```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:
1.- ALTA DEPARTAMENTO
2.- ALTA EMPLEADO
3.- LISTAR EMPLEADO
4.- TRANSFERENCIA EMPLEADO
5.- BAJA EMPLEADO
6.- MEDIA SALDO POR EMPLEADO
0.- SALIR
Opcion
```

ALTA DEPARTAMENTO

```
Opcion
1
Nombre del departamento:
Desarrollo
Departamento{id=4, nombre='Desarrollo'}
```

id	nombre
1	IT
2	Marketing
3	Diseño
4	Desarrollo
NULL	NULL

ALTA EMPLEADO

```
2
DNI del empleado:
112233446
Nombre del empleado:
Godofredo
Sueldo del empleado:
2300
Departamento{id=1, nombre='IT'}
Departamento{id=2, nombre='Marketing'}
Departamento{id=3, nombre='Diseño'}
Departamento{id=4, nombre='Desarrollo'}
Indica ID de departamento:
4
Empleado{id=5, dni='112233446', nombre='Godofredo', sueldo=2300.0, idDepartamento=Departamento{id=4, nombre='Desarrollo'}}
Godofredo dado de alta
```

id	dni	nombre	sueldo	id_departamento
1	12345678A	Pedro	1200.00	3
2	87654321Z	Roberto	1500.00	2
3	19283746D	Jacinto	1760.40	1
4	19253746F	Felipe	1300.34	2
5	11223344G	Godofredo	2300.00	4
NULL	NULL	NULL	NULL	NULL

MOSTRAR EMPLEADOS DE DEPARTAMENTO SELECCIONADO

```
3
Departamento{id=1, nombre='IT'}
Departamento{id=2, nombre='Marketing'}
Departamento{id=3, nombre='Diseño'}
Departamento{id=4, nombre='Desarrollo'}
Indica ID de departamento:
2
Empleados del departamento: Marketing
Empleado{id=2, dni='87654321Z', nombre='Roberto', sueldo=1500.0, idDepartamento=Departamento{id=2, nombre='Marketing'}}
Empleado{id=4, dni='19253746F', nombre='Felipe', sueldo=1300.34, idDepartamento=Departamento{id=2, nombre='Marketing'}}
```

TRANSFERENCIA DE EMPLEADO A OTRO DEPARTAMENTO

```
4
Empleado{id=1, dni='12345678A', nombre='Pedro', sueldo=1200.0, idDepartamento=Departamento{id=3, nombre='Diseño'}}
Empleado{id=2, dni='87654321Z', nombre='Roberto', sueldo=1500.0, idDepartamento=Departamento{id=2, nombre='Marketing'}}
Empleado{id=3, dni='19283746D', nombre='Jacinto', sueldo=1760.4, idDepartamento=Departamento{id=1, nombre='IT'}}
Empleado{id=4, dni='19253746F', nombre='Felipe', sueldo=1300.34, idDepartamento=Departamento{id=2, nombre='Marketing'}}
Empleado{id=5, dni='11223344G', nombre='Godofredo', sueldo=2300.0, idDepartamento=Departamento{id=4, nombre='Desarrollo'}}
Indica ID de empleado:
4
Departamento{id=1, nombre='IT'}
Departamento{id=2, nombre='Marketing'}
Departamento{id=3, nombre='Diseño'}
Departamento{id=4, nombre='Desarrollo'}
Indica ID de departamento:
4
Felipe transferido correctamente a Desarrollo
```

id	dni	nombre	sueldo	id_departamento
1	12345678A	Pedro	1200.00	3
2	87654321Z	Roberto	1500.00	2
3	19283746D	Jacinto	1760.40	1
4	19253746F	Felipe	1300.34	4
5	11223344G	Godofredo	2300.00	4
NULL	NULL	NULL	NULL	NULL

BAJA DE EMPLEADO

```
5
Empleado{id=1, dni='12345678A', nombre='Pedro', sueldo=1200.0, idDepartamento=Departamento{id=3, nombre='Diseño'}}
Empleado{id=2, dni='87654321Z', nombre='Roberto', sueldo=1500.0, idDepartamento=Departamento{id=2, nombre='Marketing'}}
Empleado{id=3, dni='19283746D', nombre='Jacinto', sueldo=1760.4, idDepartamento=Departamento{id=1, nombre='IT'}}
Empleado{id=4, dni='19253746F', nombre='Felipe', sueldo=1300.34, idDepartamento=Departamento{id=4, nombre='Desarrollo'}}
Empleado{id=5, dni='11223344G', nombre='Godofredo', sueldo=2300.0, idDepartamento=Departamento{id=4, nombre='Desarrollo'}}
Indica ID de empleado:
5
Godofredo dado de baja
```

id	dni	nombre	sueldo	id_departamento
1	12345678A	Pedro	1200.00	3
2	87654321Z	Roberto	1500.00	2
3	19283746D	Jacinto	1760.40	1
4	19253746F	Felipe	1300.34	4
NULL	NULL	NULL	NULL	NULL

SALARIO MEDIO POR DEPARTAMENTO

```
6
SALDO MEDIO POR DEPARTAMENTO:
                        Diseño - 1200,00€
                    Desarrollo - 1300,34€
                            IT - 1760,40€
                     Marketing - 1500,00€
```

CIERRE DEL PROGRAMA

```
0
Fin de programa.

Process finished with exit code 0
```

APP.JAVA

```java
public class App {

    public static final int ALTA_DEPARTAMENTO = 1;  1 usage
    public static final int ALTA_EMPLEADO = 2;  1 usage
    public static final int LISTAR_EMPLEADOS = 3;  1 usage
    public static final int TRANSFERENCIA_EMPLEADOS = 4;  1 usage
    public static final int BAJA_EMPLEADOS = 5;  1 usage
    public static final int SALDO_MEDIO = 6;  1 usage
    public static final int SALIR = 0;  1 usage

    public static int menu() {  1 usage
        System.out.println("1.- ALTA DEPARTAMENTO");
        System.out.println("2.- ALTA EMPLEADO");
        System.out.println("3.- LISTAR EMPLEADO");
        System.out.println("4.- TRANSFERENCIA EMPLEADO");
        System.out.println("5.- BAJA EMPLEADO");
        System.out.println("6.- MEDIA SALDO POR EMPLEADO");
        System.out.println("0.- SALIR");
        return ConsoleHelper.pedirEntero( mensaje: "Opcion",  valorMinimo: 0,  valorMaximo: 6);
    }
```

```java
    public static Departamento obtenerDepartamento() throws SQLException {  3 usages
        Servicio servicio = new ServicioImpl();
        Departamento departamentoSeleccionado = null;
        List<Departamento> departamentos = servicio.listarDepartamentos();
        departamentos.forEach(System.out::println);
        do {
            long id = ConsoleHelper.pedirEntero( mensaje: "Indica ID de departamento: ");
            departamentoSeleccionado = departamentos.stream().filter( Departamento c -> c.getId() == id).findFirst()
            if (departamentoSeleccionado == null) System.out.println("Departamento ID incorrecto.");
        } while (departamentoSeleccionado == null);
        return departamentoSeleccionado;
    }

    public static Empleado obtenerEmpleado() throws SQLException {  2 usages
        Servicio servicio = new ServicioImpl();
        Empleado empleadoSeleccionado = null;
        List<Empleado> empleados = servicio.listarEmpleados();
        empleados.forEach(System.out::println);
        do {
            long id = ConsoleHelper.pedirEntero( mensaje: "Indica ID de empleado: ");
            empleadoSeleccionado = empleados.stream().filter( Empleado c -> c.getId() == id).findFirst().orElse( other:
            if (empleadoSeleccionado == null) System.out.println("Empleado ID incorrecto.");
        } while (empleadoSeleccionado == null);
        return empleadoSeleccionado;
    }
```

```java
public static void altaDepartamento() {  1 usage
    try {
        Servicio servicio = new ServicioImpl();

        String nombre = ConsoleHelper.pedirCadena( mensaje: "Nombre del departamento: ");
        Departamento nuevoDepartamento = new Departamento(nombre);

        servicio.altaDepartamento(nuevoDepartamento);

        System.out.println(nuevoDepartamento);
        System.out.println(nuevoDepartamento.getNombre() + " dado de alta");
    } catch (SQLException ex) {
        System.out.println("Error dando de alta departamento: " + ex.getMessage());
    }
}
```

```java
public static void altaEmpleado() {  1 usage
    try {
        Servicio servicio = new ServicioImpl();
        String dni = ConsoleHelper.pedirCadena( mensaje: "DNI del empleado: ");
        String nombre = ConsoleHelper.pedirCadena( mensaje: "Nombre del empleado: ");
        double sueldo = ConsoleHelper.pedirDecimal( mensaje: "Sueldo del empleado: ");
        Departamento departamentoSeleccionado = obtenerDepartamento();
        Empleado nuevoEmpleado = new Empleado(dni, nombre, sueldo, departamentoSeleccionado);

        servicio.altaEmpleado(nuevoEmpleado);

        System.out.println(nuevoEmpleado);
        System.out.println(nuevoEmpleado.getNombre() + " dado de alta");
    } catch (SQLException ex) {
        System.out.println("Error dando de alta empleado: " + ex.getMessage());
    }
}
```

```java
public static void listarEmpleadosDepartamento() {  1 usage
    try {
        Servicio servicio = new ServicioImpl();
        Departamento departamentoSeleccionado = obtenerDepartamento();
        List<Empleado> empleadosPorDepartamento = servicio.listarEmpleados() List<Empleado>
                .stream() Stream<Empleado>
                .filter( Empleado e -> e.getIdDepartamento().getId() == departamentoSeleccionado.getId())
                .toList();
        System.out.println("Empleados del departamento: " + departamentoSeleccionado.getNombre());
        empleadosPorDepartamento.forEach(System.out::println);
    } catch (SQLException ex) {
        System.out.println("Error listando empleados por departamento: " + ex.getMessage());
    }
}
```

```java
public static void transferenciaEmpleado() {  1 usage
    try {
        Servicio servicio = new ServicioImpl();
        Empleado empleadoTransferido = obtenerEmpleado();
        Departamento departamentoNuevo = obtenerDepartamento();
        empleadoTransferido.setIdDepartamento(departamentoNuevo);
        servicio.actualizarEmpleado(empleadoTransferido);

        System.out.println(empleadoTransferido.getNombre() + " transferido correctamente a " + departamentoNuev
    } catch (SQLException ex) {
        System.out.println("Error transfiriendo empleado: " + ex.getMessage());
    }
}

public static void bajaEmpleado() {  1 usage
    try {
        Servicio servicio = new ServicioImpl();
        Empleado empleadoSeleccionado = obtenerEmpleado();
        servicio.bajaEmpleado(empleadoSeleccionado);

        System.out.println(empleadoSeleccionado.getNombre() + " dado de baja");
    } catch (SQLException ex) {
        System.out.println("Error dando de baja empleado: " + ex.getMessage());
    }
}
```

```java
public static void salarioMedio() {  1 usage
    try {
        Servicio servicio = new ServicioImpl();
        Map<String, Double> totales = servicio.saldoMedioDepartamento();
        System.out.println("SALDO MEDIO POR DEPARTAMENTO: ");
        totales.forEach(( String nombre_departamento,  Double recuento) -> {
            System.out.printf("%30s - %.2f€\n", nombre_departamento, recuento);
        });
    } catch (SQLException ex) {
        System.out.println("Error calculando salario medio: " + ex.getMessage());
    }
}
```

```java
public static void main(String[] args) {
    int opcion;
    do {
        opcion = menu();
        if (opcion == ALTA_DEPARTAMENTO)
            altaDepartamento();
        else if (opcion == ALTA_EMPLEADO)
            altaEmpleado();
        else if (opcion == LISTAR_EMPLEADOS)
            listarEmpleadosDepartamento();
        else if (opcion == TRANSFERENCIA_EMPLEADOS)
            transferenciaEmpleado();
        else if (opcion == BAJA_EMPLEADOS)
            bajaEmpleado();
        else if (opcion == SALDO_MEDIO)
            salarioMedio();
    } while (opcion != SALIR);
    System.out.println("Fin de programa.");
}
```

CRUD

```java
package dao;

import java.sql.SQLException;
import java.util.List;

public interface CrudDAO<T> {  no usages  2 implementations

    void insertar(T obj) throws SQLException;  no usages  2 implementations

    List<T> listar() throws SQLException;  no usages  2 implementations

    void eliminar( T obj ) throws SQLException;  no usages  2 implementations

    void actualizar( T obj ) throws SQLException;  no usages  2 implementations

    T obtener( long id ) throws SQLException;  no usages  2 implementations


}
```

CRUD EMPLEADO

```java
public class EmpleadoDAO implements CrudDAO<Empleado>{  no usages
    private Connection con;  8 usages

    public EmpleadoDAO( Connection con ) { this.con = con; }

    @Override  no usages
    public void insertar(Empleado obj) throws SQLException {
        String sql = "INSERT INTO empleados( id, dni, nombre, sueldo, id_departamento ) VALUES ( ?, ?, ?, ?
        try(PreparedStatement pst = con.prepareStatement(sql, PreparedStatement.RETURN_GENERATED_KEYS)) {
            pst.setLong( parameterIndex: 1, obj.getId());
            pst.setString( parameterIndex: 2, obj.getDni());
            pst.setString( parameterIndex: 3, obj.getNombre());
            pst.setDouble( parameterIndex: 4, obj.getSueldo());
            pst.setLong( parameterIndex: 5, obj.getIdDepartamento().getId());
            pst.executeUpdate();
            ResultSet rs = pst.getGeneratedKeys();
            if ( rs.next()) {
                obj.setId(rs.getLong( columnIndex: 1));
            }
        }
    }

    @Override  no usages
    public List<Empleado> listar() throws SQLException {
        List<Empleado> empleados = new ArrayList<>();
        CrudDAO<Departamento> departamentoDAO = new DepartamentoDAO(con);
        String sql = "SELECT id, dni, nombre, sueldo, id_departamento FROM empleados";
```

```java
public class EmpleadoDAO implements CrudDAO<Empleado>{  no usages
    public List<Empleado> listar() throws SQLException {
        CrudDAO<Departamento> departamentoDAO = new DepartamentoDAO(con);
        String sql = "SELECT id, dni, nombre, sueldo, id_departamento FROM empleados";
        try( Statement st = con.createStatement()) {
            ResultSet rs = st.executeQuery(sql);
            while( rs.next()) {
                long id = rs.getLong( columnLabel: "id");
                String dni = rs.getString( columnLabel: "dni");
                String nombre = rs.getString( columnLabel: "nombre");
                double sueldo = rs.getDouble( columnLabel: "sueldo");
                long idDepartamento = rs.getLong( columnLabel: "id_departamento");
                Departamento departamento = departamentoDAO.obtener(idDepartamento);
                empleados.add(new Empleado(id, dni, nombre, sueldo, departamento));
            }
        }
        return empleados;
    }

    @Override  no usages
    public void eliminar(Empleado obj) throws SQLException {
        String sql = "DELETE FROM empleados WHERE id = ?";
        try( PreparedStatement pst = con.prepareStatement(sql)) {
            pst.setLong( parameterIndex: 1, obj.getId());
            pst.executeUpdate();
        }
    }
```

CRUD DEPARTAMENTO

```java
package dao;

import modelo.Departamento;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class DepartamentoDAO implements CrudDAO<Departamento>{  6 usages

    private Connection con;  6 usages

    public DepartamentoDAO( Connection con ) { this.con = con; }

    @Override  2 usages
    public void insertar(Departamento obj) throws SQLException {
        String sql = "INSERT INTO departamentos( id, nombre ) VALUES ( ?, ? )";
        try(PreparedStatement pst = con.prepareStatement(sql, PreparedStatement.RETURN_GENERATED_KEYS)) {
            pst.setLong( parameterIndex: 1, obj.getId());
            pst.setString( parameterIndex: 2, obj.getNombre());
            pst.executeUpdate();
            ResultSet rs = pst.getGeneratedKeys();
            if ( rs.next()) {
                obj.setId(rs.getLong( columnIndex: 1));
            }
        }
    }
```

```java
@Override  2 usages
public List<Departamento> listar() throws SQLException {
    List<Departamento> departamentos = new ArrayList<>();
    String sql = "SELECT id, nombre FROM departamentos";
    try(Statement st = con.createStatement()) {
        ResultSet rs = st.executeQuery(sql);
        while( rs.next()) {
            long id = rs.getLong( columnLabel: "id");
            String nombre = rs.getString( columnLabel: "nombre");
            departamentos.add(new Departamento(id, nombre));
        }
    }
    return departamentos;
}

@Override  1 usage
public void eliminar(Departamento obj) throws SQLException {
    String sql = "DELETE FROM departamentos WHERE id = ?";
    try( PreparedStatement pst = con.prepareStatement(sql)) {
        pst.setLong( parameterIndex: 1, obj.getId());
        pst.executeUpdate();
    }
}

@Override  1 usage
public void actualizar(Departamento obj) throws SQLException {
    String sql = "UPDATE departamentos SET nombre = ? WHERE id = ?";
```

DATASOURCE

```
public class Database {  10 usages
    private static String user;  2 usages
    private static String password;  2 usages

    private static boolean configLoaded;  3 usages

    static {
        Properties propiedades = new Properties();
        try ( InputStream is = Database.class.getResourceAsStream( name: "/config.properties") ) {
            if (is == null) {
                throw new IOException("Fichero de recursos no encontrado");
            }
            // Cargar propiedades
            propiedades.load ( is );
            // Obtener valores de las propiedades
            url = propiedades.getProperty("url");
            user = propiedades.getProperty("user");
            password = propiedades.getProperty("password");
            configLoaded = true;
        } catch (IOException e) {
            configLoaded = false;
            System.out.println(e.getMessage());
        }
    }

    public static Connection getConexion() throws SQLException {  8 usages
        if ( !configLoaded ) throw new SQLException("Config not loaded");
        return DriverManager.getConnection(url, user, password);
```

CLASE DEPARTAMENTO

```java
package modelo;

public class Departamento {  5 usages

    private long id;  4 usages
    private String nombre;  5 usages

    public Departamento() {}  no usages

    public Departamento(String nombre) {  no usages

        this.nombre = nombre;
    }

    public Departamento(long id, String nombre) {  no usages
        this.id = id;
        this.nombre = nombre;
    }

    public long getId() {  no usages
        return id;
    }

    public String getNombre() {  no usages
        return nombre;
    }

    public void setId(long id) {  no usages
```

CLASE EMPLEADO

```java
public class Empleado {  34 usages                                                    4  18
    private long id;  4 usages
    private String dni;  5 usages
    private String nombre;  5 usages
    private double sueldo;  5 usages
    private Departamento idDepartamento;  5 usages

    public Empleado() {  no usages

    }

    public Empleado(String dni, String nombre, double sueldo, Departamento idDepartamento) {  1 usage
        this.dni = dni;
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.idDepartamento = idDepartamento;
    }

    public Empleado(long id, String dni, String nombre, double sueldo, Departamento idDepartamento) {  2 usages
        this.id = id;
        this.dni = dni;
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.idDepartamento = idDepartamento;
    }

    public long getId() {  4 usages
```

SERVICIO

```java
package servicios;

import modelo.Departamento;
import modelo.Empleado;

import java.sql.SQLException;
import java.util.List;
import java.util.Map;

public interface Servicio {  10 usages  1 implementation
    void altaDepartamento(Departamento departamento ) throws SQLException;  1 usage  1 implementation
    void altaEmpleado( Empleado empleado ) throws SQLException;  1 usage  1 implementation
    List<Departamento> listarDepartamentos() throws SQLException;  1 usage  1 implementation
    List<Empleado> listarEmpleados() throws SQLException;  3 usages  1 implementation
    Departamento obtenerDepartamento( long id ) throws SQLException;  no usages  1 implementation
    Empleado obtenerEmpleado( long id ) throws SQLException;  no usages  1 implementation
    void actualizarEmpleado( Empleado empleado ) throws SQLException;  1 usage  1 implementation
    void bajaEmpleado( Empleado empleado ) throws SQLException;  1 usage  1 implementation
    Rename usages
    Map<String, Double> saldoMedioDepartamento() throws SQLException;  1 implementation
}
```

SERVICIOIMPL

```java
public class ServicioImpl implements Servicio{  9 usages
    public void altaDepartamento(Departamento departamento) throws SQLException {
        try(Connection con = Database.getConexion()) {
            CrudDAO<Departamento> dao = new DepartamentoDAO(con);
            dao.insertar(departamento);
        }
    }

    @Override  1 usage
    public void altaEmpleado(Empleado empleado) throws SQLException {
        try(Connection con = Database.getConexion()) {
            CrudDAO<Empleado> dao = new EmpleadoDAO(con);
            dao.insertar(empleado);
        }
    }

    @Override  1 usage
    public List<Departamento> listarDepartamentos() throws SQLException {
        try(Connection con = Database.getConexion()) {
            CrudDAO<Departamento> dao = new DepartamentoDAO(con);
            return dao.listar();
        }
    }

    @Override  3 usages
    public List<Empleado> listarEmpleados() throws SQLException {
        try(Connection con = Database.getConexion()) {
```