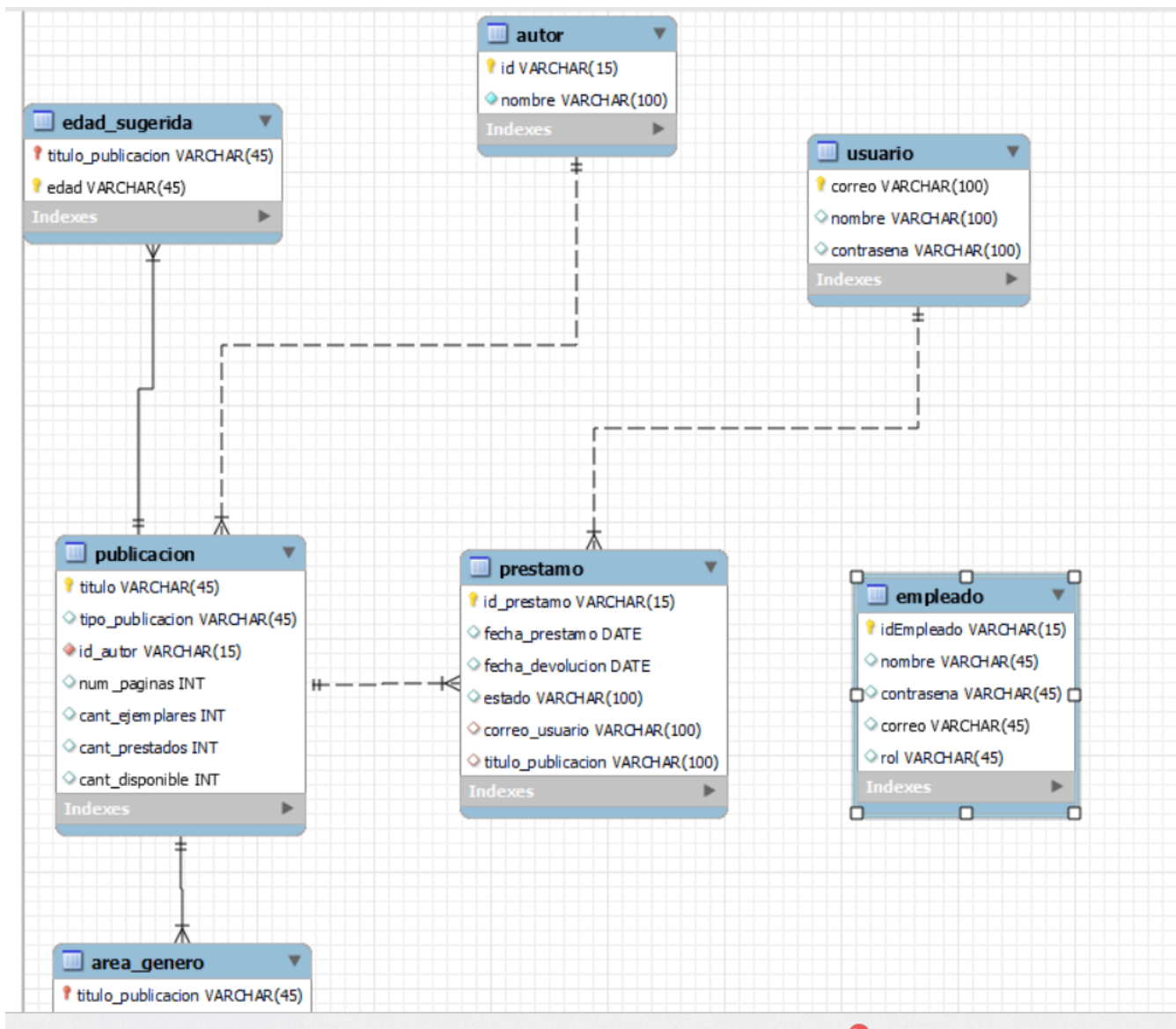
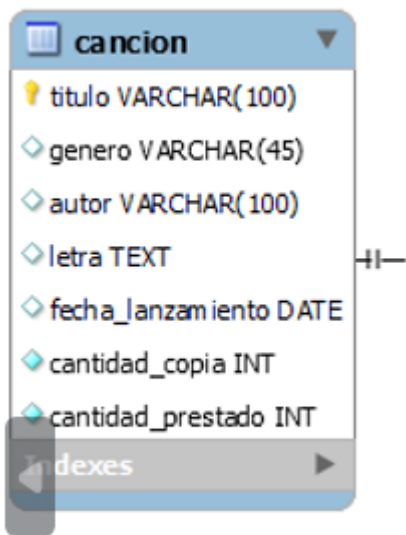


Adiciones

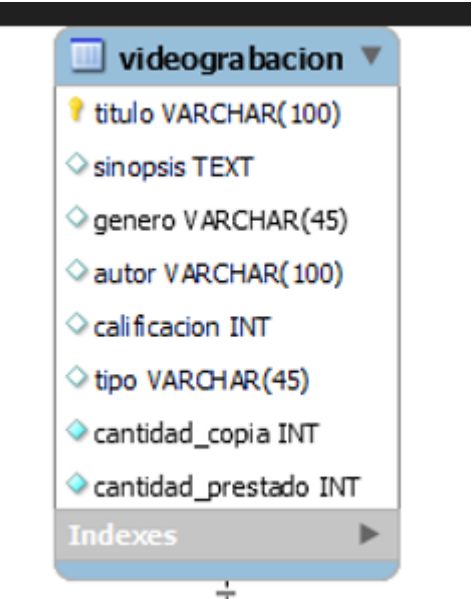
Base de datos inicial



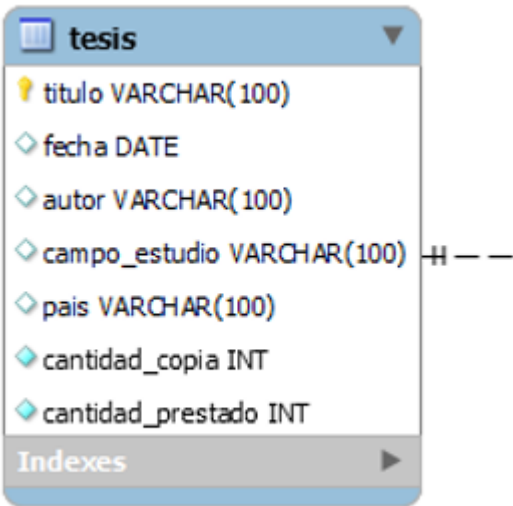
- Se crean 3 tablas para el manejo de las nuevas funcionalidades, llamadas:
  - Cancion



- Videograbacion

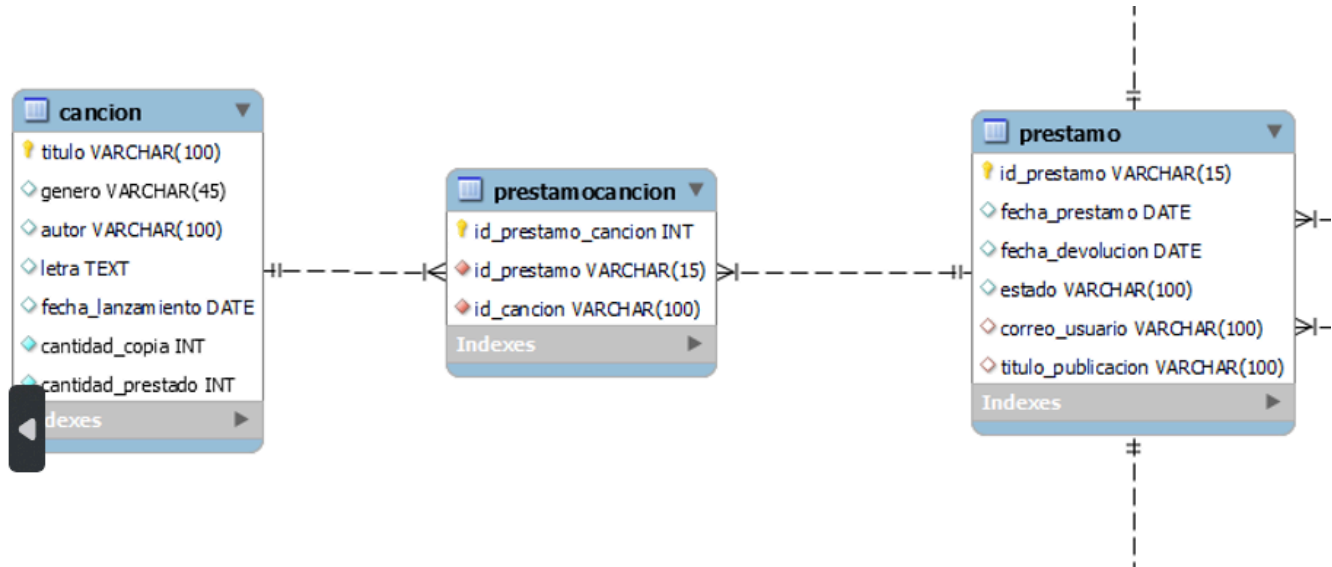


- Tesis

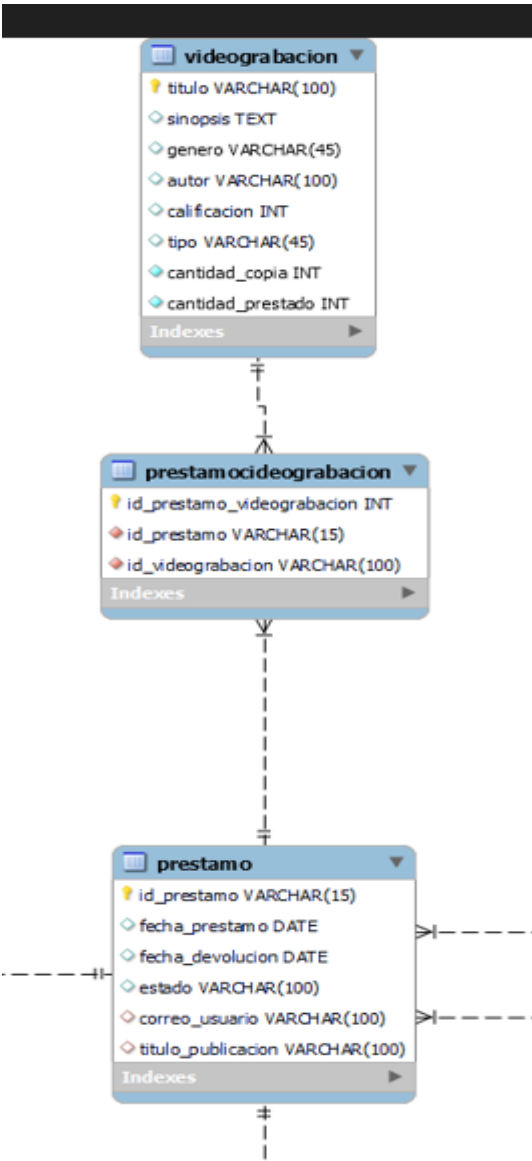


- Para que no hayan campos nulos en los préstamos, se crearon las tablas

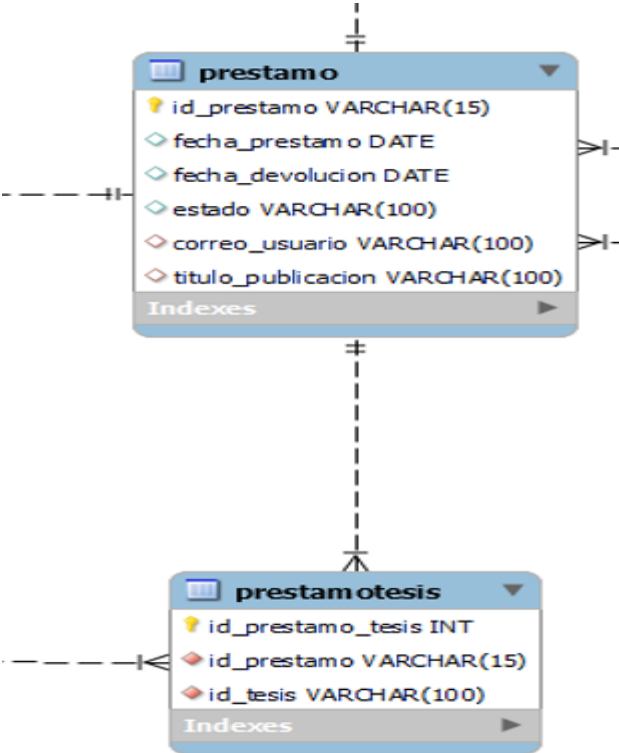
- PrestamoCancion: Relaciona el prestamo con la cancion



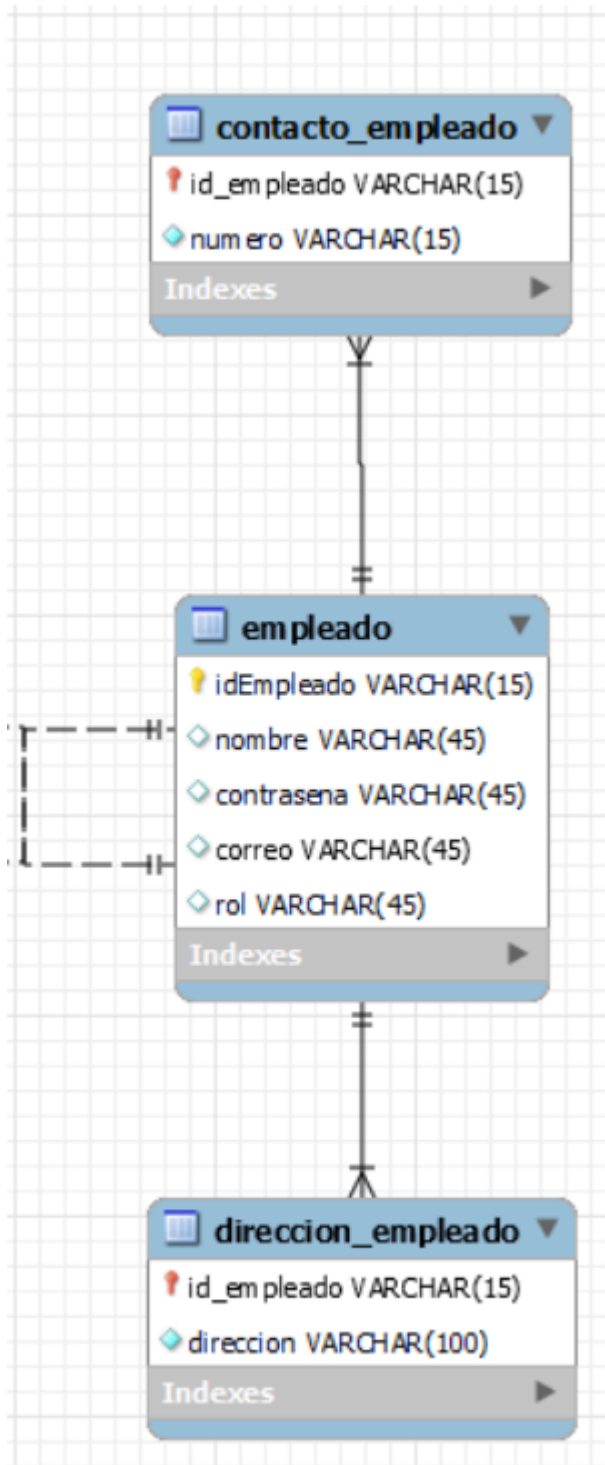
- PrestamoGrabacion: Relaciona el prestamo con la videograbacion



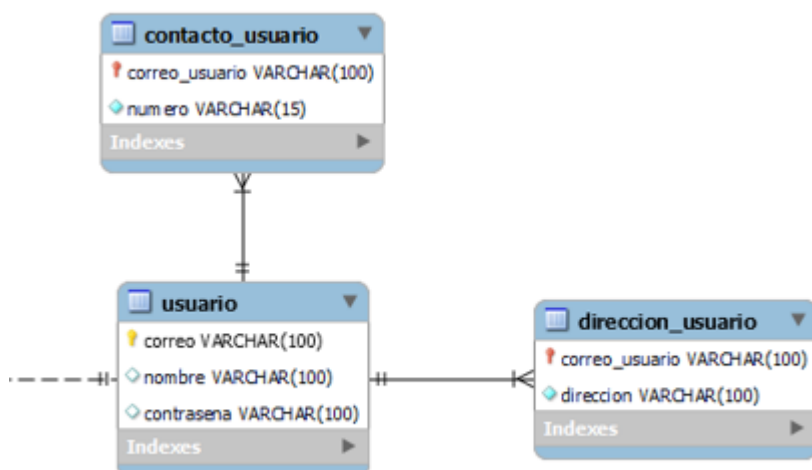
- PrestamoTesis: Relaciona el prestamo con la tesis



- Al adicionar dos campos modificables al usuario y al empleado y que estos no sean nulos surgen las tablas
  - contacto\_empleado
  - direccion\_empleado

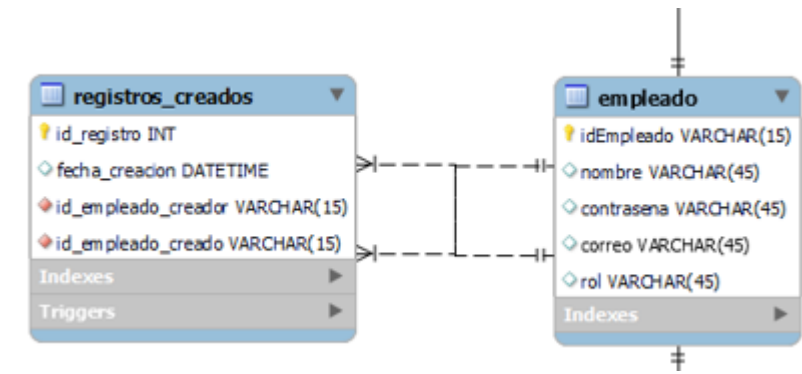


- contacto\_usuario
- direccion\_usuario



Estas tablas relacionan al tipo de usuario del sistema con su respectivo dato.

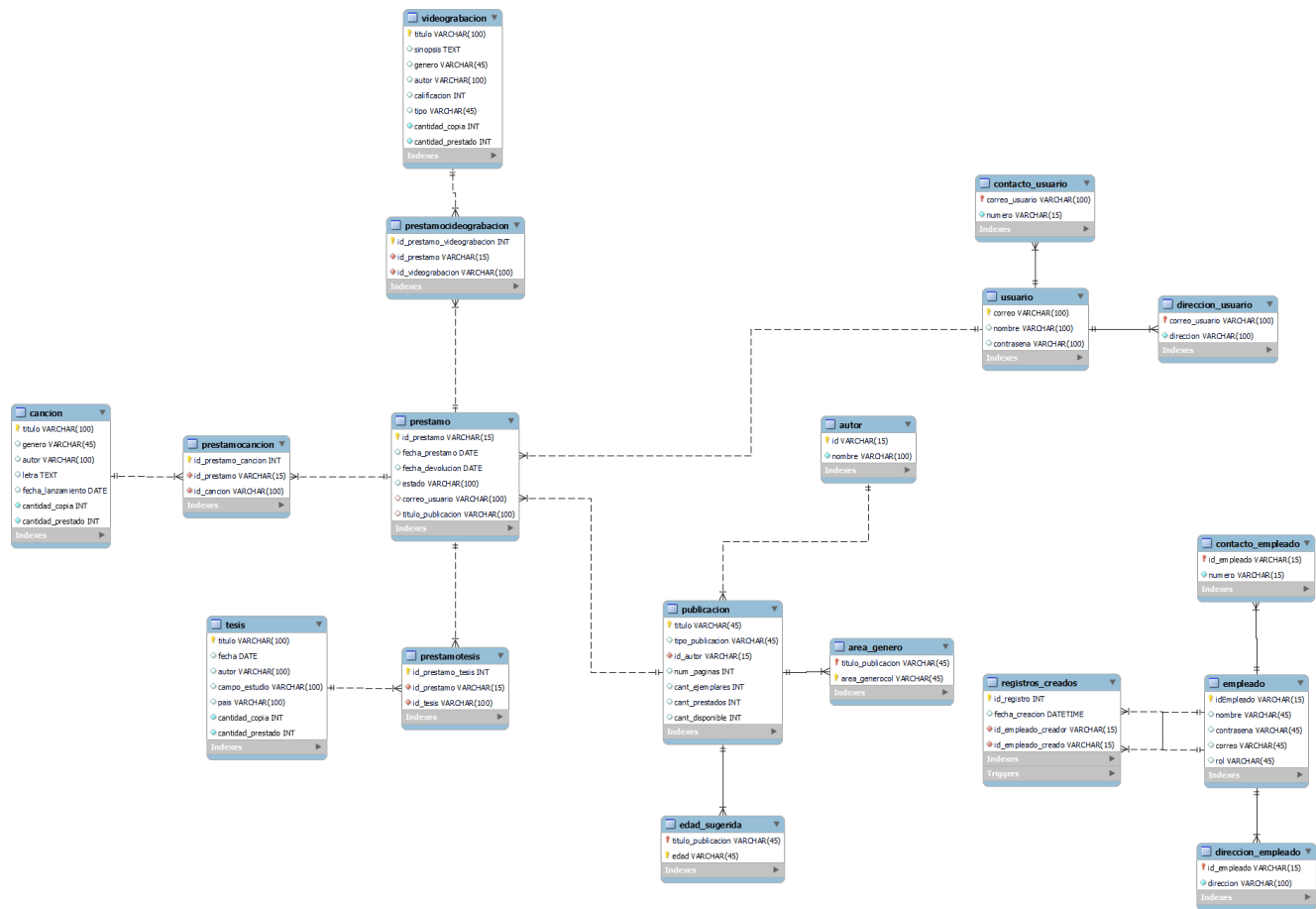
- Se crea tabla Registros\_creados para registro de la creación de usuarios



- Se crea un trigger para cada vez que se ingrese un registros guarde en el campo del tipo DateTime fecha\_creacion la hora y fecha en la que ingresa el registro

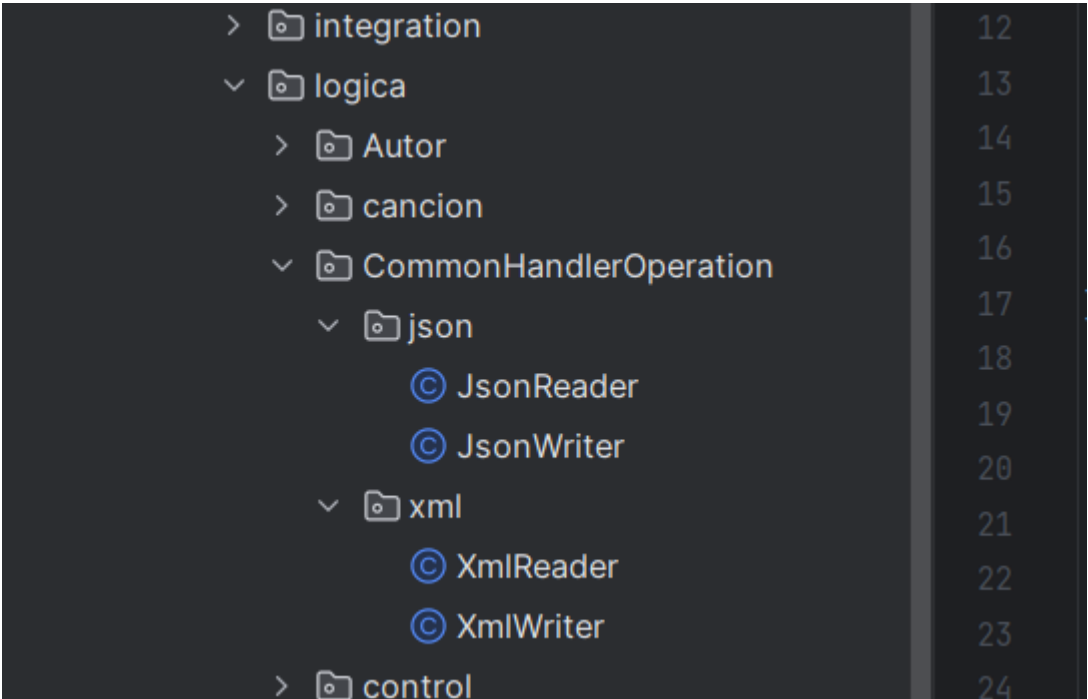
```
DELIMITER //
CREATE TRIGGER actualizar_fecha_creacion
BEFORE INSERT ON registros_creados
FOR EACH ROW
BEGIN
    SET NEW.fecha_creacion = NOW();
END;
//
DELIMITER ;
```

Base de datos actualizada nuevas funcionalidades



Lectura y escritura archivos XML y Json

- Se crearon 4 clases genéricas



en la carpeta lógica, CommonHandlerOperation encarga de las operaciones de importar desde un Json o un XML y exportar a un Json o un XML

- XmlReader y JsonReader

Estas reciben cualquier tipo de archivo XML o Json y retornan un arraylist del tipo de dato solicitado.

```
11
12 4 usages Daniel Chaparro Cano
13 public class JsonReader<T> {
14
15     // Leer archivo json y retornar un arraylist
16     1 usage Daniel Chaparro Cano
17     public ArrayList<T> leerJsonArrayArchivo(String rutaArchivo, Class<T> valueType) {
18         // Instanciar objeto ObjectMapper que permite parsear de json al tipo del objeto
19         ObjectMapper objectMapper = new ObjectMapper();
20         // Crear arraylist genérico
21         ArrayList<T> lista = new ArrayList<>();
22         try {
23             // Obtener el objeto File del archivo utilizando la ruta
24             File archivoJson = Paths.get(rutaArchivo).toFile();
25             //Indica el tipo de la lista
26             CollectionType listType = objectMapper.getTypeFactory().constructCollectionType(ArrayList.class, valueType);
27             // Se utiliza el object mapper para leer el archivo json y devolver una lista
28             // del tipo de objeto especificado
29             lista = objectMapper.readValue(archivoJson, listType);
30         } catch (IOException e) {
31             System.out.println("Error => " + e);
32         }
33     }
34 }
```

```
6 import java.io.IOException;
7 import java.util.ArrayList;
8
9 3 usages Daniel Chaparro Cano
10 public class XmlReader<T> {
11
12     // Leer archivo XML y retornar un ArrayList
13     1 usage Daniel Chaparro Cano
14     public ArrayList<T> leerXmlArrayArchivo(String rutaArchivo, Class<T> valueType) {
15         // Instanciar objeto XmlMapper que permite parsear de XML al tipo del objeto
16         XmlMapper xmlMapper = new XmlMapper();
17         // Crear ArrayList genérico
18         ArrayList<T> lista = new ArrayList<>();
19         try {
20             // Obtener el objeto File del archivo utilizando la ruta
21             File archivoXml = new File(rutaArchivo);
22             // Leer el archivo XML y convertirlo en un ArrayList del tipo especificado
23             lista = xmlMapper.readValue(archivoXml, xmlMapper.getTypeFactory().constructCollectionType(ArrayList.class, valueType));
24         } catch (IOException e) {
25             System.out.println("Error al leer el archivo XML: " + e.getMessage());
26         }
27     }
28 }
```

- XmlWriter y JsonWriter

Estas clases reciben un arrayList de cualquier tipo y lo exportan a su correspondiente archivo Json o Xml

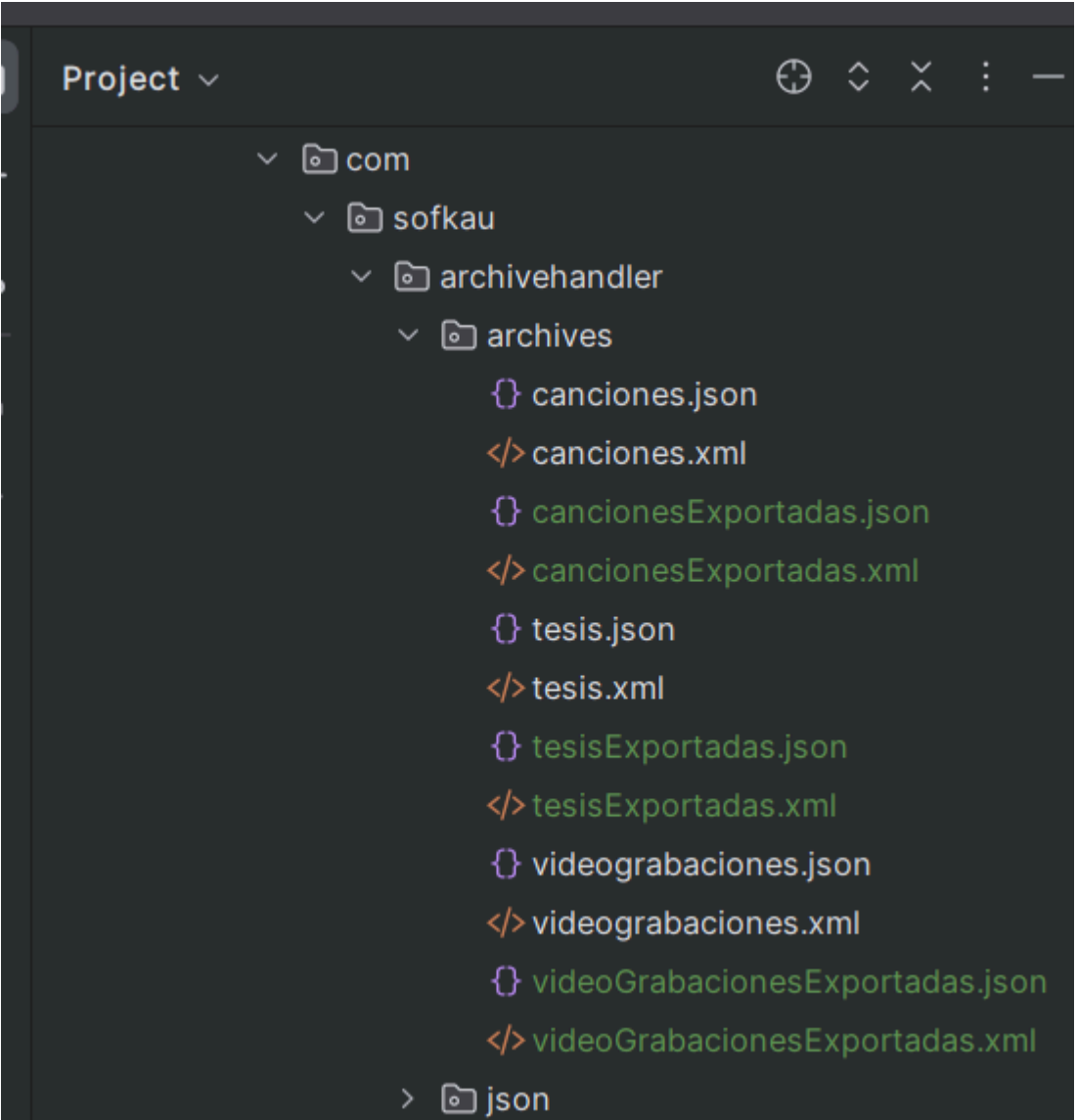
```
3 usages  Daniel Chaparro Cano
10 public class JsonWriter<T> {
11
12     // Método para escribir un ArrayList en un archivo JSON
13     1 usage  Daniel Chaparro Cano
14     public void exportarArrayListJson(ArrayList<T> arrayList, String filePath) {
15         ObjectMapper objectMapper = new ObjectMapper();
16         // Configurar ObjectMapper para que genere el JSON con formato legible
17         objectMapper.enable(SerializationFeature.INDENT_OUTPUT);
18
19         try {
20             // Convertir el ArrayList a formato JSON y escribirlo en el archivo
21             objectMapper.writeValue(new File(filePath), arrayList);
22             System.out.println("ArrayList exportado exitosamente a JSON en " + filePath);
23         } catch (IOException e) {
24             System.out.println("Error al exportar el ArrayList a JSON: " + e.getMessage());
25         }
26     }
27 }
```

```
3 usages  Daniel Chaparro Cano
public class XmlWriter<T> {

    // Método para escribir un ArrayList en un archivo XML
    1 usage  Daniel Chaparro Cano
    public void exportArrayListToXml(ArrayList<T> arrayList, String filePath) {
        // Instanciar XmlMapper
        XmlMapper xmlMapper = new XmlMapper();
        xmlMapper.enable(SerializationFeature.INDENT_OUTPUT);
        try {
            // Convertir el ArrayList a formato XML y escribirlo en el archivo
            xmlMapper.writeValue(new File(filePath), arrayList);
            System.out.println("ArrayList exportado exitosamente a XML en " + filePath);
        } catch (IOException e) {
            System.out.println("Error al exportar el ArrayList a XML: " + e.getMessage());
        }
    }
}
```

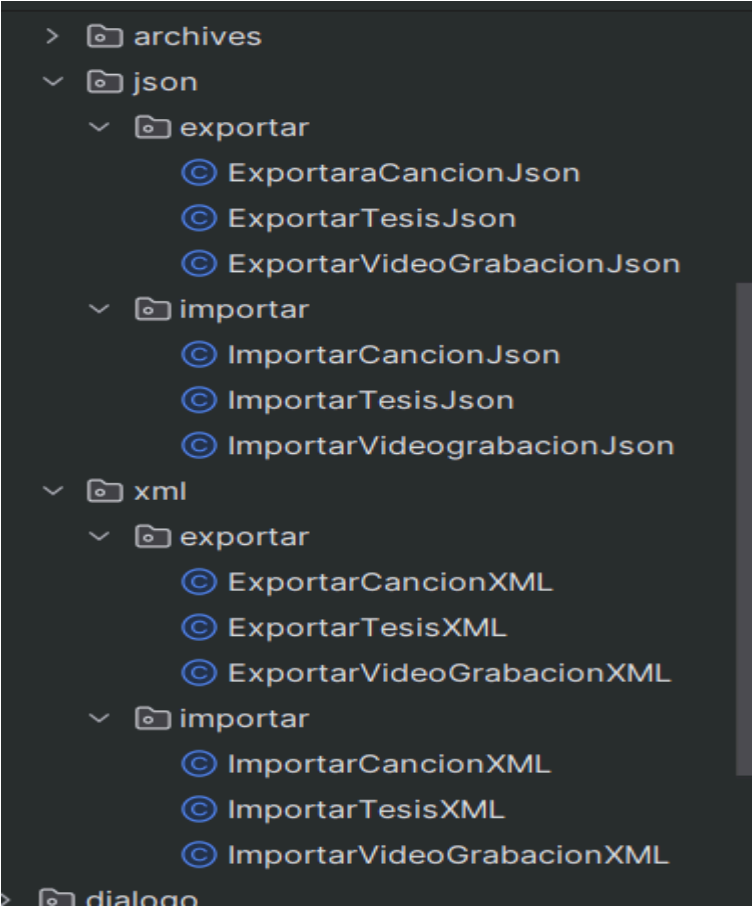
**Carpeta archive handler/ archive**

Carpeta encargada de guardar todos los archivos de donde se importan las canciones, las tesis y las videgrabaciones a la base de datos, y viceversa es decir se exportan.



Carpetas JSON y XML

encargadas de importar y exportar los archivos





```

package com.sofkau.archivehandler.xml.exportar;

> import ...

2 usages  Daniel Chaparro Cano
public class ExportarCancionXML {
    1 usage
    private static CancionOperaciones cancionOp = new CancionOperaciones();

    1 usage  Daniel Chaparro Cano
    public static void exportarCancionXml() {
        XmlWriter<Cancion> xmlWriter = new XmlWriter<>();
        xmlWriter.exportArrayListToXml(cancionOp.getListacanciones(),
            filePath: "src/main/java/com/sofkau/archivehandler/archives/cancionesExportadas.xml"
        );
    }
}

```

```

package com.sofkau.archivehandler.xml.importar;

import ...

2 usages  Daniel Chaparro Cano
public class ImportarCancionXML {
    1 usage
    private static CancionOperaciones cancionOp = new CancionOperaciones();
    2 usages
    private static ArrayList<Cancion> canciones = new ArrayList<>();

    1 usage  Daniel Chaparro Cano
    public static void guardarCancionXml() {
        XmlReader<Cancion> xmlReader = new XmlReader<>();
        canciones = xmlReader.leerXmlArrayArchivo(rutaArchivo: "src/main/java/com/sofkau/archivehandler/archives/canciones.xml",
            Cancion.class);
        for (Cancion cancion : canciones) {
            cancionOp.crearCancion(cancion);
        }
    }
}

```

```

import com.sofkau.logica.cancion.CancionOperaciones;
import com.sofkau.modelo.Cancion;

import java.util.ArrayList;

2 usages  Daniel Chaparro Cano
public class ExportaraCancionJson {

    1 usage
    private static CancionOperaciones cancionOp = new CancionOperaciones();

    1 usage  Daniel Chaparro Cano
    public static void exportarCancionJson(){
        JsonWriter jsonWriter = new JsonWriter();
        jsonWriter.exportArrayListToJson(cancionOp.getListacanciones(),
            filePath: "src/main/java/com/sofkau/archivehandler/archives/cancionesExportadas.json"
        );
    }
}

```

```
package com.sofkau.archivehandler.json.importar;

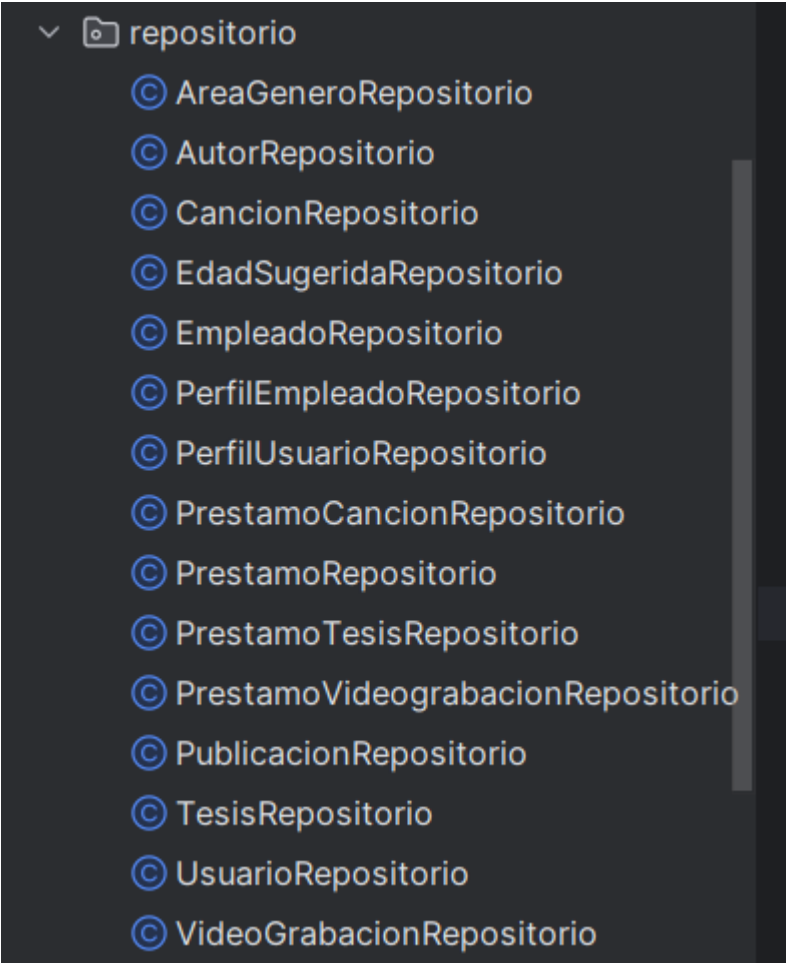
> import ...

2 usages  ▴ Daniel Chaparro Cano
public class ImportarVideoGrabacionJson {

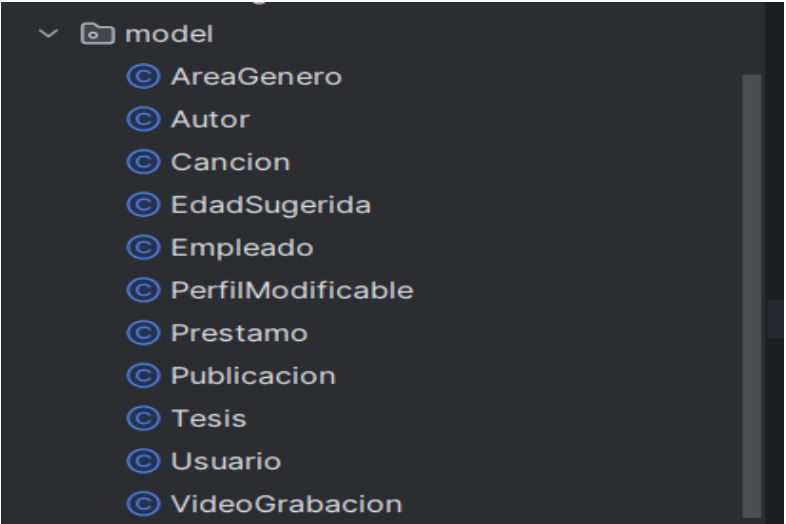
    1 usage
    private static VideoGrabacionOperaciones videograbacionOp = new VideoGrabacionOperaciones();
    2 usages
    private static ArrayList<VideoGrabacion> videograbaciones = new ArrayList<>();

    1 usage  ▴ Daniel Chaparro Cano
    public static void guardarVideoGrabacionJson() {
        JsonReader<VideoGrabacion> jsonReader = new JsonReader<>();
        videograbaciones = jsonReader.leerJsonArrayArchivo( rutaArchivo: "src/main/java/com/sofkau/archivehandler/archives/videograbaciones.json",
            VideoGrabacion.class);
        for (VideoGrabacion videograbacion : videograbaciones) {
            videograbacionOp.crearVideoGrabacion(videograbacion);
        }
    }
}
```

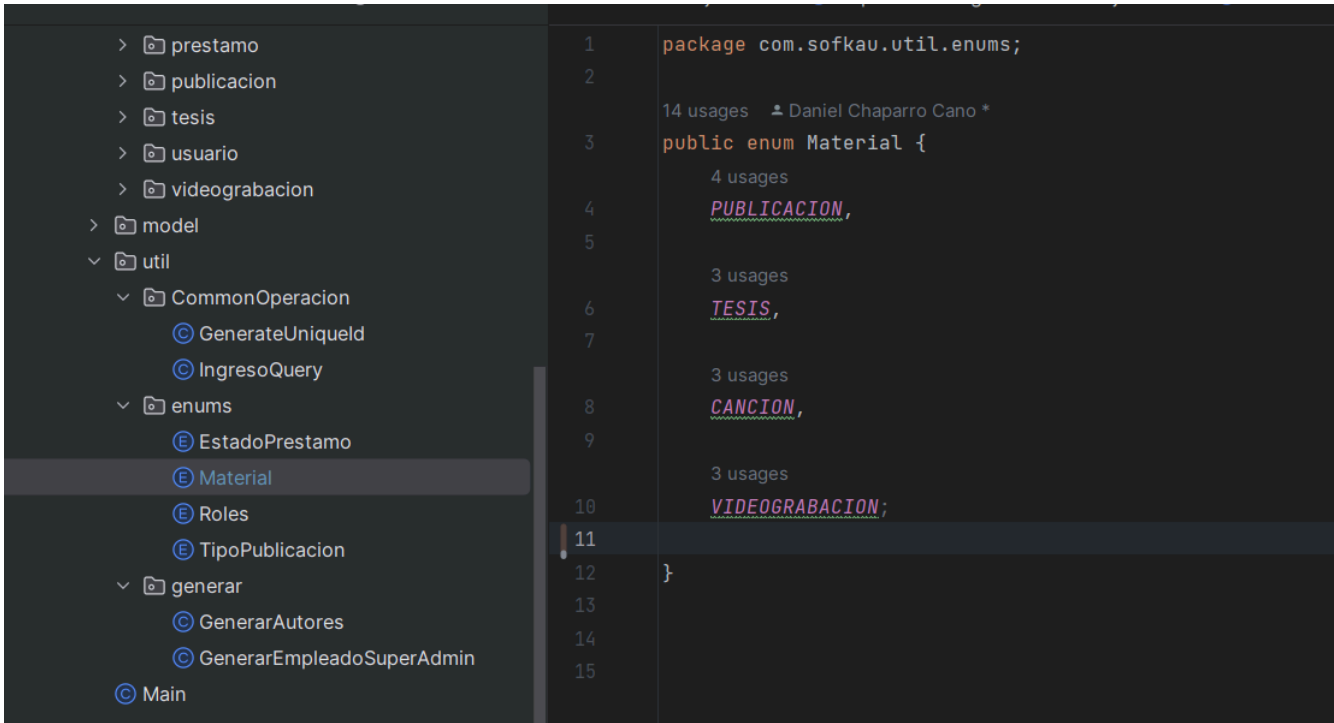
Crece la carpeta repositorio con el acceso a la bd de las nuevas tablas



Crece la cantidad de modelos

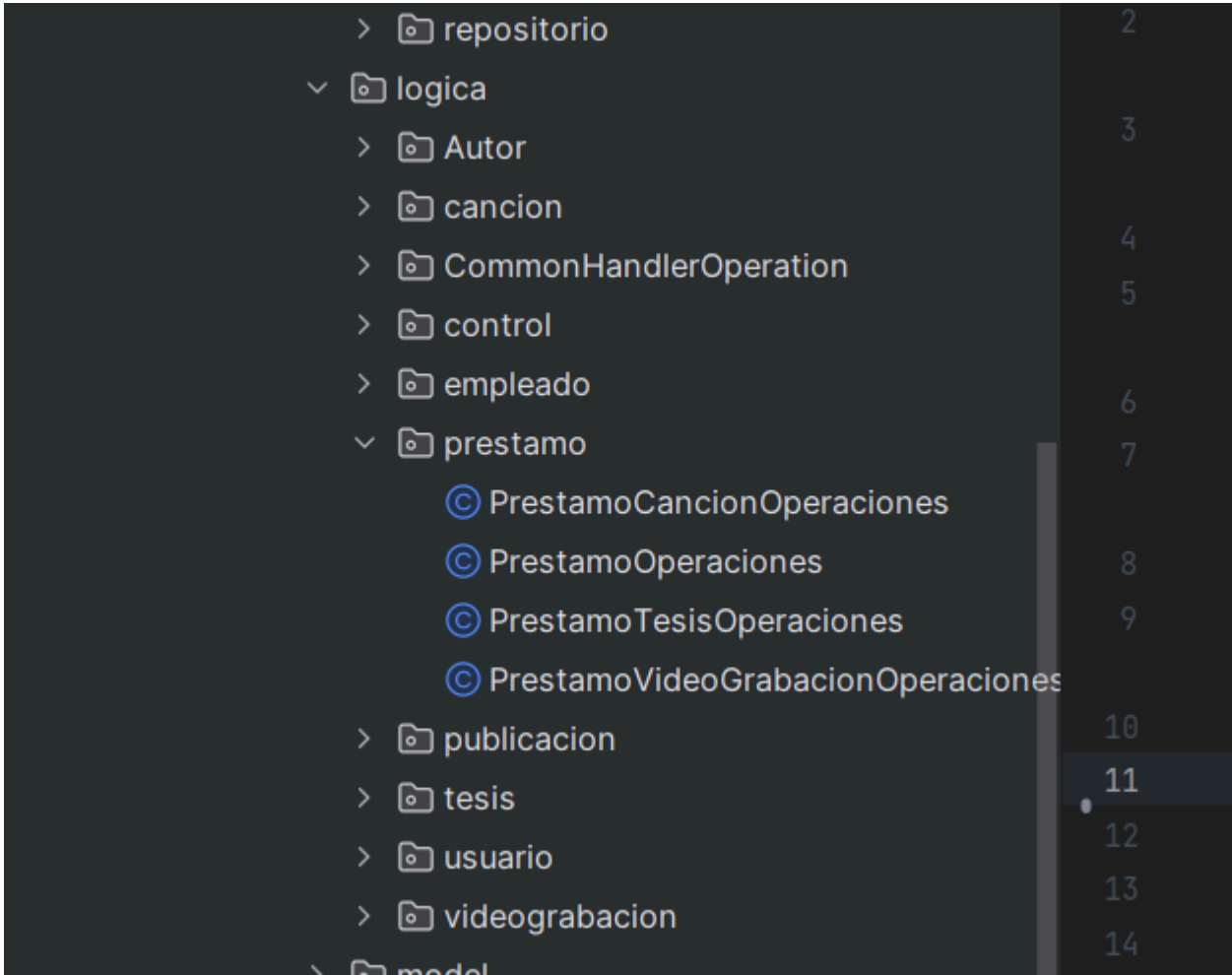


Enum material para manejar los tipos de materiales de la biblioteca

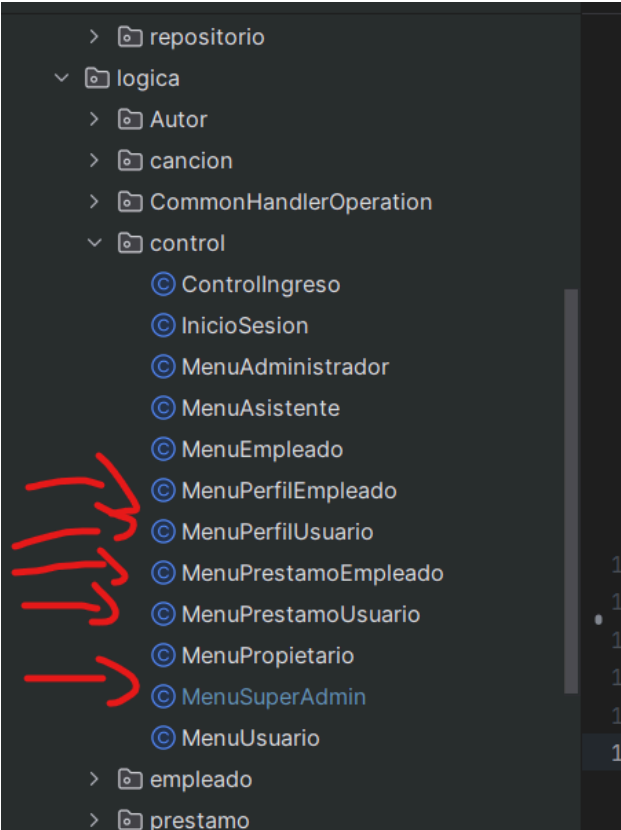


Carpeta logica

Se evidencian más clases en la lógica del prestamo



Se implementan nuevos Menu para las nuevas funcionalidades



Se implementa el menú y el modo de super administrador

```
int opcion = scannerGlobal.nextInt();
scannerGlobal.nextLine();
MenuAsistente.menuAsistente(opcion);
}
case 5 -> {
    MenuUsuario.menuUsuario();
}
case 6 -> {
    if(modosuperAdmin){
        modosuperAdmin = false;
        restaurarValores();
    }else{
        modosuperAdmin = true;
    }
}case 7 ->{
    MenuPerfilEmpleado.menuPerfilEmpleado();
}
default -> {
    System.out.println("Ha ocurrido un error por favor verifique sus credenciales");
    opcion = 0;
}
```

Este modo permite trabajar con información que no se ingresa en la base de datos pero permite el flujo normal de esta

```
videoGrabacionesExportadas.json  videoGrabacionesExportadas.xml  MenuSuperAdmin.java  IngresoQuery.java x v
1 package com.sofkau.util.CommonOperacion;
2
3 > import ...
6
7 public class IngresoQuery {
8     4 usages
9     private static MySQLOperation mySqlOperation = ConexionDatabase.getMySQLOperation();
10
11 // Metodo para ejecutar queries de ingreso de datos
12 // Daniel Chaparro Cano
13 public static void ejecutarIngresoQuery(String query){
14     try {
15         if(!MenuSuperAdmin.isModoSuperAdmin()){
16             mySqlOperation.setSqlStatement(query);
17             mySqlOperation.executeSqlStatementVoid();
18         }
19     } catch (Exception e) {
20         throw new RuntimeException(e);
21     }
22 }
```

si está activado no permite el ingreso de información a la base de datos, pero la info se maneja de forma dinámica mediante colecciones

```
nesExportadas.xml  MenuSuperAdmin.java  IngresoQuery.java  MenuAsistente.java  EmpleadoOperaciones.java x v
3 usages
19 private Empleado empleadoActual;
20
21 2 usages
22 private static Empleado empleadoPerfil;
23
24 6 usages
25 private static HashMap<String, Empleado> empleados = new HashMap<>();
26
27 4 usages Daniel Chaparro Cano *
28 @ public void registrarEmpleado(Empleado empleado, String rol) {
29     empleado.setId(GenerateUniqueId.generateID());
30     empleado.setRol(rol);
31     EmpleadoRepositorio.crearEmpleado(empleado);
32     EmpleadoRepositorio.registroCreacion(empleadoActual.getId(), empleado.getId());
33     MensajeOperacionBd.crearEmpleado();
34     System.out.println(empleado);
35     empleados.put(empleado.getId(), empleado);
36 }
37
38 2 usages Daniel Chaparro Cano
39 > public EmpleadoOperaciones() { getEmpleados(); }
```

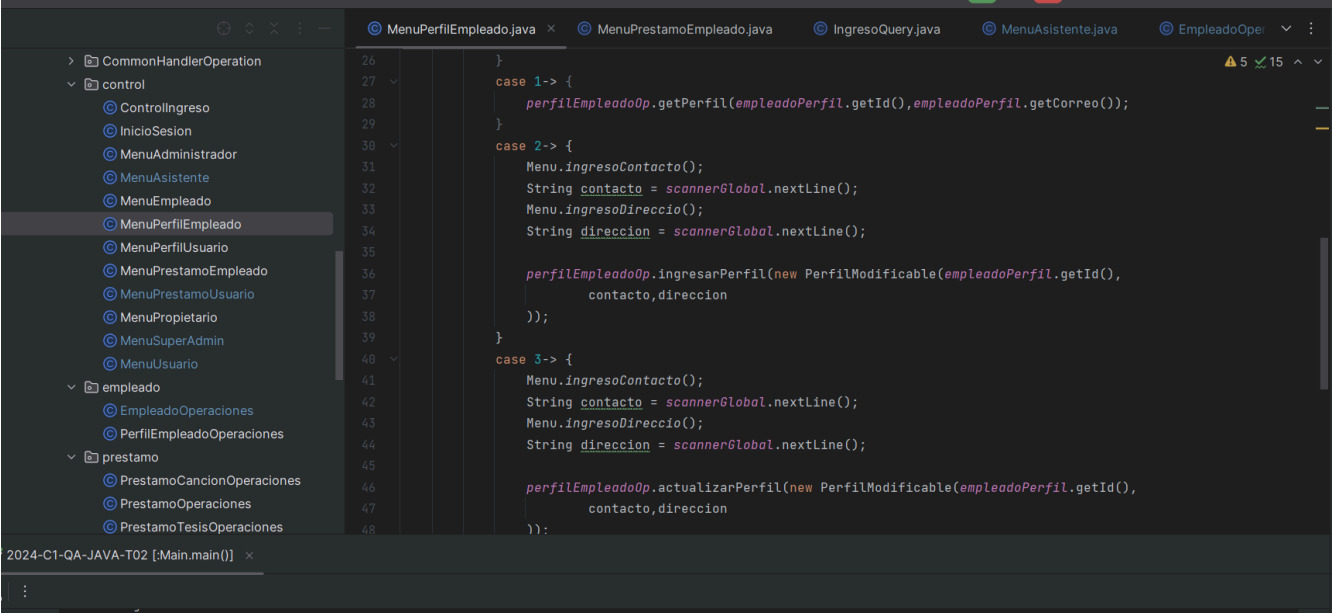
adicional cuando se desactiva se inicializa el sistema, restorando lo al momento anterior de activarse el modo super admin

```
1 usage  Daniel Chaparro Cano *
private static void restaurarValores() {

    usuarioOp = new UsuarioOperaciones();
    empleadoOp.getEmpleados();
    autorOp = new AutorOperaciones();
    publicacionOp = new PublicacionOperaciones();
    prestamoOp = new PrestamoOperaciones();
    cancionOperaciones = new CancionOperaciones();
    videoGrabacionOperaciones = new VideoGrabacionOperaciones();
    tesisOperaciones = new TesisOperaciones();
    perfilOp = new PerfilModificableOperaciones();
    prestCancionOp = new PrestamoCancionOperaciones();
    prestVideoGrabacionOp = new PrestamoVideoGrabacionOperaciones();
    prestamoTesisOp = new PrestamoTesisOperaciones();
    perfilEmpleadoOp = new PerfilEmpleadoOperaciones();

}
```

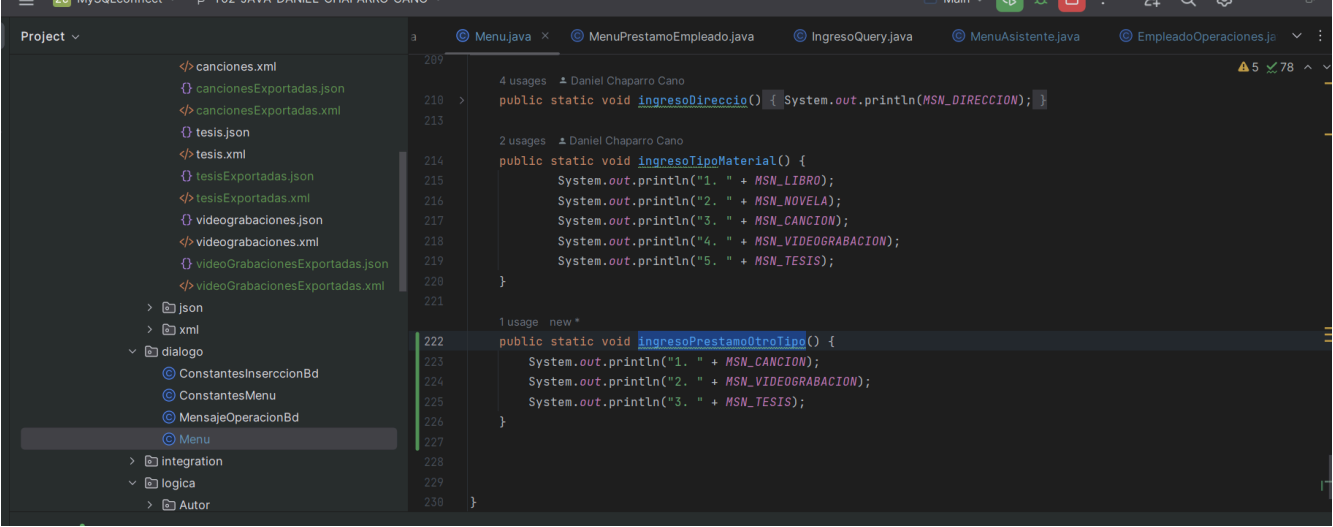
Se implementa un perfil modificable para el empleado y el usuario



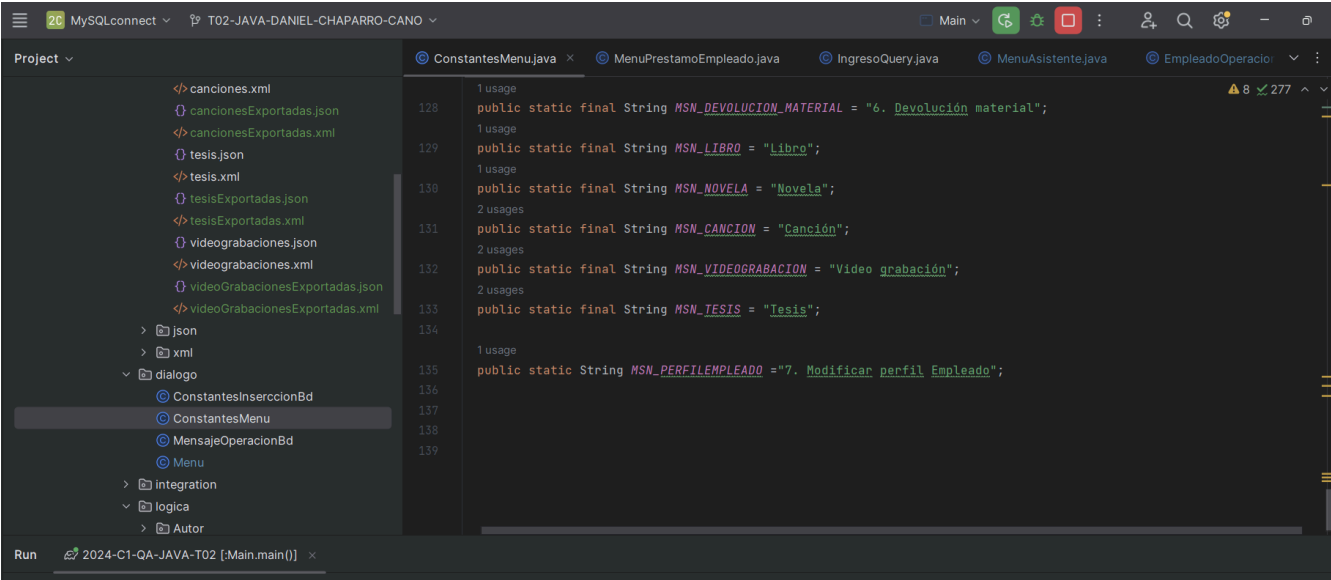
```
26 }
27
28 case 1-> {
29     perfilEmpleadoOp.getPerfil(empleadoPerfil.getId(), empleadoPerfil.getCorreo());
30 }
31
32 case 2-> {
33     Menu.ingresoContacto();
34     String contacto = scannerGlobal.nextLine();
35     Menu.ingresoDireccio();
36     String direccion = scannerGlobal.nextLine();
37
38     perfilEmpleadoOp.ingresarPerfil(new PerfilModificable(empleadoPerfil.getId(),
39         contacto, direccion));
40 }
41
42 case 3-> {
43     Menu.ingresoContacto();
44     String contacto = scannerGlobal.nextLine();
45     Menu.ingresoDireccio();
46     String direccion = scannerGlobal.nextLine();
47
48     perfilEmpleadoOp.actualizarPerfil(new PerfilModificable(empleadoPerfil.getId(),
49         contacto, direccion));
50 }
```

Oportunidades de mejora

- Clase menu y constantes

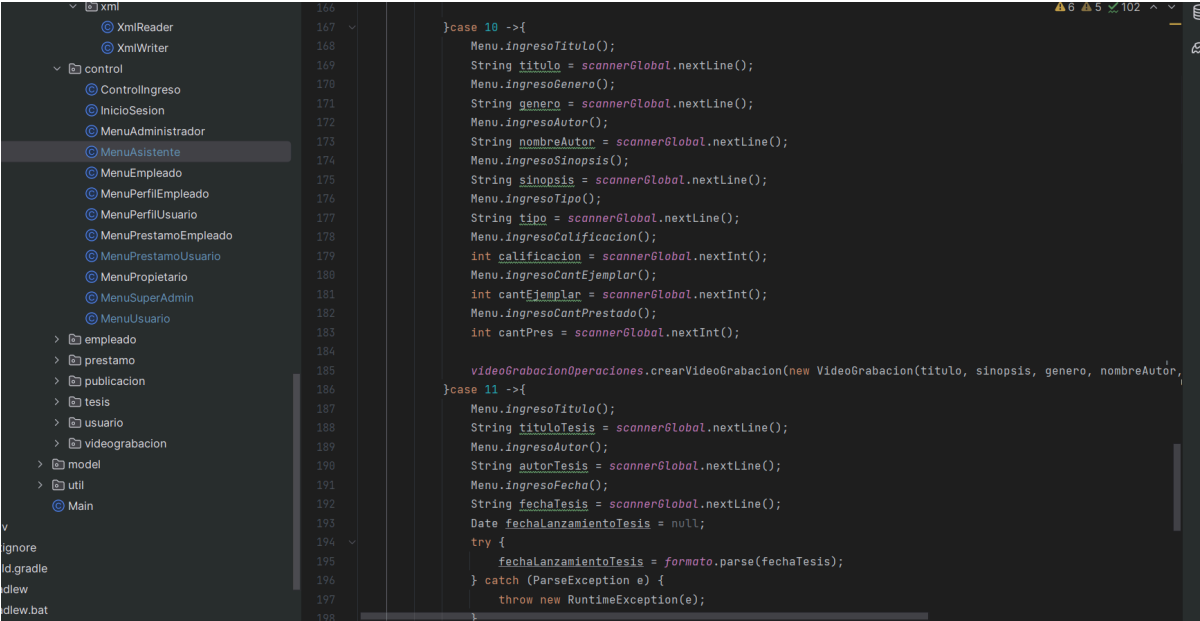
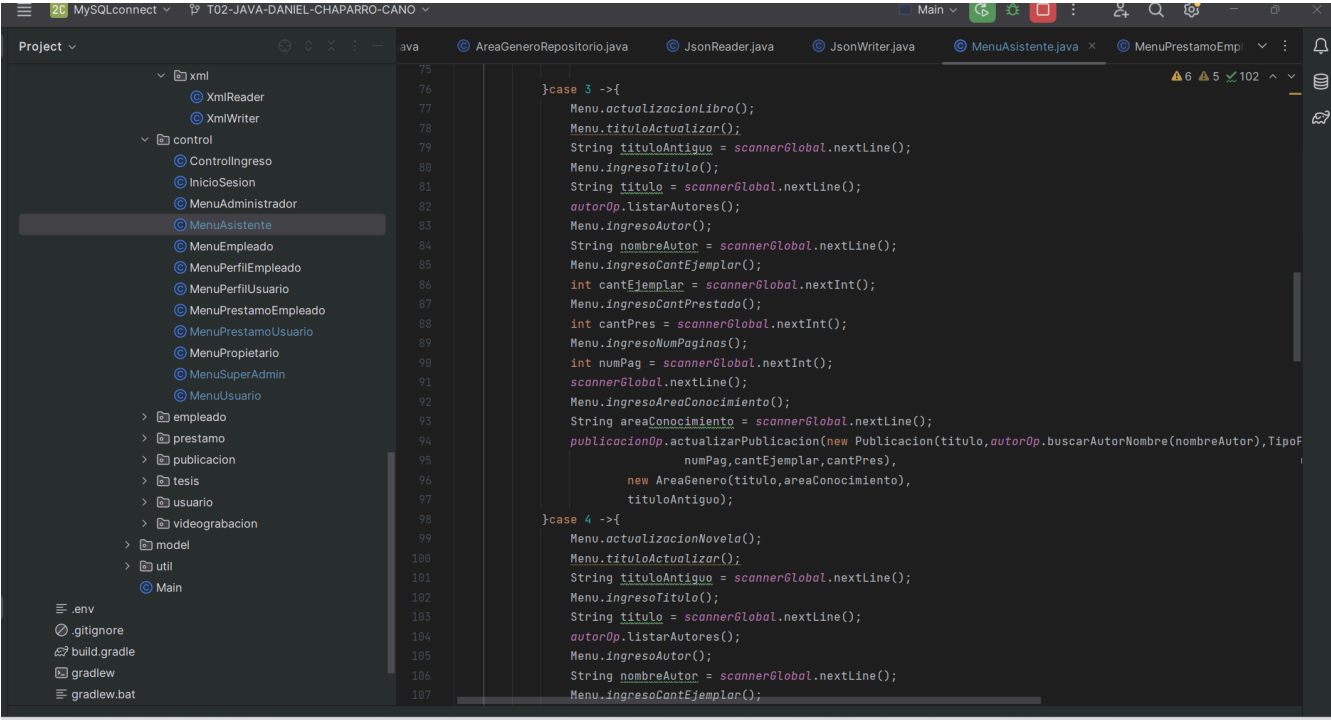


```
289
290
291 4 usages Daniel Chaparro Cano
292 public static void ingresoDireccio() { System.out.println(MSN_DIRECCION); }
293
294 2 usages Daniel Chaparro Cano
295 public static void ingresoTipoMaterial() {
296     System.out.println("1. " + MSN_LIBRO);
297     System.out.println("2. " + MSN_NOVELA);
298     System.out.println("3. " + MSN_CANCION);
299     System.out.println("4. " + MSN_VIDEOGRABACION);
300     System.out.println("5. " + MSN_TESIS);
301 }
302
303 1 usage new *
304 public static void ingresoPrestamoOtroTipo() {
305     System.out.println("1. " + MSN_CANCION);
306     System.out.println("2. " + MSN_VIDEOGRABACION);
307     System.out.println("3. " + MSN_TESIS);
308 }
309 }
```



Aunque estas clases permiten la reutilización, se puede mejorar la organización del código mediante la segregación en clases adicionales. Esto facilita la comprensión, organización y mantenimiento del código.

## - Código repetitivo al ingresar datos



En los menús, se observa una cantidad significativa de código repetitivo. Por ejemplo, al manejar la publicación, las diferencias entre un libro y una novela son mínimas en términos de datos. Sin embargo, al solicitar el ingreso de estos elementos, se repiten numerosas líneas de código. Una solución viable sería crear una capa adicional para gestionar el ingreso de estas publicaciones, separada del switch case que actualmente llama a las funciones correspondientes. Esto no solo reduciría la duplicación de código, sino que también haría que el código sea más legible y mantenible.

#### **Lista funcionalidades sin terminar**

- Exportación y lectura archivo XLSX
- Integración, Registro y consulta MongoDB

#### **Video**

**<https://youtu.be/Si8GQn5gEaY>**