Ejercicio 8: Creación aplicación CRUD

Introducción

El proyecto está compuesto de 6 clases distintas. Una para la representación de cada una de las tablas (Departamento y Empleado). Otras dos que se encargarán de la conexión a la base de datos y la realización de las ejecuciones necesarias (Create, Read, Update y Delete). Otra donde escribiremos los métodos especificados en el proyecto (Listas). Y por último la clase main (ClasesSQL), donde llamaremos a los métodos de Listas para configurar adecuadamente los requerimientos del proyecto.

Clase Departamento

En esta clase se instancian las variables referentes a los atributos de esta tabla en la base de datos.

```
public class Departamento {
    /**
     * Clase con las variables correspondientes y sus constructores, getters y setters
     */
    private int numeroDepartamento; //dept_no
    private String nombreDepartamento; //dnombre
    private String localidad; //loc
```

A continuación, se instancian los constructores, getters y setters.

```
public Departamento() {
}

public Departamento(int numeroDepartamento, String nombreDepartamento, String localidad) {
    this.numeroDepartamento = numeroDepartamento;
    this.nombreDepartamento = nombreDepartamento;
    this.localidad = localidad;
}

public int getNumeroDepartamento() {
    return numeroDepartamento;
}

public void setNumeroDepartamento(int numeroDepartamento) {
    this.numeroDepartamento = numeroDepartamento;
}

public String getNombreDepartamento() {
    return nombreDepartamento;
}

public void setNombreDepartamento(String nombreDepartamento) {
    this.nombreDepartamento = nombreDepartamento;
}

public String getLocalidad() {
    return localidad;
}

public void setLocalidad(String localidad) {
    this.localidad = localidad;
}
```

Clase Empleado

Al igual que en el caso de Departamento, en esta clase instanciamos las variables correspondientes a los atributos de la tabla Empleados.

```java
public class Empleado {
    /**
     * Clase con las variables correspondientes y sus constructores, getters y setters
     */
    private int numeroEmpleado; //emp_no
    private String apellido; //apellido
    private String oficio; //oficio
    private int direccion; //dir
    private Date fechaAlta; //fecha_alt
    private float salario; //salario
    private float comision; //comision
    private int numeroDepartamento; //dept_no
```

Y al igual que en la clase anterior, instanciamos constructores, getters y setters.

```java
public Empleado() {

}

public Empleado(int numeroEmpleado, String apellido, String oficio, int direccion, Date fechaAlta, float salario, float comision, int numeroDepartamento) {
    this.numeroEmpleado = numeroEmpleado;
    this.apellido = apellido;
    this.oficio = oficio;
    this.direccion = direccion;
    this.fechaAlta = fechaAlta;
    this.salario = salario;
    this.comision = comision;
    this.numeroDepartamento = numeroDepartamento;
}

    public int getNumeroEmpleado() {
        return numeroEmpleado;
    }

    public void setNumeroEmpleado(int numeroEmpleado) {
        this.numeroEmpleado = numeroEmpleado;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getOficio() {
        return oficio;
    }

    public void setOficio(String oficio) {
        this.oficio = oficio;
    }

    public int getDireccion() {
        return direccion;
    }

    public void setDireccion(int direccion) {
        this.direccion = direccion;
    }

    public Date getFechaAlta() {
        return fechaAlta;
    }

    public void setFechaAlta(Date fechaAlta) {
        this.fechaAlta = fechaAlta;
    }

    public float getSalario() {
        return salario;
    }

    public void setSalario(float salario) {
        this.salario = salario;
    }

    public float getComision() {
        return comision;
    }

    public void setComision(float comision) {
        this.comision = comision;
    }

    public int getNumeroDepartamento() {
        return numeroDepartamento;
    }

    public void setNumeroDepartamento(int numeroDepartamento) {
        this.numeroDepartamento = numeroDepartamento;
    }
```

Clase Departamentos

Aquí tenemos los métodos que conectan a la tabla Departamentos de la base de datos. Aquí instanciamos una variable *Connection*, además de establecer la conexión con el servidor en el constructor.

```
private Connection conexion;
//private ArrayList<Departamento> departamentos;

/**
 * Al construir un departamento se realizara la conexion a la bbdd
 */
public Departamentos() { //Contructor: Crea una conexion al MySQL. Al crear un nuevo objeto se conectará a la BBDD.
    try {
        conexion = DriverManager.getConnection("jdbc:mysql://localhost/ejemplo", "ejemplo", "ejemplo");
    } catch (SQLException ex) {
        Logger.getLogger(Departamentos.class.getName()).log(Level.SEVERE, null, ex);
        System.err.println("Error al acceder a la BBDD " + ex.getMessage());
    }
}
```

Aquí tenemos los metros para insertar nuevos datos a la tabla o para actualizarlos.

```
/**
 *
 * @param dep Departamento que se desea añadir a la base de datos
 * @return Devuelve el numero de filas afectadas
 * @throws SQLException
 */
public int Create(Departamento dep) throws SQLException {
    String sql = "INSERT INTO departamentos (dept_no, dnombre, loc) VALUES (?, ?, ?);";
    int filas;
    PreparedStatement sentencia = conexion.prepareCall(sql);
    sentencia.setInt(1, dep.getNumeroDepartamento());
    sentencia.setString(2, dep.getNombreDepartamento());
    sentencia.setString(3, dep.getLocalidad());
    filas = sentencia.executeUpdate();
    return filas;
}

/**
 *
 * @param dep_no Numero del departamento al que se actualizara su
 * informacion
 * @param dep Departamento al cual se actualizara su informacion
 * @return Devuelve el numero de filas que han sido afectadas
 * @throws SQLException
 */
public int Update(int dep_no, Departamento dep) throws SQLException {
    String sql = "UPDATE departamentos SET dnombre =  ?, loc = ? WHERE dept_no = ?;";
    int filas;
    PreparedStatement sentencia = conexion.prepareCall(sql);
    sentencia.setString(1, "PEPE");
    sentencia.setString(2, "BENAMENTE");
    sentencia.setInt(3, dep_no);
    filas = sentencia.executeUpdate();
    return filas;
}
```

Después, vemos los métodos para leer 1 solo departamento y para leerlos todos.

```java
/**
 *
 * @param dep_no Numero de departamento que se desea mostrar
 * @return devuelve el departamento a mostrar
 * @throws SQLException
 */
public Departamento Read(int dep_no) throws SQLException {
    Departamento dep;
    String sql = "SELECT * FROM departamentos WHERE dept_no = ?";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setInt(1, dep_no);
    ResultSet rs = sentencia.executeQuery();
    rs.first();
    dep = new Departamento(rs.getInt(1), rs.getString(2), rs.getString(3));
    return dep;
}

/**
 *
 * @return Devuelve un arraylist con los departamentos a mostrar
 * @throws SQLException
 */
public ArrayList<Departamento> ReadVarios() throws SQLException {
    Departamento dep;
    ArrayList<Departamento> deps = new ArrayList<>();
    String sql = "SELECT * FROM departamentos";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    ResultSet rs = sentencia.executeQuery();

    while (rs.next()) {
        dep = new Departamento(rs.getInt(1), rs.getString(2), rs.getString(3));
        deps.add(dep);
    }
    return deps;
}
```

Para terminar, tenemos los métodos para buscar departamento por nombre, borrar registros y cerrar la conexión con la base de datos.

```
     *
     * @param dep_nom Nombre del departamento que se quiere buscar
     * @return Devuelve dicho departamento
     * @throws SQLException
     */
    public Departamento ReadNombre(String dep_nom) throws SQLException {
        Departamento dep;
        String sql = "SELECT * FROM departamentos WHERE dnombre=?";
        PreparedStatement sentencia = conexion.prepareStatement(sql);
        sentencia.setString(1, dep_nom);
        ResultSet rs = sentencia.executeQuery();
        rs.first();
        dep = new Departamento(rs.getInt(1), rs.getString(2), rs.getString(3));
        return dep;
    }

    /**
     *
     * @param dep Numero del departamento que se desea borrar
     * @return Devuelve el numero de filas afectadas
     * @throws SQLException
     */
    public int Delete(int dep) throws SQLException {
        String sql = "DELETE FROM departamentos WHERE dept_no=?";
        int filas;
        PreparedStatement sentencia = conexion.prepareStatement(sql);
        sentencia.setInt(1, dep);
        filas = sentencia.executeUpdate();
        return filas;
    }

    /**
     * Cierra la conexion con la base de datos
     *
     * @throws SQLException
     */
    public void Close() throws SQLException {
        conexion.close();
    }
```

Clase Empleados

Aquí tenemos los métodos que conectan a la tabla Empleados de la base de datos. Aquí instanciamos una variable *Connection*, además de establecer la conexión con el servidor en el constructor.

```
public class Empleados {
    private Connection conexion;
    //private ArrayList<Empleado> empleados;

    /**
     * Al construir un empleado se realizara la conexion a la bbdd
     */
    public Empleados() {
        try {
            conexion = DriverManager.getConnection("jdbc:mysql://localhost/ejemplo", "ejemplo", "ejemplo");
        } catch (SQLException ex) {
            Logger.getLogger(Empleados.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    /**
     *
     * @param emp Se introduce un empleado
     * @return Devuelve el numero de filas que han sido afectadas
     * @throws SQLException
     */
    public int Create (Empleado emp) throws SQLException{
        String sql = "INSERT INTO empleados VALUES(?,?,?,?,?,?,?,?)";
        int filas;
        PreparedStatement sentencia = conexion.prepareStatement(sql);
        sentencia.setInt(1, emp.getNumeroEmpleado());
        sentencia.setString(2, emp.getApellido());
        sentencia.setString(3, emp.getOficio());
        sentencia.setInt(4, emp.getDireccion());
        sentencia.setDate(5, emp.getFechaAlta());
        sentencia.setFloat(6, emp.getSalario());
        sentencia.setFloat(7, emp.getComision());
        sentencia.setInt(8, emp.getNumeroDepartamento());
        filas = sentencia.executeUpdate();
        return filas;
    }
```

Aquí los métodos para actualizar y leer 1 registro.

```
public int Update (int emp_no, Empleado emp) throws SQLException{
    String sql = "UPDATE empleados SET emp_no = ?, apellido = ?, oficio = ?, dir = ?, fecha_alt = ?, "
            + "salario = ?,comision = ?, dpt_no = ?  WHERE emp_no = ?;";
    int filas;
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setInt(1, emp.getNumeroEmpleado());
    sentencia.setString(2, emp.getApellido());
    sentencia.setString(3, emp.getOficio());
    sentencia.setInt(4, emp.getDireccion());
    sentencia.setDate(5, emp.getFechaAlta());
    sentencia.setFloat(1, emp.getSalario());
    sentencia.setFloat(7, emp.getComision());
    sentencia.setInt(8, emp.getNumeroDepartamento());
    sentencia.setInt(9,emp_no);
    filas = sentencia.executeUpdate();
    return filas;
}
/**
 *
 * @param emp_no Numero de empleado que se desea mostrar
 * @return devuelve el empleado a mostrar
 * @throws SQLException
 */
public Empleado Read (int emp_no) throws SQLException{
    Empleado emp;
    String sql = "SELECT * FROM empleados WHERE emp_no = ?";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setInt(1, emp_no);
    ResultSet rs = sentencia.executeQuery();
    rs.first();
    emp = new Empleado(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4), rs.getDate(5), rs.getFloat(6), rs.getFloat(7), rs.getInt(8));
    return emp;
}
```

Métodos para leer todos los registros y para buscar por nombre.

```
public ArrayList<Empleado> ReadVarios() throws SQLException{
    Empleado emp;
    ArrayList<Empleado> bemps = new ArrayList<>();
    String sql = "SELECT * FROM empleados";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    ResultSet rs = sentencia.executeQuery();

    while(rs.next()){
        emp = new Empleado(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4), rs.getDate(5), rs.getFloat(6), rs.getFloat(7), rs.getInt(8));
        bemps.add(emp);
    }
    return bemps;
}
/**
 *
 * @param emp_nom Nombre del empleado que se quiere buscar
 * @return Devuelve dicho empleado
 * @throws SQLException
 */
public Empleado ReadNombre (String emp_nom) throws SQLException{
    Empleado emp;
    String sql = "SELECT * FROM empleados WHERE apellido = ?";
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setString(1, emp_nom);
    ResultSet rs = sentencia.executeQuery();
    rs.first();
    emp = new Empleado(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4), rs.getDate(5), rs.getFloat(6), rs.getFloat(7), rs.getInt(8));
    return emp;
}
```

Y, por último, métodos para borrar registros y para cerrar la conexión con la base de datos.

```
public int Delete (int emp_no) throws SQLException{
    String sql = "DELETE FROM empleados WHERE emp_no=?";
    int filas;
    PreparedStatement sentencia = conexion.prepareStatement(sql);
    sentencia.setInt(1, emp_no);
    filas = sentencia.executeUpdate();
    return filas;
}
/**
 * Cierra la conexion con la base de datos
 * @throws SQLException
 */
public void Close() throws SQLException {
    conexion.close();
}
```

Clase Lista

En esta clase es donde usamos los métodos de Departamentos y Empleados para realizar los pasos correspondientes que pide el proyecto.

El primer método nos listará 1 o todos los departamentos, según decida el usuario. Para eso le damos a opción al usuario de elegir

```
public void ListarDepartamento() throws SQLException {

    Scanner sc = new Scanner(System.in);

    Departamento dep;
    Departamentos bdep = new Departamentos();

    System.out.println("••••••••••••••••••••••••••••••••••••••••••");
    System.out.println("1- Mostrar un departamento\n2- Mostrar todos los departamentos");
    System.out.println("••••••••••••••••••••••••••••••••••••••••••");
    int n = sc.nextInt();
    System.out.println("••••••••••••••••••••••••••••••••••••••••••");
    switch (n) {
```

```
switch (n) {
    case 1:
        int s = 0;
        while (s <= 0) {
            try {
                while (s <= 0) {
                    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
                    System.out.println("O ¿Qué departamento desea ver?");
                    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
                    s = sc.nextInt();
                    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
                    if (s <= 0) {
                        System.out.println("O El numero debe ser mayor a 0");
                    }
                }
            } catch (InputMismatchException ex) {
                System.out.println("••••••••••••••••••••••••••••••••••••••••••••••");
                System.out.println("O Ha introducido un valor erróneo, vuelvalo a intentar");
                sc.nextLine();
            }
        }
        dep = bdep.Read(s);
        System.out.println("+----------------------------------------------+");
        System.out.printf("|%-15s |%-15s |%-15s| %n", "N°Departamento", "NomDepartamento", "Localidad");
        System.out.println("+----------------------------------------------+");
        System.out.printf("|%-15d |%-15s |%-15s| %n", (dep.getNumeroDepartamento()), dep.getNombreDepartamento(), dep.getLocalidad());
        System.out.println("+----------------------------------------------+");
        break;
    case 2:
        ArrayList<Departamento> bdeps = bdep.ReadVarios();
        System.out.println("+----------------------------------------------+");
        System.out.printf("|%-15s |%-15s |%-15s| %n", "N°Departamento", "NomDepartamento", "Localidad");
        System.out.println("+----------------------------------------------+");
        for (int i = 0; i < bdeps.size(); i++) {
            System.out.printf("|%-15d |%-15s |%-15s| %n", (bdeps.get(i).getNumeroDepartamento()), bdeps.get(i).getNombreDepartamento(), bdeps.get(i).getLocalidad());
        }
        System.out.println("+----------------------------------------------+");
        System.out.println("••••••••••••••••••••••••••••••••••••••••••••••");
        break;
    default:
        System.out.println("O El numero no corresponde con ninguna opcion valida");
}//fin switch
```

Para los Empleados hacemos exactamente lo mismo.

Después, tenemos los métodos para buscar Departamentos y Empleados por nombre

```java
public void DepartamentoPorNombre() throws SQLException {
    Scanner sc = new Scanner(System.in);
    Departamento dep;
    Departamentos bdep = new Departamentos();
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    System.out.println("○ Indica el nombre del departamento que desea ver");
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    String n = sc.nextLine();
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    dep = bdep.ReadNombre(n);
    System.out.println("+-------------------------------------------------+");
    System.out.printf("|%-15s |%-15s |%-15s| %n", "N°Departamento", "NomDepartamento", "Localidad");
    System.out.println("+-------------------------------------------------+");
    System.out.printf("|%-15d |%-15s |%-15s| %n", (dep.getNumeroDepartamento()), dep.getNombreDepartamento(), dep.getLocalidad());
    System.out.println("+-------------------------------------------------+");
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
}//fin DepartamentoPorNombre

/**
 * Busca un empleado por su nombre y lo muestra
 *
 * @throws SQLException
 */
public void EmpleadoPorNombre() throws SQLException {
    Scanner sc = new Scanner(System.in);
    Empleado emp;
    Empleados bemp = new Empleados();
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    System.out.println("○ Indica el nombre del empleado que desea ver");
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    String n = sc.nextLine();
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    emp = bemp.ReadNombre(n);
    System.out.println("+-----------------------------------------------------------------------------------------------------------------------+");
    System.out.printf("|%-15s |%-15s |%-15s |%-15s |%-15s |%-15s |%-15s |%-15s|%n", "N°Empleado", "Apellido", "Oficio", "Direccion", "Fecha Alta", "Salario", "Comision", "N°De
    System.out.println("+-----------------------------------------------------------------------------------------------------------------------+");
    System.out.printf("|%-15d |%-15s |%-15s |%-15s |%-15d |%-15.2f |%-15.2f |%-15d|%n", emp.getNumeroDepartamento(), emp.getApellido(), emp.getOficio(), emp.getDireccion(), em
    System.out.println("+-----------------------------------------------------------------------------------------------------------------------+");
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
}//fin EmpleadoPorNombre
```

A continuación, se muestra el método para crear un nuevo Departamento. El procedimiento sería igual con los Empleados, pero usando la clase correspondiante.

```java
public void CrearDepartamento() throws SQLException {
    System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
    Scanner sc = new Scanner(System.in);
    Departamento dep = new Departamento();
    Departamentos bdep = new Departamentos();
    boolean seguir = true;
    while (seguir) {
        try {
            while (dep.getNumeroDepartamento() <= 0) {
                System.out.println("+-------------------------------------");
                System.out.print("|N°Departamento: ");
                dep.setNumeroDepartamento(sc.nextInt());
                if (dep.getNumeroDepartamento() <= 0) {
                    System.out.println("○ El N°Departamento debe ser mayor a 0");
                }
            }
            while (dep.getNombreDepartamento() == null) {
                System.out.println("+-------------------------------------");
                System.out.print("|NomDepartamento: ");
                sc.nextLine();
                dep.setNombreDepartamento(sc.nextLine());
            }
            while (dep.getLocalidad() == null) {
                System.out.println("+-------------------------------------");
                System.out.print("|Localidad: ");
                dep.setLocalidad(sc.nextLine());
            }
            System.out.println("+-------------------------------------");
            System.out.println("••••••••••••••••••••••••••••••••••••••••••••••••••");
            System.out.println("○ Comando ejecutado, filas afectadas: " + bdep.Create(dep));
            seguir = false;
        } catch (InputMismatchException ex) {
            System.out.println("○ Ha introducido un valor erróneo, vuelvalo a intentar");
            seguir = true;
            sc.nextLine();
        }
    }//fin while
}//fin CrearDepartaento()
```

Por último, los métodos para borrar registros

```java
public void BorrarEmpleado() throws SQLException {
    Scanner sc = new Scanner(System.in);
    Empleados bemp = new Empleados();
    System.out.println("●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●");
    System.out.println("¿Que empleado desea borrar? (Numero)");
    try {
        int n = sc.nextInt();
        System.out.println("○ Comando ejecutado, filas afectadas: " + bemp.Delete(n));
    } catch (InputMismatchException ex) {
        System.out.println("○ Ha introducido un valor erróneo, vuelvalo a intentar");
    }
}//fin BorrarEmpleado()

/**
 * Borra un departamento existente
 *
 * @throws SQLException
 */
public void BorrarDepartamento() throws SQLException {
    Scanner sc = new Scanner(System.in);
    Departamentos bdep = new Departamentos();
    System.out.println("●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●");
    System.out.println("¿Que departamento desea borrar? (Numero)");
    try {
        int n = sc.nextInt();
        System.out.println("○ Comando ejecutado, filas afectadas: " + bdep.Delete(n));
    } catch (InputMismatchException ex) {
        System.out.println("○ Ha introducido un valor erróneo, vuelvalo a intentar");
    }
}//fin BorrarDepartamento()

public void CerrarConexion () {
    Departamentos bdep = new Departamentos();
    Empleados bemp = new Empleados();
    try {
        bdep.Close();
        bemp.Close();
    } catch (SQLException ex) {
        System.out.println("Error al cerrar la BBDD: " + ex.getSQLState() + " " + ex.getMessage());
    }
}//fin CerrarConexion
```

Clase main (ClasesSQL)

Aquí es donde montamos el programa. Usamos un switch, que hace la función de menú. Recogemos los posibles errores con distintos try/catch.

Ejecuciones

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
1
1- Mostrar un departamento
2- Mostrar todos los departamentos
2
+-----------------------------------------------+
|N°Departamento  |NomDepartamento |Localidad    |
+-----------------------------------------------+
|10              |CONTABILIDAD    |SEVILLA      |
|15              |ADIOS           |GRANCANARIA  |
|19              |HOLA            |LAS PALMAS   |
|20              |INVESTIGACIÓN   |MADRID       |
|30              |VENTAS          |BARCELONA    |
|40              |PRODUCCIÓN      |BILBAO       |
|60              |MARKETING       |GUADALAJARA  |
|70              |PRODUCCION      |CANARIAS     |
|99              |Buenas          |A coruña     |
|100             |Juan            |Palomero     |
+-----------------------------------------------+
1- Listar departamentos
```

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
1
1- Mostrar un departamento
2- Mostrar todos los departamentos
1
O ¿Qué departamento desea ver?
10
+-----------------------------------------------+
|N°Departamento  |NomDepartamento |Localidad    |
+-----------------------------------------------+
|10              |CONTABILIDAD    |SEVILLA      |
+-----------------------------------------------+
1- Listar departamentos
```

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
2
1- Mostrar un empleado
2- Mostrar todos los empleados
2
+-----------------------------------------------------------------------------------------------------------------------+
|N°Empleado      |Apellido        |Oficio          |Direccion       |Fecha Alta      |Salario         |Comision        |N°Departamento |
+-----------------------------------------------------------------------------------------------------------------------+
|1               |b               |a               |1               |2018-01-01      |20,000000       |20,000000       |20             |
|4457            |PEPE            |VENDEDOR        |7499            |2018-01-15      |1500,000000     |10,000000       |10             |
|7369            |SÁNCHEZ         |EMPLEADO        |7902            |1990-12-17      |3040,000000     |0,000000        |30             |
|7499            |ARROYO          |VENDEDOR        |7698            |1990-02-20      |1500,000000     |390,000000      |30             |
|7521            |SALA            |VENDEDOR        |7698            |1991-02-22      |1625,000000     |650,000000      |30             |
|7566            |JIMÉNEZ         |DIRECTOR        |7839            |1991-04-02      |2900,000000     |0,000000        |20             |
|7654            |MARTÍN          |VENDEDOR        |7698            |1991-09-29      |1600,000000     |1020,000000     |30             |
|7698            |NEGRO           |DIRECTOR        |7839            |1991-05-01      |3005,000000     |0,000000        |30             |
|7782            |CEREZO          |DIRECTOR        |7839            |1991-06-09      |2885,000000     |0,000000        |10             |
|7788            |GIL             |ANALISTA        |7566            |1991-11-09      |3000,000000     |0,000000        |20             |
|7839            |REY             |PRESIDENTE      |0               |1991-11-17      |4100,000000     |0,000000        |10             |
|7844            |TOVAR           |VENDEDOR        |7698            |1991-09-08      |1350,000000     |0,000000        |30             |
|7876            |ALONSO          |EMPLEADO        |7788            |1991-09-23      |1430,000000     |0,000000        |20             |
|7900            |JIMENO          |EMPLEADO        |7698            |1991-12-03      |1335,000000     |0,000000        |30             |
|7902            |FERNÁNDEZ       |ANALISTA        |7566            |1991-12-03      |3000,000000     |0,000000        |20             |
+-----------------------------------------------------------------------------------------------------------------------+
```

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
2
1- Mostrar un empleado
2- Mostrar todos los empleados
1
O ¿Qué empleado desea ver?
7902
```

N°Empleado	Apellido	Oficio	Direccion	Fecha Alta	Salario	Comision	N°Departamento
7902	FERNÁNDEZ	ANALISTA	7566	1991-12-03	3000,00	0,00	20

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
3
O Indica el nombre del empleado que desea ver
FERNANDEZ
```

N°Empleado	Apellido	Oficio	Direccion	Fecha Alta	Salario	Comision	N°Departamento
7902	FERNÁNDEZ	ANALISTA	7566	1991-12-03	3000,00	0,00	20

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
4
O Indica el nombre del departamento que desea ver
VENTAS
```

N°Departamento	NomDepartamento	Localidad
30	VENTAS	BARCELONA

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
5
+----------------------------------------
|N°Empleado: 55
+----------------------------------------
|Apellido: BENITEZ
+----------------------------------------
|Oficio: VENDEDOR
+----------------------------------------
|Direccion: 7499
+----------------------------------------
|Fecha Alta: 1998-10-19
+----------------------------------------
|Salario: 2200
+----------------------------------------
|Comision: 120
+----------------------------------------
|N°Departamento: 10
+----------------------------------------
O Comando ejecutado, filas afectadas: 1
```

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
6
+----------------------------------------
|N°Departamento: 110
+----------------------------------------
|NomDepartamento: CES
+----------------------------------------
|Localidad: ALCORCON
+----------------------------------------
O Comando ejecutado, filas afectadas: 1
```

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
7
¿Que empleado desea borrar? (Numero)
55
O Comando ejecutado, filas afectadas: 1
```

```
1- Listar departamentos
2- Listar empleados
3- Buscar empleado por nombre
4- Buscar departamento por nombre
5- Crear empleado
6- Crear departamento
7- Borrar empleado
8- Borrar departamento
9- Salir
8
¿Que departamento desea borrar? (Numero)
110
O Comando ejecutado, filas afectadas: 1
```