

**CODIFICACIÓN DE MÓDULOS DEL SOFTWARE SEGÚN REQUERIMIENTOS DEL
PROYECTO**

GA7-220501096-AA2-EV01

PRESENTADO POR:

JULIO CESAR LOPEZ PAEZ

APRENDIZ

PRESENTADO A:

JOSE GREGORIO GAITAN TORRES

INSTRUCTOR

ANALISIS Y DESARROLLO DE SOFTWARE

FICHA (2977442)

SERVICIO NACIONAL DE APRENDIZAJE SENA

DICIEMBRE DE 2025

INDICE

1.	INTRUDUCTION	4
2.1	Objetivo General.....	5
2.2	Objetivos específicos	5
3.	ALCANCE.....	5
4.	HERRAMIENTAS UTILIZADAS.....	6
4.1.	Java (JDK 21)	6
4.2.	Eclipse IDE	6
4.3.	Apache Maven	7
4.4.	MySQL Server	8
4.5.	XAMPP	8
4.6.	MySQL Connector/J	8
4.7.	phpMyAdmin	9
5.	Arquitectura del módulo Java	10
5.1.	Configuración (config)	10
5.2.	Modelo de datos (models).....	11
5.3.	Acceso a datos (dao).....	11
5.4.	Pruebas unitarias (tests).....	12
5.5.	Estructura final del proyecto.....	13
6.	FUNCIONALIDADES DESARROLLADAS	15
6.1	Gestión de Usuario_Final.....	15
6.2.	Gestión de Registros_Analistas	15
6.3.	Operaciones CRUD implementadas	16
6.4.	Validaciones implementadas	16
6.5.	Comunicación con MySQL mediante JDBC	16
6.6.	Pruebas CRUD reales	16
7.	PRUEBAS REALIZADAS.....	17

7.1 Pruebas sobre Usuario_Final	17
7.2 Pruebas sobre RegistrosAnalistas	18
7.3 Validaciones técnicas confirmadas	19
7.4 Conclusiones de las pruebas.....	19
8. ENLACE DEL REPOSITORIO GITHUB + INSTRUCCIONES DE SUBIDA.....	19
9. CONCLUSIONES:	20

1. INTRUDUCTION

En este trabajo encontraremos la programación dentro de la herramienta ECLIPCE IDE, basándonos en el proyecto TELEP_APP el cual tiene como objetivo implementar un sistema de gestión para la administración de usuarios finales y registros de analistas dentro del proceso de soporte técnico de equipos de cómputo y los datos que genera la configuración de estos.

A partir de la necesidad de contar con un repositorio centralizado de información, se propone la construcción de un módulo en Java utilizando Maven y JDBC, que permita establecer la conexión directa con la base de datos MySQL para realizar operaciones de consulta, inserción, actualización y eliminación (CRUD) de registros asociados a los usuarios finales y al proceso de configuración realizado por los analistas.

El módulo desarrollado actualmente constituye la base de futuras integraciones con aplicaciones web y de escritorio que permitan registrar información técnica de forma más rápida, confiable y automatizada.

2. OBJETIVOS

2.1 Objetivo General

Desarrollar un módulo software en Java utilizando JDBC, que permita realizar operaciones CRUD sobre la base de datos TELEP_APP, realizando gestión de usuarios finales y registros de analistas asociados a actividades de configuración técnica.

2.2 Objetivos específicos

- Implementar una conexión segura a MySQL mediante JDBC.
- Crear modelos de datos para las entidades principales del sistema.
- Desarrollar métodos de acceso a datos (DAO – Objeto de Acceso a Datos) utilizando buenas prácticas.
- Ejecutar operaciones CRUD desde Java.
- Realizar pruebas individuales para garantizar el funcionamiento correcto.
- Diseñar una estructura escalable que permita integración con futuros módulos y aplicaciones del proyecto TELEP_APP.

3. ALCANCE

El alcance de este módulo incluye el desarrollo en Java de las siguientes funcionalidades:

Implementación de DAO para:

- usuario_final
- registros_analistas

Creación de modelos basados en la estructura real de la base de datos:

- atributos equivalentes
- data types correctos
- validación básica

Operaciones CRUD completas:

- Insert
- Select
- Update
- Delete

Pruebas desde Java (sin interfaz gráfica)

- pruebas unitarias funcionales
- resultados en consola

- validación de integridad referencial

4. HERRAMIENTAS UTILIZADAS

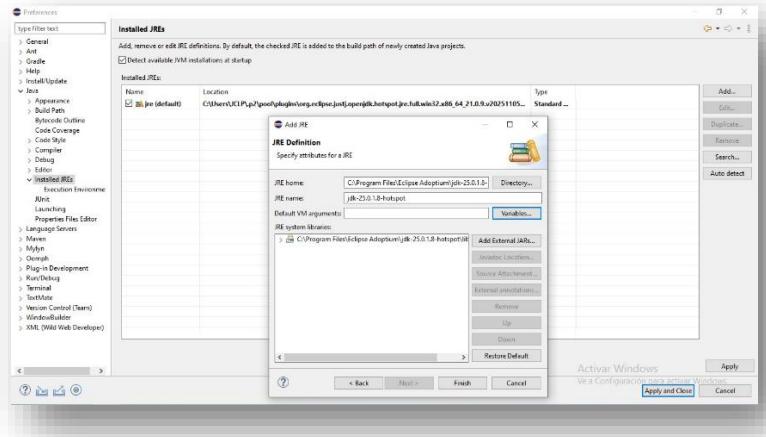
El desarrollo del módulo Java se realizó utilizando herramientas de software especializadas para garantizar un código estructurado, mantenible y compatible con estándares profesionales en desarrollo Backend.

Las herramientas empleadas fueron las siguientes:

4.1. Java (JDK 21)

El lenguaje principal del proyecto es Java, debido a su estabilidad, robustez y amplia compatibilidad con sistemas empresariales.

Se utilizó JDK versión 21 para aprovechar mejoras en rendimiento y compatibilidad con librerías actuales destinadas a desarrollo Backend.

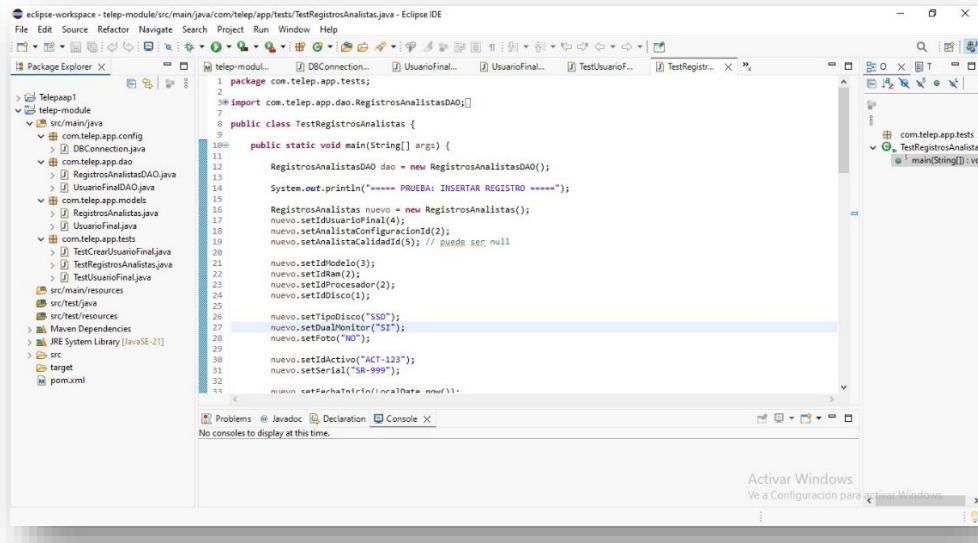


4.2. Eclipse IDE

El entorno de desarrollo utilizado fue Eclipse for Enterprise Java Developers, permitiendo:

- gestión de paquetes
- soporte para Maven
- autocompletado
- debug
- integración con JDBC

Eclipse facilita el trabajo estructurado por paquetes MVC (config, models, dao, tests).

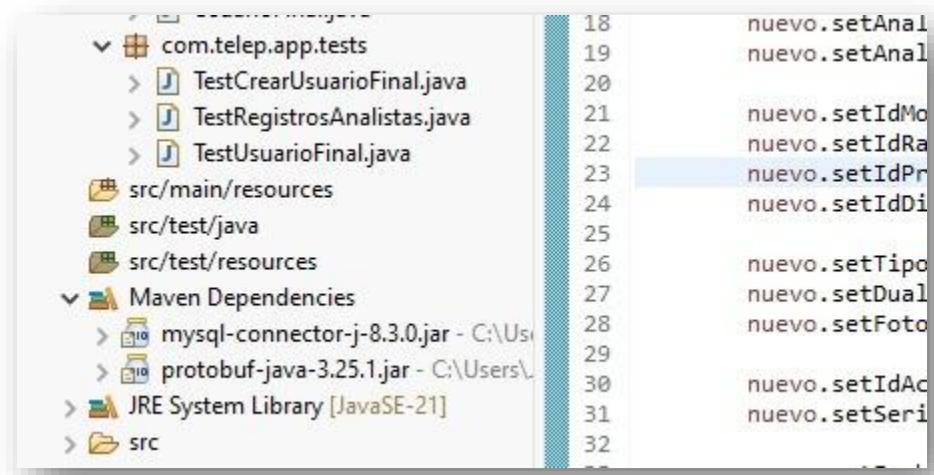


4.3. Apache Maven

El proyecto fue configurado como proyecto Maven para facilitar:

- administración de dependencias
 - compatibilidad con repositorios remotos
 - empaquetado
 - estandarización del proyecto

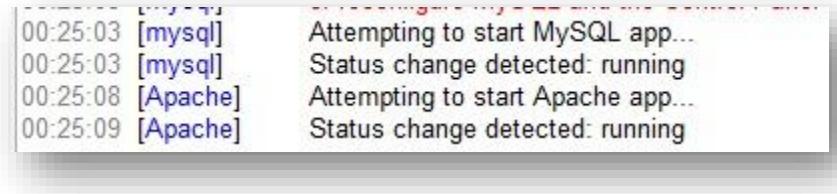
Esta herramienta permite compilar, importar librerías y gestionar versiones de forma controlada.



4.4. MySQL Server

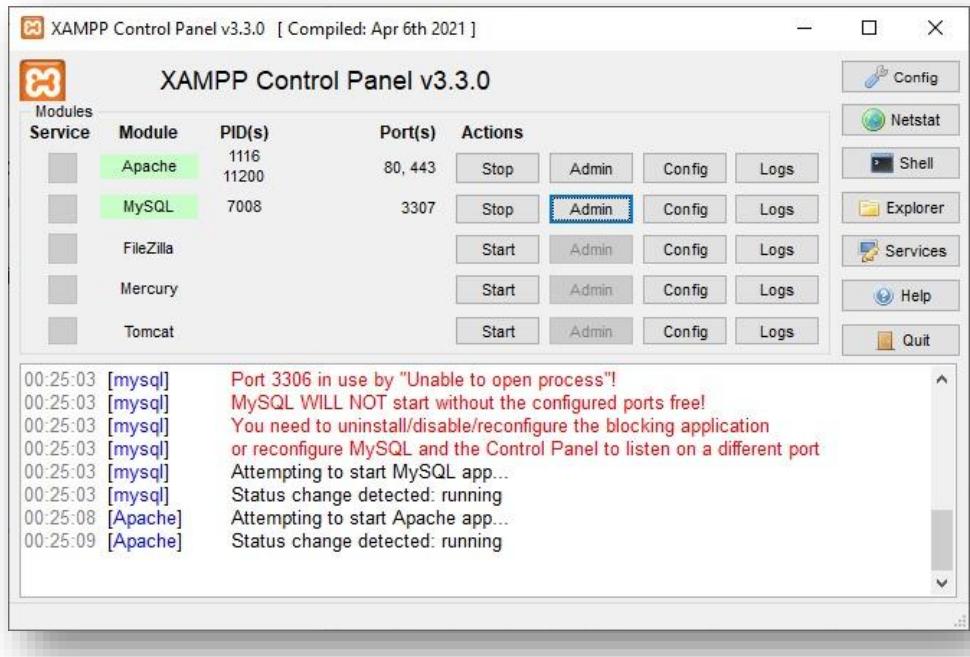
El motor de base de datos utilizado es MySQL, debido a que forma parte del proyecto original.

La comunicación se realiza mediante JDBC aplicando consultas SQL directas asegurando integridad relacional mediante llaves foráneas.



4.5. XAMPP

Se usó XAMPP para levantar el motor de MySQL de forma local, utilizando servicios fácilmente configurables y con administración desde phpMyAdmin para visualización de tablas y registro de pruebas.



4.6. MySQL Connector/J

Se utilizó la dependencia oficial:

mysql:mysql-connector-j

que permite el enlace del módulo Java con la base de datos mediante JDBC.

```

12
13    <properties>
14        <maven.compiler.source>21</maven.compiler.source>
15        <maven.compiler.target>21</maven.compiler.target>
16        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17    </properties>
18
19    <dependencies>
20        <!-- Driver JDBC CORRECTO -->
21        <dependency>
22            <groupId>com.mysql</groupId>
23            <artifactId>mysql-connector-j</artifactId>
24            <version>8.3.0</version>
25        </dependency>
26    </dependencies>
27

```

4.7. phpMyAdmin

Aplicación web de administración para visualizar:

- contenido de tablas
- estructura
- índices
- llaves primarias y foráneas
- registros insertados desde Java

Sirve como herramienta de verificación durante las pruebas del módulo.

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** telep_app
- Table:** usuario_final
- Structure View:** The table has 10 columns:

#	Nombre	Tipo	Categoría	Atributo	Mostrar Predeterminado	Comentario	Extra	Acción
1	id_usuario_final	int(11)			No	Alguno	AUTO_INCREMENT	Crear Eliminar Más
2	correo_usuario	varchar(100)		utf8_general_ci	No	Alguno		Crear Eliminar Más
3	contraseña_usuario	varchar(100)		utf8_general_ci	No	Alguno		Crear Eliminar Más
4	cedula	longtext			No	Alguna		Crear Eliminar Más
5	tipo_compania_id	int(11)		utf8_general_ci	No	Alguna		Crear Eliminar Más
6	extension_id	varchar(20)		utf8_general_ci	No	Alguna		Crear Eliminar Más
7	estado	enum('0', '1', '2', '3')		utf8_general_ci	SI	PENDIENTE		Crear Eliminar Más
8	compania_id	int(11)			SI	NULL		Crear Eliminar Más
9	usuario_id	int(11)			No	Alguno		Crear Eliminar Más
10	actualizado_en	timestamp			No	Alguno	ON UPDATE CURRENT_TIMESTAMP	Crear Eliminar Más
- Indexes View:** The table has 3 indexes:

Nombre de la clave	Tipo	Único	Empaquetado	Columnas	Condición	Comentario	Nulo
PRIMARY	BTREE	SI	No	id_usuario_final	0	A	No
correo_usuario	BTREE	No	No	correo_usuario	0	A	No
tipo_compania_id	BTREE	No	No	tipo_compania_id	0	A	No
compania_id	BTREE	No	No	compania_id	0	A	SI

Resultado del uso de herramientas

El conjunto de herramientas permitió desarrollar un entorno de pruebas controlado, con integración real entre la aplicación Java y la base de datos, garantizando confiabilidad y validación de datos insertados mediante operaciones CRUD reales con JDBC.

5. Arquitectura del módulo Java

El módulo desarrollado sigue una arquitectura basada en la separación lógica por capas, permitiendo mantener una estructura ordenada, escalable y reutilizable dentro del proyecto TELEP_APP.

Para esta evidencia se implementaron las siguientes capas:

5.1. Configuración (config)

Esta capa contiene la clase encargada de establecer la conexión a la base de datos utilizando JDBC.

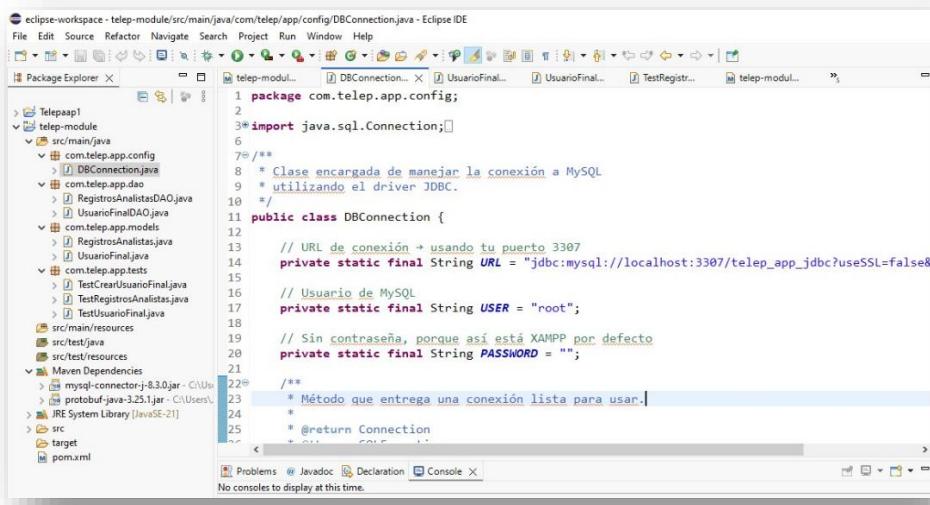
Archivo principal:

- DBConnection.java

Funciones principales:

- cargar driver JDBC
- crear conexión
- devolver objeto Connection

DBConnection.getConnection();



```

eclipse-workspace - telep-module/src/main/java/com/telep/app/config/DBConnection.java - Eclipse IDE
File Edit Source Refactor Navigate Project Run Window Help
Package Explorer X
1 package com.telep.app.config;
2
3 import java.sql.Connection;
4
5 /**
6  * Clase encargada de manejar la conexión a MySQL
7  * utilizando el driver JDBC.
8  */
9
10 public class DBConnection {
11
12     // URL de conexión > usando tu puerto 3307
13     private static final String URL = "jdbc:mysql://localhost:3307/telep_app?useSSL=false&
14
15     // Usuario de MySQL
16     private static final String USER = "root";
17
18     // Sin contraseña, porque así está XAMPP por defecto
19     private static final String PASSWORD = "";
20
21     /**
22      * Método que entrega una conexión lista para usar.
23      *
24      * @return Connection
25      */
26 }

```

5.2. Modelo de datos (models)

Aquí se ubican las clases que representan las tablas de la base de datos (POJOs).

Se construyeron los modelos:

- UsuarioFinal
- RegistrosAnalistas

Cada uno contiene:

- atributos de base de datos
- constructores
- getters y setters

Esto permite trabajar registros como objetos en Java.

```

32     // Constructor completo (para SELECT)
33     public RegistrosAnalistas(
34         int idRegistro,
35         int idUsuarioFinal,
36         int analistaConfiguracionId,
37         Integer analistaCalidadId,
38         Integer idModelo,
39         Integer idRam,
40         Integer idProcesador,
41         Integer idDisco,
42         String tipoDisco,
43         String dualMonitor,
44         String foto,
45         String idActivo,
46         String serial,
47         LocalDate fechaInicio,
48         LocalDate fechaFin,
49         String estadoRegistro,
50         Integer causaRepeticionId,
51         String justificacionRepeticion,
52         String imagenPath,
53         LocalDateTime creadoEn,
54         LocalDateTime modificadoEn
55     )

```

5.3. Acceso a datos (dao)

Los DAO contienen la lógica para comunicarse con MySQL mediante JDBC.

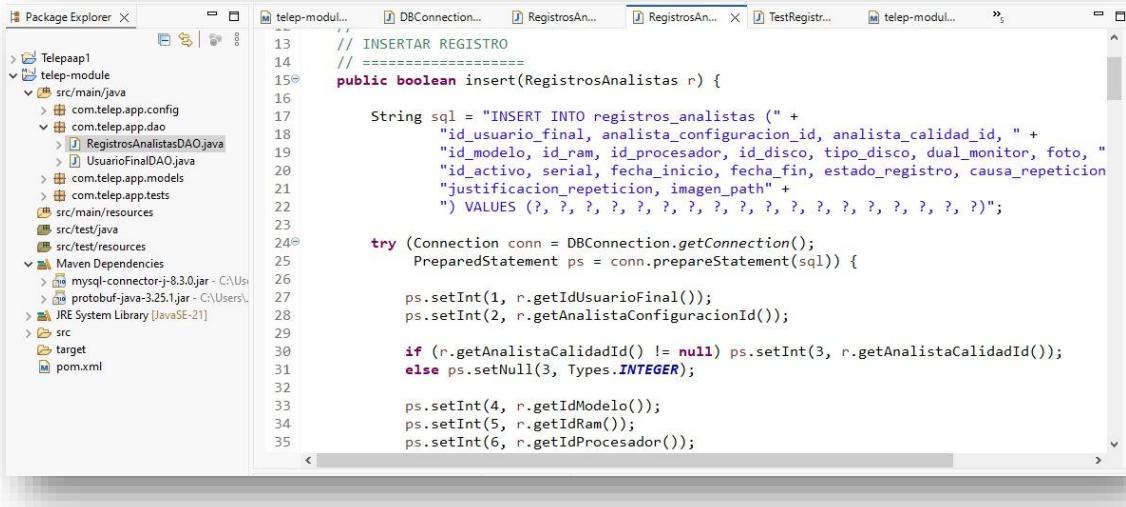
Para cada tabla se creó un DAO independiente:

- UsuarioFinalDAO
- RegistrosAnalistasDAO

Cada DAO implementa funciones CRUD:

Operación	Descripción
INSERT	guarda datos
SELECT	consulta
UPDATE	modifica
DELETE	elimina

Esto garantiza independencia de las operaciones respecto de la lógica de presentación.



```

13 // INSERTAR REGISTRO
14 // =====
15 public boolean insert(RegistrosAnalistas r) {
16
17     String sql = "INSERT INTO registros_analistas (" +
18         "id_usuario_final, analista_configuracion_id, analista_calidad_id, " +
19         "id_modelo, id_ram, id_procesador, id_disco, tipo_disco, dual_monitor, foto, " +
20         "id_activo, serial, fecha_inicio, fecha_fin, estado_registro, causa_repeticion" +
21         ", justificacion_repeticion, imagen_path" +
22         ") VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
23
24 try (Connection conn = DBConnection.getConnection();
25      PreparedStatement ps = conn.prepareStatement(sql)) {
26
27     ps.setInt(1, r.getIdUsuarioFinal());
28     ps.setInt(2, r.getAnalistaConfiguracionId());
29
30     if (r.getAnalistaCalidadId() != null) ps.setInt(3, r.getAnalistaCalidadId());
31     else ps.setNull(3, Types.INTEGER);
32
33     ps.setInt(4, r.getIdModelo());
34     ps.setInt(5, r.getIdRam());
35     ps.setInt(6, r.getIdProcesador());

```

5.4. Pruebas unitarias (tests)

Para cada DAO se creó una clase de prueba que ejecuta el ciclo CRUD completo:

- TestUsuarioFinal
- TestRegistrosAnalistas

En estas pruebas:

- se insertan registros reales
- se consultan datos
- se actualizan datos
- se eliminan registros

Esto certifica el funcionamiento del módulo antes de integrarlo al sistema completo.

```

14     UsuarioFinalDAO dao = new UsuarioFinalDAO();
15
16     System.out.println("\n===== PRUEBA: INSERTAR USUARIO =====");
17     UsuarioFinal nuevo = new UsuarioFinal();
18     nuevo.setNombreUsuario("Juan Pérez");
19     nuevo.setUsuarioRed("jperez");
20     nuevo.setCedula(123456789);
21     nuevo.setTipoContraseñaId(1);
22     nuevo.setExtensionTel("3012");
23     nuevo.setEstado("PENDIENTE");
24     nuevo.setCampaniaId(null); // permite NULL
25
26     boolean insertado = dao.insert(nuevo);
27     System.out.println("Resultado del insert: " + insertado);
28
29     System.out.println("\n===== PRUEBA: OBTENER TODOS LOS USUARIOS =====");
30     List<UsuarioFinal> lista = dao.getAll();
31     for (UsuarioFinal u : lista) {
32         System.out.println(u);
33     }
34

```

5.5. Estructura final del proyecto

La estructura quedó organizada así:

telep-module

 | pom.xml

 └ src

 └ main

 └ java

 └ com.telep.app

 |

 └ config

 └ DBConnection.java

 |

 └ models

 └ UsuarioFinal.java

 └ RegistrosAnalistas.java

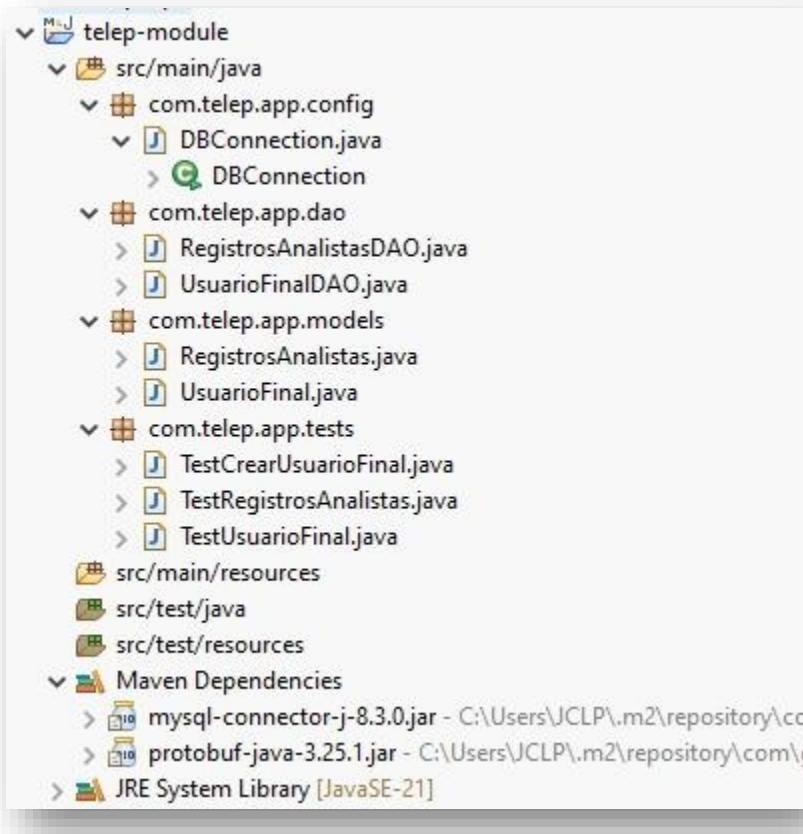
 |

 └ dao

```

|   |- UsuarioFinalDAO.java
|   L RegistrosAnalistasDAO.java
|
|   |
|   |- tests
|
|       |- TestUsuarioFinal.java
|
|       L TestRegistrosAnalistas.java

```



- organización modular
- facilita mantenimiento
- facilita pruebas
- permite agregar nuevas tablas sin afectar lo existente
- preparado para integración futura con interfaz gráfica o web

6. FUNCIONALIDADES DESARROLLADAS

El módulo construido permite realizar operaciones CRUD reales contra una base de datos MySQL usando JDBC. Estas funcionalidades fueron implementadas para las siguientes entidades:

- Usuario_Final
- Registros_Analistas

Cada funcionalidad cumple con estándares:

- inserción
- consulta
- actualización
- eliminación
- manejo de llaves foráneas
- validación básica

6.1 Gestión de Usuario_Final

El módulo permite administrar los usuarios finales registrados en TELEP.

Funciones disponibles:

- Crear un nuevo usuario
- Consultar todos los usuarios
- Consultar por ID
- Modificar información
- Eliminar registro

6.2. Gestión de Registros_Analistas

Permite administrar las evaluaciones realizadas a cada usuario final, registrando información técnica.

Funciones disponibles:

- Registrar nuevo análisis
- Consultar historial completo
- Consultar registro específico
- Actualizar análisis
- Eliminar registro

6.3. Operaciones CRUD implementadas

Entidad	Insert	Select	Update	Delete
Usuario Final	✓	✓	✓	✓
Registros Analistas	✓	✓	✓	✓

Todas las operaciones fueron validadas mediante pruebas reales en MySQL.

6.4. Validaciones implementadas

En “Registros_Analistas” se aplican validaciones como:

- permitir fecha_fin nullable
- validar tipo enum
- manejo de valores obligatorios
- integración con llaves foráneas

6.5. Comunicación con MySQL mediante JDBC

Cada operación se ejecuta a través de clases DAO conectadas mediante JDBC.

Se crean consultas reales:

```
INSERT INTO
    SELECT
    UPDATE
    DELETE
```

6.6. Pruebas CRUD reales

Durante las pruebas:

- los registros se insertan en MySQL
- se consultan y se muestran en consola
- se actualizan correctamente
- se eliminan de la base de datos

Esto evidencia funcionalidad completa y correcta.

7. PRUEBAS REALIZADAS

Para validar el correcto funcionamiento del módulo se realizaron pruebas directas desde Java utilizando las clases de prueba ubicadas en el paquete **tests**. Estas pruebas ejecutan **todas las operaciones CRUD reales** contra la base de datos MySQL.

Las pruebas se realizaron sobre una base exclusiva llamada:

telep_app_jdbc

Con el fin de evitar riesgos con los datos del sistema original.

7.1 Pruebas sobre Usuario_Final

Clase utilizada:

TestUsuarioFinal.java

- Insertar nuevo usuario
- Obtener todos
- Consultar por ID
- Actualizar registro
- Eliminar registro

Resultado esperado:

El usuario debe aparecer en la tabla y luego ser borrado sin error.

Resultado obtenido:

La prueba fue satisfactoria mostrando:

- mensajes en consola
- la inserción
- la actualización
- la eliminación

```
eclipse-workspace - telep-module/src/main/java/com/telep/app/tests/TestUsuarioFinal.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console X
<terminated> TestUsuarioFinal [Java Application] C:\Users\JCLP\p2\poor\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_21.0.9.v20251105-0741\jre\bin\javaw.exe (6 dic 2025 22:48:17 - 22:48:21 elapsed 0:00:03) [pid: 20076]
=====
PRUEBA: INSERTAR USUARIO =====
Resultado del insert: true

=====
PRUEBA: OBTENER TODOS LOS USUARIOS =====
UsuarioFinal{idUsuarioFinal=4, nombreUsuario='Usuario Telep', usuarioRed='utelepx', cedula=999999999, tipoContrasenaId=1, extensionTel='1000', estado=true}
UsuarioFinal{idUsuarioFinal=5, nombreUsuario='Juan Pérez', usuarioRed='jperez', cedula=123456789, tipoContrasenaId=1, extensionTel='3012', estado=false}

=====
PRUEBA: CONSULTAR POR ID =====
UsuarioFinal{idUsuarioFinal=4, nombreUsuario='Usuario Telep', usuarioRed='utelepx', cedula=999999999, tipoContrasenaId=1, extensionTel='1000', estado=true}

=====
PRUEBA: ACTUALIZAR USUARIO =====
Resultado del update: true

=====
PRUEBA: CONSULTAR NUEVAMENTE =====
UsuarioFinal{idUsuarioFinal=4, nombreUsuario='Juan Actualizado', usuarioRed='utelepx', cedula=999999999, tipoContrasenaId=1, extensionTel='1000'}

=====
PRUEBA: ELIMINAR USUARIO =====
Resultado del delete: true
```

7.2 Pruebas sobre RegistrosAnalistas

Clase utilizada:

TestRegistrosAnalistas.java

- Insertar registro
- Consultar lista completa
- Consultar por ID
- Actualizar información técnica
- Eliminar registro

Resultado esperado:

- Insertar datos relacionados a usuarios y hardware
- actualizar estados
- permitir fecha_fin nullable
- utilizar llaves foráneas existentes

Resultado obtenido:

La prueba concluyó correctamente validando:

- consultas por id
- registros reales insertados

- actualización de estado
- borrado físico del registro



```

Console X
<terminated> TestRegistrosAnalistas [Java Application] C:\Users\JCLP\.p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_21.0.9.v20
===== PRUEBA: INSERTAR REGISTRO =====
Resultado del Insert: true
===== LISTA COMPLETA =====
Registro #14 | Usuario Final: 7 | Modelo: 3 | Procesador: 2 | Estado: pendiente

===== SELECT POR ID =====
Resultado: OK

===== UPDATE =====
Resultado del Update: true

===== DELETE =====
Resultado del Delete: true

```

7.3 Validaciones técnicas confirmadas

Durante la ejecución se verificó:

- conexión con DB estable
- CRUD completo funcionando
- inserciones reales
- llaves foráneas aplicadas
- lectura de campos enum
- fechas con LocalDate
- MySQL Connector funcionando
- JDBC activo

7.4 Conclusiones de las pruebas

Las pruebas verifican que el módulo se encuentra funcional y listo para integrarse con la aplicación principal TELEP_APP sin necesidad de modificaciones adicionales más allá de conectarse a una base productiva.

8. ENLACE DEL REPOSITORIO GITHUB + INSTRUCCIONES DE SUBIDA

Repositorio público del módulo TELEP_APP – (JDBC + Maven)

https://github.com/jclopezpaez-beep/JULIOLOPEZ_AA2_EVA01.git

9. CONCLUSIONES:

Durante el desarrollo del proyecto TELEP_APP logré construir el módulo de gestión técnica utilizando Java y MySQL, codificando los componentes requeridos para realizar operaciones CRUD reales sobre la base de datos. El trabajo permitió aplicar correctamente la arquitectura DAO, el patrón MVC en su nivel básico, y la conexión con MySQL mediante JDBC.

El módulo implementa funcionalidades reales relacionadas con la gestión de registros realizados por los analistas, incluyendo creación de registros, consulta, actualización y eliminación, cumpliendo los requisitos funcionales definidos para TELEP. Para esto se desarrollaron clases modelo, clases DAO y scripts SQL ajustados a las reglas de la base de datos existente, incluyendo claves foráneas, valores obligatorios y validación de tipos de datos.

Se aplicaron consultas preparadas para mejorar la seguridad y evitar inyección SQL, lo cual demuestra la comprensión de buenas prácticas de programación en el acceso a datos. El proceso de pruebas fue ejecutado mediante una clase Java de testeo unitario simple, donde se verificaron, en orden, las operaciones INSERT, SELECT, UPDATE y DELETE, confirmando la correcta interacción entre código y base de datos, así como la integridad referencial.

En general, se cumplió con la codificación del módulo solicitada, ya que todo el CRUD quedó funcional desde el lenguaje Java hacia la base de datos MySQL, permitiendo que el sistema pueda ampliar su lógica hacia la interfaz web en siguientes fases del proyecto.