XMLScore:
Representación gráfica y reproducción de partituras en formato XML

Aitor Valle Allende

29 de febrero de 2016

Resumen

Este proyecto se centra en la creación de unas herramientas para la apertura e impresión de partituras sencillas, que en consonancia con el proyecto complementario de mi compañera Ane (proyecto encargado de guardarlas) y unas librerías para su visualización y manipulación, se consigue dar forma a una aplicación llamada Kosmos.

A diferencia de muchas otras aplicaciones de escritura de música, Kosmos busca la simplicidad con una interfaz muy poco recargada, pudiendo editar partituras usando mayormente gestos con el ratón.

Para poder cumplir con su cometido, Kosmos maneja varios lenguajes. Debido a la estandarización y popularidad del formato, se decidió usar MusicXML como formato de preservación de las melodías. El entramado de jMusic es el encargado de mostrar en pantalla las figuras y manejar sus cambios. Lilypond creará documentos PDF de gran calidad para que las partituras puedan ser impresas.

El desarrollo de la aplicación se realizó en Java, lenguaje que también usa la librería de jMusic. Se usó el entorno de Eclipse para programar la aplicación, junto con un plugin de Git para mantener un historial de versiones en un repositorio de BitBucket, y de paso poder compartir el código con mi compañera. LyX es el editor de textos elegido para escribir esta memoria, encargado de facilitar la creación de cuidados documentos en formato LaTeX.

Esta memoria presentará una introducción sobre la idea inicial del proyecto, la planificación inicial de funcionalidades y recursos a destinar, la investigación de aplicaciones similares, el detallado más a fondo de las funciones mencionadas anteriormente, su implementación en Kosmos y el testeo.

Los anexos amplíarán la información dada, y se podrá profundizar en temas como los formatos MusicXML, jMusic y Lilypond, el parseo de archivos XML mediante StAX, una explicación más al detalle del código de la aplicación, un manual de usuario y las licencias de jMusic y Kosmos (GPL) y del manual (Creative Commons).

Índice general

Índice de figuras

Índice de tablas

Glosario

[MusicXML] Es un formato de notación musical abierto basado en XML. Fue diseñado para el intercambio de partituras, particularmente entre diferentes editores de partituras.

Acorde [chord] Un acorde es un grupo de dos o más notas tocadas simultáneamente para crear armonía. Los acordes añaden textura a una melodía y proporcionan ritmo a una canción.

Adagio El término musical en italiano adagio es una indicación de tocar lento y calmado; con tranquilidad. Adagio es más lento que adagietto, pero más rápido que largo. Tradicionalmente, adagio tiene aproximadamente 66-76 pulsos por minuto (algunas veces está marcado como 56-76). Su rango moderno es desde 60-80.

Allegro El término musical italiano allegro es una indicación para tocar con un tempo rápido y animado. Allegro es más rápido que allegretto, pero más lento que allegrissimo. Allegro tiene aproximadamente 112-160 pulsos por minuto.

Alteración [accidental] Son signos que modifican la altura de las notas escritas en el pentagrama. Son tres: sostenido, bemol y becuadro.

Altura [pitch] Es la cualidad del sonido que nos permite identificar los sonidos como graves o agudos. Depende de la frecuencia o número de vibraciones por segundo, a mayor frecuencia, más agudo suena el sonido.

Anacrusa La anacrusa es una nota, o serie de notas, que viene antes del primer compás completo de una composición; un compás introductorio (y opcional) que no tiene el número de pulsaciones expresado por la marca de tiempo.

Andante El término musical italiano andante es una indicación para tocar con un tempo tranquilo; de una manera ligera y fluida. Andante es más rápido que el adagio , pero más lento que el allegretto; similar al moderato. Tiene alrededor de 76-108 pulsos por minuto.

Armadura [key signature] Conjunto de sostenidos o bemoles que se colocan en un pentagrama a la derecha de la clave y antes del compás e indican la tonalidad de la composición.

Becuadro [natural] Signo (♮) que se coloca delante de una nota musical previamente alterada por un sostenido o bemol para indicar que vuelve a su entonación natural: un becuadro colocado en un fa anula la alteración del fa anterior en el mismo compás.

Bemol [flat] Un bemol es una alteración que indica una pequeña bajada en la altura. Un bemol es un símbolo (♭, también cuando se teclea) colocado frente a una nota que disminuye su altura en medio tono. Re♭ está medio tono más abajo que Re.

Blanca [half / minim] Nota musical que equivale a la mitad de una redonda. Equivale a dos negras.

Clave [clef] La función de una clave musical es ubicar a quien interpreta una determinada melodía dentro de una tonalidad, utilizando aquellas notas que se encuentran asociadas armónicamente entre sí y asociandolas con los espacios y líneas del pentagrama.

Clave de fa [bass] Clave que marca dónde se coloca la nota fa que está por debajo del do central (o que el do central se encuentra en la primera línea adicional superior)

Clave de sol [treble] Clave que marca dónde se coloca la nota sol que está por encima del do central (o que el do central se encuentra en la primera línea adicional inferior)

Compás [measure] El compás es la entidad métrica musical, compuesta por varias unidades de tiempo (como la negra o la corchea). Esta división se representa gráficamente por unas líneas verticales, llamadas «líneas divisorias» o «barras de compás» que se colocan perpendicularmente a las líneas del pentagrama.

Corchea [eighth / quaver] Nota musical que equivale a la mitad de una negra. Equivale a dos semicorcheas.

Crescendo Pasaje de una composición musical que se ejecuta aumentando gradualmente la intensidad.

Decrescendo Disminución progresiva de la intensidad de una nota o un pasaje musical.

Denominador del compás/subdivisión del compás [beat-type] Es el número que se ubica abajo. Indica a la figura que representa el pulso.

Duración [duration] Es la cualidad del sonido que nos permite identificar los sonidos como largos o breves. El sonido será tan largo como sea la onda. El sonido prolongado del gong tendrá una onda más larga que el breve y seco sonido de las claves.

Etiqueta (XML) Una etiqueta o baliza (términos a veces reemplazados por el anglicismo tag) es una marca con clase que delimita una región en los lenguajes basados en XML.

Figura [note] Indican la duración del sonido, ubicadas en el pentagrama indican la altura. Las figuras musicales más usadas son: redonda, blanca, negra, corchea, semicorchea, fusa y semifusa.

Forte Forte es una indicación para tocar en voz alta; más alto que mezzo forte , pero más bajo que fortissimo. Forte se marca en la partitura musical como f.

Fortissimo (Del italiano forte, fuerte) es un término que se utiliza en notación musical para indicar un grado determinado de intensidad del sonido, es decir, un matiz dinámico. La intensidad que señala es muy alta, situándose por encima de forte y por debajo de fortissisimo.

Fusa [thirty-second / demisemiquaver] Nota musical que equivale a la mitad de una semicorchea. Equivale a dos semifusas.

Hardware Componentes físicos del ordenador, es decir, todo lo que se puede ver y tocar. Clasificaremos el hardware en dos tipos: El que se encuentra dentro de la torre o CPU, y que por lo tanto no podemos ver a simple vista y el que se encuentra alrededor de la torre o CPU, y que por lo tanto, sí que vemos a simple vista, y que denominamos periféricos (ratón, teclado...).

Instancia Se llama instancia a todo objeto que derive de algún otro. De esta forma, todos los objetos son instancias de algún otro, menos la clase Object que es la madre de todas.

Intensidad Es la cualidad del sonido que nos permite identificar los sonidos como fuertes o suaves, es pues la fuerza o volumen del sonido. Depende de la amplitud de la onda, a mayor amplitud, más fuerte suena el sonido.

Interfaz Medio que permite a un usuario comunicarse con una máquina.

Largo Indica que una composición musical o parte de ella debe interpretarse con un tempo o ritmo muy lento.

Lento En música, la indicación en italiano lento significa que hay que tocar en un tempo más lento. Literalmente más despacio. Tiene entre 52-68 pulsos por minuto.

Licencia Es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribuidor) y el licenciatario (usuario consumidor/usuario profesional o empresa) del programa informático, para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

Licencia de código abierto Una licencia de código abierto es una licencia de software que permite que tanto el código fuente como los archivos binarios sean modificados y redistribuidos libremente y sin tener que pagar al autor original. Sin embargo, ciertas licencias de código abierto pueden incorporar algunas restricciones, como el requisito de mantener el nombre de los autores y la declaración de derechos de autor en el código, o permitir la modificación del código sólo para usos personales o la redistribución del software para usos no comerciales. Un grupo popular (y a veces considerado normativo) de licencias de software de código abierto son aquellas aprobadas por la Open Source Initiative basándose en su Open Source Definition.

Licencia de código propietario Es aquel en el que un usuario tiene limitadas sus posibilidades de usarlo, modificarlo o redistribuirlo, y a menudo su licencia tiene un coste.Se le llama software propietario, no libre, privado o privativo al tipo de programas informáticas o aplicaciones en el que el usuario no puede acceder al código fuente o tiene un acceso restringido y, por tanto, se ve limitado en sus posibilidades de uso, modificación y redistribución.

Licencia GPL La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License (o simplemente sus siglas del inglés GNU GPL) es la licencia más ampliamente usada en el mundo del software y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

Ligadura [tie] Una ligadura es un arco horizontal que conecta notas musicales del mismo tono (en contraposición con el legato, que conecta dos o más de diferente tono). Las notas ligadas deben ejecutarse añadiendo el valor de ambas notas; solo se percute la primera.

Ligadura de expresión La ligadura de expresión agrupa dos o más notas de nombre diferente. También se denomina legato. El legato ha de realizarse de forma que el sonido no se interrumpa.

Ligadura de unión La ligadura de unión se coloca entre dos o más notas del mismo nombre y mantiene el mismo sonido pero uniendo su duración. Las notas han de ser del mismo nombre pero pueden ser de distinto valor, sumando siempre la duración de ambas.

Línea adicional [ledger line] Líneas que se colocan por encima o debajo del pentagrama, para extenderlo a notas que no caben dentro de ella.

Metrónomo Instrumento para medir el tiempo e indicar el compás de las composiciones musicales.

Mezzoforte El término musical italiano mezzo forte (o mf) literalmente significa "medio fuerte," y es una indicación para tocar algo fuerte; ligeramente más suave que con (f) forte.

Mezzopiano El comando musical italiano mezzo piano (o mp) literalmente significa medio suave, y es una indicación para tocar un poco fuerte; ligeramente más fuerte que (p) piano .

Moderato La indicación musical en italiano moderato indica que se debe de tocar en un tempo razonable, moderado; lit. "moderado." tiene entre 88-112 pulsos por minuto; entre andante y allegro.

Negra [quarter / crotchet] Nota musical que equivale a la mitad de una blanca. Equivale a dos corcheas.

Numerador del compás/pulsos [beat] El numerador (número de arriba) indica el número de partes (o de tercios de parte en los compases compuestos o de subdivisión ternaria) que tiene el compás.

Octava [octave] Se trata del intervalo que existe entre un par de sonidos que disponen de frecuencias que mantienen un vínculo de 2-1. Si un sonido tiene una frecuencia fundamental de 2640 Hz, se encontrará una octava más alto que aquel cuya frecuencia es de 1320 Hz.

Partitura [score] Texto escrito de una obra musical en el que se anotan los sonidos que han de ejecutar los distintos instrumentos o voces y el modo en que han de hacerlo.

Pentagrama Es una modalidad de notación musical que se basa en una estructura compuesta por cinco rectas ubicadas de manera paralela y a una misma distancia de separación. Los pentagramas se destinan a la escritura de música, es decir, para tener registro en un soporte escrito de las notas y los demás signos musicales necesarios para interpretar una melodía. Todas las líneas del pentagrama, así como sus cuatro espacios, se enumeran en dirección abajo-arriba.

Pianissimo En música, se emplea como acotación interpretativa para indicar que un fragmento o una pieza deben ejecutarse muy suavemente, con muy poca intensidad: la abreviatura de pianissimo es "pp".

Piano (Del italiano piano, suave) es un término que se utiliza en notación musical para indicar un grado determinado de intensidad del sonido, es decir, un matiz dinámico. La intensidad que señala piano es baja o suave, situándose por encima de pianissimo y por debajo de mezzopiano.

Presto Indica que el tempo de una composición es muy veloz. Esto quiere decir que la obra que debe tocarse presto tiene que ejecutarse a una velocidad superior a 180 negras por minuto, aunque inferior a 200 (ya que, a más de 200 negras por minuto, se emplea el término prestissimo). En este caso, el significado de presto está asociado a su equivalente italiano que puede traducirse como "rápido".

Puntero Los punteros permiten simular el paso por referencia, crear y manipular estructuras dinámicas de datos, tales como listas enlazadas, pilas, colas y árboles. Generalmente las variables contienen valores específicos. Los punteros son variables pero en vez de contener un valor específico, contienen las direcciones de las variables a las que apuntan. Para obtener o modificar el valor de la variable a la que apuntan se utiliza el operador de indirección. Los punteros, al ser variables deben ser declaradas como punteros antes de ser utilizadas.

Puntillo [dot] El puntillo se señala a la derecha de la nota que se pretende modificar, incrementando su valor y su duración en la mitad. Cabe destacar que tanto las notas como los silencios pueden llevar puntillos.

Redonda [whole / semibreve] Nota musical cuya duración equivale a dos blancas.

Ritmo El ritmo es la proporción existente entre el tiempo de un movimiento y el de otro diferente. La organización de los compases, los pulsos y los acentos determinan la forma en la cual el oyente percibe el ritmo y, por lo tanto, la estructura de la obra.

Ruta Una ruta (del inglés path) es la forma en que se hace referencia a un archivo o directorio dentro de un sistema de archivos. En otras palabras, la ruta señala la ubicación exacta del archivo o directorio a través de una cadena de caracteres.

Semicorchea [sixteenth / semiquaver] Nota musical que equivale a la mitad de una corchea. Equivale a dos fusas.

Semifusa [sixty-fourth / hemidemisemiquaver] Nota musical que equivale a la mitad de una fusa.

Signo de repetición Los signos de repetición son marcas y signos que tienen el objetivo de evitar volver a escribir compases que van a ser repetidos de la misma forma en que ya fueron escritos. Esto hace que los temas queden en una partitura más corta, y desde el punto de vista de lectura, el proceso es más esquemático y práctico.

Silencio [rest] La ausencia de ruido o de sonido.

Software Son las instrucciones que el ordenador necesita para funcionar. No existen físicamente, o lo que es igual, no se pueden ver ni tocar. También tenemos de dos tipos: Sistemas Operativos: Tienen como misión que el ordenador gestione sus recursos de forma eficiente, además de permitir su comunicación con el usuario. Nosotros utilizamos el Sistema Windows y las aplicaciones, programas informáticos que tratan de resolver necesidades concretas del usuario, como por ejemplo: escribir, dibujar, escuchar música...

Sonido Tiene 4 cualidades que son altura, timbre, duración e intensidad.

Sostenido [sharp] Un sostenido es una alteración que indica una pequeña subida en la altura. Un sostenido es un símbolo (♯, cuando se teclea) colocado frente a una nota que aumenta su altura en medio tono.

Tempo La velocidad con que debe ejecutarse una pieza musical. Se trata de una palabra italiana que literalmente significa «tiempo». En las partituras de una obra el tempo se suele representar al inicio de la pieza encima del pentagrama.

Timbre Es la cualidad del sonido que nos permite diferenciar las voces e instrumentos. Cada instrumento tiene un sonido característico, igual que cada uno de nosotros tiene una voz personal y distinta a la de los demás. Ese rasgo es el timbre o color característico. Por eso diferenciamos una trompeta de un xilófono aunque toquen la misma melodía.

Tonalidad [key] Sistema de sonidos que sirve de fundamento a una composición musical.

Vivace Se emplea como acotación musical para indicar que un fragmento o una pieza deben interpretarse con tempo o ritmo animado: el allegro vivace es más rápido que el allegro a secas.

Capítulo 1

Introducción

1.1. Descripción y situación del trabajo

Este proyecto forma parte de algo más complejo, ya que se irá trabajando en conjunto con otro proyecto con la finalidad de crear un único programa.

El proyecto explicado en este documento permitirá al usuario representar gráficamente y reproducir una partitura dado un formato MusicXML (para conocer más acerca del formato, ver apéndice A en la página 63). Como origen, se podrá coger una partitura creada por un proyecto complementario (*Generador automático de ítems de evaluación del lenguaje musical* [8]) y visualizar o reproducir su contenido.

Este proyecto busca agilizar el aprendizaje de la música mediante interfaces simples. La aplicación estará orientada a escuelas de música y conservatorios para una calificación y evaluación más rápida de los alumnos y por consiguiente, facilitar las tareas del profesor.

Una vez nuestro proyecto esté acabado, otro desarrollador podría ampliarlo para conseguir que sea más completo e interactivo.

1.2. Razones de elección del TFG

Dado que tengo conocimientos musicales y toco varios instrumentos (piano, y en menor medida la guitarra), me decanté por elegir el proyecto que ofrecía Javilo, mi tutor de proyecto. Me pareció interesante hacer un proyecto que más tarde puedan utilizar los conservatorios para facilitar el aprendizaje de los alumnos.

Capítulo 2

Planteamiento Inicial

2.1. Objetivos

- Representación de partituras
- Creación del sonido de la partitura
- Leerá archivos de formato MusicXML

2.2. Planificación original

Esta es la primera planificación realizada para el proyecto.

2.2.1. Alcance

El alcance tiene como objetivo definir qué trabajo del proyecto se va a realizar y cuál no.

2.2.1.1. Objetivos primarios

- Importar archivos MusicXML
- Exportar como imagen y sonido
- Partituras de una sola voz
- Clave de sol
- Compases dentro del rango: $^2/_4$, $^3/_4$, $^4/_4$, $^6/_8$, $^9/_8$
- Armadura: cualquier tonalidad
- Notas y silencios que abarquen el rango de duraciones: redonda, blanca, negra, corchea, semicorchea, fusa, semifusa
- Notas con puntillo
- Alteraciones simples (sostenido, bemol, becuadro)

2.2.1.2. Objetivos secundarios

- Clave de fa
- Más compases aparte de los mencionados arriba
- Doble puntillo
- Metrónomo

- Acordes
- Exportar a Lilypond (ver apéndice C en la página 73)
- Botones de deshacer y rehacer
- Intensidades: f, mf, mp, p, cresc., decr., ...
- Ligaduras (de unión y/o expresión)

2.2.1.3. Qué se va a hacer conjuntamente

- Investigación del formato MusicXML
- Interfaz del programa
- Pruebas

2.2.1.4. Qué no se va a hacer

- Ritmos sin altura
- Tempo (velocidades): presto, vivace, allegro, moderato, andante, adagio, largo, lento, ...
- Alteraciones múltiples (sostenido doble, bemol doble)
- Signos de repetición
- Anacrusas

2.2.2. Planificación temporal

Tarea	Subtarea	Fecha	Tiempo
Investigación y preparación previa		2013/05/27 2014/06/17	30h
	Reuniones y correos		6h
	Investigación y búsqueda		24h
Documento de viabilidad		2014/09/03 2014/10/05	20h 30min
	Redacción		20h
	Reuniones y correos		30min
Captura de requisitos		2014/10/06 2014/10/26	$\simeq 45h$
Análisis y Diseño		2014/10/27 2014/11/09	$\simeq 30h$
Implementación		2014/11/10 2015/03/29	$\simeq 300h$
	Grupal	2014/11/10 2015/03/29	$\simeq 60h$
	1. Estudio MusicXML	2014/11/10 2014/12/07	$\simeq 40h$
	2. Interfaz	2014/12/08 2015/03/29	$\simeq 20h$
	Individual	2014/12/08 2015/03/29	$\simeq 240h$
	1. Abrir MusicXML	2014/12/08 2014/12/28	$\simeq 50h$
	2. Visualizar partitura	2014/12/29 2015/03/15	$\simeq 160h$
	3. Reproducción a archivo	2015/03/16 2015/03/29	$\simeq 30h$
Documentación		2015/03/30 2015/04/03	$\simeq 20h$
Presentación		2015/04/04 2015/04/12	$\simeq 15h$
Total		2013/05/27 2015/04/12	$\simeq 460h$

Tabla 2.1: Tabla de planificación temporal

En las figuras 2.1 y 2.2 podemos ver la planificación en gráficos Gantt. El color negro representa las tareas individuales y el gris las tareas comunes.

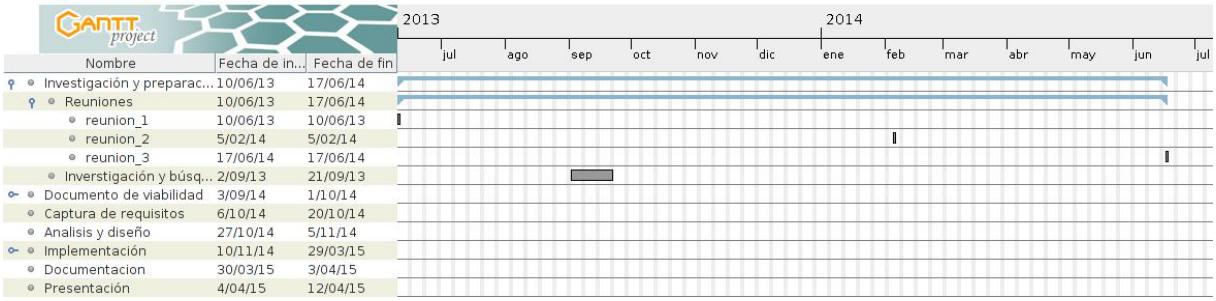


Figura 2.1: Gantt Preproyecto Aitor

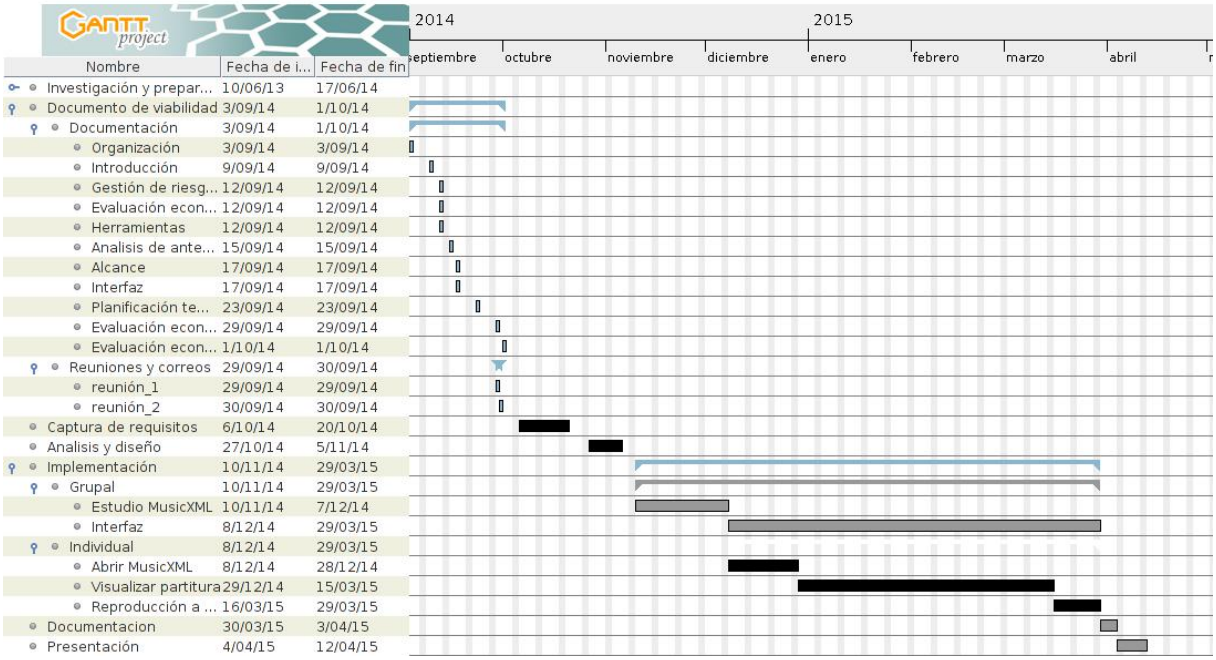


Figura 2.2: Gantt Proyecto Aitor

El diagrama de la figura 2.3 muestra la descomposición del proyecto en distintas tareas:

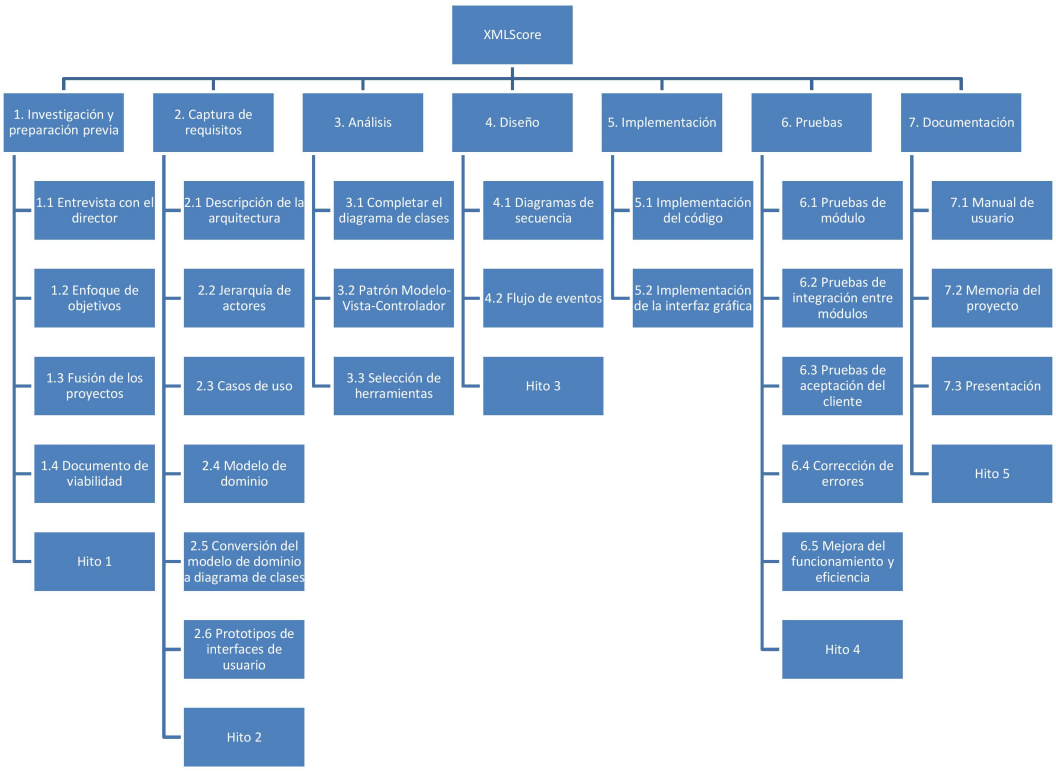


Figura 2.3: Diagrama de Estructura de Descomposición del Trabajo

2.2.3. Herramientas y lenguajes de programación utilizados

Java Lenguaje de programación orientado a objetos basado en clases.

Eclipse Entorno de desarrollo ampliable mediante plugins para diversos lenguajes de programación.

WindowBuilder Plugin que trabaja complementariamente con Eclipse, con el cual se pueden crear diferentes tipos de interfaces.

XML Lenguaje de marcado estandarizado, utilizado para almacenar datos en forma legible.

MusicXML Formato de notación musical basado en XML.

LaTeX Lenguaje de marcado utilizado para componer documentos.

LyX Herramienta de apoyo para un creación más sencilla de documentos LaTeX.

PDF Formato de documento estandarizado de formato abierto.

MIDI Protocolo estándar que permite a los dispositivos generar sonidos.

Cacoo Herramienta online para crear esquemas UML, modelos de dominio, diagrama de clases, diagramas de secuencia, ...

GanttProject Programa utilizado para generar las duraciones de las tareas gráficamente.

Word Procesador de textos usado para el diagrama de estructura de descomposición del trabajo (EDT).

Skype Aplicación para la comunicación vía llamadas a distancia

WhatsApp Herramienta de mensajeria instantánea

Dropbox Servicio online de copias de seguridad. Gracias a la aplicación de escritorio, se sincronizan los documentos con la nube, para poder almacenarlos entre los participantes de una carpeta compartida.

YouTube Servicio de vídeos utilizado para visualizar tutoriales referentes al proyecto y a las herramientas y aplicaciones utilizadas.

Git Sistema de control de versiones útil para proyectos en los que trabajen varios participantes.

BitBucket Repositorio Git en el que se almacenará el código de la aplicación a desarrollar.

Lilypond Programa de código abierto usado para crear partituras de alta calidad, que usa su propio lenguaje de marcado como código de entrada (más información acerca de la herramienta en el apéndice C en la página 73)

2.2.4. Gestión de riesgos

- **Riesgo:** Pérdida de datos
  - **Prevención:** creación de copias de seguridad de manera periódica y almacenamiento en un lugar distinto al del proyecto (nube, Dropbox, GitHub, disco duro externo, ...)
  - **Plan de contingencia:** recuperar los datos perdidos de las copias realizadas
  - **Probabilidad:** probable
  - **Impacto:** muy alto
- **Riesgo:** Retrasos en el proyecto debido a la enfermedad de alguno de los integrantes del proyecto
  - **Prevención:** No se puede prevenir
  - **Plan de contingencia:** Hacer horas extras o retrasar la fecha final
  - **Probabilidad:** Probable
  - **Impacto:** Alto
- **Riesgo:** Infección de virus en el ordenador
  - **Prevención:** Instalar antivirus y analizar periódicamente
  - **Plan de contingencia:** Realizar un analisis completo
  - **Probabilidad:** Bastante probable
  - **Impacto:** Alto, depende del daño provocado
- **Riesgo:** Poca participación por parte del compañero
  - **Prevención:** Apoyarse el uno al otro
  - **Plan de contingencia:** Llamadas de atención y dialogar el problema
  - **Probabilidad:** Probable
  - **Impacto:** Medio
- **Riesgo:** Errores en la planificación debido a la falta de experiencia
  - **Prevención:** Mirar los apuntes y tenerlos a mano en caso de duda

- **Plan de contingencia:** Organización previa
- **Probabilidad:** Bastante probable
- **Impacto:** Muy alto

- **Riesgo:** Malentendidos con el cliente
  - **Prevención:** Preguntar a la más mínima duda y asegurar todo lo explicado por el cliente
  - **Plan de contingencia:** Programar una reunión con el cliente y aclarar dudas enseñando bocetos
  - **Probabilidad:** Bastante probable
  - **Impacto:** Muy alto

- **Riesgo:** Comportamiento inesperado del programa en distintas plataformas
  - **Prevención:** Realizar casos de uso en todas las plataformas posibles
  - **Plan de contingencia:** Cambiar el código de la funcionalidad que no se ejecuta correctamente
  - **Probabilidad:** Muy probable
  - **Impacto:** Medio

- **Riesgo:** Eficiencia baja del programa
  - **Prevención:** Encontrar los puntos fuertes y débiles de los algoritmos y estructuras; y descartar los peores hasta encontrar la mejor opción
  - **Plan de contingencia:** Revisar las opciones descartadas y hacer un nuevo estudio para encontrar una solución más optima
  - **Probabilidad:** Probable
  - **Impacto:** Muy alto

2.2.5. Evaluación económica

A continuación se realizará un análisis sobre la inversión que habría que realizar si se tuviera en cuenta todos los gastos económicos relativos al proyecto.

2.2.5.1. Inversión inicial

Mano de Obra

- Precio por hora: 12€
- Horas estimadas del proyecto: 460 horas (el número de horas aproximadas por trabajador al año son 1700).
- Salario: $12 * 460 = 5520$€

Hardware

- Ordenador portátil: 900€
  - La vida de un ordenador portátil se estima que es de 5 años. La amortización anual será de:

$$\frac{900€}{5 \text{ años}} = 180€$$

  - Por lo tanto, la amortización durante el proyecto será de:

$$\frac{180€ * 7 \text{ meses}}{12 \text{ meses}} = 105€$$

Software No se hará gasto en software, dado que la mayoría de los programas utilizados son gratuitos, y en caso contrario se usarán las versiones de prueba.

Otros gastos

- Luz:

$$45€ * 7 \text{ meses} = 315€$$

- Agua[1]:
  - Cálculo mensual en base a la tarifa trimestral:

$$\frac{8€}{3 \text{ meses}} = 2{,}66€$$

  - La cantidad de litros de agua estimados al mes será de 40. Calculando el precio respecto a la cuota variable, el gasto apenas supone de varios céntimos mensuales.

$$\frac{0{,}6382€ * 40\text{l}}{1000\text{l}} = 0{,}0255€$$

  - El gasto total de agua en el proyecto será de:

$$(2{,}66€ + 0{,}0255€) * 7 \text{ meses} = 18{,}80€$$

- Desplazamientos:
  - Coste total de la tarjeta jóven del Metro durante el periodo del proyecto:

$$\frac{253€ * 7 \text{ meses}}{12 \text{ meses}} = 147{,}58€$$

- Internet:

$$30€ * 7 \text{ meses} = 210€$$

- Movil:

$$8{,}3€ * 7 = 58{,}10€$$

El total de los otros gastos es:

$$315 + 18{,}8 + 147{,}58 + 210 + 58{,}1 = 749{,}48€$$

Coste total del proyecto

$$5520 + 105 + 749{,}48 = 6374{,}48€$$

2.2.5.2. ROI

Para monetizar nuestra aplicación, se quieren poner a la venta licencias. Teniendo en cuenta las diferentes posibilidades de venta, hemos acordado las siguientes alternativas en el precio de las licencias para no tener déficit:

- Licencias a 15€: $\frac{6374{,}48€}{15€} = 425$ licencias estimadas
- Licencias a 30€: $\frac{6374{,}48€}{30€} = 213$ licencias estimadas

2.3. Replanificación

Debido a la incorporación de la herramienta jMusic (para conocer más acerca de la herramienta, ver apéndice B en la página 67), se decidió rehacer la planificacón del proyecto.

[1] https://oficinavirtual.consorciodeaguas.com/facturaelectronica/AtencionCliente/tarifas.aspx

2.3.1. Alcance

- Se mantienen los objetivos primarios como primarios.
- Se promocionan de secundarios a primarios los objetivos:
  - Clave de fa
  - Exportar a Lilypond
  - Extensión del abanico de compases a elegir.
- Se añade la investigación de jMusic como proyecto conjunto.
- Se decide que finalmente las ligaduras de unión si formarán parte del proyecto.

2.3.2. Planificación temporal

En un principio se planeó terminar el código a finales de marzo (2015), para realizar la entrega a mediados de abril (2015) y presentar el proyecto en mayo (2015).

Debido a las asignaturas que tenía pendientes, la jornada laboral de mi compañera y la búsqueda de documentación de la libreria jMusic, decidimos seguir con el código hasta finales de mayo (2015), y demorar la entrega a junio (2015) para así realizar la presentación en julio (2015).

Finalmente, al no ser posible la finalización del proyecto, se decidió volver a retormarlo en septiembre (2015). Tras haberlo finalizado del todo, se decidió presentarlo en la convocatoria de febrero (2016).

Capítulo 3

Análisis de Antecedentes

3.1. Situación actual

Existen varias aplicaciones de características similares que usaremos como referencia a lo largo de nuestro proyecto.

Por una parte está *MusikEbal* [19], un proyecto ofertado previamente por la Facultad y realizado por Lander Martínez, ex-alumno de la Universidad. La diferencia más significativa con nuestro proyecto sería el uso de MusicXML (más información en el apéndice A en la página 63), a diferencia de un formato propio creado por él mismo. MusicXML es el formato estandarizado a nivel mundial para el intercambio de partituras en formato digital. La cantidad de aplicaciones que lo soportan es muy ámplia, y la documentación está abierta a que cualquier desarrollador pueda hacer uso de ello.

Por otra parte, *LenMus* era un proyecto ambicioso que cesó su desarrollo en julio de 2014 debido a que el creador no consiguió atraer a más desarrolladores que dedicaran tiempo suficiente como para que la aplicación pudiera madurar. Su meta era crear un programa para editar documentos musicales interactivos para la enseñanza del solfeo musical.

3.2. Estudio de diferentes alternativas existentes

Denemo	Encore	Finale	Frescobaldi
Guitar Pro	Harmony Assistant	LenMus	LilyPond
MuseScore	NoteWorthy Composer	NtEd	Sibelius

Figura 3.1: Lista aplicaciones similares

Denemo Programa musical para crear y escuchar partituras sencillas de una forma rápida. Se puede usar en varios sistemas operativos: Windows, Linux y Mac.

Encore Editor de partituras para Windows y Mac diseñado por GVOX. Se caracteriza por ser el pionero en añadir la opción de añadir una nota al pentagrama mediante el ratón. Puede ser usado tanto por noveles como profesionales.

Finale Es uno de los programas más populares en el ámbito de la música. Diseñado para Windows y Mac por la empresa MakeMusic. Tiene una ámplia gama de versiones para toda clase de usuarios.

Frescobaldi Es un editor de partituras con formato Lilypond. Es de código abierto y se puede usar en Windows, Mac y Linux.

Guitar Pro Aunque esté orientado a partituras para guitarra y bajo, cabe la posibilidad de usarlo con más instrumentos. Puede ejecutarse en Windows, Mac y Linux y lo comercializa la compañía Arobas Music.

Harmony Assistant Creado por Myriad Software, es una aplicación de edición de partituras multi-plataforma (Windows, Mac y Linux). Existe una variante con menos funcionalidades llamada Melody Assistant.

LenMus Software abierto y gratuito para lenguaje musical. En julio de 2014 se anunció el cierre del programa por falta de interés público. Se caracterizó por ofrecer la posibilidad de crear documentos interactivos para aprender y practicar el lenguaje musical.

Lilypond Programa de código abierto para el grabado musical que se centra en la calidad de la notación musical, intentando imitar la tipografía de las partituras tradicionales.

MuseScore Una de las interfaces gráficas más completa para la creación de partituras en formato Lilypond. Se usa en diferentes sistemas operativos, tales como Windows, Mac y Linux.

NoteWorthy Composer Es una aplicación diseñada únicamente para Windows, popularizada por la creación sencilla de partituras.

NtEd Editor de partituras para Linux basado en NoteEdit. Importa archivos en formato MusicXML.

Sibelius Junto con Finale, una de las aplicaciones más populares del ámbito musical. Se puede ejecutar en Windows y Mac.

3.2.1. Comparación de características

Precio Como es habitual, las aplicaciones de código abierto mencionadas son gratuitas. Las de código propietario varían en precio.

	Licencia	Precios ($1 = 0,77€)		
		V. completa	V. académica	Actualizaciones
Denemo	GPL	gratis		
Encore	Propietaria	$399,99	$299,99	$129,99
Finale	Propietaria	$600	$350	$139,95
Frescobaldi	GPL	gratis		
Guitar Pro	Propietaria	59,95€	–	29,95€
Harmony A.	Propietaria	70€	–	gratis
LenMus	GPL	gratis		
Lilypond	GPL	gratis		
MuseScore	GPL	gratis		
NWC	Propietaria	$49	–	$15
NtEd	GPL	gratis		
Sibelius	Propietaria	$599,95	$295	$49,95 – $149,95

Tabla 3.1: Comparativa de precios en el mercado

Formatos de archivos soportados Algunas aplicaciones soportan múltiples formatos, pero en esta tabla vamos a mencionar los más usados.

	Entrada	Salida
Denemo	ly xml midi	png pdf midi audio ly
Encore	enc midi mus	pdf midi
Finale	mus xml midi	epub midi xml wav mp3 pdf jpg png
Frescobaldi	ly xml	ly xml pdf
Guitar Pro	midi ascii xml powerTab	midi ascii xml wav png pdf powerTab
Harmony A.	midi abc tab enc mus nwc gtp xml	wav aiff ogg mp3 midi abc tab xml
LenMus	lmb	lmb
Lilypond	ly	pdf midi
MuseScore	mscz mscx xml midi	pdf png svg wav ogg mp3 xml midi ly
NWC	nwc midi	nwc nwctxt midi
NtEd	ntd midi xml	ntd midi pdf png svg ly
Sibelius	sib midi mus tab	sib midi tab

Tabla 3.2: Formatos de archivo soportados por las aplicaciones

Interfaz

Después de observar las interfaces de los programas, hemos visto grandes diferencias entre ellos. Mientras que algunas ofrecen demasiadas funcionalidades avanzadas (Finale, Sibelius, ...), otras se centran más en simplificar la aplicación para un manejo más sencillo (MuseScore, Denemo, nted). En el otro extremo, Lilypond no tiene una interfaz gráfica, por lo que para utilizarlo hay que trabajar con la línea de comandos.

Nuestra idea sería crear una aplicación simple, del estilo de Denemo y MusikEbal.

Denemo (fig. 3.2) es una aplicación muy sencilla que tiene todas funcionalidades básicas visibles. En el lado izquierdo tiene columnas con las distintas duraciones y alturas de las notas. La barra superior, estandarizada en las aplicaciones de escritorio, nos permite conocer de antemano su función, por ejemplo el icono de imprimir.

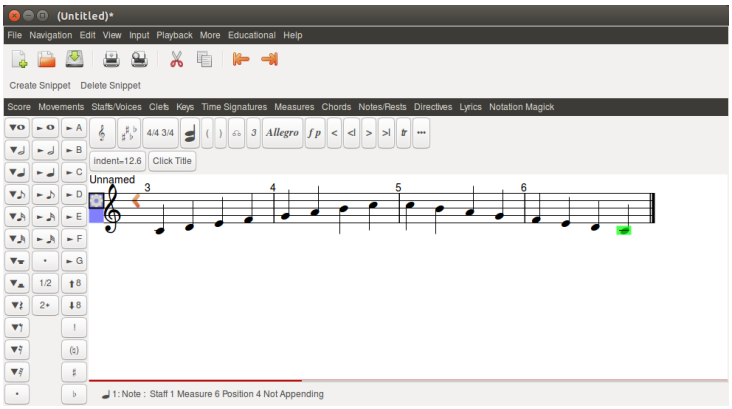


Figura 3.2: Captura de pantalla de Denemo

MusikEbal (fig. 3.3) es todavía más sencilla que Denemo. Cualquier usuario con conocimientos musicales básicos podría crear sus propias partituras. A diferencia de Denemo, las funcionalidades las tiene en la zona inferior de la ventana. Creemos que la organización de los iconos en Denemo (a la izquierda) es más cómodo de usar, ya que la zona inferior quedará libre para el resto de la partitura.

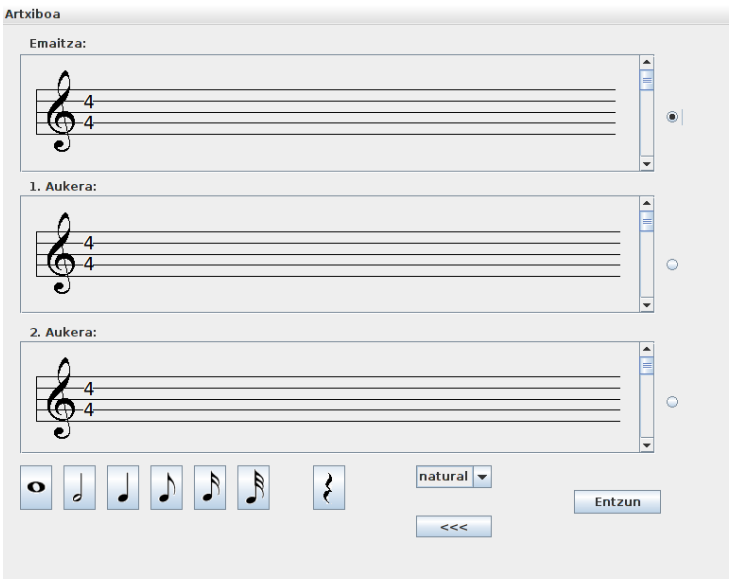


Figura 3.3: Captura de pantalla de MusikEbal

3.3. Identidad visual

Se decidió llamar "Kosmos" a la aplicación, dado que es la raíz de "Mikrokosmos", el título de una colección de obras músicales del compositor húngaro "Béla Bartók". Se ha jugado con dos colores, el blanco y el negro. La razón es que en el mundo de la música sólo se hace uso de estos colores logrando con ellos multitud de acciones.

Por otro lado, se planteó que la pantalla principal llevara algun elemento representativo de la música. En este caso se escogió el piano, instrumento que todo el mundo asocia con la música más facilmente. Otra de las razones por la que nos llevó a elegir este instrumento fue que en él se ven reflejados muchos de los atributos que la música posee (bemol, sostenido, becuadro, notas, escalas) entre otras. Si se llegara a poner otro instrumento no se podría contemplar todo esto de una manera tan sencilla. Para finalizar, se decidió no hacer uso excesivo de los botones para evitar confusiones y lograr un manejo más intuitivo e interactivo.

Capítulo 4

Captura de Requisitos

4.1. Version antigua (basada en MusicXML)

MusicXML es un formato de notación musical basado en el lenguaje de marcado XML. Gracias a que su uso ha sido estandarizado y además es de código abierto, una gran cantidad de programas lo han aplicado como medio de intercambio de partituras entre distintos programas. Para más información sobre MusicXML, ver anexo A en la página 63.

Ya que la aplicación va a manejar archivos de formato MusicXML, se analizará la estructura para organizar internamente la jerarquía de clases.

4.1.1. Actores

El único tipo de usuario que va a interactuar con nuestro programa es el profesor. En la imagen 4.1 vemos los casos de uso que el profesor podrá usar, y se explican en la próxima sección.

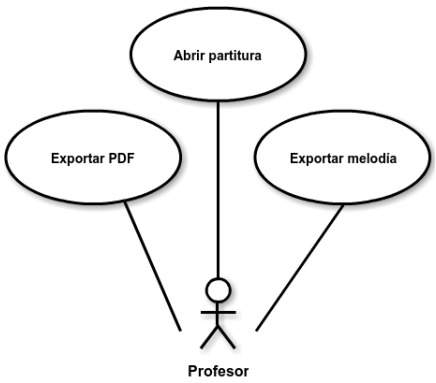


Figura 4.1: Casos de uso: profesor

4.1.2. Casos de uso del profesor

A continuación se explicarán los casos de uso en los que intervenga el profesor:

4.1.2.1. Abrir partitura

Descripción Junto con "crear partitura", este será otro botón de la pantalla inicial del programa. Mostrará una ventana emergente para elegir qué archivo abrir, y una vez abierto aparecerá una ventana similar a la de "crear partitura" pero con la partitura del archivo abierto.

Precondición Ejecutar el programa o una vez ejecutado el programa pinchar el botón abrir partitura.

Requisitos no funcionales

Postcondición Se muestra una partitura creada por el usuario guardada anteriormente.

Flujo de eventos

1. El usuario pulsa el botón de abrir partitura.
2. En la siguiente pantalla emergente el usuario indicará qué partitura desea abrir.
3. El programa leerá el archivo con formato MusicXML.
4. Se generará la partitura en nuestro programa.

4.1.2.2. Exportar PDF

Descripción Una vez terminada la partitura, se podrá exportar a un documento que el usuario podrá visualizar fácilmente.

Precondición

- La partitura a exportar estará abierta

Requisitos no funcionales

Postcondición

- El documento PDF estará creado y se abrirá

Flujo de eventos

1. El usuario pulsa en el botón "Exportar PDF".
   - Si la partitura no está guardada
     a) Se llamará a la función guardar partitura
2. En la siguiente pantalla el usuario indicará la ruta donde quiere guardar el documento.
   - Si el archivo a exportar existe:
     a) Sobrescribir el antiguo archivo.
   - Si el archivo a exportar no existe:
     a) Generar un nuevo archivo.
3. El módulo "exportar Lilypond" abrirá la partitura MusicXML e irá leyendo las etiquetas para transformarlo en un archivo Lilypond (`.ly`).
4. Una vez terminada la conversión, se procesará el archivo `.ly` con Lilypond para que se genere el documento `PDF`.

4.1.2.3. Exportar melodía

Descripción Permite guardar en un archivo MIDI el sonido de la partitura.

Precondición

- La partitura a exportar estará abierta

Requisitos no funcionales

Postcondición

- El archivo que contenga la melodía estará creado y comenzará a sonar

Flujo de eventos

1. El usuario pulsa en el botón "Exportar MIDI".
   - Si la partitura no está guardada
     a) Se llamará a la función guardar partitura
2. En la siguiente pantalla el usuario indicará la ruta donde quiere guardar la melodía.
   - Si el archivo a exportar existe:
     a) Sobrescribir el antiguo archivo.
   - Si el archivo a exportar no existe:
     a) Generar un nuevo archivo.
3. El módulo "exportar Lilypond" abrirá la partitura MusicXML e irá leyendo las etiquetas para transformarlo en un archivo Lilypond (`.ly`).
4. Una vez terminada la conversión, se procesará el archivo `.ly` con Lilypond para que se genere la melodía en formato MIDI.

4.1.3. Modelo de dominio

En la figura 4.2 podemos ver todos los tipos de objetos que vamos a utilizar en el contexto del sistema y la relación que hay entre ellos. Hemos decidido nombrar los elementos en inglés, ya que es el idioma usado por MusicXML.

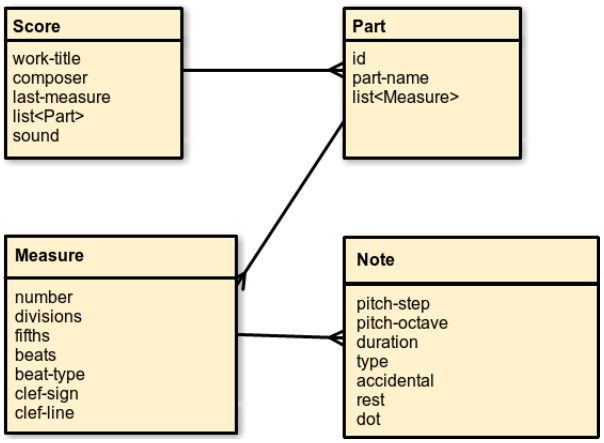


Figura 4.2: Modelo de dominio

Score Esta será la clase principal.

- work-title: título de la partitura
- composer: compositor de la partitura
- last-measure: número de compases
- list<Part>: listado de partes
- sound: compás en el que esté reproduciéndose o pausado el sonido

Part Parte de la partitura que contiene compases

- id: identificador interno de la parte
- part-name: nombre visible de la parte
- list<Measure>: listado de compases

Measure Compás

- number: número del compás. Empieza en 1, salvo que el primero sea anacrusa.
- divisions: división mínima respecto de la negra en la partitura. Ver columna *Div.* de la tabla A.3 en la página 65.
- fifths: armadura. El rango irá de -7 (7 bemoles) a 7 (7 sostenidos). 0 indicará que no tiene alteraciones. Ver tabla A.4 en la página 64.
- beats: numerador del compás. Se indicará con un número entero.
- beat-type: denominador del compás Se indicará con un número entero. Ver columna *D.C.* de la tabla A.3 en la página 65.
- clef-sign: símbolo de la clave. Para la clave de sol, será G.
- clef-line: línea de colocación de la clave. Si la clave la queremos colocar en la segunda línea, introduciremos un 2.
- list<Note>: listado de notas

Note Nota

- pitch-step: nombre de la nota. Ver tabla A.5 en la página 65.
- pitch-octave: octava de la nota. Se indicará con un número entero.
- duration: duración. Se indicará con un número entero.
- type: tipo de figura. Ver columna *Inglés (USA)* de la tabla A.3 en la página 65.
- accidental: alteración accidental.
- rest: silencio.
- dot: puntillo.

4.2. Versión nueva (basada en jMusic)

Tras el análisis de la estructura de MusicXML, se decidió buscar lenguajes de notación musical similares, que fueran más cómodos de manejar. En este caso, jMusic fue la herramienta más factible. Gracias a ella, se prescindió de las clases creadas anteriormente y se usaron las clases implementadas que ofrecía jMusic, junto con sus respectivos métodos. Para profundizar más en el ámbito de jMusic, véase el anexo B en la página 67.

4.2.1. Actores

El único tipo de usuario que va a interactuar con nuestro programa es el profesor. En la imagen 4.3 vemos los casos de uso que el profesor podrá usar, y se explican en la próxima sección.

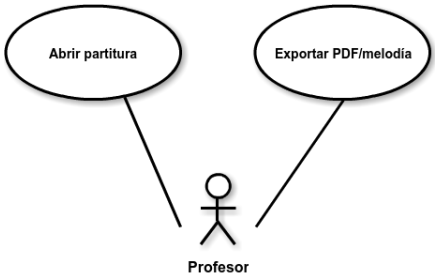


Figura 4.3: Casos de uso: profesor

4.2.2. Casos de uso del profesor

En este apartado se detallarán los nuevos casos de uso basados en jMusic, en los que intervenga el profesor:

4.2.2.1. Abrir partitura

Descripción Junto con "crear partitura", este será otro botón de la pantalla inicial del programa (fig. 4.4). Mostrará una ventana emergente para elegir qué archivo abrir, y una vez abierto aparecerá una ventana similar a la de "crear partitura" pero con la partitura del archivo abierto.

Precondición Ejecutar el programa o una vez ejecutado el programa pinchar el botón abrir partitura.

Requisitos no funcionales

Postcondición Se muestra una partitura creada por el usuario guardada anteriormente.

Flujo de eventos

1. El usuario pulsa el botón de abrir partitura.
2. En la siguiente pantalla emergente el usuario indicará qué partitura desea abrir.
3. El programa comprobará que el archivo tenga un formato MusicXML.
   - Si el formato es correcto, leerá el archivo.
   - Sino, mostrará un error.
4. Se generará la partitura en nuestro programa.

Prototipo de la interfaz Abrir partitura

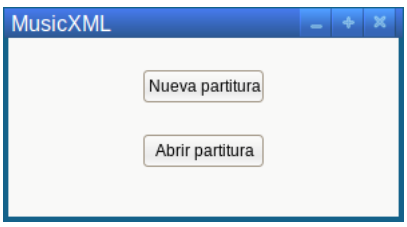


Figura 4.4: Interfaz Menú Principal

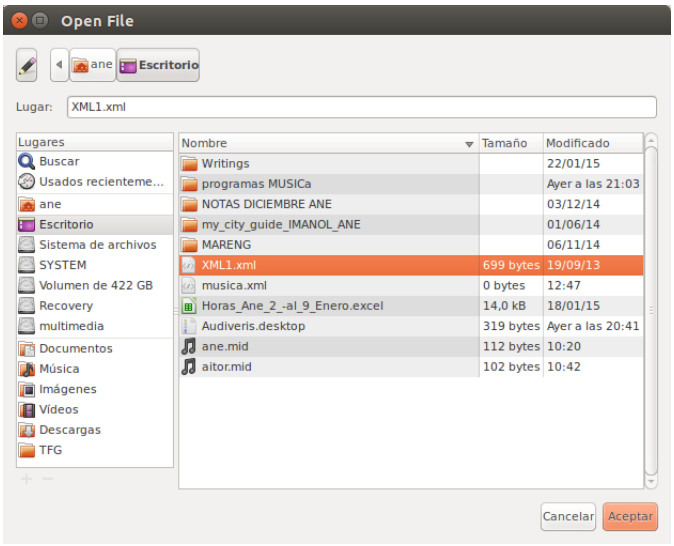


Figura 4.5: Interfaz Abrir partitura

4.2.2.2. Exportar PDF/MIDI

Descripción Una vez terminada la partitura, el usuario podrá elegir si exportarlo a un documento PDF, una melodía en formato MIDI o ambas.

Precondición

- La partitura a exportar estará abierta

Requisitos no funcionales

Postcondición

- Se creará el documento PDF, el MIDI o ambos, según haya elegido el usuario.

Flujo de eventos

1. El usuario pulsa en el botón "Exportar PDF/MIDI".
2. El usuario elige la ruta y el formato (o los formatos) a exportar la partitura.
   a) Si el usuario no ha elegido nngún formato, mostrar una alerta para que lo elija.
3. Se llamará a la función "guardar partitura" para guardarla en formato MusicXML.
4. Se abrirá el archivo MusicXML mencionado anteriormente y se irá convirtiendo a un archivo de formato Lilypond (`.ly`).
5. Una vez terminada la conversión, se procesará el archivo `.ly` con Lilypond para que se generen los archivos solicitados por el usuario.

Prototipo de la interfaz Exportar partitura

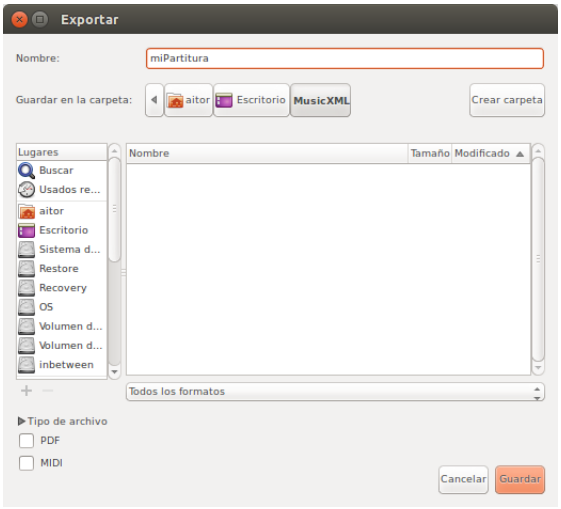


Figura 4.6: Interfaz "Exportar PDF"

4.2.3. Modelo de dominio

En la figura 4.7 podemos ver las clases que utiliza jMusic para el manejo de la partitura. Además, crearemos nuestra clase ScoreInformation para manejar datos que jMusic no nos proporciona y para tener a mano la instancia de la partitura.

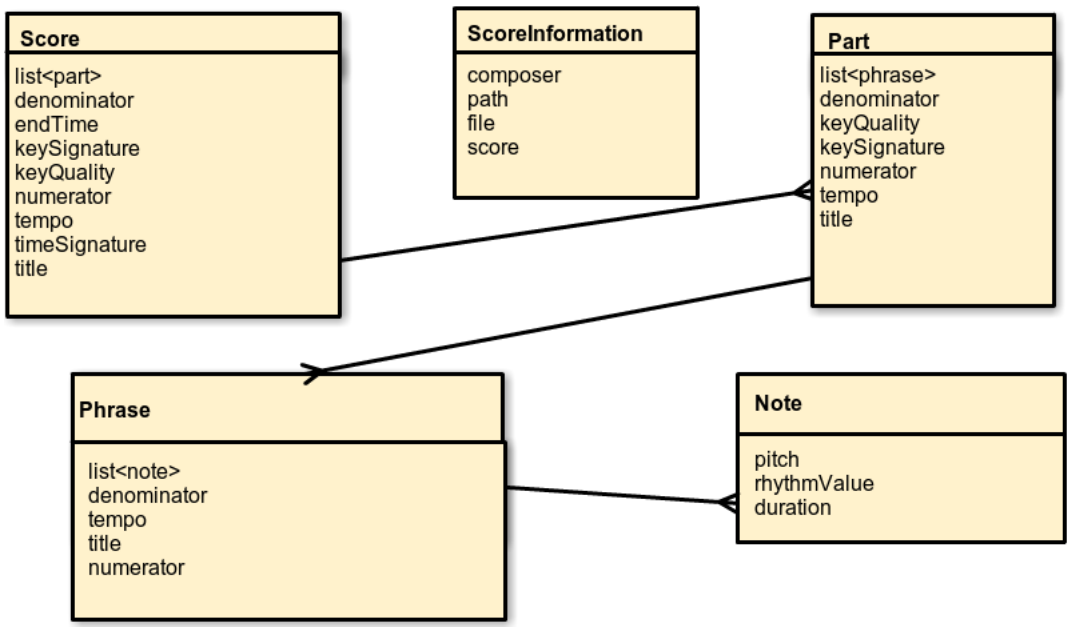


Figura 4.7: Modelo de dominio jMusic

ScoreInformation Clase principal MAE.

- composer: compositor o autor de la partitura.

- path: ruta del archivo que contiene la partitura.
- file: puntero del archivo que contiene la partitura.
- score: instancia de la partitura en formato jMusic.

Score Esta es la clase principial. Digamos que sería todo lo que conlleva una partitura. Veamos ahora sus componentes más importantes.

- list<part>: Tiene una lista de partes, o por lo que la mayoría conocemos como pentagramas.
- denominator: Dentro de la fracción del compás, es el número inferior. Indica qué figura ocupa cada parte en la que está dividido el compás.
- endTime: Nos dice el número del último compás.
- keyQuality: Nos dice en que tonalidad esta la obra. Si es 0 está en Mayor, y si es 1 la obra está en menor.
- keySignature: Armadura. Nos dice el número de sostenidos (sharps) o el número de bémoles (flats).
- numerator: Dentro de la fracción del compás, es el número superior. Indica cuantas partes tiene cada compás.
- tempo: El tempo que tiene la canción.
- timeSignature: compás. Es una fracción formada por numerador (número superior) y denominador (número inferior).
- title: Título que le hemos puesto a la obra.

Part Parte de la partitura que contiene compases.

- list<phrase>: Contiene la lista de las frases, de las melodías del pentagrama.
- denominator
- keyQuality
- keySignature
- numerator
- tempo
- title: nombre del pentagrama

Phrase Esta clase tiene una lista de las notas que contiene. Podríamos decir que es el equivalente a compás.

- list<note>: lista de las notas que contiene cada phrase.
- denominator
- numerator
- tempo
- title

Note La última clase de la lista. Pero no la menos importante.

- pitch: El tono de la nota.
- rhythmValue: La figura de la nota.
- duration: Duración de la nota.

Capítulo 5

Análisis y Diseño

5.1. Transformación del modelo de dominio

Una vez diseñado el modelo de dominio, haremos los cambios necesarios para adaptarlo a un diagrama de clases.

5.1.1. Versión antigua (basada en MusicXML)

Como se puede ver en la figura 5.1.

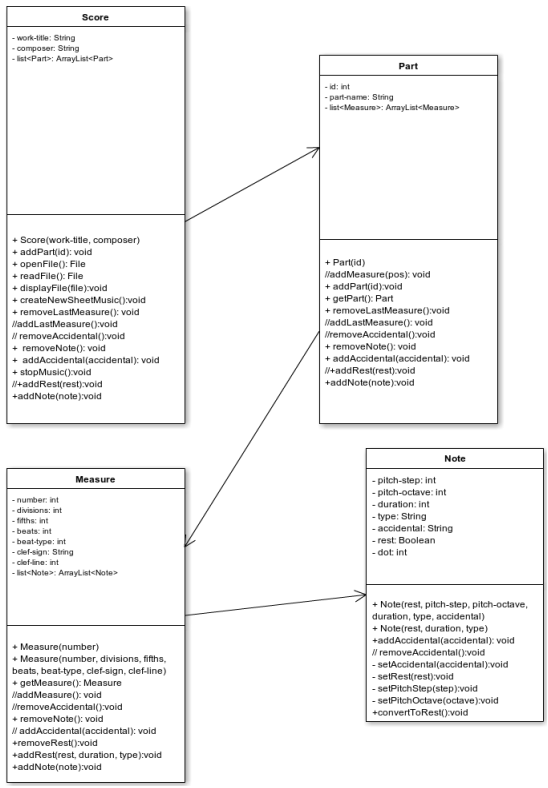


Figura 5.1: Transformación del modelo de dominio

5.1.2. Versión nueva (basada en jMusic)

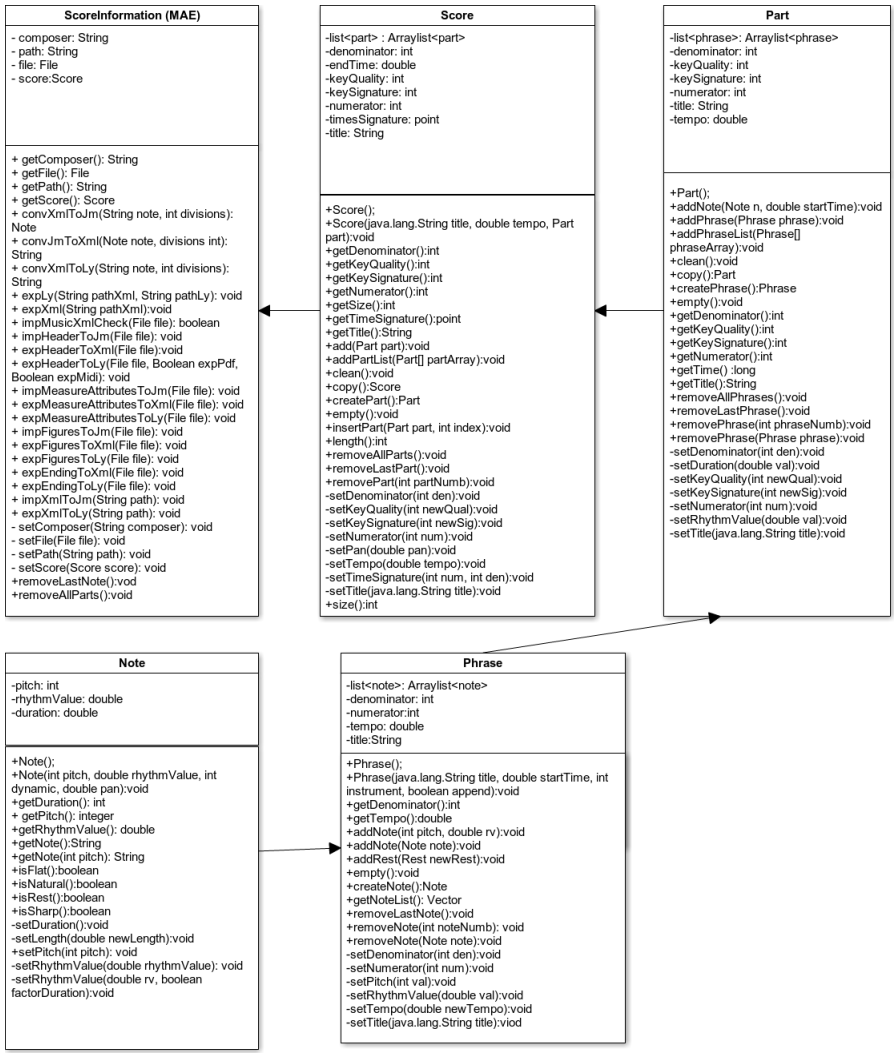


Figura 5.2: Transformación del modelo de dominio

5.2. Diagramas de secuencia

En este apartado se visualizarán los diagramas de secuencia basados en jMusic:

5.2.1. Abrir partitura

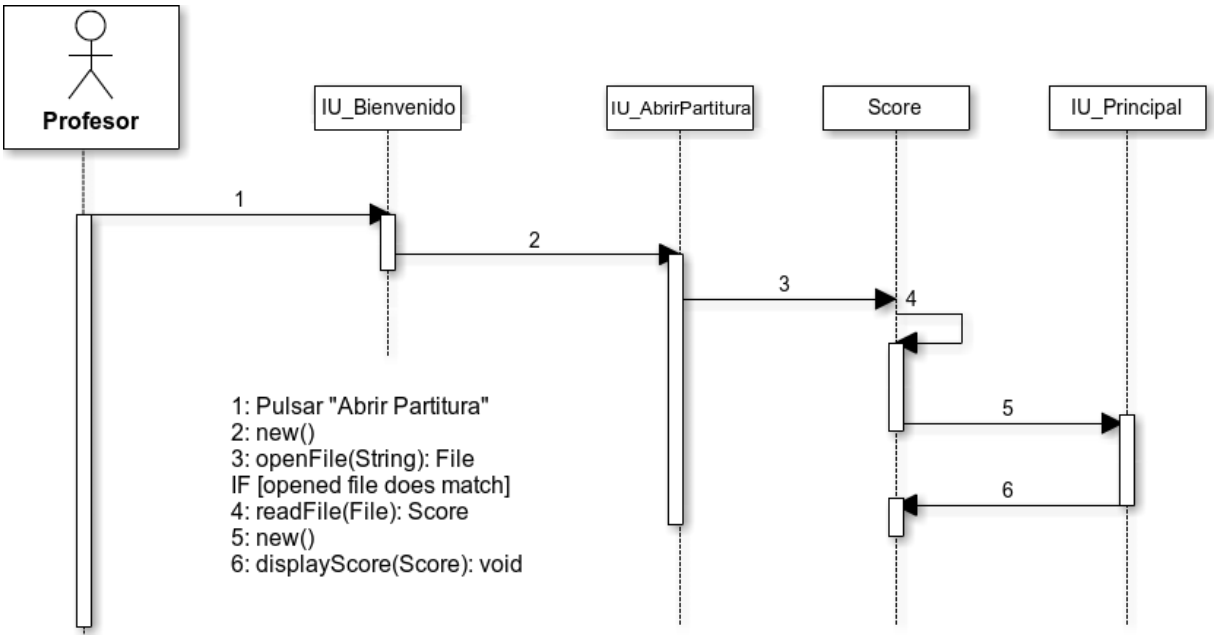


Figura 5.3: Abrir Partitura

5.2.2. Exportar PDF/melodía

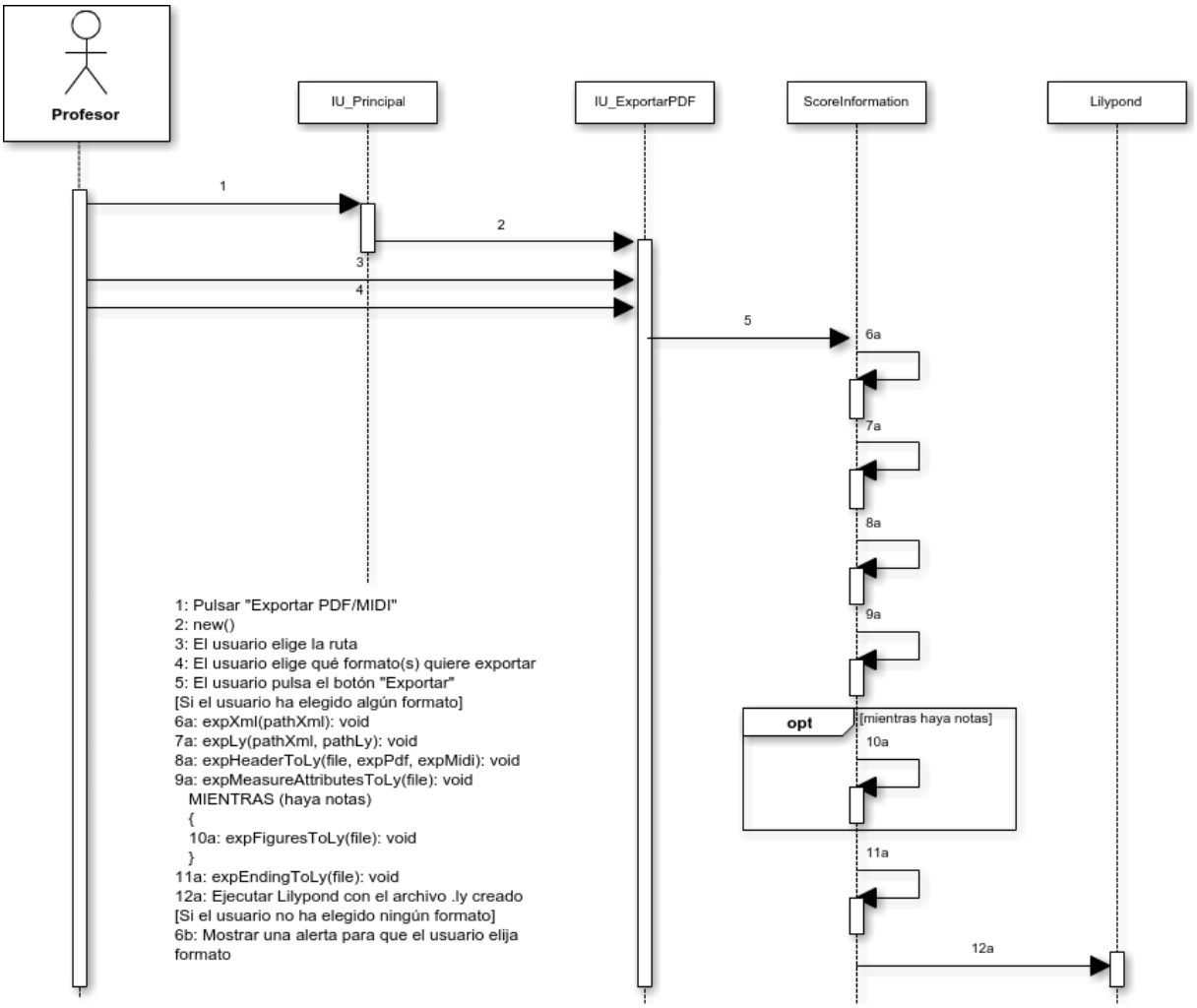


Figura 5.4: Abrir Partitura

5.3. Módulos

Los módulos son pequeñas aplicaciones en las que dividiremos cada una de las tareas que tenga el programa principal. Entre ellos podremos hallar conversores entre distintos lenguajes musicales, la apertura de partituras en formato MusicXML y el exportado a documentos PDF y melodías en formato MIDI.

5.3.1. durationConversion

Dado que para el proyecto se precisará el uso de tres lenguajes diferentes (MusicXML, JMusic, LilyPond) habrá que crear varios programas que hagan conversiones entre ellos.

Después de considerar diferentes posibilidades, las combinaciones a usar serán:

1. convXmlToJm: Este es el conversor usado para abrir las partituras. Se importará el archivo XML para añadir elementos musicales a jMusic.

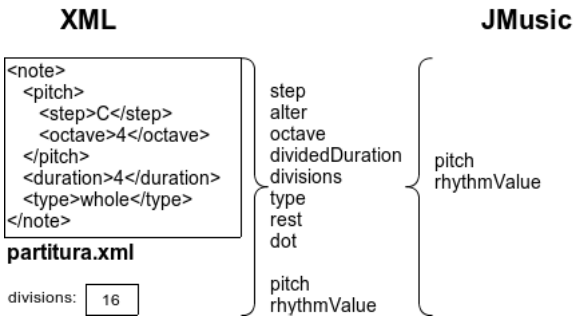


Figura 5.5: Conversor de XML a jMusic

2. convJmToXml: Este es el conversor usado para guardar las partituras. Se leerán los elementos de jMusic para exportarlos a XML.

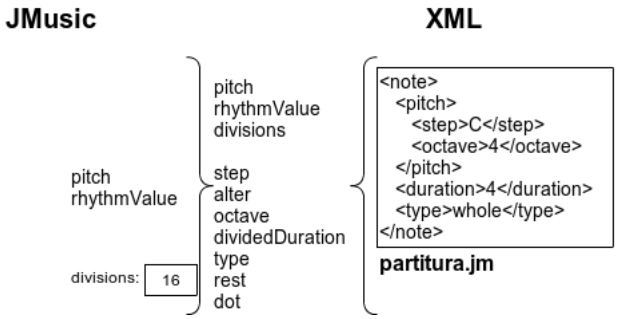


Figura 5.6: Conversor de jMusic a XML

3. convXmlToLy: Este es el conversor usado para exportar las partituras. Se abrirá el archivo XML para leer los elementos y exportar la partitura en un documento PDF.

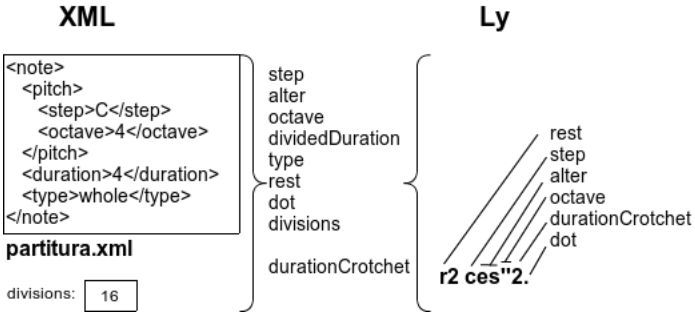


Figura 5.7: Conversor de XML a LilyPond

5.3.2. Abrir partitura (Importar MusicXML)

Nuestra aplicación hace uso de archivos XML. La función de abrir importará un archivo MusicXML e irá leyendo su contenido. Cada vez que encuentre algo significativo, irá creando y generando objetos de jMusic para luego mostrar todo el contenido de la partitura en la aplicación.

Abrir partitura se subdivide en los siguientes módulos:

5.3.2.1. impHeaderToJm

El programa buscará la etiqueta <score-partwise> para obtener los detalles de la cabecera. Podemos encontrar un ejemplo en la tabla 5.1 donde encontraremos:

- el título de la partitura (`work-title`, línea 5)
- el autor (`<creator type="composer">`, línea 8)
- el nombre interno de la parte (`<score-part>`, línea 14)
- el nombre externo de la parte (`<part-name>`, línea 15)
- la parte (`<part>`, línea 18)
- el contenido del primer compás (`<measure>`, línea 19)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD␣MusicXML␣3.0␣Partwise//EN" "http://www
       .musicxml.org/dtds/partwise.dtd">
3  <score-partwise version="3.0">
4     <work>
5        <work-title>Titulo</work-title>
6     </work>
7     <identification>
8        <creator type="composer">Autor</creator>
9        <encoding>
10          <software>XMLScore</software>
11       </encoding>
12    </identification>
13    <part-list>
14       <score-part id="P1">
15          <part-name>Voice</part-name>
16       </score-part>
17    </part-list>
18    <part id="P1">
19       <measure number="1">
```

Tabla 5.1: Ejemplo de entrada a manejar por impHeaderToJm

5.3.2.2. impMeasureAttributesToJm

El módulo impMeasureAttributesToJm se encargará de coger los atributos del compás. Podemos encontrar un ejemplo en la tabla 5.2 donde encontraremos:

- Divisiones de negra (`<divisions>`, línea 2)
- La armadura (`<fifths>`, línea 4)
- Los pulsos (`<beats>`, línea 7)
- La subdivisión del compás (`<beat-type>`, línea 8)
- Figura de la clave (`<sign>`, línea 11)
- Posición de la figura (`<line>`, línea 12)

```
1  <attributes>
2     <divisions>1</divisions>
3     <key>
4        <fifths>0</fifths>
5     </key>
6     <time>
7        <beats>4</beats>
8        <beat-type>4</beat-type>
9     </time>
10    <clef>
11       <sign>G</sign>
12       <line>2</line>
13    </clef>
14 </attributes>
```

Tabla 5.2: Ejemplo de entrada a manejar por impMeasureAttributesToJm

5.3.2.3. impFiguresToJm

Como ir creando cada una de los elementos (notas y silencios) de una partitura. Podemos encontrar un ejemplo en la tabla 5.3 donde encontraremos:

- Nombre de la nota (`<step>`, línea 3)
- Posición de la nota en el pentagrama (`<octave>`, línea 4)
- Duración de la nota (`<duration>`, línea 6)
- El tipo de nota (negra, blanca, redonda) (`<type>`, línea 7)

```
1  <note>
2     <pitch>
3        <step>C</step>
4        <octave>4</octave>
5     </pitch>
6     <duration>4</duration>
7     <type>whole</type>
8  </note>
```

Tabla 5.3: Ejemplo de entrada a manejar por impFiguresToJm

5.3.2.4. impMusicXMLCheck

Este programa tendrá que identificar si el archivo a importar está realmente en formato MusicXML o no. Para ello tendrá que identificar las etiquetas `<?xml>` y `<!DOCTYPE score-partwise>`. De ser así llevaremos a cabo la importación de dicho archivo.

5.3.3. Exportar PDF/MIDI (Exportar Lilypond)

Para exportar la partitura a un documento PDF o a una melodía en formato MIDI, la aplicación convertirá la partitura de formato MusicXML a un archivo `.ly`. Una vez completado, Lilypond procesará dicho archivo. Para más información sobre Lilypond, ver anexo C en la página 73.

5.3.3.1. expHeaderToLy

El módulo expHeaderToLy creará la cabecera del archivo Lilypond. Podemos ver un ejemplo en el cuadro 5.4. En él se anotarán:

- la versión mínima necesaria de Lilypond (`\version`, línea 1)
- la partitura (`ly-score`, línea 4)

- información para configurar el documento PDF (`\layout`, línea 9)
- información para exportar la melodía MIDI (`\midi`, líneas 10-12)
- el tempo de la partitura (`\tempo`, línea 11)
- el pentagrama (`\new Voice`, línea 15)

```
1  \version "2.18.2"
2
3  \book {
4     \score {
5
6        \header {
7           title = "Mi obra de arte"
8           composer = "Don Anónimo"
9        }
10       \layout { }
11       \midi {
12          \tempo 4 = 120
13       }
14
15       \new Staff {
16          \new Voice {
```

Tabla 5.4: Ejemplo de resultado de expHeaderToLy

5.3.3.2. expMeasureAttributesToLy

El módulo expMeasureAttributesToLy se encargará de guardar los atributos del compás. Podemos ver un ejemplo en el cuadro 5.5. Para más información sobre las variables de Lilypond para los compases, ver sección C.2.1.4 en la página 75.

```
1                \clef treble
2                \key c \major
3                \time 4/4
```

Tabla 5.5: Ejemplo de resultado de expMeasureAttributesToLy

5.3.3.3. expFiguresToLy

Con expFiguresToLy iremos creando cada una de las figuras (notas y silencios). Podemos ver un ejemplo de entrada en el cuadro 5.6. Más detalles acerca de la escritura de notas en Lilypond en la sección C.2.1.4 en la página 75.

```
1                r2 ces''2.
```

Tabla 5.6: Ejemplo de resultado de expFiguresToLy

Capítulo 6

Implementación

En esta sección se explicarán las partes más significativas e interesantes del código del proyecto.

6.1. Paquete *classes*

El paquete classes lo forman aquellas clases de Java creadas específicamente para el proyecto de Kosmos.

- FigureJm.java: se creó la clase FigureJm a modo de contenedor universal de notas, tanto para aquellas que formen parte de jMusic como MusicXML o Lilypond. Por lo tanto, esta clase carece de métodos complejos y solamente se usa como almacenamiento y asignación de ciertos valores predeterminados.
- ScoreInformation.java: es una de las clases creadas aparte para poder tener guardados ciertos datos de una partitura (score). Se creó esta clase tanto para opciones internas del programa (*divisions* para el manejo de archivos MusicXML) como para datos vitales que carecían otras clases ya existentes o no eran tan fáciles de recuperar (*composer* para guardar el autor de la partitura, *clef* con el tipo de clave)
- WizardData.java: nos encontramos ante otra clase creada para poder almacenar datos elegidos por el usuario en la interfaz *GUI_CreateScore* para crear la partitura. Se podrá tanto obtener como modificar el valor de los atributos mediante los metodos get() y set().

6.2. Paquete *gui*

Son las clases que tienen interfaz, con las que el usuario podrá interactuar.

- GUI_Splash.java: es la clase principal, la que se inicia al cargar el programa. Tiene una interfaz sencilla en la cual el usuario únicamente deberá pinchar con el ratón para pasar a la siguiente ventana.
- GUI_Welcome.java: la interfaz que nos encontramos tras haber pinchado sobre la anterior *(GUI_Splash)* es *GUI_Welcome*. En esta ventana nos encontraremos con dos botones *(Crear Partitura* y *Abrir Partitura)* sobre un fondo negro. Al pinchar sobre los botones se llevarán acabo diferentes acciones.
- GUI_CreateScore.java: esta clase contiene funciones para crear la partitura que el usuario desee en ese momento. Esto es, nos encontramos ante una ventana con multiples opciones, como *clave, compás, tempo* entre otros, que podrá elegir el usuario e irán creando una nueva partitura partiendo de los datos elegidos.
- Notate.java: la interfaz donde el usuario podrá visualizar la partitura con los datos que proporcionó en GUI_CreateScore o tras haber importado una partitura desde el botón *Abrir partitura* de la pantalla de bienvenida. Además podrá modificar la partitura (insertar elementos, borrar elementos...), escucharla, pararla, exportarla a cualquier lugar de su ordenador y podrá importar cualquier partitura, siempre y cuando este en formato *MusicXML*.

6.3. Paquete *modules*

El paquete modules incluye las clases más relevantes al proyecto. Junto con la clase para exportar la partitura en formato MusicXML (ExpJmToXml.java), encontramos las siguientes:

6.3.1. ImpXmlToJm.java

El módulo ImpXmlToJm se encargará de leer una partitura en formato MusicXML y convertirla a la estructura que maneja jMusic. Una herramienta muy útil para el manejo y parseo de archivos de formato XML es *StAX*, de la que se habla con más detalle en el anexo D en la página 77.

Está formado por tres métodos principales: `impHeaderToJm()`, `impMeasureAttributesToJm()` e `impFiguresToJm()`.

6.3.1.1. impHeaderToJm()

`impHeaderToJm` recorre la cabecera del archivo MusicXML y recoge los elementos que necesite.

Su primera tarea es recorrer el archivo hasta encontrar una etiqueta `<score-partwise>`[1]. Si no la encuentra, mostrará un error diciendo que no es un XML de formato MusicXML.

Después de eso, buscará valores generales de la partitura, tales como el autor y título.

6.3.1.2. impMeasureAttributesToJm()

`impMeasureAttributesToJm` buscará los elementos del primer compás para conocer las características generales del pentagrama.

El elemento más crítico de todos los que recogerá es «divisions» ya que servirá para hacer los cálculos de la duración de cada una de las notas. Otros elementos que buscará son «sound» (tempo), «fifths» (armadura), «mode» (modo), «beats» y «beat-type» (compás), «sign» y «line» (clave).

Toda la búsqueda la hará dentro de las etiquetas `<attributes>`. Cuando llegue a la etiqueta de cierre (`</attributes>`), procesará la clave según los elementos «sign» y «line». El resto de valores se han procesado durante la búsqueda.

6.3.1.3. impFiguresToJm()

`impFiguresToJm` recopilará los valores que formen una nota del archivo MusicXML y los irá añadiendo al pentagrama de jMusic. Para completar su cometido, se usará un objeto de la clase `FigureJm`, que almacene todas las características de las etiquetas XML y las colocará en sus respectivas variables. Cuando esté completa, el método `convXmlToJm` creará una nota de formato jMusic e introducirá todos los datos acumulados en el objeto `FigureJm`. Una vez relleno, añadirá la nota al phrase. Cuando acabe con todas las notas, creará la partitura con todas las notas.

Kosmos soporta notas que estén ligadas entre compases, por lo que si después de recopilar todos sus detalles se detecta que la nota contiene un comienzo de ligadura (`tieStart`), seguirá a la siguiente nota sin crear una nueva e irá recalculando las nuevas duraciones hasta que encuentre la última nota ligada (la que contenga la marca `tieStop`, pero no `tieStart`).

6.3.2. ExpJmToLy.java

`ExpJmToLy` es el módulo que, de manera similar a `ExpJmToXml`, realizará el trabajo opuesto a `ImpXmlToJm`: recorrerá el listado de notas y las guardará en un archivo, en este caso de formato Lilypond. Este archivo más tarde se podrá procesar con un programa de idéntico nombre para exportar la partitura en formatos `PDF` y `MIDI`. Dentro de `ExpJmToLy` podremos ver los siguientes métodos principales: `expHeaderToLy`, `expMeasureAttributesToLy`, `expFiguresToLy` y `expEndingToLy`.

[1] Si encuentra una etiqueta `<score-timewise>` mostrará un error diciendo que, aunque puede que sea un archivo MusicXML válido, Kosmos sólo soporta archivos con un formato `<score-partwise>`.

6.3.2.1. expHeaderToLy()

expHeaderToLy escribirá la cabecera del archivo Lilypond.

Todo archivo de Lilypond deberá comenzar indicando la versión mínima de Lilypond necesaria para compilar a partitura. En este caso se incluye la última versión estable que se puede descargar en estos momentos de la página oficial: 2.18.2.

Algunos de los datos que se escribirán con expHeaderToLy son «encodingsoftware» (el programa que ha exportado la partitura, en este caso: Kosmos), «title» (titulo de la partitura), «composer» (autor) y «\tempo» (velocidad).

6.3.2.2. expMeasureAttributesToLy()

expMeasureAttributesToLy exportará los datos generales del pentagrama: «\clef» (clave), «\key» (armadura) y «\time» (compás).

6.3.2.3. expFiguresToLy()

expFiguresToLy obtendrá el listado de notas de jMusic y procesará cada una de ellas para ir convirtiéndolas en notas de Lilypond. Del mismo modo que impFiguresToJm, usará un objeto de la clase FigureJm como intermediario entre jMusic y Lilypond.

Para soportar la división de notas que atraviesen compases, llevará un contador de compases (measureNumber), y otras variables que indiquen la duración máxima de un compás (maxDiv), las duraciones acumuladas del compás actual (collDiv), la duración restante por imprimir de una nota (remDur) y la duración válida que se calcula a la nota actual respecto al espacio libre disponible en el compás (divDur). Si no se va a dividir la nota, remDur y divDur coincidirán.[2]

6.3.2.4. expEndingToLy()

La labor de expEndingToLy es la más sencilla de los módulos de expJmToLy: cerrar las llaves que se hayan abierto en los módulos anteriores, y así la sintaxis del archivo sea correcta.

6.3.2.5. Finally...

Si todos los módulos anteriores se ejecutan correctamente, se cerrará la escritura del archivo y se mostrará una alerta preguntándole al usuario si quiere abrir la carpeta donde está la partitura extraída. Debido a que Lilypond es un programa externo a Kosmos, la conversión del archivo .ly a un PDF solamente la podrá realizar el usuario a mano[3].

[2] Explicado de manera tan breve puede parecer un trabalenguas, pero en el anexo "Implementación: al detalle" está explicado con código y ejemplos.

[3] Como se muestra en la ventana de alerta, para realizar dicha conversión, el usuario deberá tener el programa Lilypond instalado y hacer doble click sobre la partitura de formato .ly extraída por Kosmos.

Capítulo 7

Pruebas de Software

Las pruebas de software son parte del proceso de desarrollo de un sistema. Probar un software es ejecutarlo "con mala idea" para que falle. El objetivo de las pruebas es encontrar errores, no ver lo bien que funciona nuestro software.

7.1. Crear partitura

Cód.	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Crear partitura con los valores predeterminados	Se crea una partitura con los valores predeterminados.	Se crea la partitura con los valores predeterminados	Correcto
2	Crear partitura modificando todos los valores (armadura, clave, título, autor, compás, nota, tipo de nota, modo, intensidad, instrumento y tempo)	Se crea una partitura con los valores elegidos por el usuario	La mayoría de valores se tienen en cuenta pero algunos no varían	
2.1	Elegir fa como tipo de clave	Aparece un pentagrama con una clave de fa	Aparece un pentagrama con una clave de sol	La clave sólo se puede modificar una vez está la partitura en pantalla
2.1a		Aparece un pentagrama con una clave de fa	Aparece un pentagrama con una clave de fa	Correcto
2.2	Elegir un compás con denominador distinto a 4	El compás se muestra con un el denominador elegido	El compás se muestra con un el denominador 4	El método propio `setDenominator()` de jMusic no parece funcionar y no cambia el denominador

Tabla 7.1: Crear partitura

7.2. Modificar partitura creada

Cód.	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Cambiar clave a la partitura	Se cambia la clave pulsando en el submenu pentagrama y por lo tanto la altura de las notas para que vayan acorde con la clave	Se modifica la partitura con la nueva clave y nuevas alturas	Correcto
2	Cambiar armadura (elegir otra cantidad de bemoles o sostenidos)	Se cambian las alteraciones mediante el ratón	Se modifican las alteraciones mediante el ratón	Correcto
3	Cambiar compás	Se cambia el tipo de compás mediante el ratón	Se cambia correctamente el tipo de compás y por lo tanto la medida de cada compás	Correcto con compases entre 1/4 y 9/4, jMusic no nos permite salir de dicho listado de compases
4	Añadir nota mediante letra	Se añade una nota mediante la opción *añadir nota mediante letra*	Se añade una nota al pentagrama	Correcto
5	Borrar último elemento del pentagrama	Se borrará el último elemento del pentagrama ya sea silencio o nota	Se borra el último elemento del pentagrama	Correcto
6	Borrar todos los elementos	Se borrarán todas las notas que estén en la partitura.	Se borran todos los elementos de la partitura.	Correcto
7	Añadir una figura al final del pentagrama	La nueva figura aparecerá al final del pentagrama	La figura aparece al final del pentagrama	Correcto
8	Añadir una figura entre otras dos	La nueva figura aparecerá entre las otras dos, y las que estén más a la derecha irán recolocándose	La figura aparece en su sitio y el resto de notas se colocan automáticamente de modo correcto	Correcto

Tabla 7.2: Modificar partitura creada

7.3. Reproducir

Cód.	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Reproducir una nota	Se escucha el sonido de la nota	Se escucha el sonido de la nota	Correcto
2	Reproducir un silencio	No se deberia de escuchar nada	No se escucha nada	Correcto
3	Reproducir nota con sostenido	Se tiene que escuchar las nota alterada con su sostenido	Se escucha la nota alterada	Correcto
4	Reproducir nota con bemol	Se tiene que escuchar las nota alterada con su bemol	Se escucha la nota alterada	Correcto
5	Reproducir nota con puntillo	Se tiene que escuchar la nota con su duración más la que le añade su puntillo	Se escucha la nota con la duración correcta	Correcto
6	Reproducir silencio con puntillo	El intervalo de silencio durará lo que vale el silencio más su puntillo	El intervalo del silencio con su puntillo correctamente	Correcto
7	Reproducir último compás sólo con notas	Se escuchan solamente las notas del último compás	Se escuchan solamente las notas del último compás	Correcto
8	Reproducir último compás sólo con silencios	No se debería escuchar nada	No se escucha nada durante la duración de los silencios	Correcto

Tabla 7.3: Reproducir 1

Cód.	Descripción	Resultado esperado	Resultado obtenido	Observaciones
9	Repetir último compás sólo con notas	Se repite el último compás hasta que se pulse de nuevo *parar sonido*	Se repite el último compás hasta que se pulse de nuevo p*arar sonido*	Correcto
10	Repetir último compás sólo con silencios	Se repite el último compás pero no se oye nada	No se oye nada	Correcto
11	Repetir todo el pentagrama sin parar	Se repite todo el pentagrama hasta que se pulse de nuevo *parar sonido*	Se repite todo el pentagrama hasta que se pulse de nuevo *para sonido*	Correcto
12	Reproducir trás modificar clave	Se reproduce la partitura con distintas alturas de los elementos	Se reproduce bien	Correcto
13	Reproducir trás modificar compás	Se reproduce la partitura al cambiar el compás y los elementos cambiados	Se reproduce todo bien	Correcto
14	Reproducir trás modificar la armadura de alteraciones	Se reproduce los sonidos de las notas con las alteraciones	Se reproduce bien	Correcto
15	Reproducir notas ligadas	Se reproduce la nota con las duración de la ligadura	Se reproduce bien	Correcto

Tabla 7.4: Reproducir 2

7.4. Importar partitura de formato MusicXML a jMusic

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
1	Listado de archivos XML	Solamente se listan archivos de extensión .xml a la hora de elegir archivo	Aparecen archivos de todas las extensiones	Hay que realizar un filtrado de archivos por extensión
1a			Sólo aparecen archivos con extensión .xml	Correcto
2	Carga de un archivo sin formato XML	Aparece una alerta explicando que no tiene formato XML	Aparece una alerta, pero se cierra la aplicación	Hay que cambiar la estructura del try/catch
2a			Aparece una alerta explicando que no tiene formato XML	Correcto
3	Carga de un archivo XML sin formato MusicXML	Aparece una alerta explicando que no tiene formato MusicXML	Aparece una alerta explicando que no tiene formato MusicXML	Correcto
4	Cancelar la ventana de abrir	Desaparece el diálogo de apertura de archivo y se ve la ventana anterior	Desaparece el diálogo de apertura de archivo y se ve la ventana anterior	Correcto
5	Comprobación de alturas (escala)	Las alturas de las notas son las correctas	Ninguna altura que aparece es la correcta	Todas las notas aparecen un semitono más alto
5a			Las alturas de las notas son las correctas	Correcto
6	Comprobación de duraciones de notas	Las duraciones de las notas son corretas	No todas las duraciones se exportan correctamente	Falta la duración de las semicorcheas
6a			Las duraciones de las notas son corretas	Correcto

Tabla 7.5: Importar partitura de formato MusicXML a jMusic (1)

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
7	Comprobación de duraciones de silencios	Aparecen los silencios con sus duraciones correctas	Aparecen los silencios con sus duraciones correctas	Correcto
8	Comprobación de alteraciones	Aparecen las alteraciones	Aparecen las alteraciones	Correcto
9	Importar nota ligada	Aparece la nota ligada	Se importa incorrectamente la nota ligada	El cálculo para la división de compases es incorrecto
9a			Aparece la nota ligada	Correcto
10	Importar más de un compás	Se importan las notas correctamente	Se importan las notas correctamente	Correcto
11	Importar notas con puntillo	Las notas que tenían puntillo lo muestran	Las notas que tenían puntillo lo muestran	Correcto
12	Importar silencios con puntillo	Se exportan los silencios con sus puntillos correspondientes	Se exportan los silencios con sus puntillos correspondientes	Correcto
13	Importar distintos compases	Se exportan bien todos los compases	Se exportan bien todos los compases	Correcto
14	Importar armadura de sostenidos	Se exportan todos los sostenidos correctamente	Se exportan todos los sostenidos correctamente	Correcto
15	Importar armadura de bemoles	Se exportan todos los bemoles correctamente	Se exportan todos los bemoles correctamente	Correcto
16	Importar clave de fa y de sol	Se exporta la clave elegida	Se exporta la clave elegida	Correcto
17	Importar partitura vacía	Se exporta un pentagrama vacío	Se exporta un pentagrama vacío	Correcto

Tabla 7.6: Importar partitura de formato MusicXML a jMusic (2)

7.5. Exportar partitura de formato jMusic a Lilypond

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
1	Exportar con la extensión .ly	Se exporta la partitura con la extensión .ly	Se exporta la partitura con la extensión .ly	Correcto
2	Exportar sin la extensión .ly	Se exporta la partitura con la extensión .ly	El nombre del archivo no incluye la extensión .ly	Si el usuario modifica la extensión o la elimina, el archivo mantiene ese nombre
2a			Se exporta la partitura con la extensión .ly	Correcto. Se añade la extensión .ly si se ve que el nombre dado por el usuario no lo incluye.
3	Pulsar en exportar y luego cancelar	No se exporta el archivo	No exporta el archivo	Correcto
4	Exportar alturas mediante una escala	Se exporta toda la escala con las alturas adecuadas	La altura no se imprime correctamente	Hay que forzar que los nombres de las notas salga en minúsculas
4a			La altura de las notas está incompleta	No exporta las octavas
4b			Algunas líneas no están indentadas	Añadir idt() en las líneas que lo necesiten
4c			Se exporta toda la escala con las alturas adecuadas	Correcto
5	Exportar todas las duraciones	Se exporta con las duraciones correctas	No todas las duraciones se exportan correctamente	Falta la duración de las semicorcheas
5a			Se exporta con las duraciones correctas	Correcto

Tabla 7.7: Exportar a Lilypond (1)

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
6	Exportar todos los silencios	Los silencios se exportan correctamente	Los silencios se exportan correctamente	Correcto
7	Exportar una escala que incluya alteraciones	Se exporta con las alteraciones correctas	Se exporta con las alteraciones correctas	Correcto
8	Exportar una nota con ligadura	Se exporta la nota con la ligadura	La nota se exporta incorrectamente	El cálculo para la división de compases es incorrecto
8a			Se exporta la nota con la ligadura	Correcto
9	Exportar más de un compás	Se exportan las notas correctamente	Aparecen las ligaduras en todas las notas que acaban un compás, aunque no lo necesiten	Falta una condición en el if para las notas que rellenan el último espacio de un compás y no necesitan se divididas al siguiente compás
9a			Se exportan las notas sin ligaduras extra	Correcto

Tabla 7.8: Exportar a Lilypond (2)

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
10	Exportar todas las notas con puntillo	Se exportan las duraciones con sus puntillos	No se exporta la etiqueta <type> en las notas con puntillo	El conversor de duración a nombre de figura no tiene en cuenta las notas con puntillo
10a			Las notas con puntillo se exportan bien	Correcto
11	Exportar todos los silencios con puntillo	Se exportan los silencios con sus puntillos correspondientes	Se exporta bien	Correcto
12	Exportar armadura de sostenidos	Se exportan todos los sostenidos correctamente	Se exportan todos los sostenidos correctamente	Correcto
13	Exportar armadura de bemoles	Se exportan todos los bemoles correctamente	Se exportan todos los bemoles correctamente	Correcto
14	Exportar clave de fa y de sol	Se exporta la clave elegida	Se exporta la clave elegida	Correcto
15	Exportar partitura vacía	Se exporta un pentagrama vacío	Se exporta un pentagrama vacío	Correcto

Tabla 7.9: Exportar a Lilypond (3)

Capítulo 8

Conclusiones

Acababa de comenzar la carrera y ya estaba dándole vueltas al trabajo fin de grado (en aquellos tiempos todavía se llamaba *proyecto fin de carrera*). Puede sonar al típico dicho de *comenzar la casa por el tejado*, pero creo que nunca es demasiado pronto para ir recopilando ideas. Siendo desde hace tantos años la informática y la música mis dos mayores pasiones, sabía que tenía que hacer algo con ellas; y a ser posible, juntas. Así fue como un 29 de octubre de 2009 escribí como título en una hoja Din A4 "Reproductor de música con Recomendaciones". La idea era hacer un reproductor de música que tuviera guardadas características sobre cada canción: unas técnicas (velocidad, compás, tonalidad, ...) y otras más subjetivas (el género musical, si la canción es triste, ...). Al abrir la aplicación aparecerían varias opciones como elegir artista/disco/canción, modo aleatorio, o el más interesante de todos: un modo *encuesta rápida*. Esta opción haría que con unas pocas preguntas contestables en medio minuto o menos, se generara un listado de canciones basándose en tus gustos y estado de ánimo actual. Si la canción que suena no te motiva, la pasas y que el programa vaya mejorando la *playlist*. Después de comentárselo a varios profesores, me dijeron que no terminaban de verlo viable, y me propusieron distintas alternativas. Así que dejé la idea aparcada por si en algún momento me volvía la inspiración y me proponía hacerlo como *hobby*.

Pasaron los años y cuando se acercaba la hora de elegir proyecto, recibí un correo con el listado de los propuestos por profesores. Algunos eran bastante interesantes y prometían ser un gran reto, en el mejor de los sentidos. Pero había uno que me llamó la atención por encima de todos. El título rezaba "XMLScore: Representación gráfica y reproducción de partituras en formato XML". Y a mí que tanto me gustan la tipografía y las partituras, no lo dudé y lo elegí como primera opción. Javilo, mi tutor, acabó asignándome el proyecto, y otros compañeros de clase tuvieron la oportunidad de trabajar en otros relacionados. De hecho, después de hablarlo con Ane, decidimos trabajar en conjunto para poder llevar a cabo nuestros proyectos de manera colaborativa. Y así es cómo comenzó el *en-aquellos-momentos-todavía-sin-nombre* Kosmos.

Horas de investigación, búsqueda de herramientas, propuestas para hacer una estructura que fuera cogiendo forma, más investigación, correcciones, cambios de planes, dudas, más ideas, ... Ha sido un viaje de aventuras en el que descubríamos que jMusic, la herramienta que nos permite ver las partituras, era bastante sencilla de manejar, pero había cosas que no terminábamos de ver bien cómo funcionaban. Además, la documentación tenía algunas lagunas que no nos permitían avanzar del todo fácilmente, y encontramos cuestiones preguntadas por internet hace tiempo por otros programadores, y que no terminaron por ser resueltas. Y es que con el tiempo descubrimos que esta herramienta que comenzó por voluntarios allá por 1998, lleva 3 años sin recibir actualizaciones en su repositorio de SourceForge.

Después de haber acabado el proyecto, me ha parecido muy curioso cómo hemos ido aplicando elementos del solfeo a la programación, y cómo hemos podido hacer de *profesores de música* para estos amados y a la vez odiados cacharros llamados ordenadores. Cómo un mismo tema se puede ver desde varios puntos de vista, y cómo la música se basa tanto en las matemáticas como otras tantas áreas. Cómo una misma nota se puede representar de tantas maneras, de la misma forma que una misma palabra se puede decir en distintos idiomas, y su esencia se mantiene en todas ellas.

Y es que todos estos altibajos nos han enseñado que a pesar de los problemas encontrados por el camino, cada vez que alcanzábamos una meta, el premio nos sabía a gloria. Así que una vez aprendido eso, y con este proyecto acabado... ¿porqué no retomar viejos planes que se quedaron *hibernando* en el tintero?

Apéndice A

MusicXML

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <!DOCTYPE score-partwise PUBLIC  "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
3      "http://www.musicxml.org/dtds/partwise.dtd">
4  <score-partwise version="3.0">
5    <part-list>
6      <score-part id="P1">
7        <part-name>Music</part-name>
8      </score-part>
9    </part-list>
10   <part id="P1">
11     <measure number="1">
12       <attributes>
13         <divisions>1</divisions>
14         <key>
15           <fifths>0</fifths>
16         </key>
17         <time>
18           <beats>4</beats>
19           <beat-type>4</beat-type>
20         </time>
21         <clef>
22           <sign>G</sign>
23           <line>2</line>
24         </clef>
25       </attributes>
26       <note>
27         <pitch>
28           <step>C</step>
29           <octave>4</octave>
30         </pitch>
31         <duration>4</duration>
32         <type>whole</type>
33       </note>
34     </measure>
35   </part>
36 </score-partwise>
```

Tabla A.1: Código de ejemplo de MusicXML

El código superior (tabla A.1) es un ejemplo mínimo de archivo en formato MusicXML, disponible en la página oficial. [22] En él se pueden ver todas las etiquetas necesarias para colocar una redonda do en un compás de $^4/_4$, en clave de sol y con una armadura sin alteraciones (ver imagen A.1). El árbol de la tabla A.2 esquematiza estas partes de manera simple.

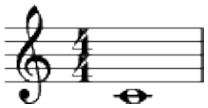


Figura A.1: Representación del ejemplo de MusicXML

- <XML>
- <Doctype>
- <Score-partwise>
  - Listado de partes
  - Parte
    - Compás
      - Cabecera pentagrama
        - División
        - Armadura
        - Compás
        - Clave
      - Notas
        - Altura
          - Nombre
          - Octava
        - Duración
        - Tipo

Tabla A.2: Esquema general de un archivo MusicXML

Como cualquier archivo XML, se empezará con una etiqueta `<?xml>` (línea 1). Para especificar que además será un archivo MusicXML, añadiremos las líneas 2 a 3.

Una partitura de partes (`score-partwise`) (l. 4–36) podrá constar de varias partes, que serán listadas en la etiqueta `<part-list>` (l. 5–9).

Cada parte (l. 10–35) deberá llevar su identificador, que debe coincidir con aquel escrito en el listado de partes. En el ejemplo, la primera (y única) parte está listada como `P1` en la línea 6, y creada en 10, también como `P1`.

Dentro de una parte podremos tener uno o varios compases. En este caso sólo hay uno (l. 11–34). Rellenaremos cada etiqueta `<measure>` con un atributo `number` que indique el número del compás (como en la l. 11).

Cada compás deberá empezar con una etiqueta `attributes` (cabecera del compás) que especifique:

- `divisions` (línea 13): La división más pequeña (respecto a la figura de negra) que queramos usar en toda la partitura. Si la nota más pequeña va a ser una negra, la división será 1, si es blanca: 2, y así sucesivamente (ver columna *Div.* de la tabla A.3). Se usará en relación con `duration` (l. 31).
- `key` (líneas 14–16): Armadura. Se representa con el número de alteraciones. Si el número es positivo, representará sostenidos. Si es negativo, bemoles. Si es 0, no tiene ninguna alteración. (ver tabla A.4).

Sostenidos		Bemoles	
0	0♯/0♭: Do M / la m		
1	1♯: Sol M / mi m	−1	1♭: Fa M / re m
2	2♯: Re M / si m	−2	2♭: Si♭ M / sol m
3	3♯: La M / fa♯ m	−3	3♭: Mi♭ M / do m
4	4♯: Mi M / do♯ m	−4	4♭: La♭ M / fa m
5	5♯: Si M / sol♯ m	−5	5♭: Re♭ M / si♭ m
6	6♯: Fa♯ M / re♯ m	−6	6♭: Sol♭ M / mi♭ m
7	7♯: Do♯ M / la♯ m	−7	7♭: Do♭ M / la♭ m

Tabla A.4: Tabla de armaduras

Figura	Silencio	Div.	D. C.	Español	Inglés (USA)	Inglés (UK)
			1	redonda	whole	semibreve
			2	blanca	half	minim
		1	4	negra	quarter	crotchet
		2	8	corchea	eighth	quaver
		4	16	semicorchea	sixteenth	semiquaver
		8	32	fusa	thirty-second	demisemiquaver
		16	64	semifusa	sixty-fourth	hemidemisemiquaver

Tabla A.3: Notas, silencios, sus códigos y nombres

- `time` (líneas 17–20): Compás. A su vez contendrá dos tags: `beats` y `beat-type`. Este último determinará la figura de subdivisión (el denominador del compás, columna *D. C.* de la tabla A.3), mientras que `beats` indicará la cantidad de notas de dicha duración para formar el compás. Es decir: si `beats` es 3 y `beat-type` es 4, el compás será de duración 3 negras, o lo que es lo mismo $^3/_4$.
- `clef` (líneas 21–24): clave. Dentro de `\clef` habrá dos tags: `sign` determinará el símbolo de la clave (`G` para clave de sol, `F` para clave de fa, `C` para clave de do), y `line` será la línea en que se centre la clave. Por lo tanto: `<sign>F</sign>` y `<line>4</line>` dibujaría una clave de fa en 4^a línea.

Una vez acabada la cabecera del compás, se irá introduciendo una etiqueta `note` por cada nota. Dentro de ella especificaremos:

- `pitch` (líneas 27-30): Altura. Se compone de dos etiquetas: `step` (nombre de la nota, ver tabla A.5) y `octave` (octava en la que se posicione la nota).

A	B	C	D	E	F	G
la	si	do	re	mi	fa	sol

Tabla A.5: Tabla de notas

- `duration` (línea 31): Duración de la nota. Cantidad de divisiones (especificadas en la línea 13 del código) equivalentes a la duración de la nota.

  Con `<divisions>4</divisions>` indicamos que la nota más breve de la partitura será una semicorchea. Si queremos que la nota tenga una duración de negra, necesitaremos `4` semicorcheas.
- `type` (línea 32): Tipo de figura. Aquí se indica la figura que tendrá la nota. Puede parecer redundante, ya que se puede deducir la duración de las notas con la etiqueta anterior (`duration`), pero esta división resulta más sencilla para los programas de notación y reproducción. Por ejemplo, para el cálculo de duraciones acumuladas en un compás se puede ir haciendo una suma en base a las etiquetas `duration`, y el generador de partituras elegirá qué figura 'imprimir' según la etiqueta `type`.
- `accidental`: Alteración accidental.
- `rest`: Silencio. Es un nodo que no necesita ni atributos ni nodos hijo. Si aparece, la figura será un silencio.
- `dot`: Puntillo.

Apéndice B

jMusic

JMusic es una biblioteca de programación musical de código abierto (licencia GPLv2, más información en la sección H en la página 103) escrita en el lenguaje de programación Java. En caso de querer profundizar más acerca de esta herramienta véase el apartado [12] de la bibliografía.

```
1  Note n = new Note(C4, CROTCHET, MF, PAN_CENTRE, CROTCHET * LEGATO);
```

```
1  import jm.JMC;
2  import jm.music.data.*;
3  import jm.util.*;
4
5     public class Dot01 implements JMC {
6        public static void main(String[] args) {
7
8           Note n;
9           n = new Note(C4, QUARTER_NOTE);
10          Phrase phrase = new Phrase();
11          phrase.addNote(n);
12          View.notate(phrase);
13       }
14    }
```

B.1. Declaración de *imports*.

En el código se aprecia que se empieza declarando los *imports*.

¿Cual es su función?

La declaración de los *imports* le indíca al compilador qué clases quieres usar para el programa (aparte de las clases de Java).

Todo programa que utilice jMusic deberá importar algunas clases de jMusic o grupos de clases llamados *packages*.

1. ***import jm.JMC;***

   Es la primera clase importada en la clase jMusic. JMC (J-Music-Constants). Contiene muchas de las instrucciones musicales que hace que el código sea más legible para los músicos, como por ejemplo, QUARTER_NOTE, CHROMATIC_SCALE, C4, MF (i.e., mezzo forte), TRUMPET y muchas más. Hay que tener en cuenta que las constantes de jMusic van siempre en Mayúscula. Esto debería hacer más fácil distinguir entre que palabras son constantes y cuales son variables u otras palabras clave.

2. ***import jm.music.data.*;***

   El siguiente import proporciona acceso completo al paquete de clases. El * indíca que todas las clases dentro del directorio jm.music.data deberían ser accesibles para el programa. Las clases Music/Data incluyen *Note, Phrase, Part* y *Score* que son usadas en casi todos los programas jMusic.

3. ***import jm.util.*;***

   Finalmente, el paquete *jm.util* es importado. Con este paquete queremos usar la clase *View*.

B.2. Arquitectura de las clases

Cada programa que escribas consistirá en una o más clases. Cada método consistirá en una o más declaraciones. Y cada declaración consistirá en una o más palabras. Familiarizarnos con la estructura de las clases es una fase importante dado que trabajas con estos ejemplos e intentas entender cual es la función de cada línea.

B.2.1. Clase (Class)

Cada clase empieza con la declaración de la misma. En este caso:

```
public class Dot01 implements JMC {
```

y acaba con una llave de finalización del programa }. Las clases son casi siempre públicas, significando que pueden ser accesibles desde cualquier otra clase en caso de requerirlo.

Después la instrucción de declaración utiliza la palabra clave *class* para declarar esta sección del código como una clase. Siguiendo esto va el nombre de la clase, en este caso *Dot01*. Por convención el nombre de las clases debe empezar con mayúscula (por el contrario, el nombre de los métodos empieza en minúscula). Opcionalmente, una clase puede implementar alguna funcionalidad adicional. En este caso la clase implementa la funcionalidad de la clase JMC, la cual le permite utilizar las constantes. Finalmente, '{' es usado para delimitar el límite de la clase.

B.2.2. Método (Method)

Cada clase tiene uno o más métodos, y cualquier clase que sea operada directamente debe contener un método llamado *main* que es donde Java empieza a operar el programa. Este simple programa tiene un sólo método *main*, declarado en la siguiente línea.

```
public static void main(String[] args) {
```

... [más y más]

B.2.3. Declaraciones (Statements)

En esta parte del código se deben declarar los objetos. Por ejemplo:

```
Note n;
n = new Note(C4, QUARTER_NOTE);
```

Un objeto del tipo *Note* esta siendo declarado aquí. El objeto es 'n'.

En la primera línea *n* es declarada siendo del tipo Note(una instancia de la clase *Note*).

En la segunda línea *n* es creada y dada valores. Los objetos son creados con la palabra clave *new*, como en *new Note()*. La constructora Note coge dos argumentos, el valor *pitch* y el valor *rythm*. En este caso las constantes de jMusic son usadas para las dos. C4 es la nota c(do) con valor de 60, y QUARTER_NOTE equivale al valor de 1.0 pulsos. Todas las constantes jMusic van en Mayúscula.

```
Phrase phrase = new Phrase();
```

En este caso se crea un objeto del tipo phrase llamado "phrase". La constructora Phrase no toma ningún argumento en este ejemplo. Esto significa que usará valores por defecto.

```
phrase.addNote(n);
```

Ahora ya teniendo la nota y la phrase, podemos añadir la nota a la phrase. Phrase puede contener muchas notas que son almacenadas, visualizadas y sonadas en el mismo orden en el que fueron añadidas, una tras otra. En esta parte del código el objeto Phrase que hemos creado, 'phrase' usa su propio *addNote()* método. Un método es una función o un procedimiento.

Cada objeto hereda los métodos de su clase, esto es, el objeto 'phrase' hereda el método addNote() de la clase *Phrase*.

```
View.notate(phrase);
```

Esta línea visualiza la nota como manuscrito. La clase *View* es una de las clases jMusic utilizadas para invocar una de las muchas y variadas visualizaciones de música en jMusic. En este caso su *notate()* método es usado para visualizar CPN y el objeto 'phrase' se le pasa al método para visualizarlo.

Nota a tener en cuenta. Hemos visto que en esta última línea podemos usar la clase *View* sin tener que crear una instancia de ella como hicimos con *Note* y *Phrase*. Esto se debe a que la clase *View* es estática, con lo que sólamente hay una copia de ella asi que la clase es llamada sólamente escribiendo su nombre.

B.3. The jMusic Data Structure

La información musical en jMusic es almacenada de una manera jerárquica basada en una puntuación convencional en el papel.

```
Score (contiene cualquier número de Parts)
   |
   +---- Part (contiene cualquier número de Phrases)
           |
           +---- Phrase (contiene cualquier número de Notes)
                    |
                    +---- Note (informa sobre cualquier evento musical)
```

Figura B.1: Jerarquía jMusic

B.3.1. Notes

La clase jm.music.data.Note es la estructura de notas usada en jMusic.
El objeto *Note* contiene mucha información:

- Pitch (tono) - El tono de la nota. Por ejemplo C4(la nota do en 4ª octava). Ojo! Aquí no sabemos la figura (negra, blanca, redonda, corchea...) sólo que es la nota do. Para más información ver sección A.5 en la página 65.
- Dynamic (dinámica) - Altitud de la nota. Para incrementar o decrementar el sonido de la nota:
  SILENT (Silencio) = 0, –> no se oye sonido alguno.
  PPP (muy pianísimo) = 10, –> se oye un poco el sonido. Pero bajísimo.
  PP (pianísimo) = 25, –> Se oye un poco más el sonido, pero aun así bajo.
  P (piano) = 50, –> Se oye el sonido, pero sin ser agresivo.
  MP (mezzo piano) = 60, –> El sonido está en la mitad.
  MF (mezzo forte) = 70 , –> Se oye un poco más el sonido, pero sin ser agresivo.
  F (forte)= 85, –> Se oye fuerte.
  FF(fortísimo) = 100, –> Se empieza a oir muy fuerte, empieza a ser agresivo.
  FFF (muy fortísimo) = 120 –> El sonido es extremadamente alto y agresivo.
- RhythmValue (Valor del ritmo) - Duración de la nota (la figura) (ej. Crotchet (Negra)). jMusic esta basado en pulsos donde un pulso equivale al valor de 1.0. Para más información ver sección A.3 en la página 65.
- Pan - La posición de las notas en el equipo de música (o más) del espectro. Esto es:
  PAN_CENTRE = 0.5, –> Si queremos oirlo en la parte derecha de los altavoces.
  PAN_LEFT = 0.0, –> Si queremos que se oiga igual en los dos lados.
  PAN_RIGHT = 1.0; –> Si queremos oirlo en la parte izquierda de los altavoces.

- Duration - Duración de la nota en milisegundos.
  STACCATO = 0.2,
  LEGATO = 0.95,
  SOSTENUTO = 1.2,
  TENUTO = 1.0;
- Offset - Desviación de la hora de inicio "normal" de la nota.

Aquí por ejemplo, vemos que se crea un do en cuarta octava (C4), su figura es la negra (CROTCHET), con una intensidad de medio fuerte (mezzo forte) (MF), queremos que el sonido esté centrado (PAN_CENTRE) y que la nota dure 0,95 milisegundos (CROTCHET * LEGATO)

```
1  Note n = new Note(C4, CROTCHET, MF, PAN_CENTRE, CROTCHET * LEGATO);
```

B.3.2. Phrases

La clase Phrase es un poco más complicada que la clase Note pero puede ser simplemente explicada como las voces.

Por ejemplo, una parte de piano es una parte única, pero puede tener múltiples voces. Toma las manos izquierda y derecha como ejemplos obvios. El objeto phrase en realidad solo puede contener un único atributo, una lista de notas. Cada objeto phrase contiene una lista de notas que puedes añadir, eliminar o mover en el pentagrama. La lista que hace todas estas maravillas se llama vector y es una clase java encontrada en el paquete *java.util.Vector*.

B.3.3. Parts

Parte es una clase que sorprendentemente tiene las notas (en frases) que se tocarán por un instrumento. Una parte contiene un vector que contiene muchas phrases. Una parte tambien tiene, título ("Violin 1" por ejemplo), el canal y un instrumento (en MIDI, el programa cambia el numero -en audio un índice en el array de instrumentos).

B.3.4. Score

La clase Score representa el nivel más alto de nuestra estructura de datos y contiene un vector de parts (partes) y un título (nombre).

B.3.5. Real-Tme audio structure

Mientras la mayoría de tutoriales introductorios se basan en componer offline (no en tiempo real), jMusic tiene una arquitectura para audio en tiempo real que incluye las clases RTLine y RTMixer.

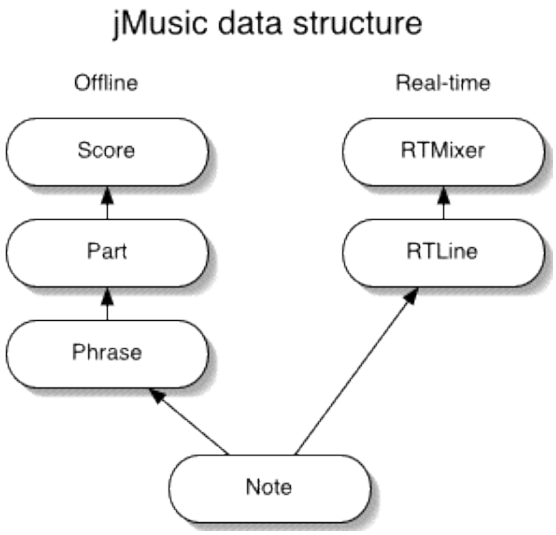


Figura B.2: jMusic data strucutre

B.3.5.1. RTLine

Esta es una clase abstracta que extiendes para proporcionar una generación (composición) lógica de la nota para el procesamiento de audio en tiempo real. RTLine es similar a Part que va constantemente a lo largo de la pieza, pero es como una frase la cual es monofónica y ofrece un flujo de notas hasta la ruta de datos.

B.3.5.2. RTMixer

Un objeto RTMixer recoge todos los flujos de audio siendo suministrados por RTLines y los suma y los pasa a la salida de audio del ordenador (a través JavaSound).

Normalmente habrá un solo objeto RTMixer por aplicación en tiempo real, pero, a menudo, habrá muchas RTLines (uno para cada parte musical).

Apéndice C

Lilypond

Lilypond es una aplicación de código abierto para la creación de partituras. Una de las características que lo distingue del resto es el cuidado por hacer que los documentos resultantes se parezcan a las partituras que se imprimían a mano, siguiendo las guías de estilo tradicionales.

La página oficial de Lilypond alberga una extensa documentación [16], acompañada siempre de ejemplos claros.

```
1  \version "2.18.2"
2
3  \book {
4     \score {
5        \header {
6           title = "Mi obra de arte"
7           composer = "Don Anónimo"
8        }
9        \layout { }
10       \midi {
11          \tempo 4 = 120
12       }
13
14       \new Staff {
15          \new Voice {
16             \relative c'' {
17                \clef treble
18                \key g \major
19                \time 3/4
20                r4 ces'. c8
21             }
22          }
23       }
24
25    }
26 }
```

Tabla C.1: Código de ejemplo de Lilypond

En la tabla C.1 podemos ver un código escrito en Lilypond para representar lo mismo que en el ejemplo de la tabla A.1 de MusicXML. Para un ejemplo tan sencillo como este, sería posible reducir el código a la parte dentro de `new Voice` (líneas 16-21, ya que Lilypond se encargaría de rellenar todas las secciones omitidas), pero de este modo podremos conocer mejor la estructura utilizada por el programa. Podemos ver un esquema general en la tabla C.2.

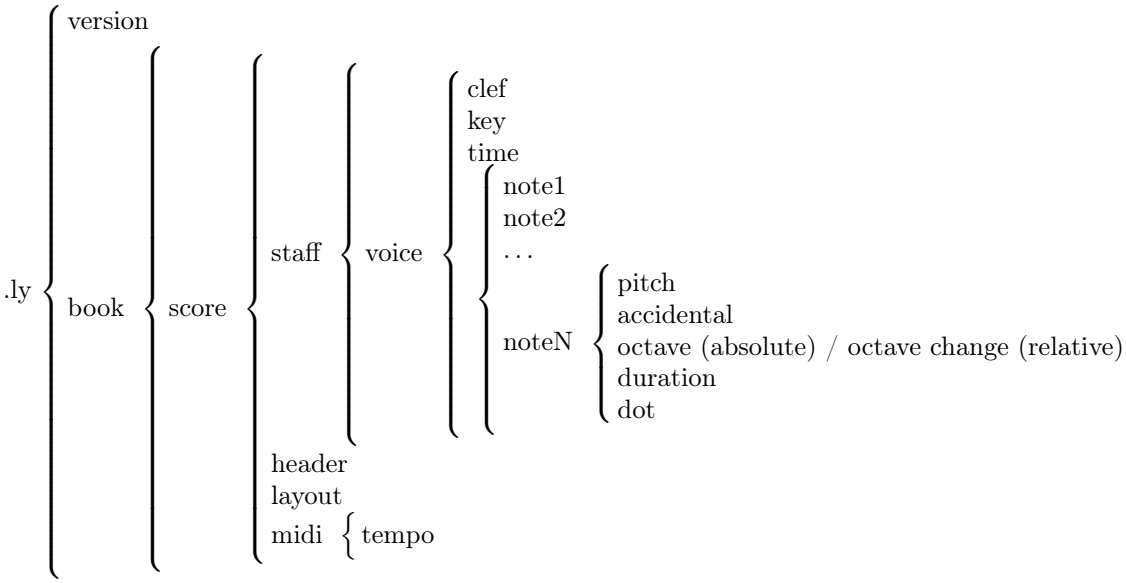


Tabla C.2: Esquema general de un archivo Lilypond

C.1. Versión (`\version`)

Debido a la constante evolución del formato Lilypond, se recomienda comenzar todo archivo con el comando `\version`, seguido de la versión de Lilypond mínima que soportaría todos los comandos que estamos utilizando. Para uso personal, con poner la versión instalada sería suficiente.

C.2. Documento (`\book`)

Lo más normal es que sólo guardemos una partitura en el archivo, que aparecerá dentro del comando `\book` (líneas 3-26). En el caso de que quisiéramos crear como salida varios documentos distintos a partir de un único archivo `.ly`, tendríamos que seguir una estructura similar a la de la tabla C.3. Cada `\book` contendrá una o varias partituras (`\score`), y determinaremos el nombre de salida de cada `book` con el comando `\bookOutputName`.

```
\version "2.18.2"

\book {
   % Primera partitura
   \bookOutputName "Partitura1"
   \score { ... }
}

\book {
   % Segunda partitura
   \bookOutputName "Partitura2"
   \score { ... }
}

...
```

Tabla C.3: Varias partituras en un archivo .ly

C.2.1. Partitura (`\score`)

Esta será la sección que contenga toda la partitura. Se subdivide en las siguientes partes:

C.2.1.1. Cabecera (`\header`)

La cabecera (líneas 5-8) podrá estar formada por varios títulos[1]. Los más comunes son:

- title: título de la obra
- composer: compositor de la obra
- copyright: línea inferior en la primera página de la partitura
- tagline: línea inferior en la última página de la partitura

C.2.1.2. Apariencia (`\layout`)

Layout (línea 9) servirá para configurar la apariencia. Aunque su contenido sea vacío, se imprimirá el documento.

C.2.1.3. MIDI (`\midi`)

MIDI (líneas 10-12) nos permitirá ajustar las opciones de la salida MIDI. Si no queremos exportar el archivo MIDI, tendremos que borrar el comando.

Una opción interesante es la de `tempo`, gracias a la que determinaremos la velocidad de la partitura.

C.2.1.4. Pentagramas (`\new Voice`)

`Voice` contendrá todos los elementos del pentagrama, como por ejemplo:

clef La figura de la clave (`\clef`, línea 17). Ver tabla C.4.

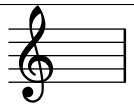
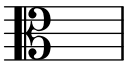
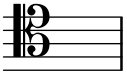
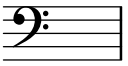
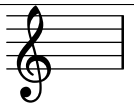
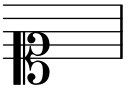
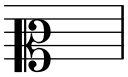
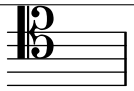
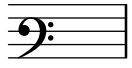
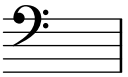
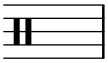
					
`treble`	`alto`	`tenor`	`bass`	`french`	`soprano`
					
`mezzosoprano`	`baritone`	`varbaritone`	`subbass`	`percussion`	

Tabla C.4: Tabla de claves

key La armadura (`\key`, línea 18), con la nota y el modo[2]. Ver tabla C.5.

`c`	do
`d`	re
`e`	mi
`f`	fa
`g`	sol
`a`	la
`b`	si

Alteración	
`es`	bemol
	natural
`is`	sostenido

Modo	
`\major`	mayor
`\minor`	menor

Tabla C.5: Tabla de armaduras

time Compás (`\time`, línea 19).

[1] El listado completo de títulos aparece en la sección "Presentación predeterminada de los títulos de partes de libro y partitura" de [17]: http://lilypond.org/doc/v2.19/Documentation/notation/creating-titles-headers-and-footers

[2] Los modos eclesiásticos aparecen en la sección "Armadura de la tonalidad" de [17]: http://lilypond.org/doc/v2.18/Documentation/notation/displaying-pitches#key-signature

note Nota. En la tabla C.6 podremos ver una explicación de qué es cada parte en las figuras r4 ces'. c8 (con `\relative` activado):

`r`			`4`	
silencio	-	-	negra	sin puntillo

`c`	`es`	`'`		`.`
do	bemol	1 octava hacia arriba	negra (misma duración que anterior)	con puntillo

`c`			`8`	
do	natural (no mantiene el bemol del do anterior)	misma octava que nota anterior	corchea	sin puntillo

Tabla C.6: Varias notas en Lilypond (`\relative` activado)

Almacenaremos en este orden:

pitch La altura

- Si es nota: do = `c`, re = `d`, mi = `e`, fa = `f`, sol = `g`, la = `a`, si = `b`
- Si es un silencio, se escribe `r`.

accidental Alteraciones

- Si es una nota natural, no se pone nada.
- Si tiene un sostenido: `is`. Si tiene dos: `isis`.
- Si tiene un bemol: `es`. Si tiene dos: `eses`.

octave La octava, con un sistema similar al de la notación musical de Helmholtz.

- Si no se especifica
  - y estamos usando octavas relativas (`\relative`): estará en la misma octava que la nota anterior. Lilypond automáticamente calcula la distancia más cercana hacia la nota especificada por el usuario, haciendo que el intervalo entre las notas sea menor que quinta. Si no es la deseada, tendremos que forzar el cambio de octava.
  - y no estamos usando octavas relativas: estará en la octava principal (en un piano de 88 teclas sería la tercera octava completa, donde recae predeterminadamente la mano izquierda).
- Por cada octava que queramos subir, añadimos una comilla (`'`).
- Por cada octava que queramos bajar, añadimos una coma (`,`).

duration La duración. Se determinará con un número que hace referencia a la duración de la nota con respecto a la de una redonda. Ver columna *D.C.* de tabla A.3 en la página 65.

dot Puntillo. Si lo tiene, un punto (`.`)

Apéndice D

StAX

Streaming API for XML (StAX) es una interfaz de programación de aplicaciones (API) para leer y escribir documentos XML, originaria de la comunidad del lenguaje de programación Java. Para aquellos que desarrollamos con XML y Java, a veces nos surge un problema a la hora de tratar con XML. Si usamos DOM nos encontramos con que es lento, y si usamos SAX nos encontramos con que no es todo lo completo que queremos. Algo intermedio entre ambos es StAX (Streaming API for XML), el cual usa tecnología Pull Parsing, para poder tratar los documentos XML de forma más rápida, con menor consumo de recursos y pudiendo saltar adelante/atrás y siendo igual para lectura y escritura. BEA fue el primero en desarrollar unas librerías para usar StAX con Java, dando un paso importante en el tratamiento de este tipo de documentos. Sus clases e interfaces están ubicadas en el paquete *javax.xml.stream*.

D.1. Lectura

StAX incluye dos modos de lectura: XMLEventReader y XMLStreamReader. El primero funciona muy al estilo de un *iterator*, mientras que el segundo es similar a *cursor*. En Kosmos, se usará la implementación de XMLEventReader.

D.1.1. XMLEventReader

Para comenzar, tendremos que recoger un archivo (`file` en el ejemplo) y crear los objetos de *Input Factory* e *Event Reader*.

```
1 FileReader fr = new FileReader(file);
2 XMLInputFactory factory = XMLInputFactory.newInstance();
3 XMLEventReader xer = factory.createXMLEventReader(fr);
```

El *Event Reader* (`xer`) será al que le vayamos pidiendo toda la información del archivo XML. Para comprobar si todavía quedan eventos que procesar en el XML, se lo pediremos mediante el método `hasNext()`. En ese caso, lo recogeremos con `nextEvent()` y más adelante lo procesaremos.

```
1 if (xer.hasNext())
2 {
3    XMLEvent event = xer.nextEvent();
```

El evento recogido puede ser de varios tipos: START_DOCUMENT (`<xml>`), START_ELEMENT (la mayoría de etiquetas sin marca de cierre `/`, como `<note>`), END_ELEMENT (lo contrario a START_ELEMENT, como `</note>`), CHARACTERS (texto plano), END_DOCUMENT (`</xml>`), ... Aquí está el listado de tipos de eventos que maneja StAX:

```
event.isAttribute()
event.isCharacters()
event.isEndDocument()
event.isEndElement()
event.isEntityReference()
event.isNameSpace()
event.isProcessingInstruction()
event.isStartDocument()
event.isStartElement()
```

Dependiendo del tipo, habrá que procesarlo de distinta manera, y cada uno tendrá sus métodos propios.

```
if (event.isStartElement())
{
   StartElement se = event.asStartElement();
   ...
}
else if (event.isEndElement())
{
   EndElement ee = event.asEndElement();
   ...
}
```

Por ejemplo, de un START_ELEMENT podemos recoger uno de sus atributos:

```
StartElement se = event.asStartElement();
localPart = se.getName().getLocalPart();

if (localPart.equals("score-part"))
{
   Attribute partId = se.getAttributeByName(new QName("id"));
```

D.2. Escritura

De una manera análoga a la de la lectura, primero inicializaremos varios objetos como el *Output Factory* y el *Stream Writer*. Este último lo crearemos con el método *IndentingXMLStreamWriter()* para nos haga automáticamente las tabulaciones (indenting).

```
FileOutputStream fos = new FileOutputStream(path);
XMLOutputFactory factory = XMLOutputFactory.newInstance();
XMLStreamWriter writer = new IndentingXMLStreamWriter(factory.createXMLStreamWriter(fos,
    "UTF-8");
```

Antes de seguir con su manejo, recordaremos que el método `flush()` escribe al archivo aquello esté todavía esperando en el buffer, y `close()` lo cierra después de hacer el flush final. Si se cierra, habrá que volver a abrirlo para trabajar con él.

```
writer.flush();
writer.close();
```

XMLStreamWriter nos permitirá escribir varios tipos de objetos, con los siguientes métodos:

```
writer.writeAttribute()
writer.writeCData()
writer.writeNamespace()
writer.writeStartElement()
writer.writeEndDocument()
writer.writeProcessingInstruction()
writer.writeCharacters()
writer.writeDTD()
writer.writeEndElement()
writer.writeStartDocument()
writer.writeComment()
writer.writeEmptyElement()
writer.writeDefaultNamespace()
```

Así, si quisiéramos crear una etiqueta con atributo, texto y otra etiqueta de cierre, podríamos escribir lo siguiente:

```
writer.writeStartElement("creator");
//<creator
writer.writeAttribute("type", "composer");
//<creator type=composer
writer.writeCharacters("Beethoven");
//<creator type=composer>Beethoven
writer.writeEndElement();
//<creator type=composer>Beethoven</creator>
```

Apéndice E

Implementación: al detalle

Para ahondar más en los módulos de Kosmos, se ha creado este anexo en el que se especificará con mayor detalle el funcionamiento de ellos.

E.1. ImpXmlToJm.java

ImpXmlToJm parseará un archivo de extensión .xml con formato XML e irá importando toda su información al entramado de jMusic, con la finalidad de visualizar y editar la partitura contenida en dicho archivo MusicXML. Las variables globales que manejará son:

```
private static ScoreInformation si = ScoreInformation.getSI();
    private static Score score;
    private static WizardData wd = new WizardData();

    private static XMLEventReader xer;
```

- si: Score Information. Clase creada para tener más a mano la partitura y guardar durante la ejecución de Kosmos valores que no se pueden guardar mediante otras clases ya implementadas.
- score: la partitura que abrirá jMusic para ser mostrada.
- wd: objeto WizardData. Aquí recogerá los datos de la cabecera y el compás inicial.
- xer: lector de eventos de XML. Herramienta de StAX para parsear archivos XML.

E.1.1. main

```
public static WizardData main() throws FileNotFoundException, XMLStreamException,
    StAXParsingError
{
    XMLInputFactory factory = XMLInputFactory.newInstance();
    xer = factory.createXMLEventReader(new FileReader(file));
    score = new Score();

    impHeaderToJm();
    impMeasureAttributesToJm();
    if (wd.getTempo() == null)
    {
        impTempoToJm();
    }
    impFiguresToJm();

    return wd;
}
```

main crea un objeto XMLInputFactory, inicializa xer indicándole la ruta del archivo XML a leer y crea la partitura.

Después, ejecuta los tres métodos principales: `impHeaderToJm()`, `impMeasureAttributesToJm()` e `impFiguresToJm()`.

Si después de `impMeasureAttributesToJm()` no ha encontrado el tempo, sigue buscándolo. Cuando termina de procesar el archivo XML, devuelve el archivo WizardData.

E.1.2. impHeaderToJm()

Pseudocódigo del método:

```
impHeaderToJm()
{
    Buscar etiqueta <score-partwise>
        Si encuentra etiqueta <score-timewise> mostrar error: archivo incompatible
        Si no encuentra etiqueta <score-partwise> mostrar error: no tiene formato
            MusicXML
        Si encuentra etiqueta <score-partwise>: continuar

    Parsear otras etiquetas de la cabecera
        Si encuentra <score-part>: poner nombre a 'part'
        Si encuentra <creator>: añadir compositor a ScoreInformation
        Si encuentra <work-title>: añadir título de partitura a 'wd'
        Si encuentra <part>: se acaba impHeaderToJm()
}
```

E.1.3. impMeasureAttributesToJm()

Pseudocódigo del método:

```
impMeasureAttributesToJm()
{
    Parsear etiquetas dentro de <attributes>
        Si encuentra <sound>: añade el tempo a 'wd'
        Si encuentra <divisions>: añadir divisions a ScoreInformation
        Si encuentra <fifths>: añade la armadura a 'wd'
        Si encuentra <mode>: añade el modo a 'wd'
        Si encuentra <beats>: añade el numerador del compás a 'wd'
        Si encuentra <beat-type>: añade el denominador del compás a 'wd'
        Si encuentra <sign>: recoge la forma de la clave en 'sign'
        Si encuentra <line>: recoge la línea de la clave en 'line'
        Si encuentra </attributes>
            Procesa 'sign' y 'line' para identificar la clave en 'wd'
}
```

E.1.4. impFiguresToJm()

Pseudocódigo del método:

```
impFiguresToJm()
{
    Phrase phrase = new Phrase();
    FigureJm figure = null;
    Note jm = null;
    boolean wasTieStart = false; // true si la nota anterior tenía tieStart

    Parsear etiquetas de todas las notas
        Si encuentra <tie>: parseTie()
        Si encuentra <duration>: parseDuration()
        Si !wasTieStart // la nota anterior no tenía tieStart
            Si encuentra <note>: crea un FigureJm nuevo [figure = new FigureJm()]
            Si encuentra <step>: parseStep()
            Si encuentra <alter>: parseAlter()
            Si encuentra <octave>: parseOctave()
            Si encuentra <type>: parseType()
            Si encuentra <rest/>: parseRest()
            Si encuentra <dot/>: parseDot()
        Si encuentra </note>
            Si la nota está ligada [isTieStart]
                wasTieStart = true
```

```
                Desactivar tieStart y tieStop en figure
            Si la nota no está ligada [!isTieStart]
                Convertir figure a nota de jMusic [jm = convXmlToJm(figure)]
                Añadir nota a phrase
        Si encuentra </part> // se han procesado todas las notas
            Añadir phrase a part
            Añadir part en score
            Asignar score a ScoreInformation
}
```

E.1.4.1. Ejemplo para las notas ligadas

```
<measure number="1">
  <attributes>
    <divisions>4</divisions>
    <key>
      <fifths>0</fifths>
      <mode>major</mode>
    </key>
    <time>
      <beats>4</beats>
      <beat-type>4</beat-type>
    </time>
    <clef>
      <sign>G</sign>
      <line>2</line>
    </clef>
  </attributes>
  <note>
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>12</duration>
    <type>half</type>
    <dot/>
  </note>
  <note>
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <tie type="start"/>
    <type>quarter</type>
    <notations>
      <tied type="start"/>
    </notations>
  </note>
</measure>
<measure number="2">
  <note>
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>2</duration>
    <tie type="stop"/>
    <type>eighth</type>
    <notations>
      <tied type="stop"/>
    </notations>
  </note>
</measure>
```

En el código de arriba podemos ver un ejemplo de notas ligadas en formato MusicXML. La representación

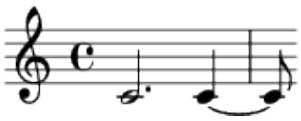


Figura E.1: Representación gráfica del ejemplo XML

step	alter	octave	dividedDuration
C	0	4	4
type	rest	dot	pitch
quarter	false	false	
rhythmValue	durationCrotchet	accidental	tieStart
			true
			tieStop
wasTieStart	false		false

Tabla E.1: Estados de wasTieStart y FigureJm con la segunda <note>

gráfica se puede ver en la figura E.1.

Después de procesar la segunda `<note>` (la negra ligada), el estado será el del cuadro E.1.

Como su estado es `tieStart=true`, se activará `wasTieStart` a `true` y se resetearán `tieStart` y `tieStop`. Al procesar el tercer `<note>` (la corchea), se detectará que `wasTieStart` está ahora a `true`, por lo que no se sobrescribirá `FigureJm` y se irán acualizando los ties (`tieStart` y `tieStop`) y la duración (`dividedDuration`). Con el tercer `<note>` el estado de `wasTieStart` y `FigureJm` quedará como en el cuadro E.2.

step	alter	octave	dividedDuration
C	0	4	**6**
type	rest	dot	pitch
quarter	false	false	
rhythmValue	durationCrotchet	accidental	tieStart
			false
			tieStop
wasTieStart	**true**		**true**

Tabla E.2: Estados de wasTieStart y FigureJm con la tercera <note>

Al detectar que `tieStart` está a `false`, Kosmos detecta que no se unirá a ninguna nota más. Por lo tanto, se crea la nueva nota de jMusic basándose en los datos de `FigureJm` y se añade la nota (`jm`) al `phrase`.

E.1.4.2. convXmlToJm

Código del método:

```
convXmlToJm(FigureJm fjm)
{
    Note f;
    int pitch = 0;
    Double rhythmValue = null;

    rhythmValue = ((double) fjm.getDividedDuration() / si.getDivisions());

    if (fjm.isRest())
    {
        f = new Rest(rhythmValue);
    }
    else
    {
```

```
        int octaves = Integer.valueOf(fjm.getOctave());
        octaves = (octaves + 1) * 12;

        int step = stepToExtraValue(fjm.getStep());

        int alter = 0;
        if (!fjm.getAlter().equals("0"))
        {
            alter = Integer.valueOf(fjm.getAlter());
        }

        pitch = octaves + step + alter;

        f = new Note(pitch, rhythmValue);
}
```

Una vez FigureJm está lleno con todos los valores, se procede la conversión a jMusic.

- rhythmValue recoge la duración de la figura
- Si es un silencio
  - se crea con la duración de rhythmValue
- Sino
  - se calculan cuántos semitonos hay hasta la octava de la nota (octaves)
  - se calculan cuántos semitonos hay en el intervalo entre DO y la nota (step)
  - si tiene alteración, se calcula la desviación (alter)
  - se suman los tres valores anteriores para calcular la altura (pitch)
  - con la altura y la duración se crea la nueva nota de jMusic

E.2. ExpJmToLy.java

ExpJmToLy recogerá todos los datos sobre la partitura que esté mostrando jMusic e irá exportándolos a un archivo Lilypond, del que más adelante el usuario podrá obtener la versión en PDF. Las variables globales que manejará son:

```
static ScoreInformation si = ScoreInformation.getSI();
static Score score = si.getScore();

static Scanner keyboard;
static String path;
static FileWriter file = null;
static PrintWriter writer = null;

static int ind = 0;

static int measureNumber = 1;
static String partId = "P1";
```

- si: Score Information.
- score: partitura en formato jMusic.
- keyboard: teclado
- path: ruta donde se escribirá el archivo Lilypond.
- file: fichero que apunte a la ruta 'path'
- writer: herramienta para escribir en archivos
- ind: nivel de indentación
- measureNumber: contador de compases
- partId: nombre de la parte

E.2.1. main

```
public static void main()
    {
        try
        {
            Phrase phrase = score.getPart(0).getPhrase(0);

            boolean chosen = getPath();

            if (chosen)
            {
                file = new FileWriter(path);
                writer = new PrintWriter(file);

                expHeaderToLy();
                writer.flush();

                expMeasureAttributesToLy();
                writer.flush();

                expFiguresToLy(phrase);
                writer.flush();

                expEndingToLy();
                writer.flush();
            }
            else
            {
                return;
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        finally
        {
            try
            {
                if (file != null)
                {
                    writer.close();
                    file.close();

                    File lyFile = new File(path);
                    String lyPath = lyFile.getAbsolutePath();
                    File lyDir = new File(lyPath).getParentFile();

                    JFrame dialFrame = new JFrame();

                    Object[] options = {"Sí, abrir carpeta", "No, continuar en Kosmos"};
                    int n = JOptionPane.showOptionDialog(dialFrame,
                            "Partitura extraida correctamente en " + path +
                            "\n\nPara convertirlo a PDF deberá tener instalado Lilypond "
                            + "y hacer doble click sobre la partitura extraida.\n\n"
                            + "¿Desea abrir la carpeta?",
                            "Extracción completa", JOptionPane.YES_NO_OPTION, JOptionPane
                                .QUESTION_MESSAGE, null,
                            options, // the titles of buttons
                            options[0]); // default button title

                    if (n == JOptionPane.YES_OPTION)
                    {
                        if (Desktop.isDesktopSupported())
                        {
                            try
                            {
                                Desktop.getDesktop().open(lyDir);
                            }
                            catch (IOException e)
```

```
                        {
                            e.printStackTrace();
                        }
                    }
                    else
                    {
                        JOptionPane.showMessageDialog(dialFrame, "Ha habido un
                            problema al abrir la carpeta.");
                    }
                }
            }
            catch (Exception e2)
            {
                e2.printStackTrace();
            }
        }
    }
```

main recoge el phrase de la partitura para poder procesar las notas introducidas. Después pide al usuario la ruta donde exportar el archivo de salida, y ejecuta los cuatro módulos de ExpJmToLy: exp`HeaderToLy, expMeasureAttributesToLy, expFiguresToLy y expEndingToLy.`

Si todo va bien, cerrará el archivo y mostrará una alerta para abrir la carpeta ddonde se haya exportado la partitura en formato Lilypond.

E.2.2. idt()

Las herramientas como StAX pueden configurarse fácilmente para poder tabular correctamente la salida de archivos XML con un solo comando. Por el contrario, si la escritura se hace a mano, también habrá que hacer el tabulado a mano. El método idt() (que proviene de la palabra 'indentation', tabulación en inglés) generará la tabulación del archivo en base al nivel que indique la variable 'ind' y lo devolverá en formato string.

```
private static String idt()
    {
        String indText = new String();

        if (ind >= 0)
        {
            indText = "";

            if (ind > 0)
            {
                for (int i=1; i <= ind; i++)
                {
                    indText = indText + "  ";
                }

            }
        }

        return indText;
    }
```

E.2.3. expHeaderToLy()

Pseudocódigo del método:

```
expHeaderToLy()
{
    Escribir comentario acerca del conversor jMusic-a-Lilypond de Kosmos
    Escribir versión mínima de Lilypond necesaria para compilar la partitura
    Comenzar \book // "libro"
        Comenzar \score // "partitura"
            Comenzar \header // "cabecera"
                Mencionar software que ha escrito la partitura
                Añadir título de la partitura
                Añadir autor de la partitura
```

```
            Finalizar \header
            Añadir \layout // "apariencia", necesario para exportar el PDF
            Comenzar \midi
                Añadir \tempo
            Finalizar \midi
            Comenzar \Staff // "pentagrama"
                Comenzar \Voice // "voz"
}
```

E.2.4. expMeasureAttributesToLy()

Pseudocódigo del método:

```
expMeasureAttributesToLy()
{
  Añadir clave    (\clef) [lyClef()]
  Añadir armadura (\key)  [lyKey()]
  Añadir compás   (\time) [lyTime()]
}
```

E.2.4.1. lyClef()

Obtiene la clave de ScoreInformation. Si es clave de sol, devuelve '*treble*'; si es clave de fa devuelve '*bass*'.

E.2.4.2. lyKey()

En base a la armadura (*keySignature*) y el modo (*keyQuality*) de la partitura de jMusic, recorre un if hasta encontrar la armadura correcta. No puede reducirse más debido a que es puramente teoría de solfeo aplicada al programa.

E.2.4.3. lyTime()

lyTime() devuelve un string con la mezcla del numerador y el denominador del compás de la partitura *score*.

E.2.5. expFiguresToLy()

El método expFiguresToLy será el que vaya imprimiendo las notas.

Debido a que jMusic añade las notas en un listado sin separación de compases (dicha separación la genera a la hora de mostrar las notas en pantalla), se tuvo que hacer un código que fuera calculando si cada nota cabía en el compás actual o no. En el caso de que la nota sobrepase el espacio libre del compás, será dividida en tantas partes como sea necesaria. Tanto el módulo de exportación de MusicXML (ExpJmToXml) como el de Lilypond (ExpJmToLy) lo usan[1], cada uno con una versión adaptada a sus necesidades.

Las variables usadas por el método expFiguresToLy son:

- measureNumber: variable global que indica el compás actual
- noteArray: listado de notas en formato jMusic
- jm: objeto FigureJm para ir acumulando los datos de cada nota, para facilitar la conversión
- cont: contador que indica la posición en *noteArray* de la nota que se está procesando
- maxDiv: número máximo de `<divisions>` que permite un compás de la partitura[2]
- collDiv: duraciones acumuladas en el compás actual

[1] MusicXML necesita realizar esta separación de compases. Lilypond no lo requiere, pero se puede usar para comprobar que la división se realiza correctamente.

[2] `<divisions>` es una idea puramente de MusicXML, pero sirve también para el procesado de notas en cualquier formato

- remDur: duración de la nota que añadir al compás. Si la nota no tiene ligaduras, será igual a su duración completa. Si la nota está en proceso de división, indicará la duración restante a añadir en el compás.
- divDur: duración de la nota que se añadirá en el compás actual. Si cabe en el compás actual, será igual a su duración completa. Si no cabe, divDur indicará la duración de la nota que quepa en el espacio libre del compás actual (y el resto se irá añadiendo en los siguientes, ligando las notas una a otra).

Pseudocódigo del método:

```
1  expFiguresToLy()
2  {
3    Note[] noteArray = phrase.getNoteArray();
4    FigureJm jm = null;
5
6    int cont = 0;
7    int maxDiv = score.getNumerator() * si.getDivisions();
8    int collDiv = 0;
9    int remDur = 0;
10   int divDur = 0;
11
12   Por cada nota en noteArray [while (noteArray.length > cont)]
13     Recoger nota en 'n'
14     Convertir 'n' a un objeto FigureJm [jm = convJmToLy(n)]
15     Recoger en 'remDur' las divisions de la nota 'n'
16
17     Mientras haya duración que añadir de la nota [remDur != 0]
18       Si no hay notas en el compás actual y la nota anterior estaba ligada:
19         Activar tieStop a jm        // imprimir la ligadura de cierre
20         y Desactivar tieStart a jm.
21
22       Si la nota no cabe en el compás:
23         divDur recogerá la parte que cabe;
24       pero si la nota cabe en el compás:
25         divDur recogerá la duración completa de la nota.
26
27       Actualizar jm con los cambios recientes de duración
28
29       Acumular en collDiv la duración (divDur) de la nota a añadir
30
31       Si collDiv indica que el compás actual está lleno:
32         Si la figura no es un silencio y faltan duraciones por añadir:
33           activar tieStart a jm  // imprimir la ligadura de apertura.
34
35         Escribir nota de 'jm' en formato Lilypond,
36         y escribir nuevo compás.
37
38         Resetear collDiv a 0.
39       pero si collDiv indiva que el compás actual no está lleno:
40         Escribir nota de 'jm' en formato Lilypond.
41
42       Quitar divDur de las duraciones restantes (remDur).
43 }
```

E.2.5.1. Ejemplo

Si seguimos la partitura de la figura E.1, expFiguresToLy() se ejecutará como en la tabla E.3.[3]
noteArray contendrá dos notas:

- La primera una do blanca con puntillo (rhythmValue=3.0, pitch=60, remDur=12 (siendo `<divisions>=4`))
- La segunda una do negra con puntillo (rhythmValue=1.5, pitch=60, remDur=6 (siendo `<divisions>=4`))

El procesado de rellenado de compases quedaría así:

[3] mN en la tabla indica la variable *measureNumber*.

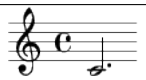
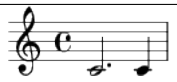
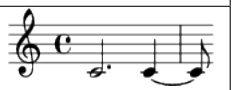
lín.	cont	maxDiv	collDiv	remDur	divDur	mN	tStart	tStop	Ly	img
14	0	16	0	0	0	1	F	F		
15	0	16	0	12	0	1	F	F		
25	0	16	0	12	12	1	F	F		
29	0	16	12	12	12	1	F	F		
40	0	16	12	12	12	1	F	F	c'2.	
42	0	16	12	0	12	1	F	F	c'2.	
14	1	16	12	0	12	1	F	F	c'2.	
15	1	16	12	6	12	1	F	F	c'2.	
23	1	16	12	6	4	1	F	F	c'2.	
29	1	16	16	6	4	1	F	F	c'2.	
33	1	16	16	6	4	1	T	F	c'2.	
36	1	16	16	6	4	1	T	F	c'2. c'4~ |	
38	1	16	0	6	4	2	T	F	c'2. c'4~ |	
42	1	16	0	2	4	2	T	F	c'2. c'4~ |	
19	1	16	0	2	4	2	F	T	c'2. c'4~ |	
25	1	16	0	2	2	2	F	T	c'2. c'4~ |	
29	1	16	2	2	2	2	F	T	c'2. c'4~ |	
40	1	16	2	2	2	2	F	T	c'2. c'4~ | c'8	
42	1	16	2	0	2	2	F	T	c'2. c'4~ | c'8	

Tabla E.3: Ejemplo de ejecución de expFiguresToLy()

- En un compás de 4/4 con `<divisions>=4` caben 16 divisiones (numerador $4 * 4$ `<divisions>`$= 16$ divisiones).
- Una blanca con puntillo ocupa 12 divisiones. (duración $3{,}0 * 4$ `<divisions>` $= 12$ divisiones).
- Una vez introducida la blanca con puntillo, quedan $16 - 12 = 4$ divisiones libres en el compás.
- Al intentar introducir una negra con puntillo (duración $1{,}5 * 4$ `<divisions>` $= 6$ divisiones), no podemos meter la figura entera porque: figura $6 > 4$ espacio libre en compás.
- Se crea figura que rellene el espacio libre del compás (espacio libre 4/4 `<divisions>` $= 1{,}0$ negra ligada)
- La figura original era una negra con puntillo (6 divisiones). La hemos dividido en una negra (4 divisiones). El resto ($6 - 4 = 2$ divisiones = corchea) irá en el siguiente compás.

E.2.5.2. convJmToLy()

Código del método:

```
convJmToLy(Note n)
    {
        FigureJm fjm = new FigureJm();

        if (n.isRest() == true)
        {
            fjm.setRest(true);
        }
        else
```

```
        {
            fjm.setRest(false);
            parseStepAlter(n, fjm);
            parseOctave(n, fjm);
        }

        parseJmDurationDot(n, fjm);

        return fjm;
}
```

Cuando FigureJm tenga todos los valores de la note de jMusic, se irá convirtiendo a notas de formato Lilypond.

- Se crea el objeto 'fjm' de tipo FigureJm.
- Si la figura es un silencio, se marca que es silencio.
- Si no lo es, se marca que no es silencio y se calculan su altura y octava.
- Tanto si es silencio como si no, se calcula la duración y si necesita puntillo.
- Se devuelve 'fjm' con todos los datos de la nota.

E.2.5.3. printFigureTags()

Código del método:

```
printFigureTags(FigureJm jm)
    {
String result = new String();

        // 1. Step/rest
        if (jm.isRest())
        {
            result = result + "r";
        }
        else
        {
            result = result + jm.getStep() + jm.getAlter() + jm.getOctave();
        }

        // 4. Duration
        result = result + jm.getDurationCrotchet();

        // 5. Dot
        if (jm.isDot())
        {
            result = result + ".";
        }

        // 6. Tie
        if (jm.isTieStart())
        {
            result = result + "~";
        }

        // Space at the ending
        result = result + " ";

        // Write the figure
        writer.print(result);
}
```

E.2.6. expEndingToLy()

Pseudocódigo del método:

```
expEndingToLy()
{
                Finalizar \Voice // "voz"
            Finalizar \Staff // "pentagrama"
        Finalizar \score // "partitura"
    Finalizar \book // "libro"
}
```

Apéndice F

Manual de Usuario

F.1. Introducción

Este programa se ha diseñado para la creación y visualización de partituras musicales al alcance de todo tipo de usuarios. Se ha programado con el lenguaje de programación Java. Por tanto, puede ejecutarse en cualquier sistema operativo que tenga instalado Java, disponible en la dirección web *http://www.java.com* o *http://www.java.com/es* en español. Para que funcione correctamente el audio también será necesario tener instalado el marco de trabajo multimedia de Java JMF, que se puede encontrar en la dirección web *http://java.sun.com/products/java-media/jmf*. Para ejecutar el programa basta con darle doble click en el ejecutable de la aplicación y se iniciará como se muestra en la figura F.1. Para seguir adelante en el programa basta con darle un click con el ratón en cualquier zona de la pantalla que se ha visualizado.

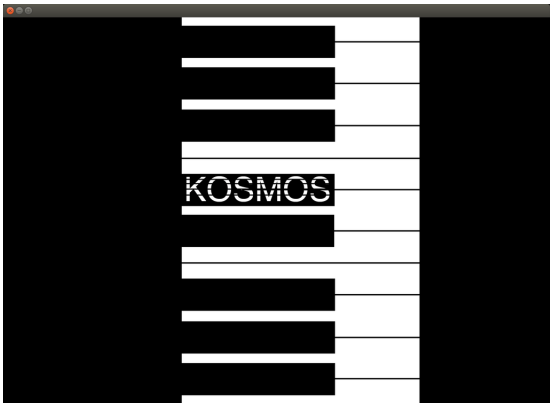


Figura F.1: Splash (Ventana al iniciar el programa)

F.2. Menu de opciones

En la siguiente ventana aparecen dos botones como se puede apreciar en la figura F.2. El primero *Crear Partitura* y el segundo *Abrir Partitura*. Ambos nos llevan a interfaces distintas con diferentes funcionalidades de acuerdo a lo que el usuario desee hacer.

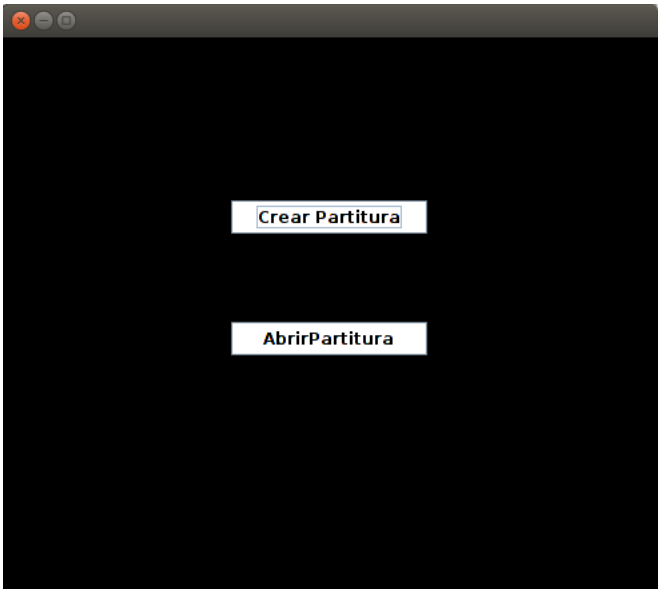


Figura F.2: Crear Partitura y Abrir Partitura

Si el usuario desea crear una partitura deberá pinchar sobre el botón *Crear Partitura*. Esta acción nos llevará a una ventana disinta como muestra la figura *F.3*. Esta interfaz ofrece al usuario un menú de opciones con el cual podrá acceder a las distintas funcionalidades. Aquí podrá empezar a crear su composición musical dado que se le proporcionan diversos componentes a elegir. Podrá escribir el nombre del autor y el título de la partitura, podrá elegir entre el sonido de 5 instrumentos (violín, acordeón, piano, flauta y guitarra), la velocidad de la pieza, dos posibles claves (*sol y fa*), tipo de armadura para su obra (*sostenidos o bemoles*), la intensidad del sonido (de muy suave a muy fuerte), el compás a elegir entre 4 posibles (*1/4, 2/4 ,3/4, 4/4*), la nota (*do, re, mi, fa, sol, la , si, do*) y el tipo de nota (*corchea, negra, blanca, redonda*). Para proseguir pinchará en el botón aceptar, en caso de querer retroceder se pulsará en el botón atrás.

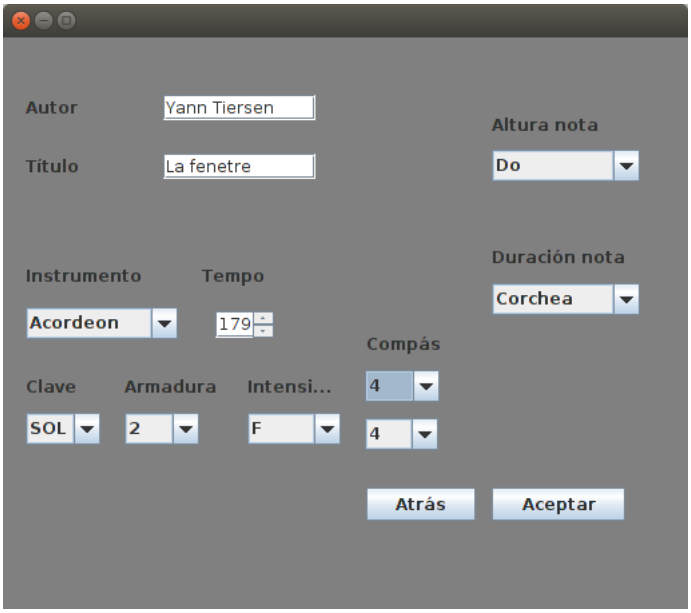


Figura F.3: Dentro de Crear Partitura

Los datos proporcionados por el usuario anteriormente se insertan en la partitura nueva como se puede apreciar en la figura F.4. Si se quiere editar algun elemento que el usuario ha introducido anteriormente podrá efectuarse con el ratón del ordenador en la nueva pantalla.

En este caso el usuario habia introducido un *do corchea*, y si desea modificarla deberá pinchar sobre la nota y sin dejar de pulsarla arrastrar el ratón de izquierda a derecha hasta llegar al tipo de nota que desee. Lo mismo pasará con la armadura y el compás. En la parte superior se halla un menu con diversas opciones (Archivo, Herramientas y Reproducción) que al seleccionarlos individualmente se desplegará un submenu con más opciones como veremos más adelante.

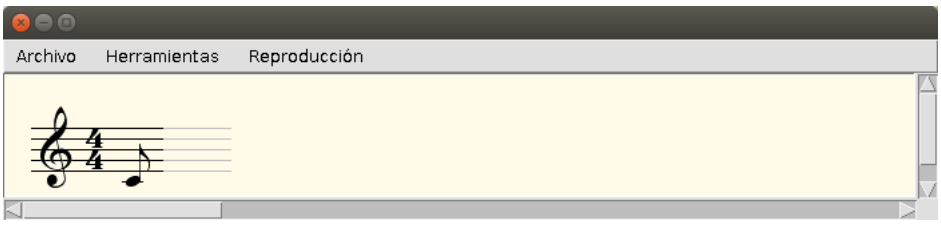


Figura F.4: Interfaz del pentagrama

A continuación se explicarán todas las funcionalidades y las subfuncionalidades que tiene esta interfaz mediante una tabla.

Opción	***Menú***	***Funcionalidad***
Nuevo pentagrama a vacío	Archivo	Crear pentagrama vacio.
Nuevo pentagrama con plantilla	Archivo	Crear pentagrama basado en una plantilla.
Abrir MusicXML	Archivo	Abrir un archivo musicXML que eliga el usuario.
Guardar MusicXML	Archivo	Guardar un archivo musicXML en la ruta elegida por el usuario.
Exportar PDF (requiere Lilypond)	Archivo	Exportar partitura en formato .ly para posterior conversor PDF.
Importar	Archivo	Contiene 3 tipos de import.
Importar MIDI	Importar	Importar un archivo midi.
Importar jMusic XML	Importar	Importar un archivo jMusic XML.
Importar archivo JM	Importar	Importar un archivo jm.
Exportar	Archivo	Contiene 3 tipos de export.
Exportar MIDI	Exportar	Exportar archivo midi.
Exportar jMusic XML	Exportar	Exportar archivo jMusic XML.
Exportar archivo JM	Exportar	Exportar archivo jm.
Cerrar	Archivo	Se cierra el programa Kosmos.

Tabla F.1: Funcionalidades del menú Archivo

Opción	***Menú***	***Funcionalidad***
Añadir nota mediante letra	Herramientas	Añadir nota mediante su correspondiente letra.
Borrar última nota	Herramientas	Borrar última nota insertada en el pentagrama.
Borrar todas las notas	Herramientas	Borrar todas las notas insertadas en el pentagrama.
Mostrar/ocultar armadura	Herramientas	Mostrar o ocultar la armadura.
Mostrar/ocultar compás	Herramientas	Mostrar o ocultar el compás.
Mostrar/ocultar número de compás	Herramientas	Mostrar o ocultar número de compáses.
Pentagrama	Herramientas	Al pulsar sobre esta opción el usuario podrá cambiar la clave.
Ver detalles técnicos	Herramientas	Todos los detalles de la partitura.
Reproducir todo	Reproducción	Empezará a sonar toda la partitura.
Repetir todo	Reproducción	Se repetirá una y otra vez la paritura.
Reproducir último compás	Reproducción	Sonará solamente el último compás.
Repetir último compás	Reproducción	Se repetirá el último compás.
Parar el sonido	Reproducción	Se parará el sonido.

Tabla F.2: Funcionalidades de menús Herramientas y Reproducción

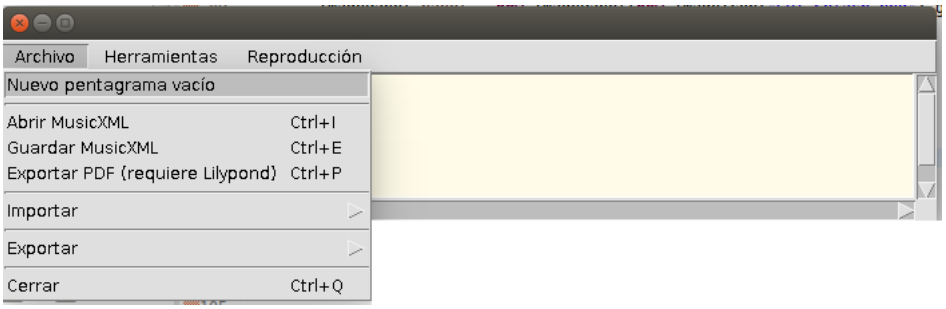


Figura F.5: Menú Archivo

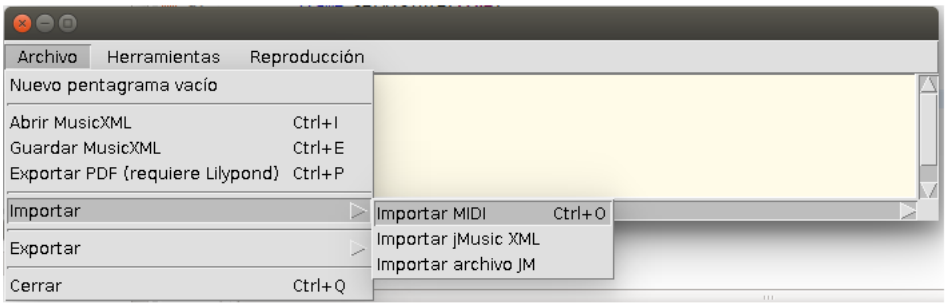


Figura F.6: Menú Archivo > Importar

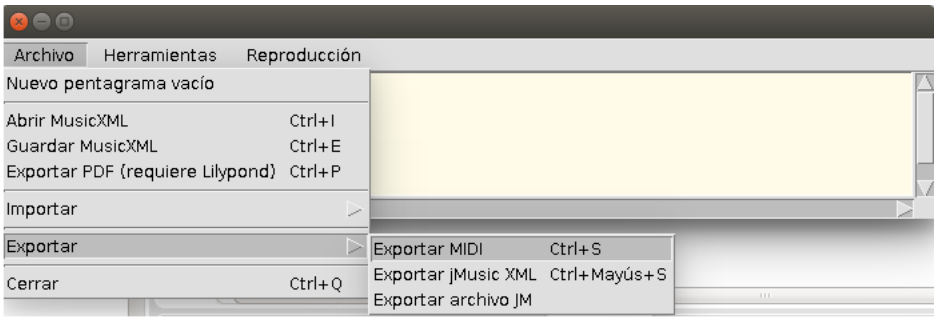


Figura F.7: Menú Archivo > Exportar

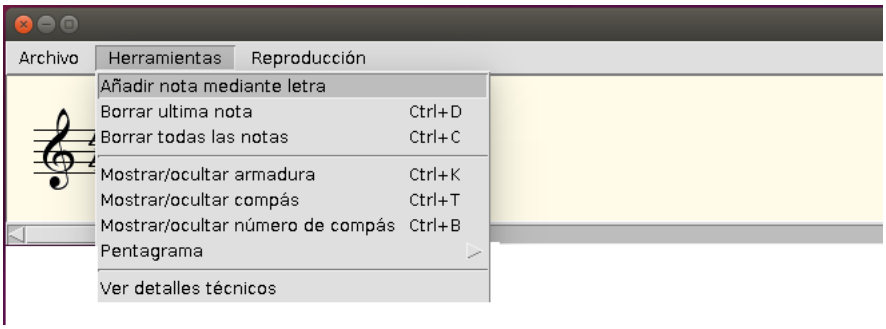


Figura F.8: Menú Herramientas

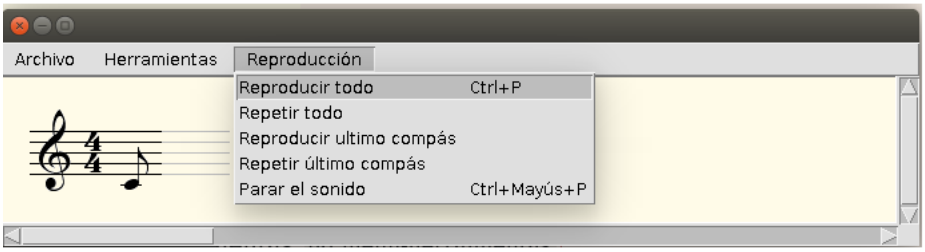


Figura F.9: Menú Reproducción

Si el usuario pincha sobre el botón Abrir partitura se desplegará un cuadro de diálogo donde se le pedirá la ruta del archivo *.xml* que desee abrir. Automáticamente le abrirá el archivo visualizando la partitura.

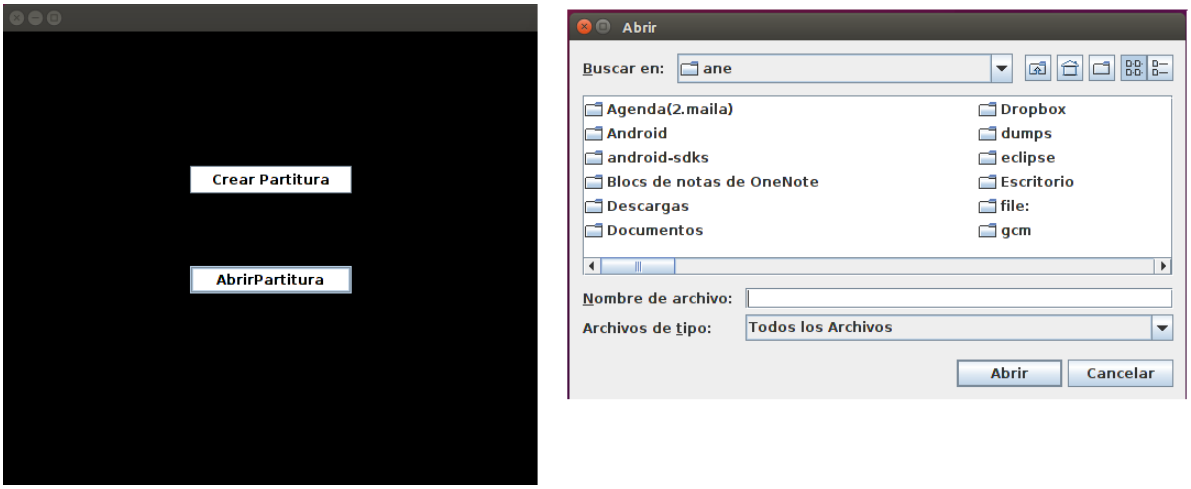


Figura F.10: Menú Abrir Partitura

Apéndice G

GNU General Public License (GPL) v2.0

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the

integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Apéndice H

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License (CC BY-NC-SA 4.0)

H.1. Commons Deed

This is a human-readable summary of (and not a substitute for) the license.

Disclaimer

This deed highlights only some of the key features and terms of the actual license. It is not a license and has no legal value. You should carefully review all of the terms and conditions of the actual license before using the licensed material.

Creative Commons is not a law firm and does not provide legal services. Distributing, displaying, or linking to this deed or the license that it summarizes does not create a lawyer-client or any other relationship.

You are free to:

Share copy and redistribute the material in any medium or format

Adapt remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution You must give *appropriate credit*[1], provide a link to the license, and *indicate if changes were made*[2]. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial You may not use the material for *commercial purposes*[3].

ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the *same license*[4] as the original.

[1] If supplied, you must provide the name of the creator and attribution parties, a copyright notice, a license notice, a disclaimer notice, and a link to the material. CC licenses prior to Version 4.0 also require you to provide the title of the material if supplied, and may have other slight differences.

[2] In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative.

[3] A commercial use is one primarily intended for commercial advantage or monetary compensation.

[4] You may also use a license listed as compatible at https://creativecommons.org/compatiblelicenses

No additional restrictions You may not apply legal terms or *technological measures*[5] that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable *exception or limitation*[6].

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as *publicity, privacy, or moral rights*[7] may limit how you use the material.

H.2. Legal code

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

H.2.1. Section 1 – Definitions.

1(a) Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

1(b) Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

1(c) BY-NC-SA Compatible License means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

1(d) Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

1(e) Effective Technological Measures means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

1(f) Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

1(g) License Elements means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution, NonCommercial, and ShareAlike.

1(h) Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

[5] The license prohibits application of effective technological measures, defined with reference to Article 11 of the WIPO Copyright Treaty.

[6] The rights of users under exceptions and limitations, such as fair use and fair dealing, are not affected by the CC licenses.

[7] You may need to get additional permissions before using the material as you intend.

1(i) Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

1(j) Licensor means the individual(s) or entity(ies) granting rights under this Public License.

1(k) NonCommercial means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange.

1(l) Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

1(m) Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

1(n) You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

H.2.2. Section 2 – Scope.

2(a) License grant.

(1) Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

(A) reproduce and Share the Licensed Material, in whole or in part, for NonCommercial purposes only; and

(B) produce, reproduce, and Share Adapted Material for NonCommercial purposes only.

(2) Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

(3) Term. The term of this Public License is specified in Section 6(a).

(4) Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

(5) Downstream recipients.

(A) Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

(B) Additional offer from the Licensor – Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

(C) No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

(6) No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

2(b) Other rights.

(1) Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

(2) Patent and trademark rights are not licensed under this Public License.

(3) To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties, including when the Licensed Material is used other than for NonCommercial purposes.

H.2.3. Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

3(a) Attribution.

(1) If You Share the Licensed Material (including in modified form), You must:

(A) retain the following if it is supplied by the Licensor with the Licensed Material:

(i) identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

(ii) a copyright notice;

(iii) a notice that refers to this Public License;

(iv) a notice that refers to the disclaimer of warranties;

(v) a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

(B) indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

(C) indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

(2) You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

(3) If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

3(b) ShareAlike.

In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

(1) The Adapter's License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-NC-SA Compatible License.

(2) You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.

(3) You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

H.2.4. Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

4(a) for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database for NonCommercial purposes only;

4(b) if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

4(c) You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

H.2.5. Section 5 – Disclaimer of Warranties and Limitation of Liability.

5(a) Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

5(b) To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

5(c) The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

H.2.6. Section 6 – Term and Termination.

6(a) This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

6(b) Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

(1) automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

(2) upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

6(c) For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

6(d) Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

H.2.7. Section 7 – Other Terms and Conditions.

7(a) The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

7(b) Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

H.2.8. Section 8 – Interpretation.

8(a) For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

8(b) To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

8(c) No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

8(d) Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Bibliografía

[1] Definición ABC. Definición ABC. http://www.definicionabc.com/tecnologia.

[2] About.com. Tocar piano (en español) - About.com. http://tocarpiano.about.com/od/musicaltermsa1/g/.

[3] Exploding Art. jMusic: documentation, downloads, programs... http://explodingart.com/jmusic/.

[4] Definiciones.de. Definiciones.de. http://definicion.de/.

[5] Eclipse. To install WindowBuilder Pro Eclipse from this update site. http://download.eclipse.org/windowbuilder/WB/integration/4.4/.

[6] Fakiro. Fakiro: Diccionario musical. http://musica.fakiro.com/diccionario/indice.html.

[7] GanttProject. GanttProject installation documentation. http://es.slideshare.net/reamari/manual-ganttproject.

[8] Ane Aliseda Ibarretxe. Generador automático de items de evaluación del lenguaje musical. Trabajo fin de grado, Escuela Universitaria de Ingeniería Técnica Industrial de la Universidad del País Vasco, 2016.

[9] Jakob Jenkov. Java XML tutorial. http://tutorials.jenkov.com/java-xml/index.html.

[10] jFugue. jFugue documentation and examples. http://www.jfugue.org/examples.html.

[11] jMusic. jMusic classes attributes and methods. http://explodingart.com/jmusic/jmDocumentation/jm/music/data/Note.html.

[12] jMusic. jMusic Tutorials and Lessons. http://explodingart.com/jmusic/jmtutorial/t1.html.

[13] Pankaj Kumar. How to read XML file in Java using Java StAX Iterator API. http://www.journaldev.com/1191/how-to-read-xml-file-in-java-using-java-stax-api.

[14] Pankaj Kumar. How to write XML file in Java using Java StAX Iterator API. http://www.journaldev.com/892/how-to-write-xml-file-in-java-using-java-stax-api.

[15] Lilypond. LilyPond - Glosario. http://www.lilypond.org/glossary.es.html.

[16] Lilypond. Manuales de LilyPond 2.18. http://lilypond.org/manuals.es.html.

[17] Lilypond. Referencia de la notación de GNU LilyPond 2.19. http://lilypond.org/doc/v2.19/Documentation/notation/index.es.html.

[18] LyX. Instrucciones para llevar a cabo el glosario. https://borrowbits.com/2013/04/plantilla-proyecto-fin-de-carrera-para-lyx/.

[19] Lander Martínez. Ariketa editorea lengoaia musikalaren ikasketa eta ebaluaketarako. Trabajo fin de grado, Escuela Universitaria de Ingeniería Técnica Industrial de la Universidad del País Vasco, 2011.

[20] Ministerio de Educación, Cultura y Deporte. Definición hardware y software. http://roble.pntic.mec.es/jprp0006/tecnologia/1eso_recursos/unidad02_componentes_ordenador/teoria/teoria1.htm.

[21] Despertar Musical. Despertar musical. http://despertarmusical.blogspot.com.es/.

[22] MusicXML. Hello World: A one-bar song with a whole note on middle C in 4/4 time. http://www.musicxml.com/tutorial/hello-world.

[23] MusicXML. MusicXML - a software engineering blog. http://sizustech.blogspot.com.es/2014/12/reading-and-writing-musicxml-files-with.html.

[24] Creando Partituras. Creando Partituras - Notación musical, software, y MIDICreando Partituras. http://www.creandopartituras.com/.

[25] Robert Piasecki. How to parse XML documents using streaming API for XML (StAX). http://softwarecave.org/2014/02/18/parse-xml-document-using-streaming-api-for-xml-stax/.

[26] Robert Piasecki. How to write XML documents using streaming API for XML (StAX). http://softwarecave.org/2014/02/15/write-xml-documents-using-streaming-api-for-xml-stax/.

[27] Wikipedia. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Wikipedia:Portada.