', 'FOR2020001');

```

La estructura de paquetes del proyecto es:

1. Javabeans
2. Daos
3. Conexión
4. Testing
5. principales

1. CLIENTES:

Consideraciones:

Hacer una clase con main estático GestionClientes con un menú con las siguientes opciones:

1. Alta del Cliente
2. Buscar un Cliente
3. Mostrar Todos.
4. Eliminar un cliente
5. Salir

Definir el javabean de Cliente, el interface para CRUD de cliente incluido findAll(), y el Interface de ClienteDaoImpl, para implementar los métodos del Interface, accediendo a bases de datos.

2. RECURSOS HUMANOS: DEPARTAMENTO, PERFIL, EMPLEADO

Crear los javabean Departamento, Perfil y Empleado, los interface y las clases que implementan el interface.

CONSIDERACIONES:

- La clase Empleado, tendrá los siguientes métodos propios:
 - salarioBruto() : double
 - salarioMensual(int meses) : double
 - literalSexo() : String. H -> Hombre, M -> mujer
 - obtenerEmail() : String. Primera letra del nombre + primer apellido, en minúsculas
 - nombreCompleto(): String. Nombre + " " + apellidos
- La clase EmpleadoDao, además de los CRUD, y findAll(), impone los siguientes métodos
 - empleadosByDepartamento(int idDepar): List<Empleado>
 - empleadosBySexo(char sexo): List<Empleado>
 - empleadosByApellido(String subcadena): List<Empleado>
 - salarioTotal(): double
 - salarioTotal(int idDepar): double. Salario Total para un

3. PROYECTOS

La clase Proyecto tiene los siguientes métodos propios:

- margenPrevisto():double. Importe de venta – coste previsto
- margenReal(): double Importe de venta – gastos reales
- diferenciaGastos(): double. Gasto real – gasto previsto
- diferenciaFinPrevistoReal(): int . Días entre fin previsto y fin real

La clase ProyectoDao, además de CRUD y findAll(), impone los siguientes métodos con acceso de base de datos:

- proyectosByEstado(String estado): List<Proyecto>
- proyectosByCliente(String cif) : List<Proyecto>
- proyectosByJefeProyectoAndByEstado(int jefeProyecto, String estado): List<Proyecto>
- importesVentaProyectosTerminados(): double
- margenBrutoProyectosTerminados():double Diferencia suma Importes venta y gastos reales.
- diasATerminoProyectoActivo(String codigoProyecto): int. Cuantos días quedan para terminar el proyecto (diferencia entre fecha_fin_previsto y la fecha de hoy)

4. EMPLEADOS EN PROYECTOS:

La clase EmpleadosEnProyectos tiene los siguientes métodos propios:

- costeHorasAsignadas(): double . Horas * precio/hora

Métodos EmpleadosEnProyectoDao, impone si quieres CRUD y findAll() y obligatoriamente impone con acceso de base de datos:

- empleadosByProyecto(String codigoProyecto): List<EmpleadosEnProyecto>
- asignarEmpleadosAProyecto(List<EmpleadosEnProyecto> empleados): int
- horasAsignadasAProyecto(String codigoProyecto): int. Suma de las horas de los empleados asignados al proyecto.
- costeActualDeProyecto(String codigoProyecto): double . horas*coste-hora de cada empleado asignado al proyecto.
- margenActualProyecto(String codigoProyecto): double. Importe_venta del proyecto – costeActual del Proyecto

Hacer clases de prueba por cada Javabeen-Interface-ImplMy8, para probar la funcionalidad.

0 Índice

	Pág.
0 Conexión con base de datos	7
1 Requerimiento 1: cliente y probando cliente	8
2 Requerimiento 2: departamento, perfil, empleado	12
3 Requerimiento 3: proyecto	21
4 Requerimiento 4: EmpleadoEnProyecto	27
5 Requerimiento 5: Probando Empleado, Proyecto, DepartamentoImpl, ...	30

Estructura de la actividad

- ▼ Basededatosejercicio
 - ▼ src
 - ▼ conexion
 - > ConexionMySQL.java
 - ▼ modelo.daojdbc
 - > AbstractMySQL.java
 - > ClienteDao.java
 - > ClienteImplMySQL.java
 - > DepartamentoDao.java
 - > DepartamentoImplMySQL.java
 - > EmpleadoDao.java
 - > EmpleadoEnProyectoDao.java
 - > EmpleadoEnProyectoImplMySQL.java
 - > EmpleadoImplMySQL.java
 - > PerfilDao.java
 - > PerfilImplMySQL.java
 - > ProyectoDao.java
 - > ProyectoImplMySQL.java
 - ▼ modelo.javabean
 - > Cliente.java
 - > Departamento.java
 - > Empleado.java
 - > EmpleadoEnProyecto.java
 - > Perfil.java
 - > Proyecto.java
 - ▼ principales
 - > GestionClientes.java
 - ▼ testing
 - > TestDepartamentoImpl.java
 - > TestEmpleado.java
 - > TestEmpleadoEnProyectoImpl.java
 - > TestEmpleadoImpl.java
 - > TestPerfilImpl.java
 - > TestProyecto.java
 - > TestProyectoImpl.java
 - > JRE System Library [JavaSE-17]
 - ▼ Referenced Libraries
 - > mysql-connector-j-8.0.32.jar - C:\Users\R

0 Conexión con base de datos

ConexionMySQL.java

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

public class ConexionMySQL {
    public static Connection conn;

    public ConexionMySQL() {
        try {
            conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/clientes_proyectos_empleados_2023?serverTimezone=UTC", "root", "melasola");
            System.out.println("Conexión establecida.");
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("Error en la conexión.");
        }
    }

    public static Connection getConexion() {
        if (conn==null) {
            new ConexionMySQL();
        }
        return conn;
    }
}
```

AbstractMySQL.java

```
package modelo.daojdbc;
import conexion.ConexionMySQL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public abstract class AbstractMySQL {

    protected Connection conn;
    protected PreparedStatement ps;
    protected ResultSet rs;
    protected String sql;
    protected int filas;

    public AbstractMySQL() {
        conn = ConexionMySQL.getConexion();
    }
}
```

1 Requerimiento 1: cliente y probando cliente

Cliente.java

```
package modelo.javabean;

public class Cliente {
    //Atributos de instancia
    private String cif, nombre, apellidos, domicilio;
    private double facturacionAnual;
    private int numeroEmpleados;

    //Métodos constructores
    public Cliente() {
    }
    public Cliente(String cif, String nombre, String apellidos, String domicilio, double facturacionAnual, int numeroEmpleados) {
        super();
        this.cif = cif;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.domicilio = domicilio;
        this.facturacionAnual = facturacionAnual;
        this.numeroEmpleados = numeroEmpleados;
    }

    //Métodos getter
    public String getCif() {
        return cif;
    }
    public String getNombre() {
        return nombre;
    }
    public String getApellidos() {
        return apellidos;
    }
    public String getDomicilio() {
        return domicilio;
    }
    public double getFacturacionAnual() {
        return facturacionAnual;
    }
    public int getNumeroEmpleados() {
        return numeroEmpleados;
    }

    //Métodos setter
    public void setCif(String cif) {
        this.cif = cif;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }
    public void setDomicilio(String domicilio) {
        this.domicilio = domicilio;
    }
    public void setFacturacionAnual(double facturacionAnual) {
        this.facturacionAnual = facturacionAnual;
    }
    public void setNumeroEmpleados(int numeroEmpleados) {
        this.numeroEmpleados = numeroEmpleados;
    }
}
```



```

@Override
public String toString() {
    return "Cliente [cif=" + cif + ", nombre=" + nombre + ", apellidos=" + apellidos
+ ", domicilio=" + domicilio+ ", facturacionAnual=" + facturacionAnual + ",
numeroEmpleados=" + numeroEmpleados + "]";
}
}

ClienteDao.java
package modelo.daojdbc;
import java.util.List;
import modelo.javabean.Cliente;

public interface ClienteDao {
    public abstract List<Cliente> buscarTodos();
    public abstract Cliente buscarUno(String cif);
    public abstract boolean altaCliente(Cliente cliente);
    public abstract boolean modificarCliente(Cliente cliente);
    public abstract boolean bajaCliente(String cif);
}

ClienteImplMySQL.java
package modelo.daojdbc;
import java.sql.SQLException;
import java.util.List; import java.util.ArrayList;
import modelo.javabean.Cliente;

public class ClienteImplMySQL extends AbstractMySQL implements ClienteDao {

    //Métodos DML-select-----

    public List<Cliente> buscarTodos() {
        List<Cliente> lista = new ArrayList<>();
        try {
            ps = conn.prepareStatement("select * from clientes");
            rs = ps.executeQuery();
            while (rs.next()) {
                Cliente cli1=new Cliente();
                cli1.setCif(rs.getString("cif"));
                cli1.setNombre(rs.getString("nombre"));
                cli1.setApellidos(rs.getString("apellidos"));
                cli1.setDomicilio(rs.getString("domicilio"));
                cli1.setFacturacionAnual(rs.getDouble("facturacion_anual"));
                cli1.setNumeroEmpleados(rs.getInt("numero_empleados"));
                lista.add(cli1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }

    public Cliente buscarUno(String cif) {
        Cliente cli1 = null;
        try {
            ps = conn.prepareStatement("select * from clientes where cif = ?");
            ps.setString(1, cif);
            rs = ps.executeQuery();
            if (rs.next()) {
                cli1 = new Cliente();
                cli1.setCif(rs.getString("cif"));
                cli1.setNombre(rs.getString("nombre"));
                cli1.setApellidos(rs.getString("apellidos"));
                cli1.setDomicilio(rs.getString("domicilio"));
                cli1.setFacturacionAnual(rs.getDouble("facturacion_anual"));
            }
        }
    }
}

```

```

        cli1.setNumeroEmpleados(rs.getInt("numero_empleados"));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return cli1;
}

//Métodos DML-insert-----
public boolean altaCliente(Cliente cliente) {
    try {
        ps = conn.prepareStatement("insert into clientes values(?,?,?,?,?,?)");
        ps.setString(1, cliente.getCif());
        ps.setString(2, cliente.getNombre());
        ps.setString(3, cliente.getApellidos());
        ps.setString(4, cliente.getDomicilio());
        ps.setDouble(5, cliente.getFacturacionAnual());
        ps.setInt(6, cliente.getNumeroEmpleados());
        ps.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}

//Métodos DML-update-----
public boolean modificarCliente(Cliente cliente) {
    try {
        ps = conn.prepareStatement("update empleados set cif = ?, nombre = ?,
apellidos = ?, domicilio = ?, facturacion_anual = ?, numero_empleados = ? where cif =
?");
        ps.setString(1, cliente.getCif());
        ps.setString(2, cliente.getNombre());
        ps.setString(3, cliente.getApellidos());
        ps.setString(4, cliente.getApellidos());
        ps.setDouble(5, cliente.getFacturacionAnual());
        ps.setInt(6, cliente.getNumeroEmpleados());
        ps.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}

//Métodos DML-delete-----
public boolean bajaCliente(String cif) {
    try {
        ps = conn.prepareStatement("delete from clientes where cif = ?");
        ps.setString(1, cif);
        ps.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
}

GestionCliente.java
package principales;
import java.util.Scanner;
import modelo.daojdbc.ClienteImplMySQL;

```

```

import modelo.javabeen.Cliente;
public class GestionClientes {

    public static void main(String[] args) {
        ClienteImplMySQL cliimpl1= new ClienteImplMySQL();
        Scanner entrada=new Scanner(System.in);
        System.out.println("Seleccione una opción:\n1. Alta del Cliente \n2. Buscar un
cliente \n3. Mostrar todos "
            + "\n4. Eliminar un cliente \n5. Salir");
        int numero=entrada.nextInt();
        switch (numero) {
            case 1:
                System.out.print("Introduzca el cif del cliente: ");
                entrada.nextLine();
                String frase1=entrada.nextLine();
                System.out.print("Introduzca el nombre del cliente: ");
                String frase2=entrada.nextLine();
                System.out.print("Introduzca los apellidos: ");
                String frase3=entrada.nextLine();
                System.out.print("Introduzca el domicilio: ");
                String frase4=entrada.nextLine();
                System.out.print("Introduzca la facturación anual: ");
                Double num1=entrada.nextDouble();
                System.out.print("Introduzca el número de empleados: ");
                int num2=entrada.nextInt();
                Cliente cli1 = new Cliente(frase1, frase2, frase3, frase4, num1, num2);
                System.out.println(cliimpl1.altaCliente(cli1));
                System.out.println(cliimpl1.buscarUno(frase1));
                break;
            case 2:
                System.out.print("Introduzca el cif del cliente. ");
                entrada.nextLine();
                String frase=entrada.nextLine();
                System.out.println(cliimpl1.buscarUno(frase));
            case 3:
                for (Cliente ele: cliimpl1.buscarTodos())
                    System.out.println(ele);
                break;
            case 4:
                System.out.print("Introduzca el cif del cliente a dar de baja: ");
                entrada.nextLine();
                String frase5=entrada.nextLine();
                System.out.println(cliimpl1.bajaCliente(frase5));
                break;
            case 5:
                System.out.println("Ha salido del sistema.");
                break;
            default:
                System.out.println("La opción seleccionada no es correcta.");
        }
    }
}

```

GestionClientes [Java Application] C:\Users\...\Desktop\Programas de instalación\Eclipse IDE for Enterprise Java

Conexión establecida.
 Seleccione una opción:
 1. Alta del Cliente
 2. Buscar un cliente
 3. Mostrar todos
 4. Eliminar un cliente
 5. Salir

2 Requerimiento 2: departamento, perfil, empleado

2.1 javabeans de Departamento, Perfil, Empleado

Departamento.java

```
package modelo.javabeans;
public class Departamento {
    //Atributos de instancia
    private int idDepar;
    private String nombre, direccion;

    //Métodos constructores
    public Departamento() {
    }
    public Departamento(int idDepar, String nombre, String direccion) {
        this.idDepar = idDepar; this.nombre = nombre; this.direccion = direccion;
    }

    //Métodos getter
    public int getIdDepar() {
        return idDepar;
    }
    public String getNombre() {
        return nombre;
    }
    public String getDireccion() {
        return direccion;
    }
    //Métodos setter
    public void setIdDepar(int idDepar) {
        this.idDepar = idDepar;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    @Override
    public String toString() {
        return "Departamento [idDepar=" + idDepar + ", nombre=" + nombre + ",
direccion=" + direccion + "]";
    }
}
```

Perfil.java

```
package modelo.javabeans;
public class Perfil {
    //Atributos de instancia
    private int idPerfil;
    private String nombre;

    //Métodos constructores
    public Perfil() {
    }
    public Perfil(int idPerfil, String nombre) {
        this.idPerfil = idPerfil;
        this.nombre = nombre;
    }

    //Métodos getter
    public int getIdPerfil() {
```

```

        return idPerfil;
    }
    public String getNombre() {
        return nombre;
    }
    //Métodos setter
    public void setIdPerfil(int idPerfil) {
        this.idPerfil = idPerfil;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return "Perfil [idPerfil=" + idPerfil + ", nombre=" + nombre + "]";
    }
}

```

Empleado.java

```

package modelo.javabeen;
import java.sql.Date;
public class Empleado {
    //Atributos de instancia
    private int idEmpl;
    private String nombre, apellidos;
    private char sexo;
    private String email, password;
    private double salario;
    private Date fechaIngreso, fechaNacimiento;
    private Perfil perfil; private Departamento departamento;

    //Métodos constructores
    public Empleado() {
    }

    public Empleado(int idEmpl, String nombre, String apellidos, char sexo, String
email, String password,
        double salario, Date fechaIngreso, Date fechaNacimiento, Perfil
perfil, Departamento departamento) {
        super();
        this.idEmpl = idEmpl;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.sexo = sexo;
        this.email = email;
        this.password = password;
        this.salario = salario;
        this.fechaIngreso = fechaIngreso;
        this.fechaNacimiento = fechaNacimiento;
        this.perfil = perfil;
        this.departamento = departamento;
    }

    //Métodos getter
    public int getIdEmpl() {
        return idEmpl;
    }
    public String getNombre() {
        return nombre;
    }
    public String getApellidos() {
        return apellidos;
    }
}

```

```

    public char getSexo() {
        return sexo;
    }
    public String getEmail() {
        return email;
    }
    public String getPassword() {
        return password;
    }
    public double getSalario() {
        return salario;
    }
    public Date getFechaIngreso() {
        return fechaIngreso;
    }
    public Date getFechaNacimiento() {
        return fechaNacimiento;
    }
    public Perfil getPerfil() {
        return perfil;
    }
    public Departamento getDepartamento() {
        return departamento;
    }
    //Métodos setter
    public void setIdEmpl(int idEmpl) {
        this.idEmpl = idEmpl;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }
    public void setSexo(char sexo) {
        this.sexo = sexo;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public void setSalario(double salario) {
        this.salario = salario;
    }
    public void setFechaIngreso(Date fechaIngreso) {
        this.fechaIngreso = fechaIngreso;
    }
    public void setFechaNacimiento(Date fechaNacimiento) {
        this.fechaNacimiento = fechaNacimiento;
    }
    public void setPerfil(Perfil perfil) {
        this.perfil = perfil;
    }
    public void setDepartamento(Departamento departamento) {
        this.departamento = departamento;
    }
}

@Override
public String toString() {
    return "Empleado [idEmpl=" + idEmpl + ", nombre=" + nombre + ",
apellidos=" + apellidos + ", sexo=" + sexo
        + ", email=" + email + ", password=" + password + ",

```

```

salario=" + salario + ", fechaIngreso="
                                + fechaIngreso + ", fechaNacimiento=" + fechaNacimiento + ",
perfil=" + perfil + ", departamento="
                                + departamento + "];
    }

    //Métodos public no static propios
    public String nombreCompleto() {
        return nombre+" "+apellidos;
    }
    public double salarioMensual(int meses) {
        return salario/meses;
    }
    public String obtenerEmail() {
        String apellidosminusculas=this.apellidos.toLowerCase();
        int posprimerapellido=apellidosminusculas.indexOf(' ');
        return
nombre.charAt(0)+apellidosminusculas.substring(0,posprimerapellido);
    }
    public String literalSexo() {
        if (this.sexo=='H') {
            return "Hombre";
        }
        if (this.sexo=='M') {
            return "Mujer";
        }
        else
            return "Desconocido";
    }
}

```

2.2 DepartamentoDao, PerfilDao, EmpleadoDao

DepartamentoDao.java

```

package modelo.daojdbc;
import java.util.List;
import modelo.javabeen.Departamento;

public interface DepartamentoDao {
    public abstract List<Departamento> buscarTodos();
    public abstract Departamento buscarUno(int idDepar);
}

```

PerfilDao.java

```

package modelo.daojdbc;
import modelo.javabeen.Perfil;
public interface PerfilDao {
    public abstract Perfil buscarUno(int idPerfil);
}

```

ClienteDao.java

```

package modelo.daojdbc;
import java.util.List;
import modelo.javabeen.Empleado;
public interface EmpleadoDao {
    public abstract List<Empleado> buscarTodos();
    public abstract Empleado buscarUno(int idEmpl);
    public abstract boolean altaEmpleado(Empleado empleado);
    public abstract boolean modificarEmpleado(Empleado empleado);
    public abstract boolean bajaEmpleado(int idEmpl);
    public abstract List<Empleado> empleadosByApellido(String apellido);
    public abstract List<Empleado> empleadosBySexo(char sexo);
    public abstract List<Empleado> empleadosByDepartamento(int idDepar);
    public abstract double salarioTotal();
}

```

```

    public abstract double salarioTotal(int idDepar);
}

```

2.3 DepartamentoImplMySQL, PerfilImplMySQL, EmpleadoImplMySQL

DepartamentoImplMySQL

```

package modelo.daojdbc;
import java.sql.SQLException;
import java.util.List; import java.util.ArrayList;
import modelo.javabean.Departamento;

public class DepartamentoImplMySQL extends AbstractMySQL implements DepartamentoDao {

    //Métodos DML-select-----
    public List<Departamento> buscarTodos() {
        List<Departamento> lista = new ArrayList<>();
        try {
            ps = conn.prepareStatement("select * from departamentos");
            rs = ps.executeQuery();
            while (rs.next()) {
                Departamento dpto1 = new Departamento();
                dpto1.setIdDepar(rs.getInt("id_depar"));
                dpto1.setNombre(rs.getString("nombre"));
                dpto1.setDireccion(rs.getString("direccion"));
                lista.add(dpto1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }

    public Departamento buscarUno(int idDepar) {
        Departamento dpto1 = null;
        try {
            ps = conn.prepareStatement("select * from departamentos where id_depar = ?");
            ps.setInt(1, idDepar);
            rs = ps.executeQuery();
            if (rs.next()) {
                dpto1 = new Departamento();
                dpto1.setIdDepar(rs.getInt("id_depar"));
                dpto1.setNombre(rs.getString("nombre"));
                dpto1.setDireccion(rs.getString("direccion"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return dpto1;
    }
}

```

PerfilImplMySQL.java

```

package modelo.daojdbc;
import java.sql.SQLException;
import modelo.javabean.Perfil;
public class PerfilImplMySQL extends AbstractMySQL implements PerfilDao{

    //Métodos DML-select-----
    public Perfil buscarUno(int idPerfil) {
        Perfil perfil1 = null;
        try {
            ps = conn.prepareStatement("select * from perfiles where id_perfil = ?");
            ps.setInt(1, idPerfil);

```



```

        rs = ps.executeQuery();
        if (rs.next()) {
            perfil1 = new Perfil();
            perfil1.setIdPerfil(rs.getInt("id_perfil"));
            perfil1.setNombre(rs.getString("nombre"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return perfil1;
}
}

```

EmpleadoImplMySQL.java

```

package modelo.daojdbc;
import java.sql.SQLException;
import java.util.List; import java.util.ArrayList;
import modelo.javabean.Empleado;

public class EmpleadoImplMySQL extends AbstractMySQL implements EmpleadoDao {

    //Métodos DML-select-----
    public List<Empleado> buscarTodos() {
        List<Empleado> lista = new ArrayList<>();
        PerfilImplMySQL perfilimpl1 = new PerfilImplMySQL();
        DepartamentoImplMySQL dptoimpl1 = new DepartamentoImplMySQL();
        try {
            ps = conn.prepareStatement("select * from empleados");
            rs = ps.executeQuery();
            while (rs.next()) {
                Empleado emp1 = new Empleado();
                emp1.setIdEmpl(rs.getInt("id_empl"));
                emp1.setNombre(rs.getString("nombre"));
                emp1.setApellidos(rs.getString("apellidos"));
                String cadena=rs.getString("sexo");
                emp1.setSexo(cadena.charAt(0));
                emp1.setEmail(rs.getString("email"));
                emp1.setPassword(rs.getString("password"));
                emp1.setSalario(rs.getDouble("salario"));
                emp1.setFechaIngreso(rs.getDate("fecha_ingreso"));
                emp1.setFechaNacimiento(rs.getDate("fecha_nacimiento"));
                emp1.setPerfil(perfilimpl1.buscarUno(rs.getInt("id_perfil")));
                emp1.setDepartamento(dptoimpl1.buscarUno(rs.getInt("id_depar")));
                lista.add(emp1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }

    public Empleado buscarUno(int idEmpl) {
        Empleado emp1 = null;
        try {
            ps = conn.prepareStatement("select * from empleados where id_empl = ?");
            ps.setInt(1, idEmpl);
            rs = ps.executeQuery();
            if (rs.next()) {
                emp1 = new Empleado();
                PerfilImplMySQL perfilimpl1 = new PerfilImplMySQL();
                DepartamentoImplMySQL dptoimpl1 = new DepartamentoImplMySQL();
                emp1.setIdEmpl(rs.getInt("id_empl"));
                emp1.setNombre(rs.getString("nombre"));
                emp1.setApellidos(rs.getString("apellidos"));
                String cadena=rs.getString("sexo");
            }
        }
    }
}

```

```

        emp1.setSexo(cadena.charAt(0));
        emp1.setEmail(rs.getString("email"));
        emp1.setPassword(rs.getString("password"));
        emp1.setSalario(rs.getDouble("salario"));
        emp1.setFechaIngreso(rs.getDate("fecha_ingreso"));
        emp1.setFechaNacimiento(rs.getDate("fecha_nacimiento"));
        emp1.setPerfil(perfilimpl1.buscarUno(rs.getInt("id_perfil")));
        emp1.setDepartamento(dptoimpl1.buscarUno(rs.getInt("id_depar")));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return emp1;
}

public List<Empleado> empleadosByApellido(String apellido) {
    List<Empleado> lista = new ArrayList<>();
    PerfilImplMySQL perfilimpl1 = new PerfilImplMySQL();
    DepartamentoImplMySQL dptoimpl1 = new DepartamentoImplMySQL();
    try {
        ps = conn.prepareStatement("select * from empleados where apellidos = ?");
        ps.setString(1, apellido);
        rs = ps.executeQuery();
        while (rs.next()) {
            Empleado emp1 = new Empleado();
            emp1.setIdEmpl(rs.getInt("id_empl"));
            emp1.setNombre(rs.getString("nombre"));
            emp1.setApellidos(rs.getString("apellidos"));
            String cadena=rs.getString("sexo");
            emp1.setSexo(cadena.charAt(0));
            emp1.setEmail(rs.getString("email"));
            emp1.setPassword(rs.getString("password"));
            emp1.setSalario(rs.getDouble("salario"));
            emp1.setFechaIngreso(rs.getDate("fecha_ingreso"));
            emp1.setFechaNacimiento(rs.getDate("fecha_nacimiento"));
            emp1.setPerfil(perfilimpl1.buscarUno(rs.getInt("id_perfil")));
            emp1.setDepartamento(dptoimpl1.buscarUno(rs.getInt("id_depar")));
            lista.add(emp1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return lista;
}

public List<Empleado> empleadosBySexo(char sexo) {
    List<Empleado> lista = new ArrayList<>();
    PerfilImplMySQL perfilimpl1 = new PerfilImplMySQL();
    DepartamentoImplMySQL dptoimpl1 = new DepartamentoImplMySQL();
    try {
        ps = conn.prepareStatement("select * from empleados where sexo = ?");
        ps.setString(1, String.valueOf(sexo));
        rs = ps.executeQuery();

        while (rs.next()) {
            Empleado emp1 = new Empleado();
            emp1.setIdEmpl(rs.getInt("id_empl"));
            emp1.setNombre(rs.getString("nombre"));
            emp1.setApellidos(rs.getString("apellidos"));
            String cadena=rs.getString("sexo");
            emp1.setSexo(cadena.charAt(0));
            emp1.setEmail(rs.getString("email"));
            emp1.setPassword(rs.getString("password"));
            emp1.setSalario(rs.getDouble("salario"));

```

```

        emp1.setFechaIngreso(rs.getDate("fecha_ingreso"));
        emp1.setFechaNacimiento(rs.getDate("fecha_nacimiento"));
        emp1.setPerfil(perfilimpl1.buscarUno(rs.getInt("id_perfil")));
        emp1.setDepartamento(dptoimpl1.buscarUno(rs.getInt("id_depar")));
        lista.add(emp1);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return lista;
}

public List<Empleado> empleadosByDepartamento(int idDepar) {
    List<Empleado> lista = new ArrayList<>();
    PerfilImplMySQL perfilimpl1 = new PerfilImplMySQL();
    DepartamentoImplMySQL dptoimpl1 = new DepartamentoImplMySQL();
    try {
        ps = conn.prepareStatement("select * from empleados where id_depar = ?");
        ps.setInt(1, idDepar);
        rs = ps.executeQuery();
        while (rs.next()) {
            Empleado emp1 = new Empleado();
            emp1.setIdEmpl(rs.getInt("id_empl"));
            emp1.setNombre(rs.getString("nombre"));
            emp1.setApellidos(rs.getString("apellidos"));
            String cadena=rs.getString("sexo");
            emp1.setSexo(cadena.charAt(0));
            emp1.setEmail(rs.getString("email"));
            emp1.setPassword(rs.getString("password"));
            emp1.setSalario(rs.getDouble("salario"));
            emp1.setFechaIngreso(rs.getDate("fecha_ingreso"));
            emp1.setFechaNacimiento(rs.getDate("fecha_nacimiento"));
            emp1.setPerfil(perfilimpl1.buscarUno(rs.getInt("id_perfil")));
            emp1.setDepartamento(dptoimpl1.buscarUno(rs.getInt("id_depar")));
            lista.add(emp1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return lista;
}

//Métodos DML-insert-----
public boolean altaEmpleado(Empleado empleado) {
    try {
        ps = conn.prepareStatement("insert into empleados values(?,?,?,?,?,?,?,?,?,?,?)");
        ps.setInt(1, empleado.getIdEmpl());
        ps.setString(2, empleado.getNombre());
        ps.setString(3, empleado.getApellidos());
        ps.setString(4, String.valueOf(empleado.getSexo()));
        ps.setString(5, empleado.getEmail());
        ps.setString(6, empleado.getPassword());
        ps.setDouble(7, empleado.getSalario());
        ps.setDate(8, empleado.getFechaIngreso());
        ps.setDate(9, empleado.getFechaNacimiento());
        ps.setInt(10, empleado.getPerfil().getIdPerfil());
        ps.setInt(11, empleado.getDepartamento().getIdDepar());
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}

```

```

//Métodos DML-delete-----
public boolean bajaEmpleado(int idEmpl) {
    try {
        ps = conn.prepareStatement("delete from empleados where id_empl = ?");
        ps.setInt(1, idEmpl);
        ps.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}

//Métodos DML-update-----
public boolean modificarEmpleado(Empleado empleado) {
    try {
        ps = conn.prepareStatement("update empleados set nombre = ?, apellidos = ?, sexo = ?,
email = ?, password = ?, salario = ?, fecha_ingreso = ?, fecha_nacimiento = ?, id_perfil = ?,
id_depar = ? where id_empl = ?");
        ps.setString(1, empleado.getNombre());
        ps.setString(2, empleado.getApellidos());
        ps.setString(3, String.valueOf(empleado.getSexo()));
        ps.setString(4, empleado.getEmail());
        ps.setString(5, empleado.getPassword());
        ps.setDouble(6, empleado.getSalario());
        ps.setDate(7, empleado.getFechaIngreso());
        ps.setDate(8, empleado.getFechaNacimiento());
        ps.setInt(9, empleado.getPerfil().getIdPerfil());
        ps.setInt(10, empleado.getDepartamento().getIdDepar());
        ps.setInt(11, empleado.getIdEmpl());
        ps.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}

public double salarioTotal() {
    double suma = 0;
    try {
        ps = conn.prepareStatement("select sum(salario) from empleados");
        rs = ps.executeQuery();
        if (rs.next())
            suma = rs.getDouble(1);
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    return suma;
}

public double salarioTotal(int idDepar) {
    double suma = 0;
    try {
        ps = conn.prepareStatement("select sum(salario) from empleados where id_depar = ?");
        ps.setDouble(1, idDepar);
        rs = ps.executeQuery();
        if (rs.next())
            suma = rs.getDouble(1);
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

    }
    return suma;
}
}

```

3 Requerimiento 3: proyecto

javabean de Proyecto, ProyectoDao.java, ProyectoImplMySql.java

```

Proyecto.java
package modelo.javabean;
import java.sql.Date;
public class Proyecto {
    //Atributos de instancia
    private String idProyecto;
    private String descripcion;
    private Date fechaInicio, fechaFinPrevisto, fechaFinReal;
    private double ventaPrevisto, costesPrevisto, costeReal;
    private String estado;
    private Empleado jefeProyecto;
    private Cliente cif;

    //Métodos constructores
    public Proyecto() {
    }
    public Proyecto(String idProyecto, String descripcion, Date fechaInicio, Date
fechaFinPrevisto, Date fechaFinReal, int ventaPrevisto, int costesPrevisto, int costeReal,
String estado, Empleado jefeProyecto, Cliente cif) {
        super();
        this.idProyecto = idProyecto;
        this.descripcion = descripcion;
        this.fechaInicio = fechaInicio;
        this.fechaFinPrevisto = fechaFinPrevisto;
        this.fechaFinReal = fechaFinReal;
        this.ventaPrevisto = ventaPrevisto;
        this.costesPrevisto = costesPrevisto;
        this.costeReal = costeReal;
        this.estado = estado;
        this.jefeProyecto = jefeProyecto;
        this.cif = cif;
    }

    //Métodos getter
    public String getIdProyecto() {
        return idProyecto;
    }
    public String getDescripcion() {
        return descripcion;
    }
    public Date getFechaInicio() {
        return fechaInicio;
    }
    public Date getFechaFinPrevisto() {
        return fechaFinPrevisto;
    }
    public Date getFechaFinReal() {
        return fechaFinReal;
    }
    public double getVentaPrevisto() {
        return ventaPrevisto;
    }
    public double getCostesPrevisto() {
        return costesPrevisto;
    }
}

```

```

    }
    public double getCosteReal() {
        return costeReal;
    }
    public String getEstado() {
        return estado;
    }
    public Empleado getJefeProyecto() {
        return jefeProyecto;
    }
    public Cliente getCif() {
        return cif;
    }
    //Métodos setter
    public void setIdProyecto(String idProyecto) {
        this.idProyecto = idProyecto;
    }
    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }
    public void setFechaInicio(Date fechaInicio) {
        this.fechaInicio = fechaInicio;
    }
    public void setFechaFinPrevisto(Date fechaFinPrevisto) {
        this.fechaFinPrevisto = fechaFinPrevisto;
    }
    public void setFechaFinReal(Date fechaFinReal) {
        this.fechaFinReal = fechaFinReal;
    }
    public void setVentaPrevisto(double ventaPrevisto) {
        this.ventaPrevisto = ventaPrevisto;
    }
    public void setCostesPrevisto(double costesPrevisto) {
        this.costesPrevisto = costesPrevisto;
    }
    public void setCosteReal(double costeReal) {
        this.costeReal = costeReal;
    }
    public void setEstado(String estado) {
        this.estado = estado;
    }
    public void setJefeProyecto(Empleado jefeProyecto) {
        this.jefeProyecto = jefeProyecto;
    }
    public void setCif(Cliente cif) {
        this.cif = cif;
    }
}

@Override
public String toString() {
    return "Proyecto [idProyecto=" + idProyecto + ", descripcion=" + descripcion + ",
fechaInicio=" + fechaInicio + ", fechaFinPrevisto=" + fechaFinPrevisto + ", fechaFinReal=" +
fechaFinReal + ", ventaPrevisto=" + ventaPrevisto + ", costesPrevisto=" + costesPrevisto +
", costeReal=" + costeReal + ", estado=" + estado + ", jefeProyecto=" + jefeProyecto + ",
cif=" + cif + "]\n";
}

//Métodos propios
public double diferenciaGastos() {
    return costeReal - costesPrevisto;
}

public long diferenciaFinPrevistoReal() {
    return (fechaFinPrevisto.getTime() - fechaFinReal.getTime())/86400000;
}

```

```

    }
}

```

ProyectoDao.java

```

package modelo.daojdbc;
import java.util.List;
import modelo.javabean.Proyecto;

public interface ProyectoDao {
    public abstract List<Proyecto> buscarTodos();
    public abstract Proyecto buscarUno(String idProyecto);
    public abstract List<Proyecto> proyectosByEstado(String estado);
    public abstract List<Proyecto> proyectosByCliente(String cif);
    public abstract List<Proyecto> proyectosByJefeProyectoAndByEstado(int jefeProyecto,
String estado);
    public abstract double importeVentaProyectosTerminados();
    public abstract double margenBrutoProyectosTerminados();
    public abstract int diasATerminoProyectoActivo(String idProyecto);
}

```

ProyectoImplMySQL.java

```

package modelo.daojdbc;
import java.sql.SQLException;
import java.util.ArrayList; import java.util.Arrays;
import java.util.List;
import modelo.javabean.Proyecto;
public class ProyectoImplMySQL extends AbstractMySQL implements ProyectoDao{

    //Métodos DML-select-----

    public List<Proyecto> buscarTodos() {
        List<Proyecto> lista = new ArrayList<>();
        ClienteImplMySQL cliimpl1 = new ClienteImplMySQL();
        EmpleadoImplMySQL empimpl1 = new EmpleadoImplMySQL();
        try {
            ps=conn.prepareStatement("select * from proyectos");
            rs = ps.executeQuery();
            while (rs.next()) {
                Proyecto pro1 = new Proyecto();
                pro1.setIdProyecto(rs.getString("id_proyecto"));
                pro1.setDescripcion(rs.getString("descripcion"));
                pro1.setFechaInicio(rs.getDate("fecha_inicio"));
                pro1.setFechaFinPrevisto(rs.getDate("fecha_fin_previsto"));
                pro1.setFechaFinReal(rs.getDate("fecha_fin_real"));
                pro1.setVentaPrevisto(rs.getDouble("venta_previsto"));
                pro1.setCostesPrevisto(rs.getDouble("costes_previsto"));
                pro1.setCosteReal(rs.getDouble("coste_real"));
                pro1.setEstado(rs.getString("estado"));
                pro1.setJefeProyecto(empimpl1.buscarUno(rs.getInt("jefe_proyecto")));
                pro1.setCif(cliimpl1.buscarUno(rs.getString("cif")));
                lista.add(pro1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }

    public Proyecto buscarUno(String idProyecto) {
        Proyecto pro1 = null;
        ClienteImplMySQL cliimpl1 = new ClienteImplMySQL();
        EmpleadoImplMySQL empimpl1 = new EmpleadoImplMySQL();
        try {

```

```

        ps = conn.prepareStatement("select * from proyectos where id_proyecto = ?");
        ps.setString(1, idProyecto);
        rs = ps.executeQuery();
        if (rs.next()) {
            pro1 = new Proyecto();
            pro1.setIdProyecto(rs.getString("id_proyecto"));
            pro1.setDescripcion(rs.getString("descripcion"));
            pro1.setFechaInicio(rs.getDate("fecha_inicio"));
            pro1.setFechaFinPrevisto(rs.getDate("fecha_fin_previsto"));
            pro1.setFechaFinReal(rs.getDate("fecha_fin_real"));
            pro1.setVentaPrevisto(rs.getDouble("venta_previsto"));
            pro1.setCostesPrevisto(rs.getDouble("costes_previsto"));
            pro1.setCosteReal(rs.getDouble("coste_real"));
            pro1.setEstado(rs.getString("estado"));
            pro1.setJefeProyecto(empimpl1.buscarUno(rs.getInt("jefe_proyecto")));
            pro1.setCif(cliimpl1.buscarUno(rs.getString("cif")));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return pro1;
}

public List<Proyecto> proyectosByEstado(String estado) {
    List<Proyecto> lista = new ArrayList<>();
    ClienteImplMySQL cliimpl1 = new ClienteImplMySQL();
    EmpleadoImplMySQL empimpl1 = new EmpleadoImplMySQL();
    try {
        ps=conn.prepareStatement("select * from proyectos where estado = ?");
        ps.setString(1, estado);
        rs = ps.executeQuery();
        while (rs.next()) {
            Proyecto pro1 = new Proyecto();
            pro1.setIdProyecto(rs.getString("id_proyecto"));
            pro1.setDescripcion(rs.getString("descripcion"));
            pro1.setFechaInicio(rs.getDate("fecha_inicio"));
            pro1.setFechaFinPrevisto(rs.getDate("fecha_fin_previsto"));
            pro1.setFechaFinReal(rs.getDate("fecha_fin_real"));
            pro1.setVentaPrevisto(rs.getDouble("venta_previsto"));
            pro1.setCostesPrevisto(rs.getDouble("costes_previsto"));
            pro1.setCosteReal(rs.getDouble("coste_real"));
            pro1.setEstado(rs.getString("estado"));
            pro1.setJefeProyecto(empimpl1.buscarUno(rs.getInt("jefe_proyecto")));
            pro1.setCif(cliimpl1.buscarUno(rs.getString("cif")));
            lista.add(pro1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return lista;
}

public List<Proyecto> proyectosByCliente(String cif) {
    List<Proyecto> lista = new ArrayList<>();
    ClienteImplMySQL cliimpl1 = new ClienteImplMySQL();
    EmpleadoImplMySQL empimpl1 = new EmpleadoImplMySQL();
    try {
        ps=conn.prepareStatement("select * from proyectos where cif = ?");
        ps.setString(1, cif);
        rs = ps.executeQuery();
        while (rs.next()) {
            Proyecto pro1 = new Proyecto();
            pro1.setIdProyecto(rs.getString("id_proyecto"));
            pro1.setDescripcion(rs.getString("descripcion"));

```



```

        pro1.setFechaInicio(rs.getDate("fecha_inicio"));
        pro1.setFechaFinPrevisto(rs.getDate("fecha_fin_previsto"));
        pro1.setFechaFinReal(rs.getDate("fecha_fin_real"));
        pro1.setVentaPrevisto(rs.getDouble("venta_previsto"));
        pro1.setCostesPrevisto(rs.getDouble("costes_previsto"));
        pro1.setCosteReal(rs.getDouble("coste_real"));
        pro1.setEstado(rs.getString("estado"));
        pro1.setJefeProyecto(empimpl1.buscarUno(rs.getInt("jefe_proyecto")));
        pro1.setCif(cliimpl1.buscarUno(rs.getString("cif")));
        lista.add(pro1);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

return lista;
}

public List<Proyecto> proyectosByJefeProyectoAndByEstado(int jefeProyecto, String estado) {
    List<Proyecto> lista = new ArrayList<>();
    ClienteImplMySQL cliimpl1 = new ClienteImplMySQL();
    EmpleadoImplMySQL empimpl1 = new EmpleadoImplMySQL();
    try {
        ps=conn.prepareStatement("select * from proyectos where jefe_proyecto = ? and estado =
? ");

        ps.setInt(1, jefeProyecto);
        ps.setString(2, estado);
        rs = ps.executeQuery();
        while (rs.next()) {
            Proyecto pro1 = new Proyecto();
            pro1.setIdProyecto(rs.getString("id_proyecto"));
            pro1.setDescripcion(rs.getString("descripcion"));
            pro1.setFechaInicio(rs.getDate("fecha_inicio"));
            pro1.setFechaFinPrevisto(rs.getDate("fecha_fin_previsto"));
            pro1.setFechaFinReal(rs.getDate("fecha_fin_real"));
            pro1.setVentaPrevisto(rs.getDouble("venta_previsto"));
            pro1.setCostesPrevisto(rs.getDouble("costes_previsto"));
            pro1.setCosteReal(rs.getDouble("coste_real"));
            pro1.setEstado(rs.getString("estado"));
            pro1.setJefeProyecto(empimpl1.buscarUno(rs.getInt("jefe_proyecto")));
            pro1.setCif(cliimpl1.buscarUno(rs.getString("cif")));
            lista.add(pro1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return lista;
}

public double importeVentaProyectosTerminados() {
    double suma = 0;
    try {
        ps = conn.prepareStatement("select sum(venta_previsto) from proyectos where estado =
?");

        ps.setString(1, "terminado");
        rs = ps.executeQuery();
        if (rs.next())
            suma = rs.getDouble(1);
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    return suma;
}
}

```

```

    public double margenBrutoProyectosTerminados() {
        double suma = 0;
        try {
            ps = conn.prepareStatement("select sum(venta_previsto)-sum(coste_real) from proyectos
where estado = ?");
            ps.setString(1, "terminado");
            rs = ps.executeQuery();
            if (rs.next())
                suma = rs.getDouble(1);
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
        return suma;
    }

    public int diasATerminoProyectoActivo(String idProyecto) {
        int diferencia=0;
        try {
            ps = conn.prepareStatement("select datediff(fecha_fin_previsto, current_date()) from
proyectos where id_proyecto = ?");
            ps.setString(1, idProyecto);
            rs = ps.executeQuery();
            if (rs.next())
                diferencia = rs.getInt(1);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return diferencia;
    }

    public int diferenciaFinPrevistoReal(String idProyecto) {
        int diferencia=0;
        try {
            ps = conn.prepareStatement("select datediff(fecha_fin_previsto, fecha_fin_real) from
proyectos where id_proyecto = ?");
            ps.setString(1, idProyecto);
            rs = ps.executeQuery();
            if (rs.next())
                diferencia = rs.getInt(1);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return diferencia;
    }

    /*public List<Integer> diferenciaFinPrevistoReal() {
        List<Integer> lista = new ArrayList<>();
        int diferencia=0;
        try {
            ps = conn.prepareStatement("select datediff(fecha_fin_previsto, fecha_fin_real) from
proyectos");
            rs = ps.executeQuery();
            while (rs.next()) {
                diferencia = rs.getInt(1);
                lista.add(diferencia);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }*/

    //Devuelve un list donde cada elemento es a su vez un arraylist que contiene el idProyecto y

```

```

la diferencia
    public List<ArrayList> diferenciaFinPrevistoReal2() {
        List<ArrayList> lista = new ArrayList<>();
        int diferencia=0;
        try {
            ps = conn.prepareStatement("select id_proyecto, datediff(fecha_fin_previsto,
fecha_fin_real) from proyectos");
            rs = ps.executeQuery();
            while (rs.next()) {
                diferencia = rs.getInt(2);
                ArrayList<Object> lista1=new
ArrayList<>(Arrays.asList(rs.getString("id_proyecto"),diferencia));
                lista.add(lista1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }
}

```

4 Requerimiento 4: EmpleadoEnProyecto

```
EmpleadoEnProyecto.java
package modelo.javabeen;

import java.sql.Date;
public class EmpleadoEnProyecto {
    //Atributos de instancia
    private int numeroOrden;
    private Proyecto proyecto;
    private Empleado empleado;
    private int horasAsignadas;
    private Date fechaIncorporacion;

    public EmpleadoEnProyecto() {
    }
    public EmpleadoEnProyecto(int numeroOrden, Proyecto proyecto, Empleado empleado, int
    horasAsignadas, Date fechaIncorporacion) {
        this.numeroOrden = numeroOrden;
        this.proyecto = proyecto;
        this.empleado = empleado;
        this.horasAsignadas = horasAsignadas;
        this.fechaIncorporacion = fechaIncorporacion;
    }

    //Métodos getter
    public int getNumeroOrden() {
        return numeroOrden;
    }
    public Proyecto getProyecto() {
        return proyecto;
    }
    public Empleado getEmpleado() {
        return empleado;
    }
    public int getHorasAsignadas() {
        return horasAsignadas;
    }
    public Date getFechaIncorporacion() {
        return fechaIncorporacion;
    }
    //Métodos setter
    public void setNumeroOrden(int numeroOrden) {
        this.numeroOrden = numeroOrden;
    }
    public void setProyecto(Proyecto proyecto) {
        this.proyecto = proyecto;
    }
    public void setEmpleado(Empleado empleado) {
        this.empleado = empleado;
    }
    public void setHorasAsignadas(int horasAsignadas) {
        this.horasAsignadas = horasAsignadas;
    }
    public void setFechaIncorporacion(Date fechaIncorporacion) {
        this.fechaIncorporacion = fechaIncorporacion;
    }

    @Override
    public String toString() {
        return "ProyectoConEmpleados [numeroOrden=" + numeroOrden + ", proyecto=" +
        proyecto + ", empleado=" + empleado + ", horasAsignadas=" + horasAsignadas + ",
        fechaIncorporacion=" + fechaIncorporacion + "];"
    }
}
```

```
}
```

EmpleadoEnProyectoDao.java

```
package modelo.daojdbc;
import java.util.List;
import modelo.javabean.EmpleadoEnProyecto;

public interface EmpleadoEnProyectoDao {
    public abstract List<EmpleadoEnProyecto> empleadosByProyecto(String codigoProyecto);
    public abstract int asignarEmpleadosAProyecto(List<EmpleadoEnProyecto> empleados);
    public abstract int horasAsignadasAProyecto(String codigoProyecto);
    public abstract double costeActualDeProyecto(String codigoProyecto);
    public abstract double margenActualProyecto(String codigoProyecto);
}
```

EmpleadoEnProyectoMySQL.java

```
package modelo.daojdbc;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import modelo.javabean.EmpleadoEnProyecto;

public class EmpleadoEnProyectoImplMySQL extends AbstractMySQL implements EmpleadoEnProyectoDao {

    public List<EmpleadoEnProyecto> empleadosByProyecto(String codigoProyecto) {
        List<EmpleadoEnProyecto> lista = new ArrayList<>();
        EmpleadoImplMySQL empleadoimpl1 = new EmpleadoImplMySQL();
        ProyectoImplMySQL proyectoimpl1 = new ProyectoImplMySQL();
        try {
            ps = conn.prepareStatement("select * from proyecto_con_empleados where id_proyecto = ?");
            ps.setString(1, codigoProyecto);
            rs = ps.executeQuery();
            while (rs.next()) {
                EmpleadoEnProyecto empep1 = new EmpleadoEnProyecto();
                empep1.setNumeroOrden(rs.getInt("numero_orden"));
                empep1.setEmpleado(empleadoimpl1.buscarUno(rs.getInt("id_empl")));
                empep1.setProyecto(proyectoimpl1.buscarUno(rs.getString("id_proyecto")));
                empep1.setHorasAsignadas(rs.getInt("horas_asignadas"));
                empep1.setFechaIncorporacion(rs.getDate("fecha_incorporacion"));
                lista.add(empep1);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return lista;
    }

    public int asignarEmpleadosAProyecto(List<EmpleadoEnProyecto> empleados) {
        //Por hacer
        return 0;
    }

    public int horasAsignadasAProyecto(String codigoProyecto) {
        //Por hacer
        return 0;
    }

    public double costeActualDeProyecto(String codigoProyecto) {
        //Por hacer
        return 0;
    }

    public double margenActualProyecto(String codigoProyecto) {
        //Por hacer
    }
}
```

```

    }
    return 0;
}

```

5 Requerimiento 5: Probando Empleado, Proyecto, DepartamentoImpl, ...

TestEmpleado.java

```

package testing;
import java.sql.Date;
import modelo.javabeen.Departamento;
import modelo.javabeen.Perfil;
import modelo.javabeen.Empleado;

public class TestEmpleado {
    public static void main (String[] args) {

        Perfil perfil1 = new Perfil(9, "N/A");
        Departamento dpto1 = new Departamento(99, "Atención al cliente", "N/A");
        Empleado empleado1 = new Empleado(999, "Olivia", "Pérez Martínez", 'M',
"olpema@tt.com", null,
30000, Date.valueOf("2011-02-15"), Date.valueOf("2011-02-15"), perfil1,
dpto1);

        //Probando que los métodos de la clase Empleado funcionan
        System.out.println(empleado1.salarioMensual(12));
        System.out.println(empleado1.literalSexo());
        System.out.println(empleado1.obtenerEmail());
        System.out.println(empleado1.nombreCompleto());
    }
}

```

```

<terminated> TestEmpleado [Java Application] C:\Users\
2500.0
Mujer
Opérez
Olivia Pérez Martínez

```

TestProyecto.java

```

package testing;
import java.sql.Date;
import modelo.javabeen.Cliente;
import modelo.javabeen.Departamento;
import modelo.javabeen.Empleado;
import modelo.javabeen.Perfil;
import modelo.javabeen.Proyecto;
public class TestProyecto {
    public static void main (String[] args) {

        Perfil perfil1 = new Perfil(9, "N/A");
        Departamento dpto1 = new Departamento(99, "Atención al cliente", "N/A");
        Cliente cliente1 = new Cliente("Z00000000", "Zacarías", "Zárate", "Zaragoza",
9999999, 200);
        Empleado empleado1 = new Empleado(999, "Olivia", "Pérez Martínez", 'M',
"olpema@tt.com", null,
30000, Date.valueOf("2011-02-15"), Date.valueOf("2011-02-15"), perfil1,
dpto1);
        Proyecto proy1 = new Proyecto ("FOR2021999", null, Date.valueOf("2020-01-01"),
Date.valueOf("2020-09-01"), Date.valueOf("2020-11-01"),
99999, 75000, 80000, "terminado", empleado1, cliente1);

        //Probando que los métodos de la clase Empleado funcionan
        System.out.println(proy1.diferenciaGastos());
        System.out.println(proy1.diferenciaFinPrevistoReal());
    }
}

```

```

    }
}
<terminated> TestProyecto [Java Application] C:\Users\... \Desktop\Programas de instalación\Eclipse IDE for Enterprise Java
5000.0
-61

```

```

TestDepartamentoImpl.java
package testing;
import modelo.daojdbc.DepartamentoImplMySQL;
public class TestDepartamentoImpl {

    public static void main(String[] args) {

        DepartamentoImplMySQL dpto1= new DepartamentoImplMySQL();
        System.out.println("Datos de ese departamento:");
        System.out.println(dpto1.buscarUno(20));

    }
}
<terminated> TestDepartamentoImpl [Java Application] C:\Users\... \Desktop\Programas de instalación\Eclipse IDE for Enterprise Java
Conexión establecida.
Datos de ese departamento:
Departamento [idDepar=20, nombre=Software, direccion=Madrid]

```

```

TestPerfilImpl.java
package testing;
import modelo.daojdbc.PerfilImplMySQL;

public class TestPerfilImpl {
    public static void main(String[] args) {
        PerfilImplMySQL perfil1= new PerfilImplMySQL();
        System.out.println("Datos de ese perfil:");
        System.out.println(perfil1.buscarUno(1));
    }
}
<terminated> TestPerfillImpl [Java Application] C:\Users\... \Desktop\Programas de instalación\Eclipse IDE for Enterprise Java
Conexión establecida.
Datos de ese perfil:
Perfil [idPerfil=1, nombre=Control de Gestion]

```

```

TestEmpleadoImpl
package testing;
import java.sql.Date;
import modelo.daojdbc.DepartamentoImplMySQL;
import modelo.daojdbc.EmpleadoImplMySQL;
import modelo.daojdbc.PerfilImplMySQL;
import modelo.javabean.Empleado;

public class TestEmpleadoImpl {

    public static void main(String[] args) {

        EmpleadoImplMySQL empimpl1 = new EmpleadoImplMySQL();

        //Métodos DML-select-----
        //1
        System.out.println("Lista de todos los empleados:");
        for (Empleado ele: empimpl1.buscarTodos())
            System.out.println(ele);
        //2
        System.out.println("Datos de un empleado dado su idEmpl:");
        System.out.println(empimpl1.buscarUno(116));
        //3
    }
}

```

```

System.out.println("Empleados de ese apellido:");
for (Empleado ele: empimpl1.empleadosByApellido("koo"))
    System.out.println(ele);
//4
System.out.println("Empleados de ese sexo:");
for (Empleado ele: empimpl1.empleadosBySexo('H'))
    System.out.println(ele);

//5
System.out.println("Empleados de ese departamento:");
for (Empleado ele: empimpl1.empleadosByDepartamento(10))
    System.out.println(ele);

//Métodos DML-insert-----
PerfilImplMySQL perfilimpl1 = new PerfilImplMySQL();
DepartamentoImplMySQL dptoimpl1 = new DepartamentoImplMySQL();
System.out.println("Nuevo empleado a nivel de java");
Empleado empleado1 = new Empleado(999, "Olivia", "Pérez Martínez", 'M',
"olpema@tt.com", "contraseña", 30000, Date.valueOf("2011-02-15"), Date.valueOf("2011-02-
15"), perfilimpl1.buscarUno(4), dptoimpl1.buscarUno(40));
System.out.println("Insercción en sql de ese empleado (devuelve true si el insert
es satisfactorio)");
System.out.println(empimpl1.altaEmpleado(empleado1));

System.out.println("Buscar un empleado por idEmpl");
System.out.println(empimpl1.buscarUno(999));

//Métodos DML-update-----
System.out.println("Modificación en java de datos de ese empleado");
/*Ojo NO funciona empimpl1.buscarUno(999).setSalario(35000)*/;
empleado1.setSalario(33000);
System.out.println(empleado1.getSalario());
System.out.println("Actualización en sql de datos de ese empleado");
System.out.println(empimpl1.modificarEmpleado(empleado1));
System.out.println(empimpl1.buscarUno(999));

//Métodos DML-delete-----
System.out.println("Dar de baja a ese empleado (devuelve true si el delete es
satisfactorio)");
System.out.println(empimpl1.bajaEmpleado(999));

//9
System.out.println("Suma de todos los salarios:");
System.out.println(empimpl1.salarioTotal());
//10
System.out.println("Suma de todos los salarios de ese departamento:");
System.out.println(empimpl1.salarioTotal(10));
}

```

```

}
<terminated> TestEmpleadoImpl [Java Application] C:\Users\... \Desktop\Programas de instalación\Eclipse IDE for Enterprise Java and Web Developers jee-2022-06-R-wins32-x86_64
Conexión establecida.
Lista de todos los empleados:
Empleado [idEmpl=100, nombre=esteban, apellidos=Diaz, sexo=H, email=ediaz@tt.com, password=esteban, salario=90000.0, fechaIngreso=1990-01-1
Empleado [idEmpl=101, nombre=Sara, apellidos=Hernandez, sexo=M, email=shernandez@tt.com, password=sara, salario=45000.0, fechaIngreso=2005-
Empleado [idEmpl=114, nombre=Rafael, apellidos=Raphaelly, sexo=H, email=rraphaelly@tt.com, password=rafael, salario=78000.0, fechaIngreso=2
Empleado [idEmpl=115, nombre=Carlos, apellidos=Koo, sexo=H, email=ckoo@tt.com, password=carlos, salario=33000.0, fechaIngreso=2015-09-07, f
Empleado [idEmpl=116, nombre=Carmen, apellidos=Baida, sexo=M, email=cbaida@tt.com, password=carmen, salario=32000.0, fechaIngreso=2015-09-0
Empleado [idEmpl=117, nombre=Alejandro, apellidos=Himuro, sexo=H, email=ahimuro@tt.com, password=alejandros, salario=25000.0, fechaIngreso=2
Empleado [idEmpl=118, nombre=Eva, apellidos=Colmenares, sexo=M, email=ecolmenares@tt.com, password=eva, salario=25000.0, fechaIngreso=2015-
Empleado [idEmpl=119, nombre=Eva, apellidos=Tobias, sexo=M, email=etobias@tt.com, password=eva, salario=25000.0, fechaIngreso=2015-09-09, f
Empleado [idEmpl=120, nombre=Raqueel, apellidos=Oliva, sexo=M, email=roliva@tt.com, password=raqueel, salario=38000.0, fechaIngreso=2015-09-0
Datos de un empleado dado su idEmpl:
Empleado [idEmpl=116, nombre=Carmen, apellidos=Baida, sexo=M, email=cbaida@tt.com, password=carmen, salario=32000.0, fechaIngreso=2015-09-0
Empleados de ese apellido:
Empleado [idEmpl=115, nombre=Carlos, apellidos=Koo, sexo=H, email=ckoo@tt.com, password=carlos, salario=33000.0, fechaIngreso=2015-09-07, f

```



```

Empleados de ese sexo:
Empleado [idEmpl=100, nombre=esteban, apellidos=Diaz, sexo=H, email=ediaz@tt.com, password=esteban, salario=90000.0, fechaIngreso=1990-01-1
Empleado [idEmpl=114, nombre=Rafael, apellidos=Raphaelly, sexo=H, email=rraphaelly@tt.com, password=rafael, salario=78000.0, fechaIngreso=2
Empleado [idEmpl=115, nombre=Carlos, apellidos=Koo, sexo=H, email=ckoo@tt.com, password=carlos, salario=33000.0, fechaIngreso=2015-09-07, f
Empleado [idEmpl=117, nombre=Alejandro, apellidos=Himuro, sexo=H, email=ahimuro@tt.com, password=alejandro, salario=25000.0, fechaIngreso=2
Empleados de ese departamento:
Empleado [idEmpl=100, nombre=esteban, apellidos=Diaz, sexo=H, email=ediaz@tt.com, password=esteban, salario=90000.0, fechaIngreso=1990-01-1
Empleado [idEmpl=101, nombre=Sara, apellidos=Hernandez, sexo=M, email=shernandez@tt.com, password=sara, salario=45000.0, fechaIngreso=2005-
Nuevo empleado a nivel de java
Insercción en sql de ese empleado (devuelve true si el insert es satisfactorio)
true
Buscar un empleado por idEmpl
null
Modificación en java de datos de ese empleado
33000.0
Actualización en sql de datos de ese empleado
true
null
Dar de baja a ese empleado (devuelve true si el delete es satisfactorio)
true
Suma de todos los salarios:
391000.0

```

TestProyectoImpl

```

package testing;
import java.sql.Date;
import modelo.javabeen.Cliente;
import modelo.javabeen.Departamento;
import modelo.javabeen.Empleado;
import modelo.javabeen.Perfil;
import modelo.javabeen.Proyecto;
public class TestProyecto {
    public static void main (String[] args) {

        Perfil perfil1 = new Perfil(9, "N/A");
        Departamento dpto1 = new Departamento(99, "Atención al cliente", "N/A");
        Cliente cliente1 = new Cliente("Z00000000", "Zacarías", "Zárate", "Zaragoza",
9999999, 200);
        Empleado empleado1 = new Empleado(999, "Olivia", "Pérez Martínez", 'M',
"olpema@tt.com", null,
30000, Date.valueOf("2011-02-15"), Date.valueOf("2011-02-15"), perfil1,
dpto1);
        Proyecto proy1 = new Proyecto ("FOR2021999", null, Date.valueOf("2020-01-01"),
Date.valueOf("2020-09-01"), Date.valueOf("2020-11-01"),
99999, 75000, 80000, "terminado", empleado1, cliente1);

        //Probando que los métodos de la clase Empleado funcionan
        System.out.println(proy1.diferenciaGastos());
        System.out.println(proy1.diferenciaFinPrevistoReal());

    }
}

```

```

<terminated> TestProyecto [Java Application] C:\Users\
5000.0
-61

```

TestEmpleadoEnProyectoImpl

```

package testing;
import modelo.daojdbc.EmpleadoEnProyectoImplMySQL;
import modelo.javabeen.EmpleadoEnProyecto;

public class TestEmpleadoEnProyectoImpl {

    public static void main(String[] args) {

        EmpleadoEnProyectoImplMySQL empepimpl1 = new EmpleadoEnProyectoImplMySQL();

        //Métodos DML-select-----
        System.out.println("Empleados en ese proyecto:");
        for (EmpleadoEnProyecto elemento: empepimpl1.empleadosByProyecto("FOR2020001"))

```

```
System.out.println(elemento);
```

```
/*Falta probar asignarEmpleadosAProyecto(List<EmpleadoEnProyecto> empleados);  
* horasAsignadasAProyecto(String codigoProyecto);  
* costeActualDeProyecto(String codigoProyecto);  
* margenActualProyecto(String codigoProyecto);  
*/  
  
}
```

```
}
```

```
<terminated> TestEmpleadoEnProyectoImpl [Java Application] C:\Users\... \Desktop\Programas de instalación\Eclipse IDE for Enterprise Java and Web Developers jee-2022-06-R-wins32-x86_64\plugins\org.e  
Conexión establecida.  
Empleados en ese proyecto:  
ProyectoConEmpleados [numeroOrden=1, proyecto=Proyecto [idProyecto=FOR2020001, descripcion=Formacion de habilidades directivas, fechaInicio=2020-01-15, fechaF  
ProyectoConEmpleados [numeroOrden=2, proyecto=Proyecto [idProyecto=FOR2020001, descripcion=Formacion de habilidades directivas, fechaInicio=2020-01-15, fechaF
```