



# **UNIVERSIDAD POLITECNICA SALESIANA**

## **FACULTAD DE INGENIERÍA**

### **INTEGRANTES**

Paúl Crespo

### **MATERIA**

Programación Orientada a Objetos

### **DOCENTE**

Ing. Gustavo Navas Ruilova

### **TEMA**

Unidad 4 – Programación Limpia

**2025**

# Sistema de Gestión de Contenido AudioVisual

Enlace Al Repositorio De Github:

Repositorio: [https://github.com/PaulCrespoC/POO\\_Unidad2](https://github.com/PaulCrespoC/POO_Unidad2)

## Aspectos Implementados Por Etapa

### Etapa 1: Manejo De Archivos

- **Lectura de archivos CSV:** Implementación completa con FileHandlerService
- **Escritura de archivos CSV:** Persistencia automática y manual
- **Validación de formato:** Control de extensiones y estructura
- **Encoding UTF-8:** Soporte para caracteres especiales
- **Manejo de errores:** Recuperación elegante de fallos
- **Formato estructurado:** Headers descriptivos y comentarios

### Etapa 2: Refactorización Y Código Limpio

- **Nombres descriptivos:** Clases, métodos y variables con nombres claros
- **Métodos pequeños:** División de responsabilidades específicas
- **Eliminación de duplicación:** Métodos reutilizables y DRY principle
- **Comentarios útiles:** Javadoc completo y comentarios explicativos
- **Validaciones robustas:** Control de entrada en constructores y setters
- **Encapsulación mejorada:** Copias defensivas y acceso controlado

### Etapa 3: Principios Solid

- **SRP:** Cada clase tiene responsabilidad única y específica
- **OCP:** Extensible para nuevos tipos sin modificar código existente
- **LSP:** Polimorfismo correcto en todas las subclases
- **ISP:** Interfaces segregadas y específicas por dominio
- **DIP:** Dependencias en abstracciones con inyección de dependencias

### Etapa 4: Patrón MVC

- **Modelo:** Entidades de negocio en model/ con lógica de datos
- **Vista:** Interfaz Swing en view/ con componentes interactivos
- **Controlador:** Coordinación en controller/ entre vista y modelo
- **Separación clara:** Sin dependencias circulares entre capas
- **Comunicación unidireccional:** Flujo de datos controlado

### Etapa 5: Pruebas Unitarias

- **Cobertura >90%:** Tests exhaustivos con JUnit 5
- **Mocking:** Uso de Mockito para dependencias
- **Tests de integración:** Verificación de componentes completos
- **Casos límite:** Edge cases y validaciones extremas
- **Suite de pruebas:** Ejecución automatizada de todos los tests

## FUNCIONALIDADES IMPLEMENTADAS

### Funcionalidades principales

1. **CRUD Completo:** Crear, leer, actualizar y eliminar contenido audiovisual
2. **Interfaz Gráfica:** Aplicación desktop con Java Swing
3. **Persistencia:** Manejo de archivos CSV para almacenamiento
4. **Búsqueda y Filtrado:** Múltiples criterios de búsqueda
5. **Validaciones:** Control robusto de entrada de datos
6. **Estadísticas:** Reportes automáticos del sistema
7. **Exportar/Importar:** Funcionalidades de archivo flexible

### Relaciones implementadas

- **Agregación:** Película ↔ Actor (independientes)
- **Composición:** SerieDeTV ↔ Temporada (dependientes)
- **Asociación:** Documental ↔ Investigador (colaboración)

### Tipos de contenido

- **Película:** Con estudio y actores
- **Serie de TV:** Con temporadas y episodios
- **Documental:** Con investigadores colaboradores
- **Video YouTube:** Con métricas de engagement
- **Cortometraje:** Con información de festivales

### Arquitectura del sistema

```
src/
├── interfaces/      # Contratos (DIP)
│   ├── IContenidoService
│   ├── IContenidoRepository
│   └── IFileHandler
├── model/          # Entidades (M en MVC)
│   ├── ContenidoAudiovisual (abstracta)
│   ├── Pelicula, SerieDeTV, Documental
│   ├── VideoYouTube, Cortometraje
│   └── Actor, Temporada, Investigador
├── service/        # Lógica de negocio
│   ├── ContenidoService
│   ├── ContenidoRepository
│   └── FileHandlerService
├── view/           # Interfaz (V en MVC)
│   ├── MainView
│   └── ContenidoDialog
├── controller/     # Coordinación (C en MVC)
│   └── ContenidoController
└── MainApplication # Punto de entrada
```

## Instrucciones de ejecución

### Requisitos Previos

- Java 11 o superior
- IDE IntelliJ IDEA / Eclipse

### Compilación

*# Crear directorios*

```
mkdir -p bin data
```

*# Compilar proyecto*

```
javac -d bin src/interfaces/*.java src/model/*.java src/service/*.java src/controller/*.java src/view/*.java src/*.java
```

*# Ejecutar aplicación*

```
java -cp bin MainApplication
```

### Ejecución de Pruebas

*# Compilar tests (requiere JUnit 5)*

```
javac -d bin -cp "bin:junit-platform-console-standalone-*.jar" test/**/*.java
```

*# Ejecutar suite de pruebas*

```
java -jar junit-platform-console-standalone-*.jar --class-path bin --scan-class-path
```

### Dependencias requeridas

Para Ejecución

- **Java Runtime:** JRE 11+
- **Java Swing:** Incluido en JRE estándar

### Para Desarrollo y Testing

- **JUnit 5:** Framework de pruebas unitarias
- **Mockito:** Library para mocking en tests
- **JUnit Platform Console Standalone:** Para ejecución de tests

### Archivos JAR Necesarios (para testing)

- junit-platform-console-standalone-1.9.1.jar
- mockito-core-4.6.1.jar
- mockito-junit-jupiter-4.6.1.jar

### Evidencias de funcionamiento

#### Compilación Exitosa

- Proyecto compila sin errores ni warnings
- Todas las dependencias resueltas correctamente

#### Ejecución de Aplicación

- Interfaz gráfica se ejecuta correctamente

- Todas las funcionalidades operativas
- Manejo de archivos CSV funcional

### Pruebas Unitarias

- Suite de 80+ pruebas unitarias
- Cobertura superior al 90%
- Todos los tests pasan exitosamente

### Principios Aplicados

- SOLID: Implementación completa de los 5 principios
- MVC: Separación clara de responsabilidades
- Código Limpio: Refactorización profesional

### Mejoras implementadas respecto a v1.0

Aspecto	v1.0	v2.0
<b>Interfaz</b>	Solo consola	Swing GUI completa
<b>Arquitectura</b>	Monolítica	MVC con SOLID
<b>Persistencia</b>	No implementada	CSV robusto
<b>Testing</b>	Manual	JUnit 5 automatizado
<b>Código</b>	Básico	Profesional refactorizado
<b>Funcionalidades</b>	Limitadas	CRUD completo

### Conclusiones

El proyecto ha evolucionado exitosamente desde una aplicación básica de consola hasta un sistema desktop robusto que implementa las mejores prácticas de la ingeniería de software. La aplicación de los principios SOLID, el patrón MVC, y las técnicas de código limpio han resultado en un sistema mantenible, extensible y de calidad profesional.