CRUD (JAVA and MySQL (DAO))

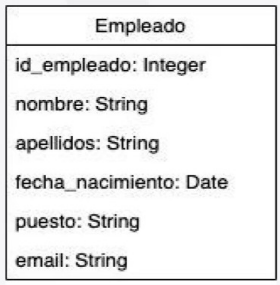
CRUD (JAVA and MySQL (DAO))

ÍNDICE

CRUD (JAVA and MySQL (DAO))

La base de datos se llama empresa y la tabla empleado

```sql
CREATE TABLE empleado (
    id_empleado         MEDIUMINT NOT NULL AUTO_INCREMENT,
    nombre              VARCHAR(100) NOT NULL,
    apellidos           VARCHAR(100) NOT NULL,
    fecha_nacimiento    DATE NOT NULL,
    puesto              VARCHAR(100) NOT NULL,
    email               VARCHAR(320),
    PRIMARY KEY (id_empleado)
);
```



1.0. POOL de CONEXIONES (1ª Conexión) (Frameworks HikariCP)

1.1. Código POOL

```
public class MyDataSource {

   private static HikariConfig config = new  HikariConfig();
   private static HikariDataSource dataSource;

   static {
    config.setJdbcUrl(
        “jdbc:mysql://localhost:3306/empresa?
allowPublicKeyRetrieval=true&useSSL=false&useUnicode=true&serverTi
mezone=Europe/Madrid”);
     config.setUsername(“user”);
     config.setPassword(“password”);
     config.addDataSourceProperty(“maximunPoolSize”,1);
     config.addDataSourceProperty(“cachePrepStmts”,”true”);
     config.addDataSourceProperty(“prepStmtCacheSize”,”250”);
     config.addDataSourceProperty(“prepStmtCacheSqlLimit”,”2048”);
     dataSource = new HikariDataSource(config);
   }
```

CRUD (JAVA and MySQL (DAO))

```
   private MyDataSource() {}

   public static Connection getConnection() throws SQLException {
      return dataSource.getConnection();
   }

}
```

1.2. Ejemplo utilizar POOL(metaDatos)

```
public class App {
 public static void main(String[] args) {
    /*
     * Imprime los nombres de todas las tablas de la conexión
     * realizada en este caso solo empleado.
    */
    try (Connection conn = MyDataSource.getConnection()) {
       DatabaseMetaData metaData = conn.getMetaData();
       String[] types = {“TABLE”};
       ResultSet tables = metaData.getTables(null,null,”%”,types);
       while (tables.next()) {
          System.out.println(tables.getString(“TABLE_NAME”));
       }
    } catch (SQLException e) {
         e.printStackTrace();
    }
 }
}
```

CRUD (JAVA and MySQL (DAO))

2.0. DAO

2.1. Creación del objeto, interfaz y uso en una clase

```
public class Empleado {
    private int id_empleado;
    private String nombre;
    private String apellidos;
    private LocalDate fechaNacimiento;
    private String puesto;
    private String email;

    public Empleado() { }

    public Empleado(String nombre, String apellidos, LocalDate
fechaNacimiento, String puesto, String email) {
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.fechaNacimiento = fechaNacimiento;
        this.puesto = puesto;
        this.email = email;
    }

    public Empleado(int id_empleado, String nombre, String
apellidos, LocalDate fechaNacimiento, String puesto, String email)
{
        this(nombre, apellidos, fechaNacimiento, puesto, email);
        this. id_empleado = id_empleado;
    }

    public int getId_empleado() {
```

```
    return id_empleado;
}

public void setId_empleado(int id_empleado) {
    this.id_empleado = id_empleado;
}

public String getNombre() {
    return nombre;
}

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

public String getApellidos() {
    return apellidos;
}

public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
}

public LocalDate getFechaNacimiento() {
    return fechaNacimiento;
}

public void setFechaNacimiento(LocalDate fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
```

CRUD (JAVA and MySQL (DAO))

```
    }

    public String getPuesto() {
        return puesto;
    }

    public void setPuesto(String puesto) {
        this.puesto = puesto;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Empleado empleado = (Empleado) o;
        return id_empleado == empleado.id_empleado &&
Objects.equals(nombre, empleado.nombre) &&
Objects.equals(apellidos, empleado.apellidos) &&
Objects.equals(fechaNacimiento, empleado.fechaNacimiento) &&
Objects.equals(puesto, empleado.puesto) && Objects.equals(email,
empleado.email);
    }
```

CRUD (JAVA and MySQL (DAO))

```
    @Override
    public int hashCode() {
        return Objects.hash(id_empleado, nombre, apellidos,
fechaNacimiento, puesto, email);
    }

    @Override
    public String toString() {
        return "Empleado{" +
                "id_empleado=" + id_empleado +
                ", nombre='" + nombre + '\'' +
                ", apellidos='" + apellidos + '\'' +
                ", fechaNacimiento=" + fechaNacimiento +
                ", puesto='" + puesto + '\'' +
                ", email='" + email + '\'' +
                '}';
    }
}
```

```
public interface EmpleadoDao {
    int add(Empleado emp) throws SQLException;
    Empleado getById(int id) throws SQLException;
    List<Empleado> getAll() throws SQLException;
    int update(Empleado emp) throws SQLException;
    void delete(int id) throws SQLException;
}
```

CRUD (JAVA and MySQL (DAO))

```
public class EmpleadoDaoImpl implements EmpleadoDao {
    private static EmpleadoDaoImpl instance;
    static {
        instance = new EmpleadoDaoImpl();
    }
    private EmpleadoDaoImpl() {}
    public static EmpleadoDaoImpl getInstance() {
        return instance;
    }

    @Override
    public int add(Empleado emp) throws SQLException {
        String sql = """
                    INSERT INTO empleado (nombre, apellidos, fecha_nacimiento,
puesto, email)
                    VALUES (?, ?, ?, ?, ?);
                """;
        int result;
        try(Connection conn = MyDataSource.getConnection();
            PreparedStatement pstm = conn.prepareStatement(sql)) {
            pstm.setString(1, emp.getNombre());
            pstm.setString(2, emp.getApellidos());
            pstm.setDate(3, Date.valueOf(emp.getFechaNacimiento()));
            pstm.setString(4, emp.getPuesto());
            pstm.setString(5, emp.getEmail());
            result = pstm.executeUpdate();
        }
        return result;
    }

    @Override
    public Empleado getById(int id) throws SQLException {
        Empleado result = null;
        String sql = "SELECT * FROM empleado WHERE id_empleado = ?";
```

CRUD (JAVA and MySQL (DAO))

```
        try(Connection conn = MyDataSource.getConnection();
            PreparedStatement pstm = conn.prepareStatement(sql)) {
            pstm.setInt(1, id);
            try(ResultSet rs = pstm.executeQuery()) {
                while(rs.next()) {
                    result = new Empleado();
                   result.setId_empleado(rs.getInt("id_empleado"));
                    result.setNombre(rs.getString("nombre"));
                    result.setApellidos(rs.getString("apellidos"));
result.setFechaNacimiento(rs.getDate("fecha_nacimiento").toLocalDate());
                    result.setPuesto(rs.getString("puesto"));
                    result.setEmail(rs.getString("email"));
                }
            }
        }
        return result;
    }

    @Override
    public List<Empleado> getAll() throws SQLException {
        String sql = "SELECT * FROM empleado";
        List<Empleado> result = new ArrayList<>();
        try(Connection conn = MyDataSource.getConnection();
            PreparedStatement pstm = conn.prepareStatement(sql);
            ResultSet rs = pstm.executeQuery()) {
            Empleado emp;
            while(rs.next()) {
                emp = new Empleado();
                emp.setId_empleado(rs.getInt("id_empleado"));
                emp.setNombre(rs.getString("nombre"));
                emp.setApellidos(rs.getString("apellidos"));
emp.setFechaNacimiento(rs.getDate("fecha_nacimiento").toLocalDate());
                emp.setPuesto(rs.getString("puesto"));
                emp.setEmail(rs.getString("email"));
```

CRUD (JAVA and MySQL (DAO))

```
            result.add(emp);
        }
    }
    return result;
}

@Override
public int update(Empleado emp) throws SQLException {
    String sql = """
                UPDATE empleado SET
                    nombre = ?, apellidos = ?,
                    fecha_nacimiento = ?,
                    puesto = ?, email = ?
                WHERE id_empleado = ?
            """;
    int result;
    try(Connection conn = MyDataSource.getConnection();
        PreparedStatement pstm = conn.prepareStatement(sql)) {
        pstm.setString(1, emp.getNombre());
        pstm.setString(2, emp.getApellidos());
        pstm.setDate(3, Date.valueOf(emp.getFechaNacimiento()));
        pstm.setString(4, emp.getPuesto());
        pstm.setString(5, emp.getEmail());
        pstm.setInt(6, emp.getId_empleado());
        result = pstm.executeUpdate();
    }
    return result;
}

@Override
public void delete(int id) throws SQLException {
    String sql = "DELETE FROM empleado WHERE id_empleado = ?";
    try (Connection conn = MyDataSource.getConnection();
```

```
            PreparedStatement pstm = conn.prepareStatement(sql)){

            pstm.setInt(1, id);
            pstm.executeUpdate();

        }
    }
}
```

2.2. Método de inserción

```
public int add(Empleado emp) throws SQLException {
   String sql = """
      INSERT INTO empleado (nombre, apellidos, fecha_nacimiento, puesto, email)
      VALUES (?, ?, ?, ?, ?);
                """;
   int result;
   try(Connection conn = MyDataSource.getConnection();
      PreparedStatement pstm = conn.prepareStatement(sql)) {
      pstm.setString(1, emp.getNombre());
      pstm.setString(2, emp.getApellidos());
      pstm.setDate(3, Date.valueOf(emp.getFechaNacimiento()));
      pstm.setString(4, emp.getPuesto());
      pstm.setString(5, emp.getEmail());
      result = pstm.executeUpdate();
   }
   return result;
}
```

EJEMPLO DE COMO UTILIZARLO

EmpleadoDao dao = EmpleadoDaoImpl.getInstance();

Empleado emp = new Empleado(“Adrián”, “Leal Vacas”, LocalDate.of(2004,4,6), “Profesor”, “leal.adrian.vacas@gmail.com”);

```
try {
    int n =  dao.add(emp);
    System.out.println(“El numero de registros insertados es: ” + n);
} catch (SQLException e) {
    e.printStackTrace();
}
```

2.3. Método de consulta (getAll())

```
public List<Empleado> getAll() throws SQLException {
   String sql = "SELECT * FROM empleado";
   List<Empleado> result = new ArrayList<>();
   try(Connection conn = MyDataSource.getConnection();
      PreparedStatement pstm = conn.prepareStatement(sql);
      ResultSet rs = pstm.executeQuery()) {
         Empleado emp;
         while(rs.next()) {
            emp = new Empleado();
            emp.setId_empleado(rs.getInt("id_empleado"));
            emp.setNombre(rs.getString("nombre"));
            emp.setApellidos(rs.getString("apellidos"));
emp.setFechaNacimiento(rs.getDate("fecha_nacimiento").toLocalDate());
            emp.setPuesto(rs.getString("puesto"));
            emp.setEmail(rs.getString("email"));
            result.add(emp);
         }
      }
   return result;
}
```

EJEMPLO DE COMO UTILIZARLO

```
List<Empleado> empleados = dao.getAll();

if (empleados.isEmpty())
    System.out.println(“No hay empleados registrados”);
```

else
empleados.forEach(System.out::println)

2.4. Método de consulta (getById(int id))

```
public Empleado getById(int id) throws SQLException {
   Empleado result = null;
   String sql = "SELECT * FROM empleado WHERE id_empleado = ?";
   try(Connection conn = MyDataSource.getConnection();
      PreparedStatement pstm = conn.prepareStatement(sql)) {
         pstm.setInt(1, id);
         try(ResultSet rs = pstm.executeQuery()) {
            while(rs.next()) {
               result = new Empleado();
               result.setId_empleado(rs.getInt("id_empleado"));
               result.setNombre(rs.getString("nombre"));
               result.setApellidos(rs.getString("apellidos"));
result.setFechaNacimiento(rs.getDate("fecha_nacimiento").toLocalDate());
               result.setPuesto(rs.getString("puesto"));
               result.setEmail(rs.getString("email"));
            }
         }
      }
   return result;
}
```

EJEMPLO DE COMO UTILIZARLO

Empleado emp1 = dao.getById(1);

System.out.println(emp1);

CRUD (JAVA and MySQL (DAO))

2.5. Método de actualización

```
public int update(Empleado emp) throws SQLException {
   String sql = """
                  UPDATE empleado SET
                       nombre = ?,
                       apellidos = ?,
                       fecha_nacimiento = ?,
                       puesto = ?,
                       email = ?
                    WHERE id_empleado = ?
                """;
   int result;
   try(Connection conn = MyDataSource.getConnection();
      PreparedStatement pstm = conn.prepareStatement(sql)) {
         pstm.setString(1, emp.getNombre());
         pstm.setString(2, emp.getApellidos());
         pstm.setDate(3, Date.valueOf(emp.getFechaNacimiento()));
         pstm.setString(4, emp.getPuesto());
         pstm.setString(5, emp.getEmail());
         pstm.setInt(6, emp.getId_empleado());
         result = pstm.executeUpdate();
      }
   return result;
}
```

EJEMPLO DE COMO UTILIZARLO

```
emp1.setFechaNacimiento(LocalDate.of(1992,9,19));

n = dao.update(emp1);

emp1 = dao.getById(1);

System.out.println(emp1);
```

CRUD (JAVA and MySQL (DAO))

2.5. Método de borrado

```
public void delete(int id) throws SQLException {
   String sql = "DELETE FROM empleado WHERE id_empleado = ?";
   try (Connection conn = MyDataSource.getConnection();
      PreparedStatement pstm = conn.prepareStatement(sql)){
         pstm.setInt(1, id);
         pstm.executeUpdate();
      }
}
```

EJEMPLO DE COMO UTILIZARLO

dao.delete(1);

empleados = dao.getAll();

if (empleados.isEmpty())
    System.out.println(“No hay empleados registrados”);
else
    empleados.forEach(System.out::println)

Realizado por: Adrián Leal Vacas