



Matias Cedeño, Anthony Cochea y Matheo Chávez

Validación y Verificación de software

2024-2025

Facultad de Ingeniería y Ciencias Aplicadas, Universidad De Las Américas

martes 21 de enero del 2025

Contenido

1.	Introducción.....	4
2.	Descripción General.....	4
2.1.	Alcance:	4
2.2.	Actores:.....	5
2.3.	Restricciones:.....	5
3.	Requisitos Específicos.....	5
3.1.	Requisitos funcionales:	5
3.2.	Requisitos no funcionales:	6
4.	Propuesta de Solución.....	7
4.1.	Uso de OpenXava para Desarrollo Rápido	7
4.2.	Modularidad en el Diseño.....	7
4.3.	Optimización del Tiempo y Recursos	7
4.4.	Validaciones y Consistencia.....	8
4.6.	Estrategias de Validación	8
4.7.	Gestión de Seguridad y Desempeño.....	9
5.	Historias de Usuario	9
6.	Funcionalidad y Time to Market	12
7.	Técnicas de Prueba	14
7.1.	Prueba de Caja Negra: Clases de Equivalencia	14
7.2.	Prueba de Caja Negra: Valores Límite	14
7.3.	Prueba de Caja Blanca: Cobertura de Decisiones	15
7.4.	Prueba de Caja Blanca: Cobertura de Caminos	15
8.	Casos de Uso	17
9.	Diseño de Clases.....	18
10.	Descripción de implementación	22
10.1.	Diagrama de Datos.....	22
10.2.	Herramientas y Plataformas Utilizadas	23
10.3.	Descripción de la Implementación	23
11.	Casos de Prueba Técnica 1.....	25
12.	Casos de Prueba Técnica 2.....	28
13.	Casos de Prueba Técnica 3.....	30
14.	Casos de Prueba Técnica 4.....	35
	CP_REGCAN1: Registrar Canción con campos vacíos.....	36

2. CP_REGCAN2: Registrar Canción con duración inválida	36
3. CP_REGCAN3: Registrar Canción con datos válidos	37
4.CP_REGCAN4: Registrar Canción sin Género asociado.....	37
5. CP_EDITCAN1: Editar Canción con campos obligatorios vacíos ...	38
6. CP_REGALB1: Registrar Álbum con campos vacíos	38
7.-CP_REGALB2: Registrar Álbum con artista inexistente	38
8. CP_REGALB3: Registrar Álbum con datos válidos.....	39
9. CP_EDITALB1: Editar Álbum con campos obligatorios vacíos	39
15. Resultado de Test.....	49
15.1. Representación Gráfica de los Resultados	49
15.2. Análisis de los Resultados	50
15.3. Recomendaciones	50
15.4. Conclusión	51
16. Links.....	51

1. Introducción

El presente documento detalla las fases iniciales de desarrollo y validación del sistema "Musicly". Este proyecto tiene como objetivo principal ofrecer una solución eficiente para organizar y administrar información relacionada con canciones, álbumes, géneros, artistas y playlists, en un entorno académico orientado a la validación y verificación de software. A través del uso del framework OpenXava, el sistema busca garantizar una experiencia de usuario estructurada y de alta calidad.

El desarrollo de Musicly está alineado con las mejores prácticas en gestión de proyectos de software, con un enfoque iterativo para implementar módulos funcionales clave. Este documento incluye la introducción general al problema a resolver, una descripción detallada del sistema y sus funcionalidades principales, y un listado de los requisitos específicos que garantizan su implementación exitosa.

2. Descripción General

Musicly es un sistema diseñado para operar como una solución autónoma y flexible dentro del contexto de validación de software. Su principal objetivo es proporcionar a los usuarios herramientas eficientes para gestionar bibliotecas musicales, permitiendo el registro, organización y consulta de información sobre canciones, álbumes, géneros, artistas y playlists. Además, el sistema se centra en garantizar la consistencia de los datos y cumplir con las reglas de negocio establecidas.

El proyecto también tiene potencial para escalar y adaptarse a entornos más complejos, como integraciones con plataformas de streaming, ampliaciones en el análisis de popularidad o desarrollo de funcionalidades móviles. Actualmente, su desarrollo está orientado a satisfacer los requisitos de un semestre académico.

2.1. Alcance:

- Registro de canciones y clasificación por álbum y género.
- Gestión de información de artistas, incluyendo roles en canciones y álbumes.
- Creación y administración de playlists personalizadas con cálculo automático de duración.
- Creación y administración de roles personalizados.

2.2.Actores:

- Usuarios finales: Personas interesadas en gestionar sus bibliotecas musicales personales.
- Administradores del sistema: Responsables de supervisar, ingreso, modificación y mantener la calidad de los datos.
- Validadores o testers: Profesionales encargados de garantizar que el sistema cumple con los requisitos definidos mediante pruebas funcionales y técnicas.

2.3.Restricciones:

Rol único de Usuario

- El sistema operará bajo un único rol de usuario con privilegios completos para realizar todas las operaciones disponibles.
- No se implementarán diferentes niveles de acceso.

Reglas de negocio inalterables

- Cada canción debe pertenecer a un solo álbum.
- Las canciones no pueden exceder a 60 minutos.
- Todas las operaciones deben respetar las reglas de negocio establecidas para mantener la consistencia de los datos.

3. Requisitos Específicos

3.1.Requisitos funcionales:

1.1.1. Gestión de canciones:

- RF1.1: Registro de canciones con título, duración, género, álbum asociado y artistas.
- RF1.2: Edición y eliminación de canciones existentes.
- RF1.3: Validación de unicidad de álbum por canción.
- RF1.4: Búsqueda y filtrado de canciones por palabras clave, género o popularidad.

- RF1.5: Listado de canciones asociadas a álbum y género.
- RF1.6: La duración de una canción no puede durar más de una hora.

1.1.2. Gestión de álbumes:

- RF2.1: Registro de álbumes con nombre, fecha de lanzamiento y artista principal.
- RF2.2: Edición y eliminación de álbumes existentes.
- RF2.3: Listado de canciones asociadas a cada álbum.

1.1.3. Gestión de géneros:

- RF3.1: Creación, edición y eliminación de géneros musicales.
- RF3.2: Validación de nombres únicos para cada género.

1.1.4. Gestión de artistas:

- RF4.1: Registro de artistas con información como nombre, rol y biografía.
- RF4.2: Asignación de rol a un artista.
- RF4.3: Edición y eliminación de información de artistas.
- RF4.4: Asociación de artistas con múltiples álbumes.

1.1.5. Gestión de playlists:

- 1.1.1.1. RF5.1: Creación de playlists personalizadas.
- 1.1.1.2. RF5.2: Cálculo automático de la duración total de la playlist.
- 1.1.1.3. RF5.3: Edición y eliminación de playlists.

3.2.Requisitos no funcionales:

- Rendimiento: Respuesta del sistema en menos de 3 segundos para el 90% de las operaciones.
- Seguridad: Protección de ataques comunes como validaciones contra inyección SQL, CSRF y XSS.
- Disponibilidad: Disponibilidad del sistema durante las horas de uso, con un objetivo del 95%.
- Usabilidad: Diseño de interfaz intuitivo con validaciones claras para el usuario.
- Compatibilidad: Operación garantizada en navegadores modernos y dispositivos de escritorio.

4. Propuesta de Solución

Para garantizar que el sistema Musiely cumpla con los requisitos funcionales y no funcionales establecidos, se plantean las siguientes propuestas de solución, alineadas con las necesidades del proyecto:

4.1. Uso de OpenXava para Desarrollo Rápido

Razonamiento: OpenXava es un framework diseñado para el desarrollo rápido de aplicaciones empresariales, lo que permite reducir los tiempos de implementación y validar rápidamente las funcionalidades del sistema.

Aplicación:

- Generar automáticamente interfaces y formularios CRUD para las entidades definidas (canciones, álbumes, artistas, géneros y playlists).
- Aprovechar las herramientas de validación declarativa para cumplir con reglas de negocio, como la unicidad de géneros y el límite de duración de playlists.

4.2. Modularidad en el Diseño

- **Razonamiento:** Dividir el desarrollo en módulos funcionales permite un avance incremental y facilita las pruebas y el mantenimiento del sistema.
- **Aplicación:**
 - Crear módulos independientes para la gestión de canciones, álbumes, géneros, playlists y estadísticas de popularidad.
 - Integrar los módulos mediante APIs internas y asegurarse de que sean cohesivos pero desacoplados.

4.3. Optimización del Tiempo y Recursos

- **Razonamiento:** Utilizar herramientas y tecnologías que minimicen la carga de trabajo y maximicen la eficiencia del equipo.
- **Aplicación:**
 - Implementar una base de datos relacional (PostgreSQL o MySQL) con esquemas predefinidos para gestionar relaciones entre entidades.
 - Emplear un enfoque iterativo basado en metodologías ágiles, con entregas parciales y revisión constante.

4.4. Validaciones y Consistencia

- **Razonamiento:** Garantizar la calidad de los datos es esencial para mantener la confiabilidad del sistema.
- **Aplicación:**
 - Configurar validaciones declarativas en OpenXava para asegurar que:
 - Las canciones tengan asignado un único género y álbum.
 - Las playlists respeten el límite de duración de 3 horas.
 - Los datos de género sean únicos y consistentes.
 - Implementar validaciones adicionales a nivel de código para garantizar la integridad de los datos.

4.5. Escalabilidad y Usabilidad

- **Razonamiento:** El sistema debe ser fácil de usar y capaz de evolucionar con futuras necesidades.
- **Aplicación:**
 - Diseñar una interfaz de usuario intuitiva utilizando los componentes de OpenXava para facilitar la navegación y la interacción.
 - Asegurar la compatibilidad del sistema con navegadores web modernos para un acceso universal.
 - Preparar el sistema para futuras ampliaciones, como integración con plataformas de streaming o análisis avanzado de datos.

4.6. Estrategias de Validación

- **Razonamiento:** Validar las funcionalidades en cada etapa asegura que el sistema cumpla con los objetivos planteados.
- **Aplicación:**
 - Diseñar y ejecutar casos de prueba para los módulos más importantes, validando desde la funcionalidad básica hasta la consistencia de datos.
 - Registrar resultados y realizar ajustes oportunos basados en retroalimentación.

4.7. Gestión de Seguridad y Desempeño

- **Razonamiento:** Asegurar la protección de datos y la eficiencia del sistema es fundamental para su confiabilidad.
- **Aplicación:**
 - Optimizar consultas a la base de datos para tiempos de respuesta inferiores a 3 segundos en el 90% de las operaciones.

5. Historias de Usuario

Gestión de Canciones

1. Historia de Usuario RF 1.1

- Como administrador del sistema,
- quiero registrar nuevas canciones proporcionando el título, duración, género, álbum y artistas asociados.
- para asegurar que la biblioteca musical contiene todas las canciones necesarias con datos completos y correctos.

2. Historia de Usuario RF 1.2

- Como administrador del sistema,
- quiero modificar o eliminar canciones previamente registradas,
- para corregir errores, actualizar datos o eliminar canciones obsoletas de la biblioteca musical.

3. Historia de Usuario RF 1.3

- Como administrador del sistema,
- quiero que el sistema valide que cada canción pertenezca a un único álbum al momento de registrarla,
- para mantener la consistencia y organización lógica en la clasificación de las canciones.

Gestión de Álbumes

4. Historia de Usuario RF 2.1

- Como administrador del sistema,

- quiero registrar nuevos álbumes proporcionando el nombre, fecha de lanzamiento y el artista,
- para organizar las canciones en sus álbumes correspondientes y facilitar su consulta.

5. Historia de Usuario RF 2.2

- Como administrador del sistema,
- quiero editar la información o eliminar álbumes existentes,
- para mantener actualizados los datos y eliminar información innecesaria.

6. Historia de Usuario RF 2.3

- Como administrador del sistema,
- quiero visualizar la lista completa de canciones asociadas a un álbum específico,
- para identificar rápidamente el contenido de cada álbum.

Gestión de Géneros

7. Historia de Usuario RF 3.1

- Como administrador del sistema,
- quiero agregar, editar y eliminar géneros musicales en el sistema,
- para clasificar las canciones de manera precisa y organizada.

8. Historia de Usuario RF 3.2

- Como administrador del sistema,
- quiero asegurar que cada género musical registrado tenga un nombre único,
- para evitar confusiones y mantener la integridad de los datos en el sistema.

Gestión de Artistas

9. Historia de Usuario RF 4.1

- Como administrador del sistema,

- quiero registrar nuevos artistas con detalles como nombre, rol (e.g., vocalista, productor) y una biografía breve,
- para documentar la participación de cada artista en canciones y álbumes.

10. Historia de Usuario RF 4.2

- Como administrador del sistema,
- quiero asignar un rol a un artista en una canción o álbum,
- para reflejar con precisión su contribución en cada proyecto musical.

11. Historia de Usuario RF 4.3

- Como administrador del sistema,
- quiero actualizar o eliminar la información de artistas registrados,
- para garantizar que los datos reflejen su actividad actual en el sistema.

Gestión de Playlists

12. Historia de Usuario RF 5.1

- Como usuario final,
- quiero crear playlists personalizadas seleccionando canciones,
- para organizar mis canciones según mis preferencias.

13. Historia de Usuario RF 5.2

- Como usuario final,
- quiero que el sistema calcule automáticamente la duración total de una playlist,
- para asegurar que mis listas sean prácticas y ajustadas a mis necesidades de reproducción.

14. Historia de Usuario RF 5.3

- Como usuario final,
- quiero editar el contenido o eliminar playlists existentes,
- para personalizar mis listas musicales según mis necesidades actuales.

Consulta de Popularidad

15. Historia de Usuario RF 6.1

- Como usuario final,
- quiero consultar la popularidad de canciones basándome en la cantidad de reproducciones registradas,
- para identificar cuáles son las canciones más escuchadas en mi biblioteca.

16. Historia de Usuario RF 6.2

- Como usuario final,
- quiero consultar la popularidad de playlists según sus datos de reproducción,
- para descubrir cuáles son las listas más exitosas en mi colección.

Búsqueda y Filtros Avanzados

17. Historia de Usuario RF 7.1

- Como usuario final,
- quiero buscar canciones, álbumes, géneros y artistas utilizando palabras clave,
- para localizar rápidamente la información que necesito.

18. Historia de Usuario RF 7.2

- Como usuario final,
- quiero aplicar filtros avanzados por género, artista, fecha de lanzamiento o popularidad,
- para refinar mis búsquedas y obtener resultados más relevantes.

6. Funcionalidad y Time to Market

El concepto de Time to Market, aplicado en el desarrollo del sistema Musicly, se centra en reducir el tiempo necesario para que el producto esté disponible para pruebas y uso final. La clave para alcanzar un TTM óptimo en este proyecto es la implementación en OpenXava, un framework que permite el desarrollo rápido y eficiente de aplicaciones empresariales en Java. A continuación, se describen las principales funcionalidades y el uso de esta estrategia:

1. Desarrollo Rápido con OpenXava

- **Automatización del Proceso de Desarrollo:** OpenXava permite generar automáticamente interfaces y formularios para operaciones CRUD (Crear,

Leer, Actualizar y Eliminar), lo que reduce significativamente el tiempo invertido en programación manual.

- **Soporte Declarativo:** Las reglas de negocio y validaciones se definen de manera declarativa, facilitando la implementación de restricciones como:
 - Límite de duración en playlists.
 - Clasificación única de canciones por género y álbum.

2. Modularidad y Flexibilidad

- **Desarrollo por Módulos:** La implementación se divide en módulos funcionales independientes (gestión de canciones, géneros, álbumes, artistas y playlists). Esto permite avanzar de manera incremental y entregar funcionalidades clave rápidamente.
- **Evolución Rápida:** El diseño modular facilita la incorporación de nuevas características en futuras iteraciones, como análisis avanzado de popularidad o integración con servicios externos.

3. Minimización de Recursos y Esfuerzo

- **Reutilización de Componentes Predefinidos:** OpenXava proporciona un conjunto de herramientas estándar que eliminan la necesidad de desarrollar interfaces y funcionalidades desde cero.
- **Integración Simplificada:** La compatibilidad con bases de datos relacionales, como MySQL o PostgreSQL, acelera la configuración y gestión de datos.

4. Alineación con los Plazos Académicos

- **Cronograma Ajustado:** La elección de OpenXava como plataforma permite cumplir con los plazos establecidos en el cronograma del proyecto, entregando un sistema funcional en cada etapa clave.
- **Entrega Temprana para Validación:** Las iteraciones rápidas aseguran que las funcionalidades estén disponibles para pruebas y ajustes antes de la finalización del periodo académico.

5. Beneficios del Enfoque TTM

- **Reducción del Tiempo Total de Desarrollo:** Gracias a OpenXava, se logra un avance acelerado desde la concepción del sistema hasta la entrega funcional.
- **Mayor Enfoque en Reglas de Negocio:** Al simplificar los aspectos técnicos, el equipo puede dedicar más tiempo a garantizar que las funcionalidades cumplan con los objetivos del sistema.

- **Incremento en la Calidad y Validación:** La disponibilidad temprana permite identificar y corregir errores en etapas iniciales, mejorando la calidad del producto final.

7. Técnicas de Prueba

Para el sistema Musicly, se seleccionaron las siguientes técnicas de prueba, asegurando que cubran tanto la validación desde la perspectiva del usuario como la lógica interna del sistema. Estas técnicas son fundamentales para garantizar que el software cumpla con los requisitos funcionales y no funcionales especificados.

7.1. Prueba de Caja Negra: Clases de Equivalencia

Descripción: Esta técnica se enfoca en dividir las entradas posibles del sistema en clases de equivalencia (válidas e inválidas) y seleccionar valores representativos de cada clase para las pruebas.

- **Ejemplo en Musicly:**
 - Caso de prueba: Registrar canciones con datos válidos e inválidos.
 - Aplicación: Verificar que el sistema maneje correctamente las entradas válidas (registro exitoso) e inválidas (mostrar mensajes de error).
- **Razón para Selección:**
 - Permite validar que las funcionalidades operen correctamente desde la perspectiva del usuario, sin necesidad de conocer la lógica interna del código.
 - Asegura que todas las posibles condiciones de entrada sean manejadas adecuadamente, mejorando la experiencia del usuario.

7.2. Prueba de Caja Negra: Valores Límite

Descripción: Evalúa el comportamiento del sistema en los extremos de los rangos de entrada aceptables y justo fuera de estos límites.

- **Ejemplo en Musicly:**
 - Caso de prueba: Verificar que las canciones no superen la duración máxima de una hora.

- Aplicación: Probar con el ingreso de una canción que tenga exactamente 60 minutos y otro ingreso que exceda.
- **Razón para Selección:**
 - Identifica errores en los bordes de las reglas de negocio, donde los sistemas suelen fallar.
 - Es especialmente útil para asegurar que las restricciones de negocio definidas se implementen correctamente.

7.3. Prueba de Caja Blanca: Cobertura de Decisiones

Descripción: Examina la lógica interna del código, asegurándose de que todas las decisiones posibles (condiciones) se evalúen al menos una vez.

- **Ejemplo en Musicly:**
 - Caso de prueba: Validar que la lógica de eliminación de álbumes asociada a canciones funcione correctamente.
 - Aplicación: Verificar que el sistema implemente adecuadamente los flujos alternativos, como bloquear la eliminación si hay asociaciones activas.
- **Razón para Selección:**
 - Asegura que la lógica interna sea robusta y no contenga caminos no probados.
 - Identifica errores en la implementación del flujo lógico del sistema, especialmente en escenarios complejos con múltiples condiciones.

7.4. Prueba de Caja Blanca: Cobertura de Caminos

Descripción: Esta técnica examina todos los posibles flujos de ejecución dentro de un método, asegurando que cada camino independiente en el código sea ejecutado al menos una vez.

- **Ejemplo en Musicly:**
 - Caso de prueba: Validar el cálculo de la duración total en segundos de una lista de canciones.
 - Aplicación: Se analizó el método `getDuracionTotalEnSegundos()` que calcula la duración total de una lista de canciones, identificando y probando todos los caminos posibles.

Justificación de las Técnicas Seleccionadas

1. Clases de Equivalencia:

- a. Facilita la identificación de escenarios representativos de entradas válidas e inválidas.
- b. Optimiza el número de pruebas al eliminar redundancias.

2. Valores Límite:

- a. Ayuda a detectar errores en los bordes del rango aceptable, donde es más probable que ocurran problemas.

3. Cobertura de Decisiones:

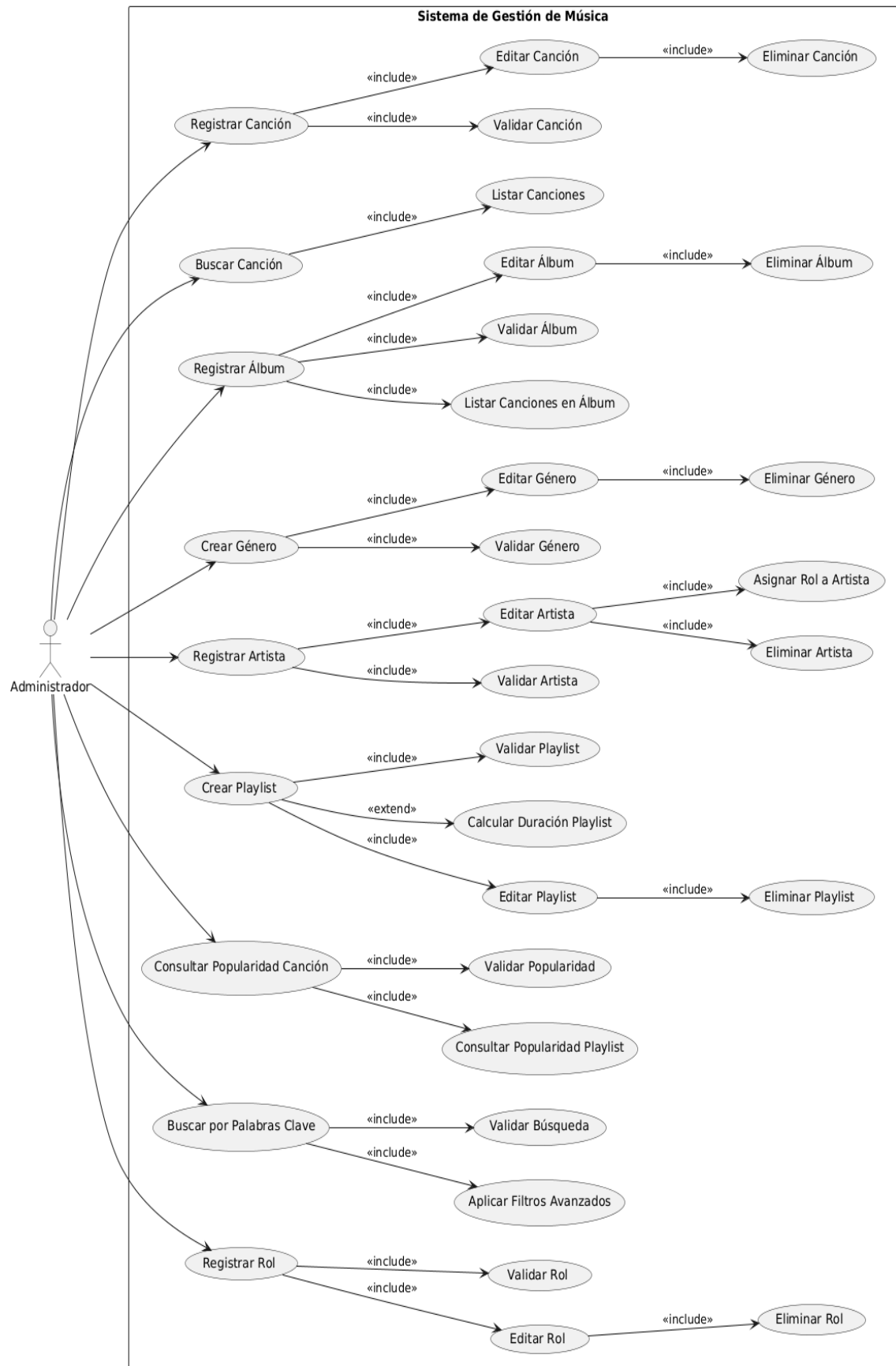
- a. Complementa las pruebas de caja negra al garantizar que se evalúe la lógica interna del sistema.
- b. Proporciona confianza adicional en la estabilidad y confiabilidad del código.

4. Cobertura de Camino:

- a. Garantiza que todas las rutas de ejecución posibles sean probadas.
- b. Identifica y valida casos especiales en el procesamiento de duraciones.

Estas técnicas, combinadas, ofrecen una cobertura completa para garantizar que el sistema funcione correctamente desde la perspectiva del usuario y de la lógica interna, maximizando la calidad y confiabilidad de Musicly.

8. Casos de Uso



9. Diseño de Clases

1. Clase Canción

- **Descripción:** Representa una canción en el sistema, incluyendo detalles como su título, duración, género y álbum asociado.
- **Atributos:**

@Entity

```
public class Cancion {
```

```
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;
```

```
    @Column(length = 100, nullable = false)
```

```
    private String titulo;
```

```
    private Integer duracion; // Duración en segundos
```

```
    @ManyToOne
```

```
    private Genero genero;
```

```
    @ManyToOne
```

```
    private Album album;
```

```
    @ManyToMany
```

```
    private List<Artista> artistas;
```

```
}
```

- **Relaciones:**
 - @ManyToOne: Relación con un único Género y Álbum.
 - @ManyToMany: Relación con múltiples Artistas.

2. Clase Álbum

- **Descripción:** Representa un álbum musical, agrupando canciones y vinculándose a un artista principal.

- **Atributos:**

@Entity

```
public class Album {
```

```
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
```

```

private Long id;

@Column(length = 100, nullable = false)

private String nombre;

private LocalDate fechaLanzamiento;

@ManyToOne

private Artista artistaPrincipal;

@OneToMany(mappedBy = "album")

private List<Cancion> canciones;
}

```

- **Relaciones:**

- @ManyToOne: Relación con un único Artista.
- @OneToMany: Relación con múltiples Canciones.

3. Clase Género

- **Descripción:** Define los géneros musicales dentro del sistema.

- **Atributos:**

```

@Entity

public class Genero {

    @Id @GeneratedValue(strategy = GenerationType.AUTO)

    private Long id;

    @Column(length = 50, unique = true, nullable = false)

    private String nombre;

    private String descripcion;
}

```

- **Relaciones:**

- Se vincula con la clase Cancion mediante una relación @OneToMany.

4. Clase Artista

- **Descripción:** Representa a los artistas musicales, permitiendo su asociación con canciones y álbumes.

- **Atributos:**

@Entity

```
public class Artista {
```

```
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;
```

```
    @Column(length = 100, nullable = false)
```

```
    private String nombre;
```

```
    @OneToMany(mappedBy = "artista")
```

```
    private List<Album> albumes;
```

```
    @OneToOne(mappedBy = "artista" , cascade = CascadeType.ALL,
    orphanRemoval = true)
```

```
    private Role rol;
```

```
}
```

- **Relaciones:**

- @OneToMany: Relación con múltiples Álbumes.
- @OneToOne: Relación con la clase Rol, un artista tiene un rol

5. Clase Playlist

- 1) **Descripción:** Permite a los usuarios crear listas de reproducción personalizadas.

- 2) **Atributos:**

@Entity

```
public class Playlist {
```

```
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;
```

```
    @Column(length = 100, nullable = false)
```

```
    private String nombre;
```

```
    @ManyToMany
```

```

private List<Cancion> canciones;

public Integer getDuracionTotal() {
    return canciones.stream().mapToInt(Cancion::getDuracion).sum();
}
}

```

3) **Relaciones:**

4) @ManyToMany: Relación con múltiples Canciones.

5) **Métodos:**

6) getDuracionTotal(): Calcula la duración total de la playlist.

6. Clase Role

- **Descripción:** Define roles específicos para los artistas en álbumes y canciones, como vocalista o productor.

- **Atributos:**

@Entity

```
public class Role {
```

```
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;
```

```
    @Column(length = 50, nullable = false)
```

```
    private String descripcion;
```

```
    @ManyToOne
```

```
    private Artista artista;
```

```
}
```

- **Relaciones:**

- @ManyToOne: Relación con la clase Artista.

Optimización del Diseño para el Time-to-Market

El enfoque Time-to-Market se ha priorizado en la implementación de este diseño mediante:

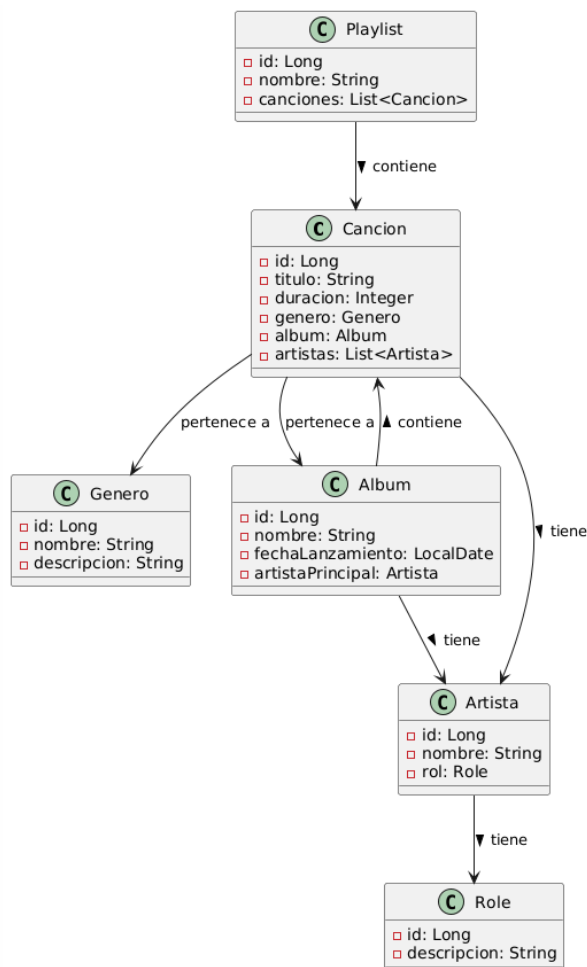
- **Automatización con OpenXava:**

- Generación automática de interfaces CRUD a partir de las entidades JPA, reduciendo significativamente el tiempo de desarrollo.

- Formularios preconfigurados para cada entidad, permitiendo una rápida interacción con la base de datos.
- **Conexión simplificada con MySQL:**
 - Configuración centralizada en context.xml para conexión rápida a la base de datos sin necesidad de configuraciones complejas.
- **Manejo eficiente de relaciones:**
 - Uso de anotaciones como @ManyToOne, @OneToMany y @ManyToMany para definir fácilmente las relaciones entre entidades.
- **Escalabilidad y mantenibilidad:**
 - Diseño modular que permite agregar nuevas entidades o mejorar la funcionalidad sin afectar la estructura existente.
- **Validaciones automáticas:**
 - Uso de anotaciones como @Column(nullable=false, unique=true) para evitar inconsistencias de datos y garantizar la calidad desde el inicio.

10. Descripción de implementación

10.1. Diagrama de Datos



10.2. Herramientas y Plataformas Utilizadas

Para la implementación de Musicly, se utilizaron las siguientes herramientas y plataformas:

Herramienta/Plataforma	Descripción
OpenXava	Framework de desarrollo rápido en Java que genera interfaces y operaciones CRUD automáticamente.
Java (JDK 17)	Lenguaje de programación utilizado para la lógica del negocio del sistema.
MySQL	Sistema de gestión de base de datos relacional para almacenar la información del sistema.
Maven	Herramienta de gestión de dependencias y construcción del proyecto.
Git/GitHub	Control de versiones para gestión colaborativa del código.

10.3. Descripción de la Implementación

La implementación del sistema se realizó en varias fases, asegurando la correcta integración de todas las funcionalidades esenciales:

Fase 1: Configuración del Proyecto

- Se configuró el entorno de desarrollo con OpenXava, asegurando la compatibilidad con MySQL.
- Se editaron los archivos de configuración principales:
 - pom.xml para incluir dependencias necesarias.
 - context.xml para establecer la conexión con la base de datos.
- Se ejecutó mvn install para descargar las dependencias y verificar la configuración.

Fase 2: Creación del Modelo de Datos

- Se definieron las entidades utilizando anotaciones de JPA:
 - @Entity, @Id, @GeneratedValue, @ManyToOne, @OneToMany, @ManyToMany.
- Se diseñaron las clases Cancion, Album, Genero, Artista, Playlist, Role, asegurando relaciones correctas y restricciones de negocio.

Fase 3: Implementación de Funcionalidades CRUD

- Se generaron automáticamente las interfaces para la gestión de las entidades usando OpenXava.

- Se probaron las operaciones de creación, actualización, eliminación y consulta para garantizar la integridad de los datos.

Fase 4: Desarrollo de la Lógica de Negocio

- Se implementaron métodos específicos, como el cálculo automático de la duración de una playlist.
- Se añadieron validaciones adicionales en los formularios para mejorar la experiencia del usuario.

Fase 5: Pruebas y Validación

- Se realizaron pruebas unitarias para verificar la funcionalidad de cada módulo.
- Se aplicaron pruebas de integración para evaluar el correcto funcionamiento del sistema con la base de datos.

Fase 6: Despliegue

- El sistema fue desplegado accediendo directamente mediante un navegador en localhost.

4. Resultados Obtenidos

La implementación del sistema Musicly ha dado como resultado:

- **Interfaz Generada Automáticamente:** A partir de las clases de dominio, se han generado formularios funcionales sin necesidad de desarrollo manual.
- **Gestión Completa de la Biblioteca Musical:** Los usuarios pueden registrar, editar y eliminar canciones, álbumes, géneros y playlists de manera sencilla.
- **Conexión Exitosa con MySQL:** Los datos se almacenan de manera eficiente y cumplen con las reglas de negocio definidas.
- **Plazo de Entrega Cumplido:** Gracias al uso de herramientas de desarrollo rápido como OpenXava, se ha optimizado el tiempo de implementación.

11. Casos de Prueba Técnica 1

Prueba de Caja Blanca (Cobertura de Caminos)

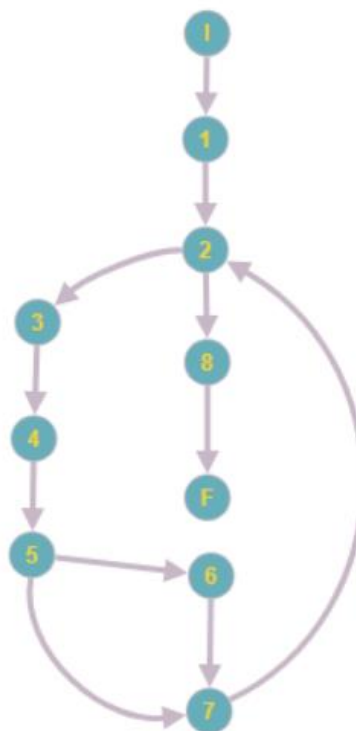
Este análisis se centra en la evaluación del método `getDuracionTotalEnSegundos()`, que forma parte del sistema de gestión de canciones. La técnica de cobertura de caminos se aplica para garantizar que todas las rutas de ejecución posibles sean validadas, asegurando la precisión en el cálculo de la duración total de una lista de canciones.

Descripción del Método: El método `getDuracionTotalEnSegundos()` cumple la función de calcular la duración total en segundos de una colección de canciones, realizando las siguientes operaciones:

- Conversión de duraciones de formato decimal a minutos y segundos
- Validación y ajuste de segundos cuando exceden 60
- Acumulación del tiempo total
- Registro de información detallada mediante logs

```
public int getDuracionTotalEnSegundos() {  
    int total = 0;  
    for (Cancion cancion : canciones) {  
        float duracion = cancion.getDuracion();  
        int minutos = (int) Math.floor(duracion);  
        int segundos = Math.round((duracion - minutos) * 100);  
        log.info("Canción: " + cancion.getTitulo() + ", Duración: " + duracion + " minutos (" + minutos + "m " + segundos + "s");  
  
        // Validación adicional para asegurar que los segundos no excedan 59  
        if (segundos >= 60) {  
            minutos += segundos / 60;  
            segundos = segundos % 60;  
            log.warn("Duración corregida: " + minutos + "m " + segundos + "s");  
        }  
  
        total += minutos * 60 + segundos;  
    }  
    log.info("Duración total en segundos calculada: " + total);  
    return total;  
}
```

Grafo de control



Estructura del Grafo de Control:

Descripción de los nodos:

- Nodo I: Inicio del método
- Nodo 1: Inicialización de total = 0
- Nodo 2: Evaluación del bucle for (¿hay más canciones?)
- Nodo 3: Obtener duración de la canción
- Nodo 4: Calcular minutos y segundos
- Nodo 5: Evaluación de la condición (segundos \geq 60)
- Nodo 6: Ajuste de minutos y segundos + log warn
- Nodo 7: Actualización del total
- Nodo 8: Log info del total
- Nodo F: Retorno del total y fin del método

Aristas del Grafo:

- De I a 1: Ingreso al método
- De 1 a 2: Inicio del bucle
- De 2 a 3: Iteración del bucle
- De 3 a 4: Flujo de cálculo
- De 4 a 5: Evaluación de segundos
- De 5 a 6: Segundos \geq 60
- De 5 a 7: Segundos $<$ 60
- De 6 a 7: Después de ajuste
- De 7 a 2: Retorno al bucle
- De 2 a 8: Fin del bucle
- De 8 a F: Fin del método

Complejidad Ciclomática:

La complejidad del método se determina mediante tres métricas que coinciden:

- Fórmula de McCabe: $V = 12 \text{ aristas} - 11 \text{ nodos} + 2 = 3$
- Número de nodos predicado + 1 = $2 + 1 = 3$ (bucle for y condición if)
- Número de regiones = 3

Caminos Independientes:

Se identifican tres caminos básicos que deben ser probados:

1. Lista vacía de canciones: $I \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow F$
2. Canciones con segundos $<$ 60: $I \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 2 \rightarrow 8 \rightarrow F$
3. Canciones con segundos \geq 60: $I \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 2 \rightarrow 8 \rightarrow F$

La implementación de pruebas que cubran estos caminos asegurará:

- Manejo correcto de listas vacías
- Cálculo preciso de duraciones estándar
- Ajuste adecuado cuando los segundos exceden 60
- Acumulación correcta del tiempo total
- Registro apropiado de información en los logs

Casos de prueba

	Entrada	Ejecución Esperada	Camino Cubierto
Caso 1	Lista vacía de canciones	Se inicializa total = 0, no se ejecuta el bucle y se retorna 0. El log muestra "Duración total en segundos calculada: 0"	I - 1 - 2 - 8 - F
Caso 2	Una canción con duración 2.30 (2 minutos, 30 segundos)	Se procesa la canción, los segundos son < 60, se acumula el total (150 segundos) y se retorna. El log muestra la duración original y el total	I - 1 - 2 - 3 - 4 - 5 - 7 - 2 - 8 - F
Caso 3	Una canción con duración 2.65 (2 minutos, 65 segundos)	Se procesa la canción, los segundos son ≥ 60, se ajusta a 3:05, se registra warning, se acumula el total (185 segundos) y se retorna	I - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 2 - 8 - F

Resultados de las Pruebas

Los tres casos de prueba garantizan una cobertura completa de caminos para este algoritmo, validando:

1. El manejo correcto de listas vacías
2. El procesamiento normal de duraciones sin ajuste de segundos
3. El ajuste automático cuando los segundos exceden 60

El análisis de cobertura de caminos para el método `getDuracionTotalEnSegundos()` confirma que todos los flujos posibles han sido evaluados mediante pruebas específicas, asegurando su correcto comportamiento en las siguientes condiciones:

- Entrada vacía
- Procesamiento de duraciones estándar
- Procesamiento de duraciones que requieren ajuste
- Registro correcto de logs en cada situación

12. Casos de Prueba Técnica 2

Casos de Prueba Técnica 2 - Caja Negra Cobertura de Valores Límites

El enfoque de la cobertura de valores límites es realizar pruebas en los puntos justo por debajo y por encima de los valores límite esperados. A continuación, se presentan los casos de prueba para validar las entidades del sistema de gestión musical.

El enfoque de la cobertura de valores límites es realizar pruebas en los puntos justo por debajo y por encima de los valores límite esperados. Para la entidad 'Album', los límites a evaluar son:

- **Fecha de lanzamiento:** No puede ser una fecha futura.
- **Lista de artistas:** Debe contener al menos un artista.

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado	Observaciones
TC 1	fecha válida	2023-01-01	True	Fecha de lanzamiento válida.
TC 2	fecha futura	2030-05-10	False	La fecha no puede ser futura.
TC 3	artistas vacíos	[]	False	Debe haber al menos un artista.
TC 4	artistas válidos	[Artista1, Artista2]	True	Mínimo un artista requerido.

Para la entidad 'Cancion', la cobertura de valores límites se centra en:

- **Duración:** Debe ser entre 1 y 60 minutos, y los segundos no deben superar los 59.
- **Título:** No debe exceder los 100 caracteres y no debe estar vacío.

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado	Observaciones
TC 3	duración válida	5.30	True	Duración dentro del rango válido.
TC 4	duración negativa	-3.00	False	La duración no puede ser negativa.
TC 5	duración excesiva	61.00	False	La duración no puede superar 60 min.
TC 6	segundos inválidos	5.60	False	Los segundos no pueden exceder 59.

Para la entidad 'Genero', la validación se centra en:

- **Descripción:** No debe estar vacía y no debe exceder los 50 caracteres.

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado	Observaciones
TC 1	descripción válida	Rock	True	Descripción dentro del límite.
TC 2	descripción vacía	“”	False	No puede estar vacía.
TC 3	descripción muy larga	XXXXXXXXXXXXXXXXXXXXX X	False	Excede la longitud permitida.

Para la entidad 'Role', se verifica la validez de la descripción asegurando que:

- **Descripción:** Esté dentro de los límites de longitud adecuados y no sea vacía.

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado	Observaciones
TC 1	descripción válida	Vocalista	True	Descripción dentro del límite.
TC 2	descripción vacía	“”	False	No puede estar vacía.
TC 3	descripción muy larga	XXXXXXXXXXXXX	False	Excede la longitud permitida.

13. Casos de Prueba Técnica 3

Casos de Prueba Técnica 3 - Caja Negra Cobertura en Equivalencia

Se ha diseñado un conjunto de pruebas basado en clases de equivalencia, aplicando la técnica de caja negra para validar restricciones específicas dentro del sistema de gestión musical. El objetivo es garantizar que los datos ingresados en las distintas entidades del sistema cumplan con los requisitos establecidos en términos de longitud, formato y reglas de negocio.

A continuación, se presenta una descripción detallada de los casos de prueba diseñados para evaluar el cumplimiento de estas restricciones.

Validación de la entidad Álbum

El objetivo de estas pruebas es asegurar que la información de los álbumes sea válida de acuerdo con las siguientes reglas de negocio:

- El nombre debe tener una longitud entre 1 y 100 caracteres.
- La fecha de lanzamiento debe ser una fecha pasada o presente.
- Debe haber al menos un artista asociado al álbum.

Casos de prueba:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: Nombre válido	"Mi Álbum"	True
2	EC2: Nombre inválido	""	False
3	EC3: Nombre muy largo	"A".repeat(101)	False
4	EC4: Fecha válida	2023-01-01	True
5	EC5: Fecha inválida	2030-05-10	False
6	EC6: Artistas vacíos	[]	False

Luego de tener los casos de prueba, se analiza con el caso de equivalencia para poder observar qué condiciones se necesitan cumplir para que el caso de prueba se apruebe, donde:

Las clases de equivalencia se agrupan en dos categorías principales:

Valores Numéricos: Representan valores válidos e inválidos relacionados con el álbum.

- EC1: Nombre dentro de los límites establecidos (válido).
- EC2: Nombre vacío (no válido).

- EC3: Nombre demasiado largo (no válido).

Valores No Reales: Representan datos no válidos.

- EC4: Fecha de lanzamiento válida (válido).
- EC5: Fecha de lanzamiento futura (no válido).
- EC6: Artistas vacíos (no válido).

Matriz de Cobertura de la entidad Album:

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3	TC 4	TC 5
Nombre	EC1: Nombre válido	Válido	"Mi Álbum"	*				
	EC2: Nombre inválido	No válido	""		*			
	EC3: Nombre muy largo	No válido	"A".repeat(101)			*		
Fecha Lanzamiento	EC4: Fecha válida	Válido	2023-01-01				*	
	EC5: Fecha inválida	No válido	2030-05-10					*
Artistas	EC6: Lista vacía	No válido	[]					*

Validación de la entidad Cancion

El objetivo de estas pruebas es verificar que la información de las canciones sea válida de acuerdo con las siguientes reglas de negocio:

- La duración debe estar entre 1 y 60 minutos.
- El título debe estar entre 1 y 100 caracteres.
- La popularidad debe ser un número positivo.

Casos de prueba:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
----------------	-----------------------	---------------	--------------------

1	EC1: Duración válida	5.30	True
2	EC2: Duración negativa	-3.00	False
3	EC3: Duración excedida	61.00	False
4	EC4: Título válido	"Mi Canción"	True
5	EC5: Título vacío	""	False
6	EC6: Popularidad negativa	-50	False

Las clases de equivalencia se agrupan en dos categorías principales:

Valores Numéricos: Representan valores válidos e inválidos relacionados con la canción.

- EC1: Duración válida (válido).
- EC2: Duración negativa (no válido).
- EC3: Duración excedida (no válido).

Valores No Reales: Representan datos no válidos.

- EC4: Título válido (válido).
- EC5: Título vacío (no válido).
- EC6: Popularidad negativa (no válido).

Matriz de Cobertura de la entidad Canción:

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3	TC 4	TC 5
Duración	EC1: Duración válida	Válido	5.30	*				
	EC2: Duración negativa	No válido	-3.00		*			
	EC3: Duración excedida	No válido	61.00			*		
Título	EC4: Título válido	Válido	"Mi Canción"				*	
	EC5: Título vacío	No válido	""					*
Popularidad	EC6: Popularidad negativa	No válido	-50					*

Validación de la entidad Playlist

El objetivo de estas pruebas es asegurar que la información de las playlists sea válida de acuerdo con las siguientes reglas de negocio:

- El nombre debe tener una longitud entre 1 y 100 caracteres.
- Debe haber al menos una canción asociada a la playlist.

Casos de prueba:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: Nombre válido	"Mi Playlist"	True
2	EC2: Nombre vacío	""	False
3	EC3: Nombre con caracteres especiales	"@Playlist!"	False
6	EC6: Lista de canciones vacía	[]	False

Las clases de equivalencia se agrupan en dos categorías principales:

Valores Numéricos: Representan valores válidos e inválidos relacionados con la playlist.

- **EC1:** Nombre dentro del límite permitido (válido).
- **EC2:** Nombre vacío (no válido).
- **EC6:** Lista de canciones vacía (no válido).

Matriz de Cobertura de la entidad Playlist:

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3
Nombre	EC1: Nombre válido	Válido	"Mi Playlist"	*		
	EC2: Nombre vacío	No válido	""		*	
Canciones	EC6: Lista de canciones vacía	No válido	[]			*

Validación de la entidad Genero

El objetivo de estas pruebas es asegurar que la información de los géneros musicales sea válida de acuerdo con las siguientes reglas de negocio:

- La descripción no debe estar vacía.
- La descripción no debe exceder los 50 caracteres.

Casos de prueba:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: Descripción válida	"Rock"	True
2	EC2: Descripción vacía	""	False
3	EC3: Descripción larga	izk[3U!RD(fV-[Bg5RjpVL(y8+9@M,4RWmah=h!/N&Y?&ZMJ+R\$	False

14.Casos de Prueba Técnica 4

ID	Nombre	Descripción	Descripción	Pasos	Resultado esperado	Resultado Actual	Estado	Referencias
CP_REGCAN1	Registrar Canción con campos vacíos	Registrar una canción dejando todos los campos obligatorios vacíos	El usuario debe estar autenticado como administrador	1. Seleccionar "Registrar Canción" en el menú principal. 2. Dejar todos los campos vacíos. 3. Presionar "Guardar". 4. Seleccionar "Registrar Canción". 5. Ingresar título válido. 6. Ingresar duración: 5 segundos. 7. Completar otros campos válidos. 8. Presionar "Guardar".	Mostrar notificaciones solicitando completar todos los campos	Notificaciones mostradas correctamente.	Aprobado	Caso de Uso [CU-01]
CP_REGCAN2	Registrar Canción con duración inválida	Registrar una canción con duración negativa	El usuario debe estar autenticado como administrador	1. Seleccionar "Registrar Canción". 2. Ingresar título: "Nueva Canción". 3. Ingresar duración: 3:45. 4. Seleccionar género existente. 5. Seleccionar álbum existente. 6. Seleccionar artista existentes. 7. Presionar "Guardar".	Mostrar mensaje de error indicando que la duración debe ser mayor a 0	Mensaje de error mostrado correctamente.	Aprobado	Caso de Uso [CU-01]
CP_REGCAN3	Registrar Canción con datos válidos	Registrar una canción con todos los datos correctos, menos uno	El usuario debe estar autenticado como administrador	1. Seleccionar "Registrar Canción". 2. Ingresar título válido. 3. Ingresar duración válida. 4. Presionar "Guardar".	La canción se registra exitosamente pero no es visible en futuras búsquedas	La canción no se registró correctamente.	No Aprobado	Caso de Uso [CU-01]
CP_REGCAN4	Registrar Canción sin género	Registrar una canción sin género	El usuario debe estar autenticado como administrador	1. Seleccionar una canción existente para editar. 2. Dejar el campo "Título" vacío. 3. Presionar "Guardar Cambios".	Mostrar mensaje de error indicando que falta el género	Mensaje de error mostrado correctamente.	Aprobado	Caso de Uso [CU-01]
CP_EDITCAN1	Editar Canción con campos obligatorios vacíos	Editar una canción dejando algunos campos obligatorios vacíos	La canción debe existir en el sistema	1. Seleccionar "Registrar Álbum" en el menú principal. 2. Dejar todos los campos vacíos. 3. Presionar "Guardar".	Mostrar notificación solicitando completar el campo "Título"	Notificación mostrada correctamente.	Aprobado	Caso de Uso [CU-02]
CP_REGALB1	Registrar Álbum con campos vacíos	Registrar un álbum dejando todos los campos obligatorios vacíos	El usuario debe estar autenticado como usuario	1. Seleccionar "Registrar Álbum". 2. Ingresar nombre válido. 3. Ingresar fecha de lanzamiento válida.	Mostrar notificaciones solicitando completar todos los campos	Notificaciones mostradas correctamente.	Aprobado	Caso de Uso [CU-04]
CP_REGALB2	Registrar Álbum sin artista	Registrar un álbum sin asociar artista	El usuario debe estar autenticado como usuario	1. Seleccionar "Registrar Álbum". 2. Ingresar nombre: "Nuevo Álbum". 3. Ingresar fecha de lanzamiento: "2025-01-01". 4. Asociar artista existente.	Mostrar mensaje de error indicando que el artista no existe	Mensaje de error mostrado correctamente.	Aprobado	Caso de Uso [CU-04]
CP_REGALB3	Registrar Álbum con datos válidos	Registrar un álbum con todos los datos correctos	El usuario debe estar autenticado como usuario	1. Seleccionar un álbum existente para editar. 2. Dejar el campo "Nombre" vacío. 3. Presionar "Guardar Cambios".	El álbum se registra exitosamente y está disponible para futuras búsquedas	El álbum fue registrado correctamente.	Aprobado	Caso de Uso [CU-04]
CP_EDITALB1	Editar Álbum con campos obligatorios vacíos	Editar un álbum dejando algunos campos obligatorios vacíos	El álbum debe estar registrado previamente en el sistema	1. Seleccionar "Gestionar Álbumes" en el menú principal. 2. Elegir un álbum existente. 3. Presionar "Eliminar".	Mostrar notificación solicitando completar el campo "Nombre"	Notificación mostrada correctamente.	Aprobado	Caso de Uso [CU-05]
CP_ELIMALB1	Eliminar Álbum exitosamente	Eliminar un álbum existente correctamente	El álbum debe estar registrado en el sistema	4. Confirmar la eliminación en la ventana emergente. 5. Seleccionar "Gestionar Álbumes". 6. Intentar eliminar un álbum con un ID inexistente.	El álbum se elimina del sistema y ya no está visible en la lista.	Álbum eliminado correctamente.	Aprobado	Caso de Uso [CU-06]
CP_ELIMALB2	Eliminar álbum inexistente	Intentar eliminar un álbum que no existe en el sistema	-	1. Seleccionar "Gestionar Álbumes". 2. Elegir un álbum asociado a canciones. 3. Presionar "Eliminar". 4. Confirmar la eliminación.	Mostrar mensaje de error indicando que el álbum no existe	Mensaje de error mostrado correctamente.	Aprobado	Caso de Uso [CU-06]
CP_ELIMALB4	Eliminar álbum asociado a canciones	Intentar eliminar un álbum que está asociado a canciones	El álbum debe estar asociado a al menos una canción	1. Seleccionar "Gestionar Géneros" en el menú principal. 2. Presionar "Agregar Género". 3. Ingresar nombre válido.	Mostrar advertencia indicando que el álbum está asociado a canciones y solicitar confirmación adicional o reasignación.	Advertencia mostrada correctamente.	Aprobado	Caso de Uso [CU-06]
CP_GENGEN1	Agregar género con datos válidos	Agregar un nuevo género con todos los datos completos	El usuario debe estar autenticado y autorizado	1. Seleccionar "Gestionar Géneros". 2. Presionar "Agregar Género". 3. Dejar el campo "Nombre" vacío.	El género se agrega correctamente y aparece en la lista de géneros.	Género agregado correctamente.	Aprobado	Caso de Uso [CU-07]
CP_GENGEN2	Agregar género sin nombre	Intentar agregar un género dejando el campo "Nombre" vacío	El usuario debe estar autenticado y autorizado	4. Presionar "Guardar". 5. Seleccionar "Gestionar Géneros". 6. Presionar "Agregar Género". 7. Ingresar nombre duplicado.	Mostrar notificación solicitando completar el campo "Nombre"	Notificación mostrada correctamente.	Aprobado	Caso de Uso [CU-07]
CP_GENGEN3	Agregar género duplicado	Intentar agregar un género con un nombre que ya existe	El género a duplicar debe estar registrado	1. Seleccionar "Gestionar Géneros". 2. Presionar "Agregar Género". 3. Ingresar nombre duplicado.	Mostrar mensaje de error indicando duplicidad del género.	Mensaje de error mostrado correctamente.	Aprobado	Caso de Uso [CU-07]
CP_GENGEN4	Editar género con datos válidos	Editar un género existente con datos correctos	El género debe estar registrado en el sistema	1. Seleccionar "Gestionar Géneros". 2. Elegir un género existente. 3. Presionar "Editar". 4. Modificar el nombre.	El género se actualiza correctamente en la lista.	Género editado correctamente.	Aprobado	Caso de Uso [CU-07]
CP_GENGEN5	Editar género dejando campos obligatorios vacíos	Intentar editar un género dejando el campo "Nombre" vacío	El género debe estar registrado en el sistema	5. Presionar "Guardar Cambios". 6. Seleccionar "Gestionar Géneros". 7. Elegir un género existente. 8. Presionar "Editar".	Mostrar notificación solicitando completar el campo "Nombre"	Notificación mostrada correctamente.	Aprobado	Caso de Uso [CU-07]
CP_GENGEN6	Editar género a uno duplicado	Intentar editar un género cambiando su nombre a uno ya existente	El género de destino debe estar registrado	1. Seleccionar "Gestionar Géneros". 2. Elegir un género existente. 3. Presionar "Editar". 4. Cambiar el nombre a uno duplicado.	Mostrar mensaje de error indicando duplicidad del género.	Mensaje de error mostrado correctamente.	Aprobado	Caso de Uso [CU-07]
CP_GENGEN7	Eliminar género no asociado a canciones	Eliminar un género que no está asociado a ninguna canción	El género no debe estar asociado a canciones	5. Presionar "Guardar Cambios". 6. Seleccionar "Gestionar Géneros". 7. Elegir un género no asociado. 8. Presionar "Eliminar".	El género se elimina correctamente y ya no aparece en la lista.	Género eliminado correctamente.	Aprobado	Caso de Uso [CU-07]
CP_EDITART1	Editar artista con datos completos	Editar un artista existente con todos los campos llenos	El artista debe estar registrado en el sistema	1. Seleccionar "Gestionar Artistas" en el menú principal. 2. Elegir un artista existente. 3. Presionar "Editar".	El artista se actualiza correctamente en la lista de artistas.	Artista editado correctamente.	Aprobado	Caso de Uso [CU-09]
CP_EDITART2	Editar artista dejando campos obligatorios vacíos	Intentar editar un artista dejando campos obligatorios vacíos	El artista debe estar registrado en el sistema	4. Modificar los campos "Nombre", "Rol" y/o "Biografía". 5. Presionar "Guardar Cambios". 6. Seleccionar "Gestionar Artistas". 7. Elegir un artista existente.	Mostrar notificación solicitando completar los campos obligatorios.	Notificación mostrada correctamente.	Aprobado	Caso de Uso [CU-09]
CP_CREPLAY1	Crear playlist con datos completos	Crear una playlist con nombre semejante y canciones válidas	Las canciones deben estar registradas en el sistema	1. Seleccionar "Gestionar Artistas". 2. Elegir un artista existente. 3. Presionar "Editar". 4. Dejar uno o más campos obligatorios vacíos.	La playlist se crea correctamente y aparece en la lista de playlists.	Playlist creada correctamente.	No Aprobado	Caso de Uso [CU-10]
CP_CREPLAY2	Crear playlist dejando campos obligatorios vacíos	Intentar crear una playlist dejando el campo "Nombre" vacío	Las canciones deben estar registradas en el sistema	5. Presionar "Guardar Cambios". 6. Seleccionar "Crear Playlist". 7. Dejar el campo "Nombre" vacío.	Mostrar notificación solicitando completar el campo "Nombre"	Notificación mostrada correctamente.	Aprobado	Caso de Uso [CU-10]
CP_EDITPLAY1	Editar Playlist con campos vacíos	Editar una playlist dejando algunos campos obligatorios vacíos	La playlist debe estar registrada previamente	1. Seleccionar "Gestionar canciones". 2. Presionar "Editar". 3. Seleccionar una playlist. 4. No completar los campos obligatorios.	El sistema solicita completar los campos obligatorios.	Notificaciones mostradas correctamente.	Aprobado	Caso de Uso [CU-11]
CP_DELPLAY1	Eliminar Playlist existente	Eliminar una playlist existente en el sistema	La playlist debe estar registrada previamente	1. Seleccionar "Gestionar canciones". 2. Confirmar la eliminación. 3. Validar que fue eliminada.	La playlist se elimina correctamente.	Playlist eliminada correctamente.	Aprobado	Caso de Uso [CU-12]
CP_FILTER2	Filtrar resultados sin coincidencias	Aplicar filtros que generan coincidencias	El usuario debe haber realizado una búsqueda previa	2. Seleccionar filtros sin coincidencias. 3. Confirmar.	El sistema muestra un mensaje indicando que no hay coincidencias	Mensaje de "sin coincidencias" mostrado correctamente.	No Aprobado	Caso de Uso [CU-14]

CP_REGCAN1: Registrar Canción con campos vacíos

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Todos los campos vacíos.
- **Resultado Esperado:** El sistema identifica que los campos obligatorios están vacíos y muestra notificaciones para completarlos.
- **Estado:** APROBADO

The screenshot shows a web application interface for registering a song. The form includes fields for 'Titulo', 'Duracion', 'Genero', 'Descripción', 'Album' (with sub-fields 'Nombre' and 'Fecha lanzamiento'), 'Artista principal' (with sub-fields 'Nombre' and 'Biografía'), and 'Roles'. At the top, there are two red error messages: 'Es obligatorio que Album en Cancion tenga valor' and 'Es obligatorio que Genero en Cancion tenga valor'. The interface also features a navigation bar with tabs for 'Cancion', 'Genero', 'Artista', 'Album', and 'Roles', and a toolbar with buttons for 'Lista', 'Nuevo', 'Grabar', and 'Refrescar'. At the bottom, there is a table with columns 'Id' and 'Descripción', and a pagination control showing '10' items per page.

2. CP_REGCAN2: Registrar Canción con duración inválida

- **Tipo de Prueba:** Caja Negra (Valores Límite)
- **Clases de Equivalencia:**
 - **Valores Límite:** Duración mínima válida es 0 segundos.
- **Resultado Esperado:** El sistema muestra un mensaje de error indicando que la duración debe ser un valor positivo.
- **Estado:** APROBADO

The screenshot shows the same web application interface as before, but with the 'Duracion' field set to '-1'. A red error message at the top states: 'Ha sido imposible ejecutar la acción Grabar. La duración debe ser un valor positivo mayor a 0.' The other fields are filled with test data: 'Titulo' is 'Eventually', 'Descripción' is 'pop', 'Album' 'Nombre' is 'Lolabum', and 'Artista principal' 'Nombre' is 'Pancho'. The interface elements (navigation bar, toolbar, table) are consistent with the previous screenshot.

3. CP_REGCAN3: Registrar Canción con datos válidos

- **Tipo de Prueba:** Caja Negra (Caso de Uso exitoso)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Casi todos los campos se llenan correctamente con datos válidos.
 - **Entrada Invalida:** Un campo falta de llenar correctamente
- **Resultado Esperado:** La canción no se va a registrar exitosamente y no está disponible para futuras búsquedas.
- **Estado:** NO APROBADO

Id	Título	Duración	Id de género	Género	Id de album	Album	Reproducciones	Popularidad
2	Eventually	4	2	pop	1	Lolabum	0	0
Σ		Σ	Σ	Σ	Σ		Σ	Σ

4.CP_REGCAN4: Registrar Canción sin Género asociado

- **Tipo de Prueba:** Caja Negra (Asociaciones inválidas)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** No asociar a un género registrado.
- **Resultado Esperado:** El sistema muestra un mensaje de error indicando que falta el género.
- **Estado:** APROBADO

Es obligatorio que Genero en Cancion tenga valor

Título: Redbone

Duración: 4

Genero: [empty]

Descripción: [empty]

Album: Lolabum

Fecha lanzamiento: [empty]

Artista principal: Pancho

Biografía: pancho

5. CP_EDITCAN1: Editar Canción con campos obligatorios vacíos

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios en edición)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Campo "Título" vacío al editar.
- **Resultado Esperado:** El sistema solicita al usuario completar el campo "Título".
- **Estado:** APROBADO

The screenshot shows a web application interface for editing a song. At the top, there are tabs for 'Cancion', 'Artista', 'Album', 'Genero', and 'Roles'. A red error message at the top right states: 'no es un valor válido para Título de Cancion: El título no puede estar vacío.' Below the tabs, there are navigation buttons: '< Lista', '<< >>', 'Nuevo', 'Grabar', and 'Refrescar'. The form fields include: 'Título' (empty), 'Duración' (with a calendar icon), 'Genero' (dropdown menu), 'Descripción' (text field with 'pop'), 'Album' (dropdown menu), 'Nombre' (text field with 'Lolabum'), and 'Fecha lanzamiento' (with a calendar icon).

6. CP_REGALB1: Registrar Álbum con campos vacíos

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Todos los campos vacíos.
- **Resultado Esperado:** El sistema identifica que los campos obligatorios están vacíos y muestra notificaciones para completarlos.
- **Estado:** APROBADO

The screenshot shows a web application interface for registering an album. At the top, there are tabs for 'Album', 'Cancion', 'Artista', 'Genero', and 'Roles'. A red error message at the top right states: 'Ha sido imposible ejecutar la acción Grabar: El nombre del álbum no puede estar vacío.' Below the tabs, there are navigation buttons: '< Lista', '<< >>', 'Nuevo', 'Grabar', and 'Refrescar'. The form fields include: 'Nombre' (empty), 'Fecha lanzamiento' (with a calendar icon), 'Artista principal' (dropdown menu), 'Nombre' (text field with 'Pancho'), 'Biografía' (text field with 'pancho'), and 'Roles' (dropdown menu).

7.-CP_REGALB2: Registrar Álbum con artista inexistente

- **Tipo de Prueba:** Caja Negra (Asociaciones inválidas)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** No asociar un artista .

- **Resultado Esperado:** El sistema muestra un mensaje de error indicando que el artista se requiere
- **Estado:** APROBADO

The screenshot shows a web application interface for creating an album. At the top, there is a navigation bar with tabs: Album, Cancion, Artista, Genero, and Roles. A red error message banner at the top right states: "Es obligado que Artista principal en Album tenga valor". Below the navigation bar, there is a toolbar with buttons: Lista, Nuevo, Grabar, and Refrescar. The form fields include:

- Nombre:** A text input field containing "Bocanada".
- Fecha lanzamiento:** A date picker showing "07/01/2025".
- Artista principal:** A dropdown menu with a search icon and a plus icon.
- Nombre:** A text input field for the main artist's name, which is currently empty and has a red border, indicating it is required.
- Biografia:** A text area for the artist's biography.
- Roles:** A dropdown menu for selecting roles.

8. CP_REGALB3: Registrar Álbum con datos válidos

- **Tipo de Prueba:** Caja Negra (Caso de Uso exitoso)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Todos los campos se llenan correctamente con datos válidos.
- **Resultado Esperado:** El álbum se registra exitosamente y está disponible para futuras asociaciones.
- **Estado:** APROBADO

The screenshot shows a web application interface for managing albums. At the top, there is a navigation bar with tabs: Album, Cancion, Artista, Genero, and Roles. Below the navigation bar, there is a toolbar with buttons: Nuevo, Borrar, Generar PDF, Generar Excel, and Importar datos. The table displays a list of albums with the following columns: Id, Nombre, Fecha lanzamiento, Id de artista principal, and Artista principal. The table contains two rows of data:

Id	Nombre	Fecha lanzamiento	Id de artista principal	Artista principal
1	Lolabum			1 Pancho
3	Bocanada	07/01/2025		1 Pancho

 The table also includes a summary row with a sigma symbol (Σ) and a footer showing "10 filas por página".

9. CP_EDITALB1: Editar Álbum con campos obligatorios vacíos

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios en edición)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Campo "Nombre" vacío al editar.
- **Resultado Esperado:** El sistema solicita al usuario completar el campo "Nombre".
- **Estado:** APROBADO

Album x Cancion x Artista x Genero x Roles x

Ha sido imposible ejecutar la acción Grabar: El nombre del álbum no puede estar vacío. x

< Lista < < > > Nuevo Grabar Refrescar Borrar

Nombre

Fecha lanzamiento 07/01/2025

Artista principal

Nombre Pancho

Biografía pancho

Roles

Generar PDF Generar Excel

10. CP_ELIMALB1: Eliminar Álbum exitosamente

- **Tipo de Prueba:** Caja Negra (Flujo normal de eliminación)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Seleccionar y eliminar un álbum existente.
- **Resultado Esperado:** Si el álbum no tiene ninguna asociación con una canción, se elimina correctamente y no aparece en la lista.
- **Estado:** Aprobado

Album x Roles x Playlist x Genero x Cancion x Artista x

Nuevo Borrar Generar PDF Generar Excel Importar datos

Todos

	Id	Nombre	Fecha lanzamiento	Id de artista principal	Artista principal
	1	EP	08/11/2024	1	Bruno Mars
	Σ			Σ	

Album x Cancion x Playlist x Roles x Genero x Artista x

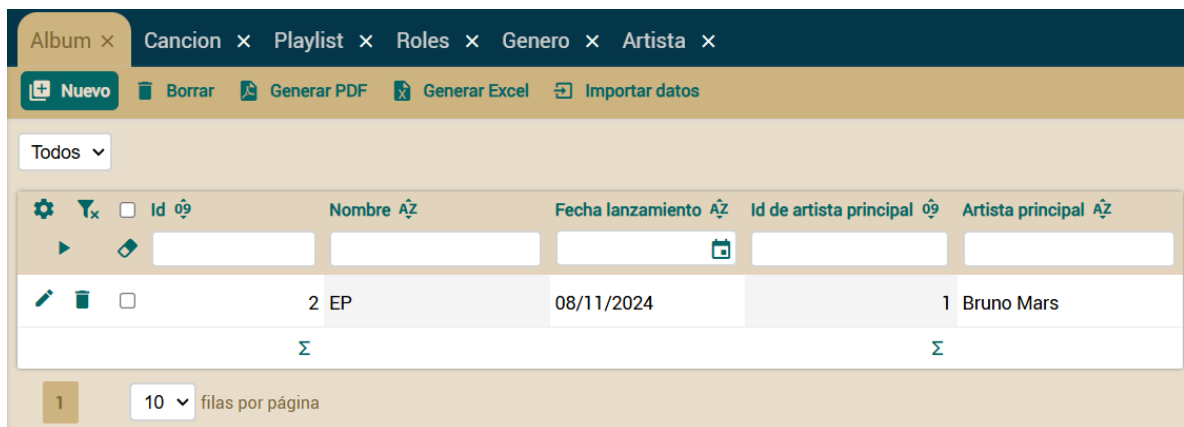
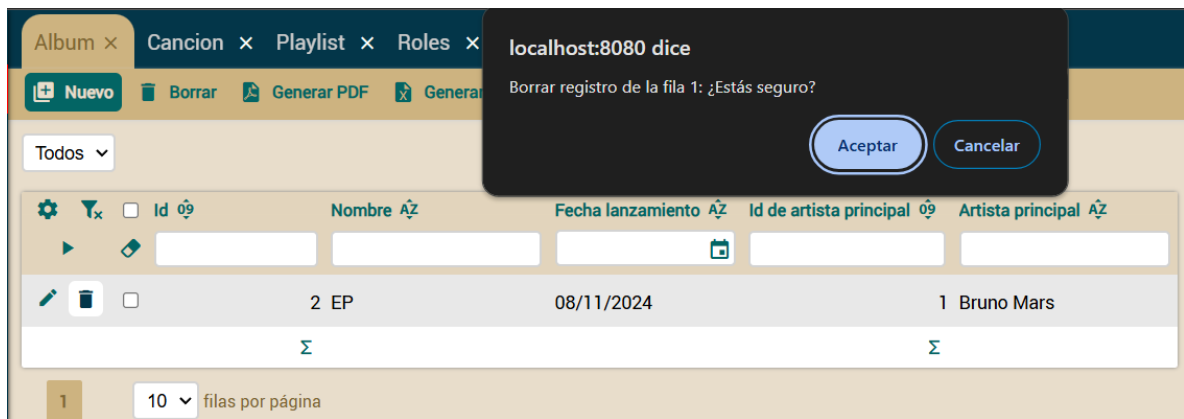
Nuevo Borrar Generar PDF Generar Excel Importar datos

Todos

	Id	Nombre	Fecha lanzamiento	Id de artista principal	Artista principal
No hay registros					
1	10	filas por página			

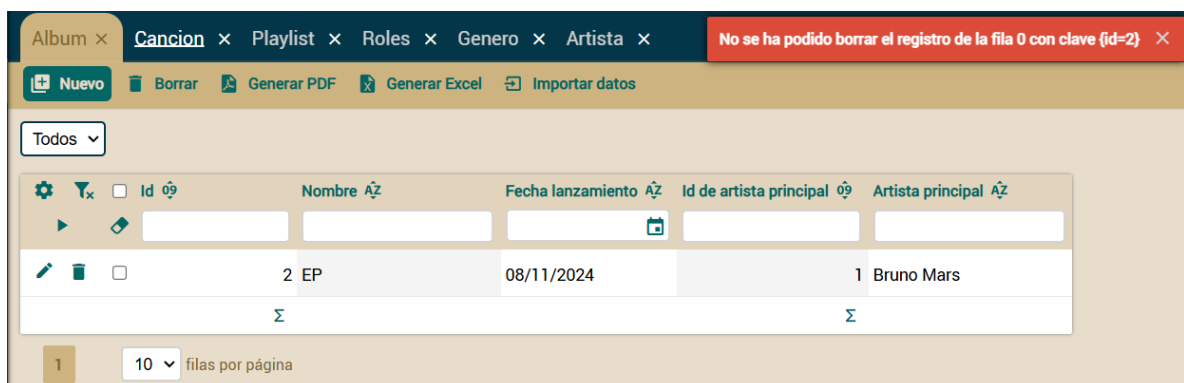
11. CP_ELIMALB2: Cancelar eliminación de álbum

- **Tipo de Prueba:** Caja Negra (Cancelación de acción)
- **Clases de Equivalencia:**
 - **Flujo Alternativo:** El usuario decide no eliminar el álbum.
- **Resultado Esperado:** El álbum permanece en el sistema y la eliminación se cancela.
- **Estado:** Aprobado



12. CP_ELIMALB4: Eliminar álbum asociado a canciones

- **Tipo de Prueba:** Caja Negra (Restricciones de integridad)
- **Clases de Equivalencia:**
- **Restricción de Negocio:** No se puede eliminar un álbum que tiene canciones asociadas sin reasignar o eliminar dichas canciones.
- **Resultado Esperado:** Mostrar advertencia y solicitar confirmación adicional o acciones previas.
- **Estado:** Aprobado



12. CP_GENGEN1: Agregar género con datos válidos

- **Tipo de Prueba:** Caja Negra (Flujo normal de agregar)
- **Clases de Equivalencia:**

- **Entrada Válida:** Nombre de género correctamente ingresado.
- **Resultado Esperado:** El género se agrega correctamente y aparece en la lista.
- **Estado:** Aprobado

The screenshot shows the application's top navigation bar with tabs for 'Genero', 'Album', 'Cancion', and 'Play'. A green notification banner at the top right states 'Genero creado/a satisfactoriamente'. Below the navigation bar is a toolbar with icons for 'Lista', 'Nuevo', 'Grabar', and 'Refrescar'. The main content area features a 'Descripción' label and an empty text input field. A dark blue 'GRABAR' button is positioned below the input field.

13. CP_GENGEN2: Agregar género sin nombre

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Campo "Descripción" vacío.
- **Resultado Esperado:** El sistema solicita completar el campo "Descripción".
- **Estado:** Aprobado

The screenshot shows the application's top navigation bar with tabs for 'Genero', 'Album', 'Cancion', 'Playlist', 'Roles', and 'Artista'. A red notification banner at the top right displays the error message: 'Es obligado que Descripción en Genero tenga valor'. The main content area has the 'Descripción' label and an empty text input field. A dark blue 'GRABAR' button is located below the input field.

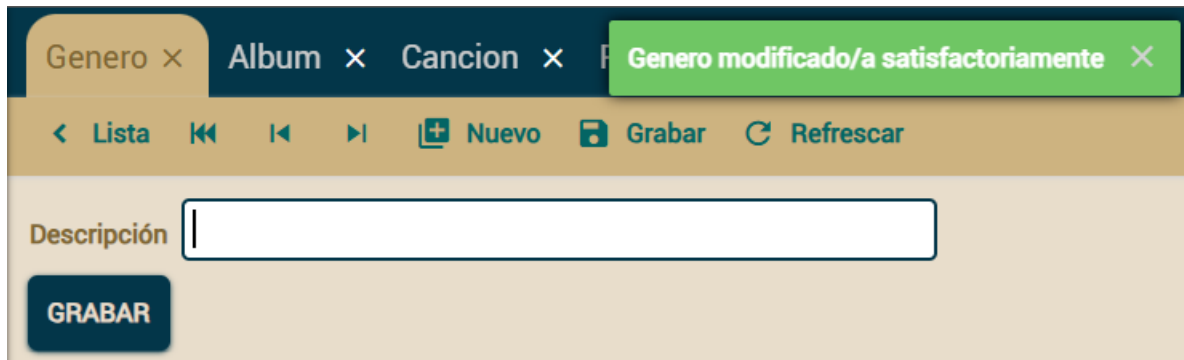
14. CP_GENGEN3: Agregar género duplicado

- **Tipo de Prueba:** Caja Negra (Validación de duplicidad)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Nombre de género que ya existe.
- **Resultado Esperado:** Mostrar mensaje de error indicando duplicidad.
- **Estado:** Aprobado

The screenshot shows the application's top navigation bar with tabs for 'Genero', 'Album', 'Cancion', 'Playlist', 'Roles', and 'Artista'. A red notification banner at the top right displays the error message: 'genero.uk_jilya70lqfcusw9ao7foi59sp'. The main content area has the 'Descripción' label and a text input field containing the word 'Dance'. A dark blue 'GRABAR' button is positioned below the input field.

14. CP_GENGEN4: Editar género con datos válidos

- **Tipo de Prueba:** Caja Negra (Flujo normal de edición)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Modificación de nombre a uno válido y no duplicado.
- **Resultado Esperado:** El género se actualiza correctamente en la lista.
- **Estado:** Aprobado



The screenshot shows the application's top navigation bar with tabs for 'Genero', 'Album', 'Cancion', and 'Playlist'. A green notification banner at the top right states 'Genero modificado/a satisfactoriamente'. Below the navigation bar is a toolbar with buttons for 'Lista', 'Nuevo', 'Grabar', and 'Refrescar'. The main content area features a 'Descripción' label and an empty text input field. A 'GRABAR' button is located below the input field.

15. CP_GENGEN5: Editar género dejando campos obligatorios vacíos

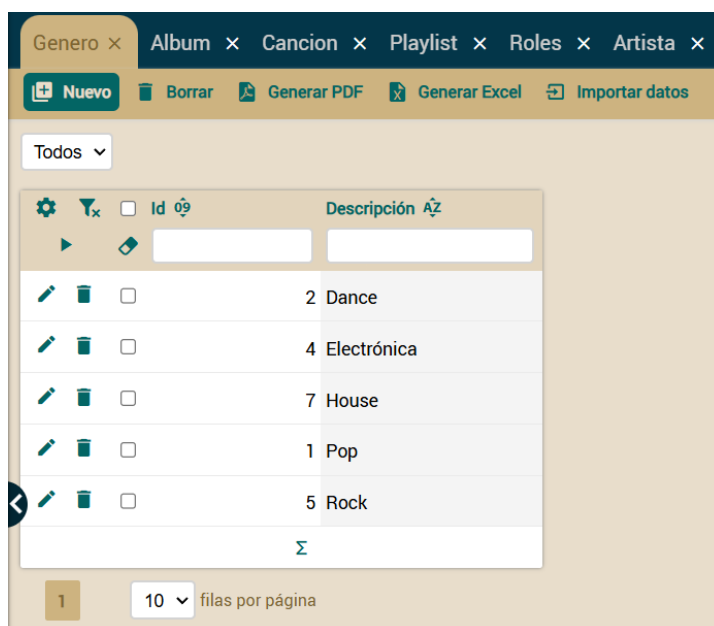
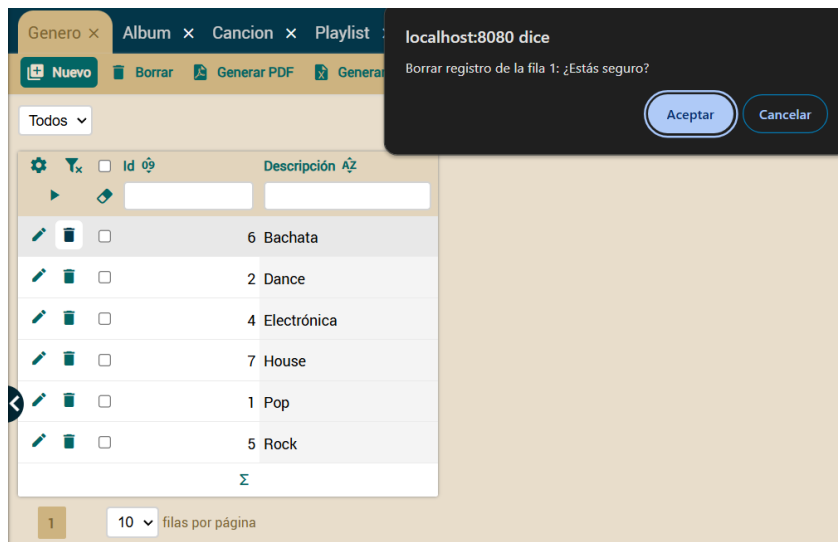
- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios en edición)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Campo "Descripción" vacío durante la edición.
- **Resultado Esperado:** El sistema solicita completar el campo "Descripción".
- **Estado:** Aprobado



The screenshot shows the application's top navigation bar with tabs for 'Genero', 'Album', 'Cancion', 'Playlist', 'Roles', and 'Artista'. A red notification banner at the top right states 'Es obligado que Descripción en Genero tenga valor'. Below the navigation bar is a toolbar with buttons for 'Lista', 'Nuevo', 'Grabar', 'Refrescar', and 'Borrar'. The main content area features a 'Descripción' label and an empty text input field. A 'GRABAR' button is located below the input field.

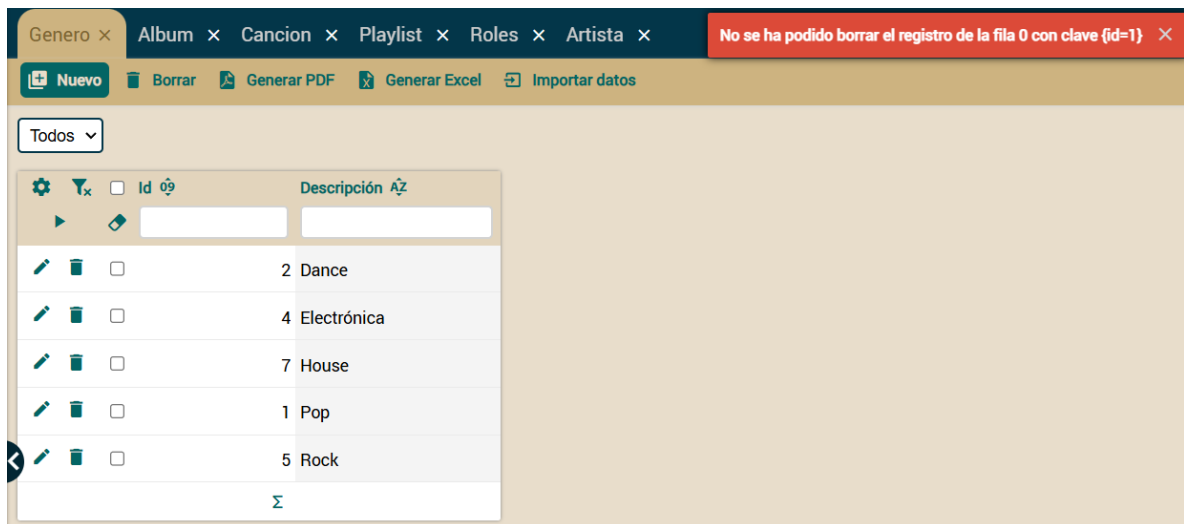
16. CP_GENGEN6: Eliminar género no asociado a canciones

- **Tipo de Prueba:** Caja Negra (Flujo normal de eliminación)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Género sin asociaciones.
- **Resultado Esperado:** El género se elimina correctamente.
- **Estado:** Aprobado



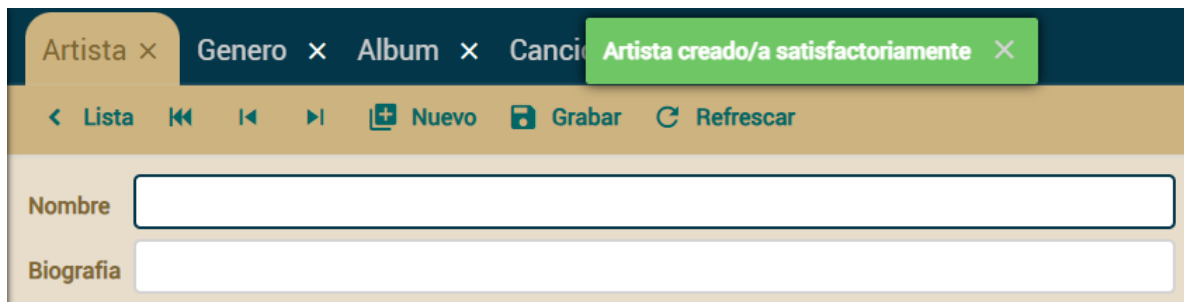
17. CP_GENGEN7: Eliminar género asociado a canciones

- **Tipo de Prueba:** Caja Negra (Restricciones de integridad)
- **Clases de Equivalencia:**
 - **Restricción de Negocio:** No se puede eliminar un género asociado a canciones sin reasignar o eliminar dichas canciones.
- **Resultado Esperado:** Mostrar advertencia y solicitar acción previa.
- **Estado:** Aprobado



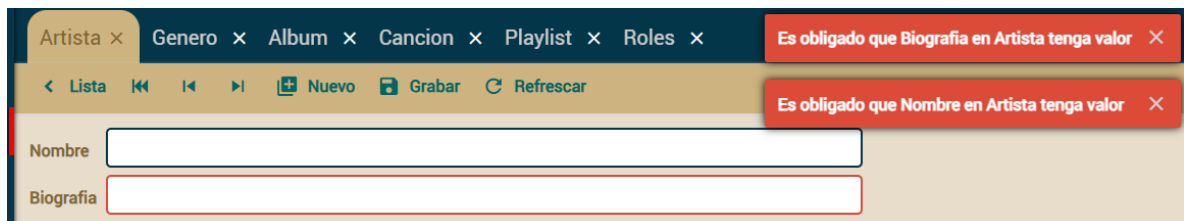
18. CP_REGART1: Registrar artista con datos completos

- **Tipo de Prueba:** Caja Negra (Flujo normal de registro)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Todos los campos completados correctamente.
- **Resultado Esperado:** El artista se registra exitosamente y aparece en la lista.
- **Estado:** Aprobado



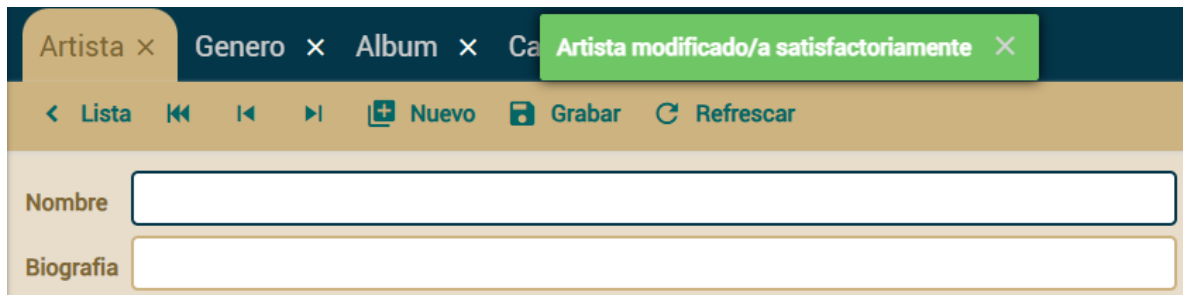
19. CP_REGART2: Registrar artista con campos vacíos

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Uno o más campos obligatorios vacíos.
- **Resultado Esperado:** El sistema solicita completar los campos obligatorios.
- **Estado:** Aprobado



20. CP_EDITART1: Editar artista con datos completos

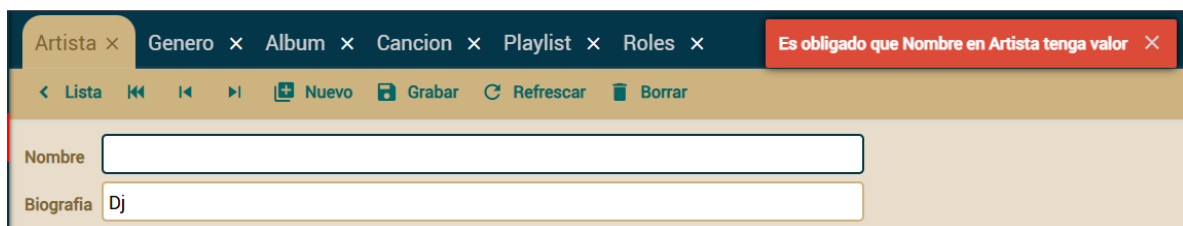
- **Tipo de Prueba:** Caja Negra (Flujo normal de edición)
- **Clases de Equivalencia:**
 - **Entrada Válida:** Todos los campos modificados correctamente.
- **Resultado Esperado:** El artista se actualiza correctamente en la lista.
- **Estado:** Aprobado



The screenshot shows a web application interface for editing an artist. At the top, there is a navigation bar with tabs: 'Artista', 'Genero', 'Album', 'Cancion', and 'Playlist'. The 'Artista' tab is active. Below the navigation bar, there is a green notification box that says 'Artista modificado/a satisfactoriamente'. Below this, there is a form with two input fields: 'Nombre' and 'Biografia'. The 'Nombre' field is empty, and the 'Biografia' field contains the text 'Dj'. At the bottom of the form, there are buttons for 'Nuevo', 'Grabar', and 'Refrescar'.

21. CP_EDITART2: Editar artista dejando campos obligatorios vacíos

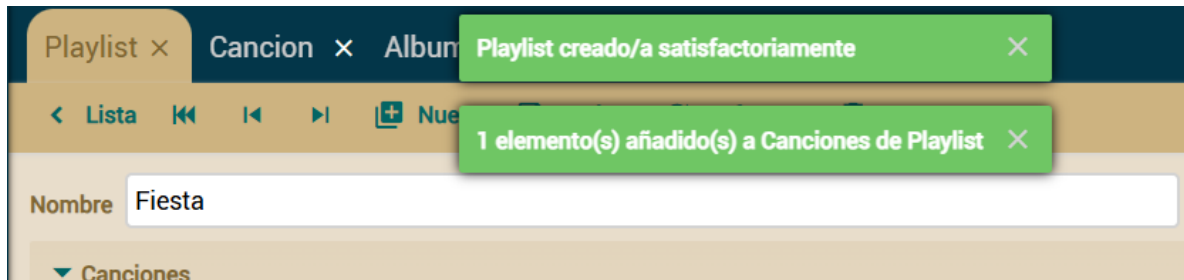
- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios en edición)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Uno o más campos obligatorios vacíos.
- **Resultado Esperado:** El sistema solicita completar los campos obligatorios.
- **Estado:** Aprobado



The screenshot shows the same web application interface as in the previous image. However, the 'Nombre' field is now empty, and the 'Biografia' field contains the text 'Dj'. A red notification box at the top right of the form says 'Es obligatorio que Nombre en Artista tenga valor'. At the bottom of the form, there are buttons for 'Nuevo', 'Grabar', 'Refrescar', and 'Borrar'.

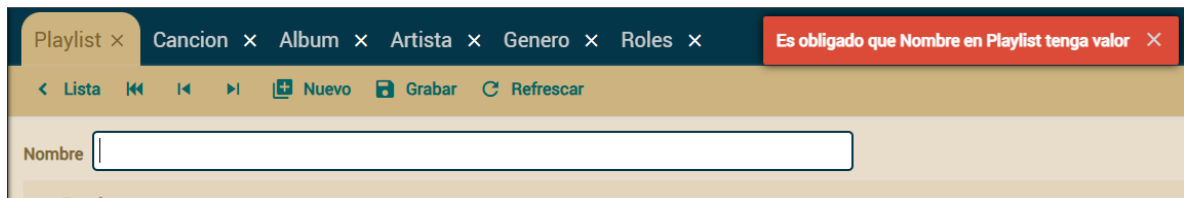
22. CP_CREPLAY1: Crear playlist con datos completos

- **Tipo de Prueba:** Caja Negra (Flujo normal de creación)
- **Clases de Equivalencia:**
 - **Entrada Válida:** selección de canciones válidas.
 - **Entrada Invalida:** Nombre semejante.
- **Resultado Esperado:** La playlist se crea correctamente y aparece en la lista.
- **Estado:** No Aprobado



23. CP_CREPLAY2: Crear playlist dejando campos obligatorios vacíos

- **Tipo de Prueba:** Caja Negra (Validación de campos obligatorios)
- **Clases de Equivalencia:**
 - **Entrada Inválida:** Campo "Nombre" vacío.
- **Resultado Esperado:** El sistema solicita completar el campo "Nombre".
- **Estado:** Aprobado



24. CP_EDITPLAY1: Editar Playlist dejando campos obligatorios vacíos

Tipo de Prueba: Caja Negra (Validación de campos obligatorios en edición)

Clases de Equivalencia: Entrada Inválida: Uno o más campos obligatorios vacíos.

Resultado Esperado: El sistema solicita completar los campos obligatorios.

Estado: Aprobado

Playlist × Album × Cancion × Artista × Genero × Roles × Es obligado que Nombre en Playlist tenga valor ×

< Lista ⏮ ⏪ ⏩ ⏭ + Nuevo + Grabar ↻ Refrescar 🗑 Borrar

Nombre

▼ Canciones

+ Añadir + Nuevo - Quitar de la lista 🗑 Borrar ✂ Cortar

<input type="checkbox"/>	Id	Título	Duracion	Reproducciones	Popularidad
	2		4	0	0
		Σ	Σ	Σ	Σ

Duracion total

Popularidad

GRABAR

25. CP_DELPLAY1: Eliminar Playlist existente

- Tipo de Prueba: Caja Negra (Flujo normal de eliminación)
- Clases de Equivalencia: Entrada Válida: Playlist existente seleccionada para eliminación.
- Resultado Esperado: La playlist se elimina correctamente y desaparece de la lista.
- Estado: Aprobado

Playlist × Album × Cancion × Artista × Genero × Roles ×

+ Nuevo 🗑 Borrar 📄 Generar PDF 📊 Generar Excel 📁 Importar datos

Todos ▾

⚙	🔍	<input type="checkbox"/>	Id 🔍	Nombre A-Z	Popularidad 🔍	Duracion total
▶	🔍		<input type="text"/>	<input type="text"/>	<input type="text"/>	

No hay registros

1 10 ▾ filas por página

26. CP_FILTER2: Filtrar resultados sin coincidencias

- Tipo de Prueba: Caja Negra (Validación de resultados filtrados)
- Clases de Equivalencia: Entrada Válida: Filtros seleccionados que generan coincidencias.
- Resultado Esperado: El sistema muestra un mensaje indicando "Sin coincidencias encontradas".
- Estado: No Aprobado

		<input type="checkbox"/>	Id	Titulo	Duracion	Id de genero	Genero	Id de album	Album
			<input type="text"/>	<div>contiene</div> <div>jkh</div>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

No hay registros

Se ejecutaron un total de 26 casos de prueba, de los cuales 23 fueron exitosos y 3 resultaron fallidos, lo que representa una tasa de éxito del 88.46%. Los casos fallidos han sido analizados minuciosamente para identificar sus causas subyacentes y definir las acciones correctivas necesarias. Estos resultados evidencian un alto nivel de eficacia en la implementación y funcionalidad del sistema; sin embargo, se han identificado áreas específicas que requieren atención para asegurar una cobertura integral y un rendimiento óptimo.

15.Resultado de Test

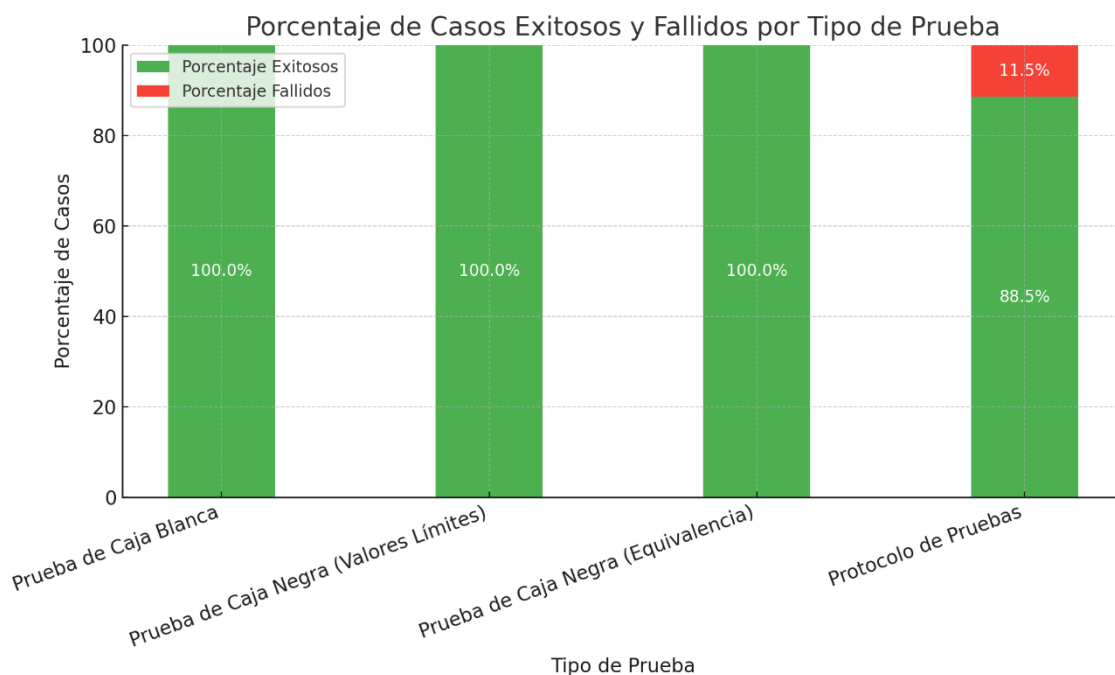
Durante el proceso de evaluación del software, se llevaron a cabo un total de 26 casos de prueba distribuidos en 4 tipos de pruebas, con el objetivo de garantizar la calidad y funcionalidad del sistema antes de su lanzamiento a producción. A continuación, se presenta un resumen detallado de las pruebas realizadas:

Tipo de Prueba	Descripción	Número de Casos	Casos Exitosos	Casos Fallidos
Prueba de Caja Blanca	Cobertura de Caminos en el método getDuracionTotalEnSegundos()	3	3	0
Prueba de Caja Negra	Cobertura de Valores Límites en entidades como Álbum, Canción, Género, Playlist	12	12	0
Prueba de Caja Negra	Cobertura en Equivalencia	8	8	0
Protocolo de Pruebas	Casos de prueba específicos para evaluar la funcionalidad global	3	0	3
Total		26	23	3

15.1. Representación Gráfica de los Resultados

A continuación, se presenta un gráfico que ilustra la distribución de los casos de prueba exitosos y fallidos:

Tipos de Casos de Prueba Exitosos y Fallidos



15.2. Análisis de los Resultados

- **Cobertura de Caja Blanca:** Todas las rutas de ejecución posibles en el método `getDuracionTotalEnSegundos()` fueron validadas exitosamente, asegurando la precisión en el cálculo de la duración total de las canciones.
- **Cobertura de Caja Negra (Valores Límites y Equivalencia):** Se validaron exhaustivamente las entidades principales del sistema, garantizando que los datos ingresados cumplan con las restricciones establecidas en términos de longitud, formato y reglas de negocio.
- **Protocolo de Pruebas:** De los 3 casos ejecutados bajo esta técnica, **3 resultaron fallidos**, lo que representa una tasa de fallos del **11.54%**. Estos fallos indican áreas específicas que requieren atención adicional antes de considerar la aceptación del software para producción.

15.3. Recomendaciones

Basándonos en los resultados obtenidos, se emiten las siguientes recomendaciones:

1. Corrección de Fallos Identificados:

- a. **Análisis Detallado:** Investigar las causas subyacentes de los 3 casos fallidos para determinar si se trata de errores críticos que afectan funcionalidades esenciales o si son problemas menores que pueden ser resueltos rápidamente.

- b. **Implementación de Soluciones:** Proceder a corregir los defectos identificados y realizar pruebas adicionales para asegurar que las soluciones implementadas resuelvan los problemas sin introducir nuevos errores.

2. Reevaluación Posterior a Correcciones:

- a. **Pruebas Adicionales:** Una vez realizadas las correcciones, ejecutar nuevamente los casos de prueba fallidos y otros casos relevantes para verificar la estabilidad y funcionalidad del sistema.
- b. **Validación Completa:** Asegurar que todas las pruebas, incluyendo las de caja blanca y caja negra, sigan siendo exitosas tras las modificaciones.

3. Consideración para Producción:

- a. **Tasa de Éxito Alta:** Con un **88.46%** de casos de prueba exitosos, el sistema demuestra un alto nivel de eficacia en su implementación y funcionalidad.
- b. **Salvaguardas Adicionales:** Si los fallos son menores y no comprometen la integridad del sistema, se puede considerar una salida controlada a producción, acompañada de monitoreo continuo para detectar y resolver cualquier incidencia que pueda surgir post-lanzamiento.

4. Mejora Continua:

- a. **Refinamiento de Pruebas:** Ampliar la cobertura de pruebas incluyendo escenarios adicionales que puedan no haber sido considerados inicialmente.
- b. **Automatización de Pruebas:** Implementar herramientas de automatización para facilitar pruebas regulares y reducir el tiempo de validación en futuras versiones del software.

15.4. Conclusión

El proceso de pruebas realizado ha demostrado una sólida cobertura y una alta tasa de éxito, validando la mayoría de las funcionalidades críticas del sistema de gestión musical. Sin embargo, los 3 casos fallidos identificados requieren atención inmediata para garantizar la calidad y confiabilidad del software antes de su lanzamiento a producción. Al abordar estos fallos y seguir las recomendaciones propuestas, se puede asegurar una implementación exitosa y un rendimiento óptimo del sistema.

16. Links

Link del video en Youtube: <https://youtu.be/qnTWvokiViM>

Link del Github: <https://github.com/thonyDeveloperSoftware77/Musicly-OpenXava.git>