Generador automático de items de evaluación del lenguaje musical

Ane Aliseda Ibarretxe

29 de febrero de 2016

RESUMEN

La intención de este proyecto es la creación de un sencillo e interactivo programa para facilitar el aprendizaje de la música pudiendo crear, editar, importar y exportar composiciones propias. No se ha querido crear un programa demasiado llamativo para que el usuario centrara la atención y resultara más sencilla la interacción. Nos encontramos ante una aplicación al alcance de cualquier usuario debido a su fácil manejo y coste gratuito.

La gran mayoría de programas músicales que podemos encontrar navegando en la red son más complejos puesto que existe una gran variedad de opciones que nos aportan y que muchas de ellas tienen una finalidad quizá demasiado específica. Sin embargo, todas las opciones ofrecidas por el programa son útiles. Para llevar a cabo este programa se buscaron fuentes de investigación que proporcionasen datos importantes para el desarrollo y finalidad del proyecto como la libreria "jMusic" y el formato "MusicXML". Además se ha empleado "eclipse", un programa ya conocido por haberlo utilizado a lo largo de la carrera, el cual ha facilitado el camino para hacer el código. El programa más usado ha sido el "Skype" para facilitar la comunicación entre mi compañero e proyecto y yo. Por otro lado, la utilización de aplicaciones como "Dropbox" o "Gmail" entre otros fueron de gran ayuda para evitar posibles perdidas de documentación. Gracias a la fase de investigación se han podido emplear varias librerias halladas en internet que han servido para la estructura de la aplicación. La aportación de estas ha servido para la creación de los elementos de todo el programa.

Para finalizar, hay que decir que me ha resultado complicado realizar el trabajo ya que nunca habia realizado ninguna tarea relacionada con la creación de partituras. Además, se ha invertido mucho tiempo en las fases de investigación. Sin embargo, me ha parecido muy agradecido y provechoso trabajar en este tema. Lo más complejo ha sido la realización del código por no tener experiencia programando esta clase de aplicaciones.

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, bemol u otra alteración 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 fortissisissimo.

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

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 b> 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 icono que sirve para...

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.

MusicXML Formato de notación musical de código abierto basado en XML.

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 fúsas.

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 es una pieza de un trabajo más complejo, puesto que se precisa la creación de dos subproyectos complementarios entre sí. Ambos se irán realizando a la vez para dar como resultado con un único programa.

El proyecto explicado en este documento permite al usuario crear partituras mediante una sencilla interfaz. Una vez terminada la partitura, el usuario podrá guardar el resultado en un archivo con formato MusicXML (para conocer más acerca del formato, ver apéndice B en la página 97). El otro proyecto *(XMLScore: Representación gráfica y reproducción de partituras musicales en formato XML* [1]) podrá reproducir la partitura tanto de manera gráfica como sonora. Esto lo conseguirá mediante un archivo MusicXML generado por el proyecto previamente explicado.

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.

Acabado el proyecto, otro desarrollador podría ampliarlo para conseguir que sea más completo e interactivo.

1.2. Razones de elección del TFG

Mi tutor, Javilo Lopez, conocedor de mis años en la escuela de música y que toco dos instrumentos musicales me ofreció una opción de proyecto que no dude en elegir ya que me pareció interesante aportar mis conocimientos a un proyecto que podrá ser utilizado por conservatorios y escuelas de música para evaluar a los alumnos y facilitar el aprendizaje en las aulas.

Capítulo 2

PLANTEAMIENTO INICIAL

2.1. Objetivos

- Creación de partituras
- Interfaz sencilla
- Al alcance de cualquier usuario con conocimientos musicales
- Se guardará en formato MusicXML

2.2. Planificación original

Esta es la primera planificación realizada para el proyecto.

Alcance

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

Objetivos primarios

- Creación y reproducción de una partitura
- Borrado de notas ya posicionadas
- Guardado en formato MusicXML
- Partituras de una sola voz
- Clave de sol
- Compases dentro del rango: $^2/_4, ^3/_4, ^4/_4, ^6/_8, ^9/_8$
- Armadura en cualquier tonalidad
- Figuras y silencios que marquen el rango de duraciones: redonda, blanca, negra, corchea, semicorchea, fusa, semifusa
- Notas con puntillo y alteraciones simples (sostenido,bemol,becuadro)
- Intensidades tales como: ff,f,mf,p,pp,ppp

Objeticos secundarios

- Clave de fa
- Metrónomo
- Más compases a parte de los mencionados arriba
- Doble puntillo
- Acordes
- Botones de deshacer y rehacer
- Ligaduras

Qué se va a hacer conjuntamente

- Investigación del formato MusicXML
- Investigación del formato jMusic
- Investigación del formato lilypond
- Interfaz del programa
- Pruebas

Qué no se va a hacer

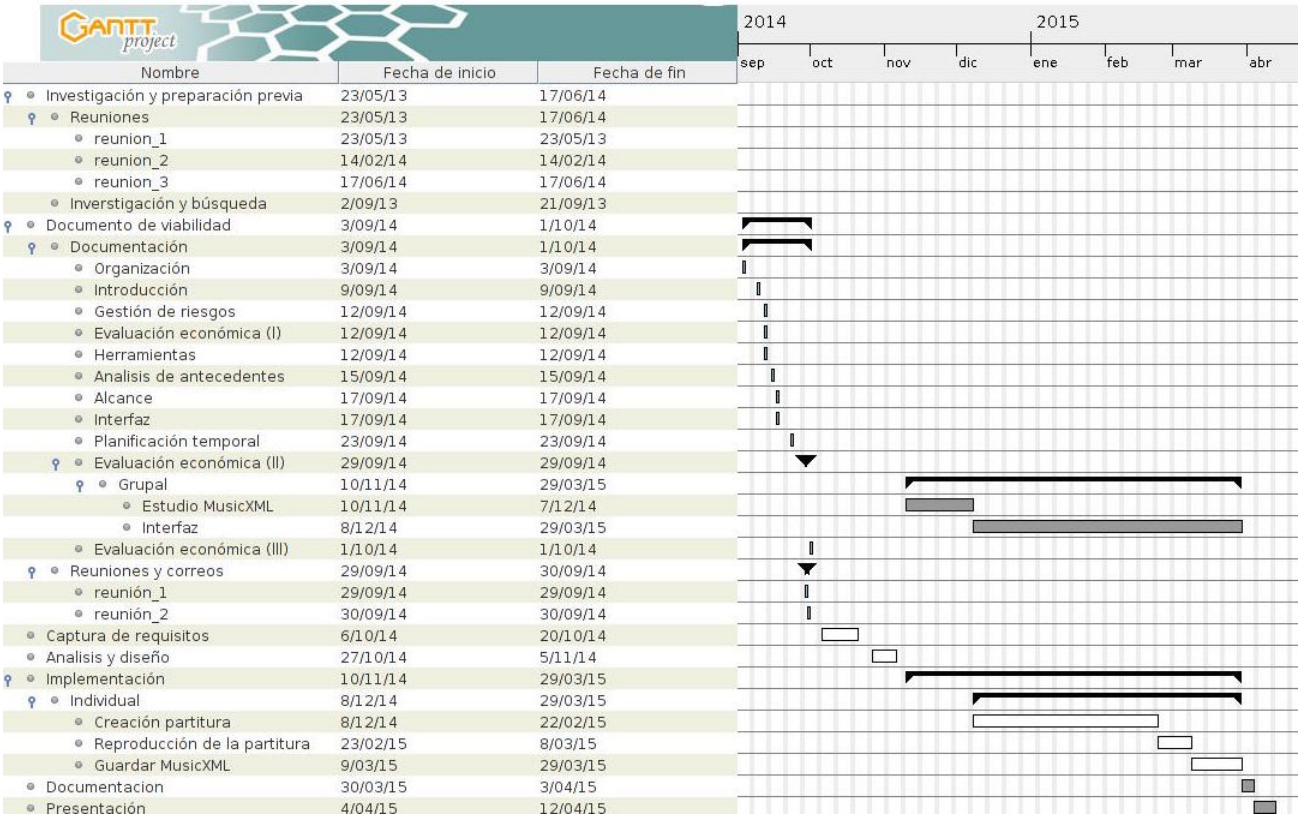
- Ritmos sin altura
- Velocidades tales como: allegro, moderato, vivace, presto, andante, adagio, largo, lento, ...
- Alteraciones múltiples: Doble sostenido, doble bemol
- Signos de repetición
- Anacrusas

Planificación temporal

Tarea	Subtarea	Fecha	Tiempo
Investigación y preparación previa		2013/05/23 2014/06/17	30h
	Reuniones y correos		6h
	Investigación y búsqueda		24h
Documento de viabilidad		2014/09/03 2014/10/05	20h30min
	Redacción		20h
	Reuniones		30min
Captura de Requisitos		2014/10/06 2014/10/26	$\simeq 45h$
Análisis y Diseño		2014/10/27 2014/11/09	$\simeq 30h$
Implementación	**Total G e I**	2014/11/10 2015/03/29	$\simeq 300h$
	Grupal	2014/11/10 2015/03/29	$\simeq 60h$
	1.Estudio del MusicXML	2014/11/10 2014/12/07	$\simeq 40h$
	2.Interfaz principal	2014/12/08 2015/03/29	$\simeq 20h$
	Individual	2014/12/08 2015/03/29	$\simeq 240h$
	1.Creación partitura	2014/12/08 2015/02/22	$\simeq 160h$
	2.Reproducción de la partitura	2015/02/23 2015/03/08	$\simeq 30h$
	3.Guardar musicXML	2015/03/09 2015/03/29	$\simeq 50h$
Documentación		2015/03/30 2015/04/03	$\simeq 20h$
Presentación		2015/04/04 2015/04/12	$\simeq 15h$
TOTAL		2013/05/23 2015/04/12	$\simeq 460h$

Tabla 2.1: Tabla de planificación temporal

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



Nombre	Fecha de inicio	Fecha de fin
Investigación y preparación previa	23/05/13	17/06/14
Reuniones	23/05/13	17/06/14
reunion_1	23/05/13	23/05/13
reunion_2	14/02/14	14/02/14
reunion_3	17/06/14	17/06/14
Inverstigación y búsqueda	2/09/13	21/09/13
Documento de viabilidad	3/09/14	1/10/14
Documentación	3/09/14	1/10/14
Organización	3/09/14	3/09/14
Introducción	9/09/14	9/09/14
Gestión de riesgos	12/09/14	12/09/14
Evaluación económica (I)	12/09/14	12/09/14
Herramientas	12/09/14	12/09/14
Analisis de antecedentes	15/09/14	15/09/14
Alcance	17/09/14	17/09/14
Interfaz	17/09/14	17/09/14
Planificación temporal	23/09/14	23/09/14
Evaluación económica (II)	29/09/14	29/09/14
Grupal	10/11/14	29/03/15
Estudio MusicXML	10/11/14	7/12/14
Interfaz	8/12/14	29/03/15
Evaluación económica (III)	1/10/14	1/10/14
Reuniones y correos	29/09/14	30/09/14
reunión_1	29/09/14	29/09/14
reunión_2	30/09/14	30/09/14
Captura de requisitos	6/10/14	20/10/14
Analisis y diseño	27/10/14	5/11/14
Implementación	10/11/14	29/03/15
Individual	8/12/14	29/03/15
Creación partitura	8/12/14	22/02/15
Reproducción de la partitura	23/02/15	8/03/15
Guardar MusicXML	9/03/15	29/03/15
Documentacion	30/03/15	3/04/15
Presentación	4/04/15	12/04/15

Figura 2.1: Gantt Preproyecto Ane

Herramientas y lenguajes de programación utilizados

Java Lenguaje de programación orientado a objetos basado en clases multiplataforma.

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 una creación más sencilla de documentos LaTeX.

MIDI Protocolo estándar que permite a los dispositivos generar sonidos.

Cacoo Herramienta online para crear esquemas uml, modelo de dominios, diagramas de clases, diagramas de secuencia entre otros.

Gantt project 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 via 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 videos utilizado para visualizar tutoriales referentes a nuestro 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.

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

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.

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=540€
  - La vida de un ordenador portátil se estima que es de 5 años. La amortización anual será de:
$$\frac{540€}{5 \text{ años}} = 108€$$
  - Por lo tanto, la amortización durante el proyecto será de:
$$\frac{108€ * 7\text{meses}}{12 \text{ meses}} = 63 €$$

Software No se hará gasto en Software, dado que, la mayoria de los programas que utilizamos 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:
$$\frac{0{,}6382€ * 40\text{l}}{1000\text{l}} = 0{,}0255€$$

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

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

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

- Desplazamientos: 16€ por mes. Por lo tanto,

$$16€ * 7\text{meses}{=}112€$$

- Internet:

$$30€ * 7 \text{ meses} = 210€$$

- Movil:

$$11€/\text{mes} * 7\text{meses} = 77€$$

- Total de otros gastos:

$$315 + 18{,}8 + 112 + 210 + 77 = 732{,}8€$$

Coste total del proyecto:

$$5520 + 63 + 732{,}8 = 6315{,}8€$$

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 30€: $\frac{6315{,}8€ \text{ total}}{30€ \text{ por licencia}} = 211$ licencias estimadas.
- Licencias a 15€: $\frac{6315{,}8€ \text{ total}}{15€ \text{ por licencia}} = 421$ licencias estimadas.

2.3. Replanificación

Debido a la incorporación de la herramienta jMusic (para conocer más acerca de la herramienta, ver apéndice A en la página 91), se decidió rehacer la planificacón del proye

Alcance

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

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 pendientes de mi compañero de proyecto, la jornada laboral y la búsqueda de documentación de la libreria jMusic, se decidió seguir con el código hasta finales de mayo(2015), y demorar la entrega a junio(2015) para realizar la presentación en julio(2015).

Finalmente, al no ser posible la finalización total 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 B en la página 97), a diferencia de un formato propio creado por el 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. La documentación está abierta a que cualquier desarrollador pueda hacer uso de ella.

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

3.2. Estudio de diferentes alternativas existentes

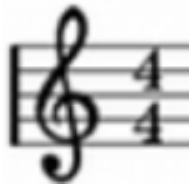
			
Denemo	Encore	Finale	Frescobaldi
			
Guitar Pro	Harmony Assistant	LenMus	LilyPond
			
MuseScore	NoteWorthy Composer	Nted	Sibelius

Figura 3.1: Lista aplicaciones similares

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

Encore Editor de partituras para Windows y OS X. Diseñado por GVOX. Se caracteriza por ser el pionero en añadir la opción de arrastrar una nota al pentagrama mediate el ratón del PC. Usado tanto por noveles como profesionales.

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

Frescobaldi Editor de partituras con formato LilyPond. Es de código abierto. Usado en Windows, OS X y Linux.

Guitar Pro Aunque esté orientado a partituras para guitarra y bajo también cabe la posibilidad de usarlo con más instrumentos.

Diseñado para OS X, Windows y Linux. Lo comercializa la empresa Arobas Music.

Harmony Assistant Editado por Myriad Software. Usado en Windows, OS X y Linux. Es un editor de partituras para diferentes usuarios.

Existe una variante del programa llamada Melody Assistant.

LenMus Software abierto y gratuito para lenguaje musical. En julio del 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 grabado musical que se centra en la calidad de la notación musical intentando imitar la tipografia de las partituras tradicionales. Es de código abierto y forma parte del proyecto GNU.

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, OS X y Linux.

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

Nted Es un 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.

Comparación de características

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

	Licencia	**Precios** (1USD = 0.77 EUR)		
		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		
GuitarPro	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 de las aplicaciones soportan multiples formatos pero en esta tabla vamos a mencionar los más usados.

	Entrada	**Salida**
Denemo	Ly, xml,midi	png,pdf,midi,audio,ly
Encore	midi,enc,mus	pdf,midi
Finale	mus,xml,midi	wav,mp3,epub,midi,xml,pdf,jpg,png
Frescobaldi	xml,ly	xml,pdf,ly
GuitarPro	midi,ascii,mxl,powerTab,tablEdit	midi,ascii,xml,wav,png,pdf
Harmony A.	midi,abc,tablEdit,enc,mus,nwc,gtp,xml	wav,aiff,ogg,mp3,midi,abc,tablEdit,xml
LenMus	lmb	lmb
LilyPond	ly	pdf,midi
MuseScore	mscz,mscx,xml,midi	pdf,png,svg,wav,mp3,ogg,xml,midi,ly,
NWC	nwc,midi	nwc,nwctxt,midi
Nted	midi,xml,ntd	midi,pdf,png,svg,ly,ntd
Sibelius	sib,midi,mus	sib,midi

Tabla 3.2: Formatos de archivos 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 utilizarla hay que trabajar con la línea de comandos.

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

Denemo (Figura 3.2) es una aplicación muy sencilla y tiene todas las 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 que todo el mundo conoce.

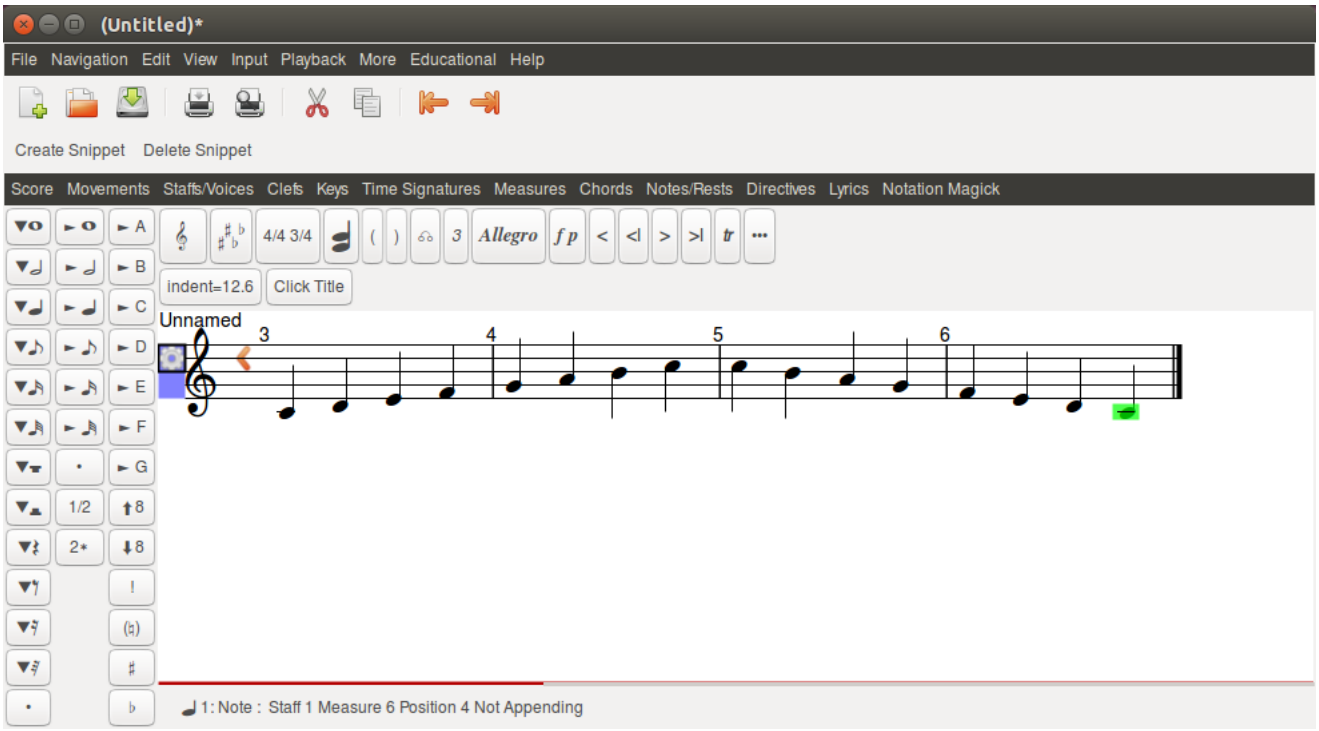


Figura 3.2: Captura de pantalla de Denemo

MusikEbal (Figura 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 parte 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.

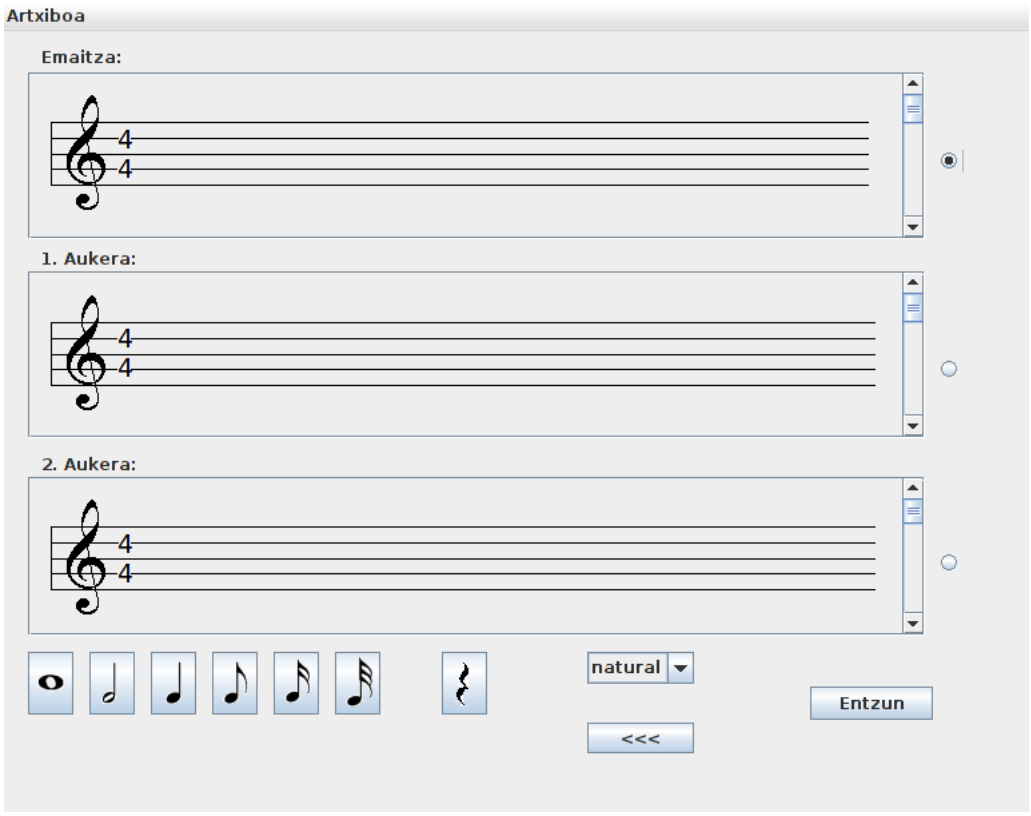


Figura 3.3: Captura de pantalla de MusikEbal

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 musicales del compositor hungaro "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 algún 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. Versión antigua (basada en MusicXML)

MusicXML es un formato de notación musical basado en 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.

Ya que la aplicación va a manejar archivos de formato MusicXML, se analizará la estructura para organizar internamente la jerarquía de clases. Para más información sobre MusicXML ver anexo B en la página 97.

Actores

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

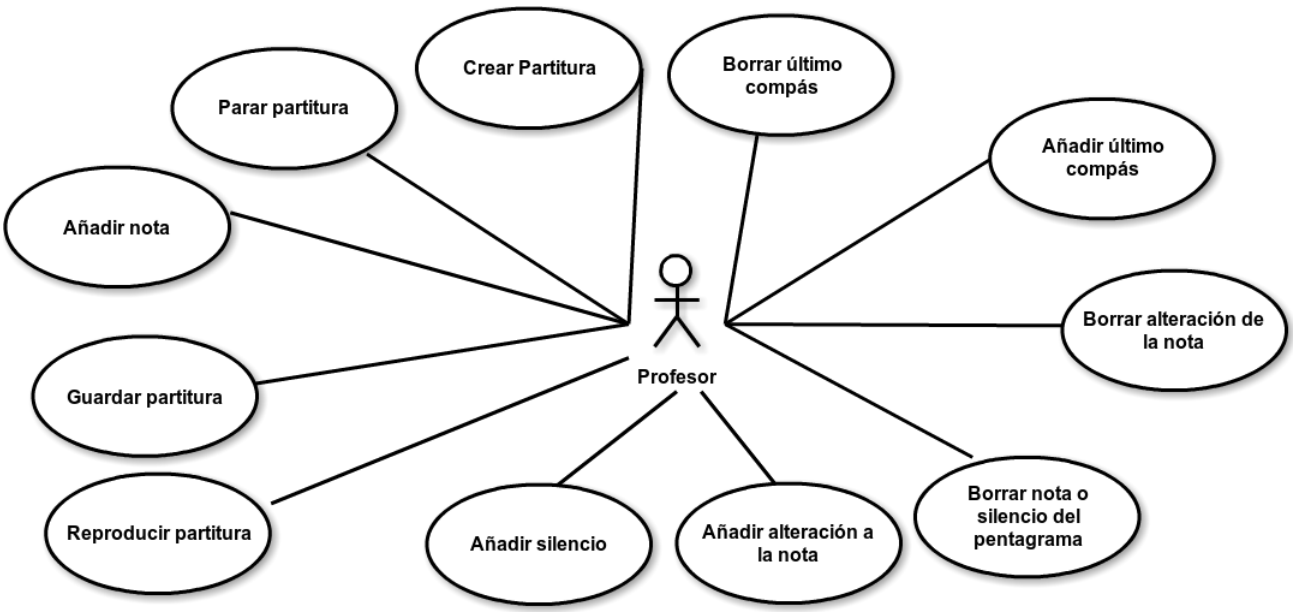


Figura 4.1: Actor: Profesor

Casos de uso del profesor

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

Crear partitura

Descripción Creará una partitura vacía para poder insertar en ella elementos musicales. Aparecerá como un botón en la ventana de bienvenida

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

Requisitos no funcionales

Postcondición Se muestra una partitura vacia.

Flujo de eventos

1. El usuario pulsa en el botón nueva partitura.
2. En la siguiente ventana aparecerá el pentagrama vacío para que el usuario pueda añadirle elementos

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.

Guardar partitura

Descripción El programa guardará todo el contenido de la partitura en un archivo MusicXML, que más tarde se podrá volver a recuperar.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Todos los cambios de la partitura han sido guardados.

Flujo de eventos

1. El usuario pulsa el icono guardar partitura.
2. En la siguiente ventana el usuario indicará el lugar donde desea guardar la partitura.
   - Si el archivo a guardar existe:
     a) Sobrescribir el antiguo archivo.
   - Si el archivo a guardar no existe:
     a) Guardarlo en el sitio indicado por el usuario.
3. Se guardará la partitura en el lugar indicado por el usuario.

Reproducir partitura

Descripción Mediante este botón podremos reproducir nuestra partitura. Cuando se pinche sobre él empezará desde el principio del pentagrama o seguirá desde la última posición que fue pausado, y el botón se convertirá en el icono pausa.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Se escuchará la melodía de la partitura.

Flujo de eventos

1. El usuario pulsa el icono de "*play*".
2. El módulo de reproducción leerá la partitura para transformar las notas en melodía.
   - Si hay una marca de compás guardada:
     a) Se reproducirá la melodía desde ese punto en adelante.
   - Si no hay una marca de compás guardada:
     a) Se reproducirá la melodía desde el principio de la partitura.
3. Se escuchará la melodía.
4. El icono "*play*" se convertirá en "pause".

Parar partitura

Descripción Si ya no queremos escuchar más la partitura, este botón parará todo el proceso de reproducción.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Se detendrá la melodía de la partitura.

Flujo de eventos

1. El usuario pulsa el icono de "*stop*".
2. Se inicializa la marca de compás (Para que la reproducción se haga desde el principio).
3. Se detendrá la melodía.

Añadir nota

Descripción Esta funcionalidad estará colocada en la parte izquierda de la pantalla. Veremos todas las notas visibles, que al pinchar sobre cualquiera de ellas podremos colocarla en el pentagrama.

Precondición Tener abierta una partitura.

Requisitos no funcionales

Postcondición Se añadirá una nota en la partitura.

Flujo de eventos

1. El usuario pulsará en la nota que quiera y la arrastrará al pentagrama.
   - Si la duración de la nota sobrepasa la capacidad restante del compás:
     a) Una advertencia tipo (sonido, parpadeo...).
   - Si la duración de la nota no sobrepasa la capacidad restante del compás:
     a) Añadir la nota al pentagrama.
2. El pentagrama tendrá un elemento más.

Añadir silencio

Descripción Al igual que con la funcionalidad "añadir nota al pentagrama", ocurre lo mismo pero con los silencios. Utilizaremos este botón cuando no queramos que en esa parte de la partitura vaya cualquier sonido.

Precondición Tener abierta una partitura.

Requisitos no funcionales

Postcondición Se añadirá un nuevo silencio en la partitura.

Flujo de eventos

1. El usuario pulsará en el silencio que quiera y lo arrastrará al pentagrama.
   - Si la duración del silencio sobrepasa la capacidad restante del compás:
     a) Una advertencia tipo (sonido, parpadeo...).
   - Si la duración del silencio no sobrepasa la capacidad restante del compás:
     a) Añadir silenio al pentagrama.
2. El pentagrama tendrá un silencio más.

Añadir alteración a la nota

Descripción Para subir o bajar la tonalidad de las notas pincharemos sobre este botón. Al pinchar sobre él, tendremos tres opciones, una para insertar bemol, otra para insertar sostenido y la última para insertar becuadro.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Se añadirá una alteración a la nota que quiera el usuario.

Flujo de eventos

1. El usuario pulsará sobre la alteración que quiera.
2. La arrastrará al lado de la nota que quiere alterar.
   - Si la nota contiene la misma alteración:
     a) No hacer nada.
   - Si la nota contiene otra diferente:
     a) Cambiarla por la nueva.
   - Si la nota no tiene ninguna alteración:
     a) Añadir la alteración a la nota.
3. Finalmente se añadirá una alteración a la nota del pentagrama.

Borrar nota del pentagrama

Descripción Mediante esta funcionalidad se podrá eliminar una nota añadida en el pentagrama. En el caso de que la nota tuviese alguna alteración también se borrará.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El pentagrama contará con una nota menos.

Flujo de eventos

1. El usuario selecciona la nota que quiere borrar.
2. A continuación pulsa la tecla de borrar del teclado.
3. Se borrará la nota.

Borrar silencio del pentagrama

Descripción De manera similar a la funcionalidad anterior podremos borrar silencios añadidos al pentagrama.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El programa contará con un silencio menos.

Flujo de eventos

1. El usuario selecciona el silencio que quiere borrar.
2. Una vez seleccionado el silencio pulsar la tecla borrar del teclado.
3. Finalmente se borrará ese silencio de la partitura.

Borrar alteración de la nota

Descripción Esta funcionalidad al igual que las dos anteriores borrará cualquier alteración que tuviese la nota. Sólo se borrará la alteración, no la nota.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Se borrará la alteración de la nota que el usuario decida.

Flujo de eventos

1. El usuario elegirá que alteración quiere borrar.
2. A continuación pulsará sobre la tecla borrar del teclado.
   - Si la nota se encuentra en la armadura:
     a) Alterar la nota con la alteración de la armadura
   - Si la nota no se encuentra en al armadura:
     a) Dejar la nota sin alteraciones.
3. Finalmente se borrará la alteración de la nota.

Añadir último compás

Descripción Otra de las funcionalidades situadas en la parte superior del programa. En el caso de querer añadir otro compás más a la partitura se hará pulsando sobre él.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El pentagrama tendrá un compás más.

Flujo de eventos

1. El usuario pulsará sobre el botón superior de añadir compás.
2. Se añadira un compás nuevo al final de la partitura.

Borrar último compás

Descripción La funcionalidad opuesta a la anterior mencionada. Borrará el último compás de la partitura con todo su contenido.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El pentagrama contará con un compás menos.

Flujo de eventos

1. El usuario pulsará sobre el botón superior de eliminar compás.
2. A continuación se borrará el último compás de la partitura con todos los elementos que contenga.
3. El pentagrama tendrá un compás menos.

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.

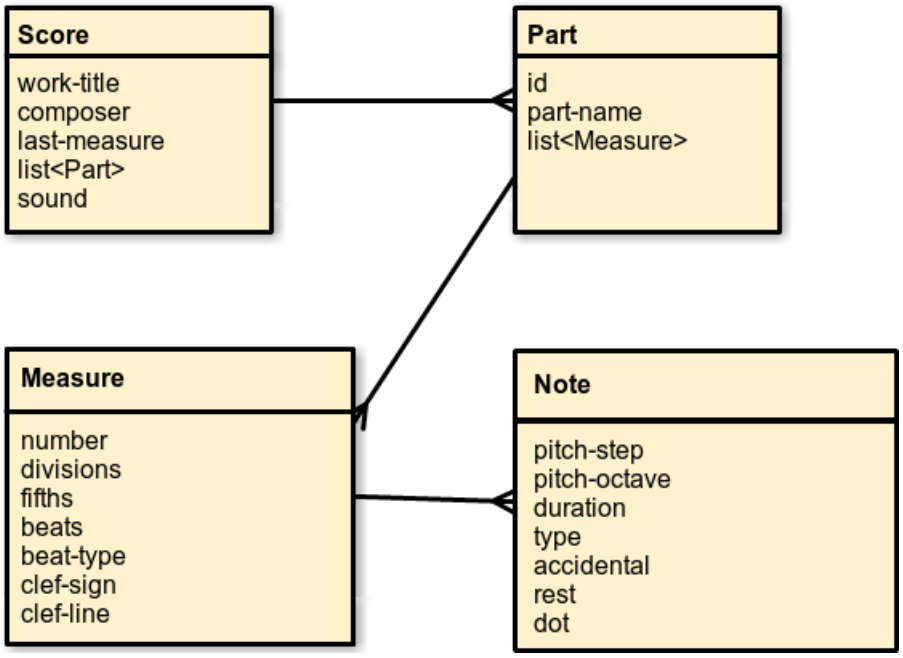


Figura 4.2: Modelo de dominio

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 cada parte
- part-name: nombre visible de la parte
- last<measure>: listado de compases

Measure Compás

- number: número del compás. Empieza en 1, salvo que, el primero sea anacrusa.
- divisions: división mínima respecto a la negra en la partitura. Ver columna *Div.* de la tabla B.3 en la página 99.
- fifths: armadura. El rango irá de -7 (7 bemoles) a 7 (7 sostenidos). 0 indicará que no tiene alteraciones. Ver tabla B.4 en la página 99.
- *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 B.3 en la página 99.
- *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*<node>: listado de notas

Note Nota

- *pitch-step*: nombre de la nota. Ver tabla B.5 en la página 100.
- *pitch-octave*: en que octava está 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 B.3 en la página 99.
- *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 A en la página 91.

Actores

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

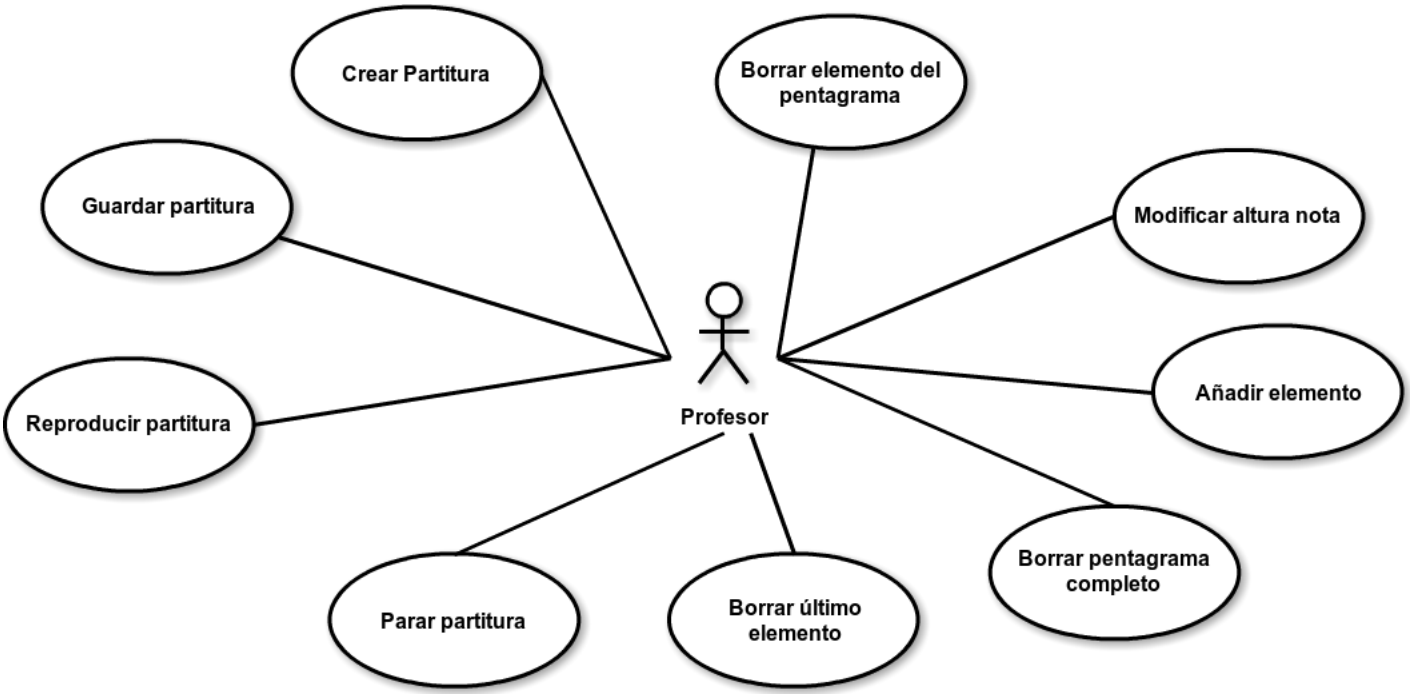


Figura 4.3: Actor: Profesor

Casos de uso del profesor

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

Crear partitura

Descripción Creará una partitura vacía para poder insertar en ella elementos musicales. Aparecerá como un botón en la ventana de bienvenida.

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

Requisitos no funcionales

Postcondición Se muestra una partitura vacia.

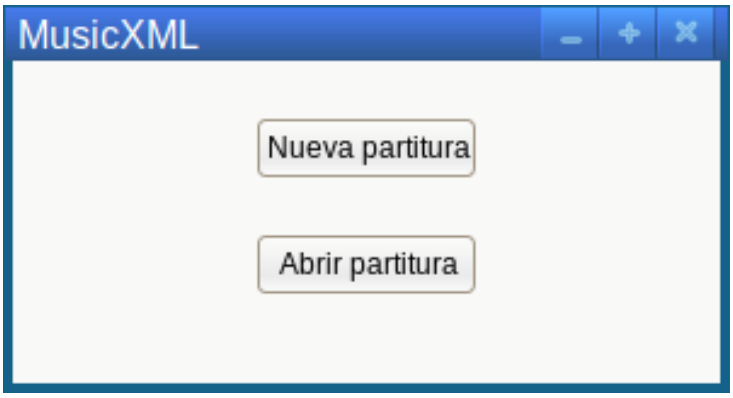


Figura 4.4: Interfaz Menú Principal

Flujo de eventos

1. El usuario pulsa en el botón nueva partitura (Fig. 4.4).
2. Aparece una ventana emergente que el usuario deberá rellenar (Fig. 4.5).
3. El usuario rellenará el título, autor, tempo, clave, armadura y compás.
4. Se crea la partitura.
5. Se modificar los datos de la partitura por los rellenados por el usuario.
6. En la siguiente ventana aparecerá un pentagrama vacío en el que el usuario irá añadiendo los elementos.

Prototipo de la interfaz Crear partitura

Guardar partitura

Descripción El programa guardará todo el contenido de la partitura en un archivo MusicXML, que más tarde se podrá volver a recuperar.

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Todos los cambios de la partitura han sido guardados.

Flujo de eventos

1. El usuario pulsa el icono guardar partitura.
2. En la siguiente ventana (Fig. 4.6) el usuario indicará el lugar donde desea guardar la partitura.
    - Si el archivo a guardar existe:

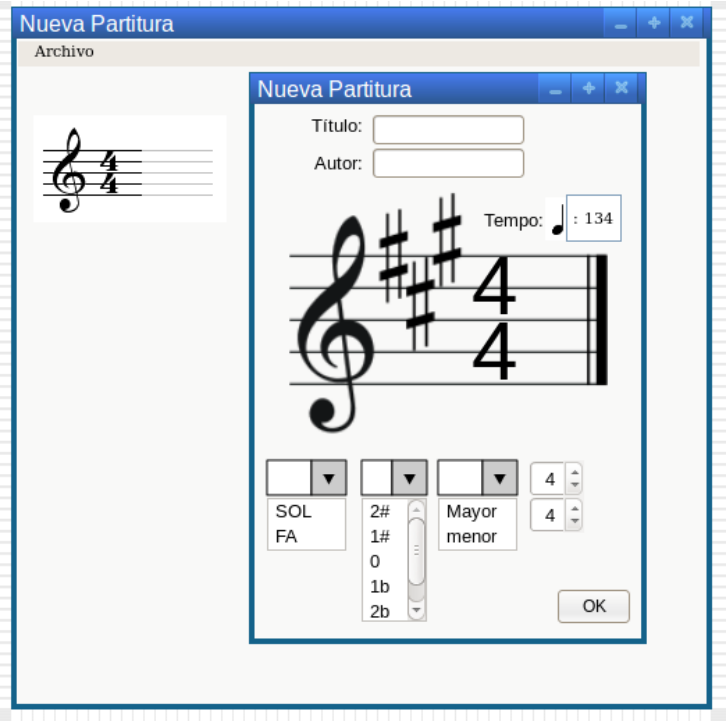


Figura 4.5: Interfaz partitura vacía

a) Sobrescribir el antiguo archivo.

- Si el archivo a guardar no existe:

a) Guardarlo en el sitio indicado por el usuario.

3. Se guardará la partitura en el lugar indicado por el usuario.

Prototipo de la interfaz Guardar partitura

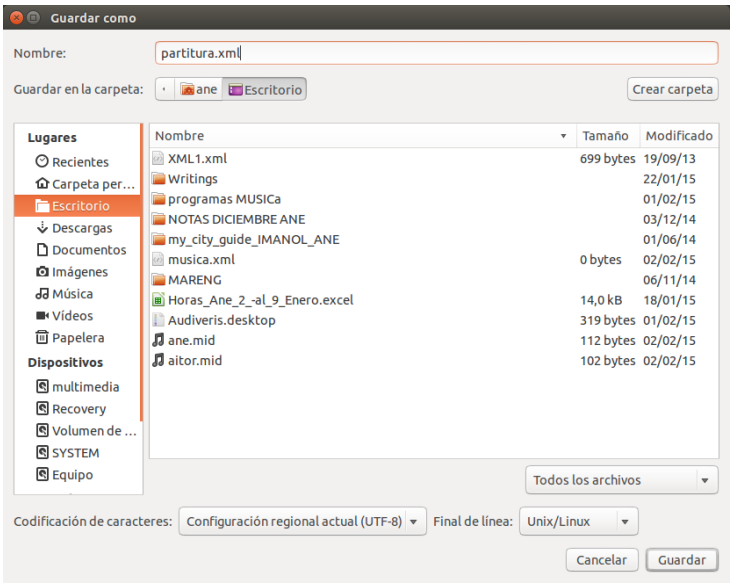


Figura 4.6: Interfaz guardar partitura

Reproducir partitura

Descripción Mediante este botón podremos reproducir nuestra partitura. Cuando se pinche sobre él empezará la reproducción (Fig. 4.7).

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Se escuchará la melodía de la partitura.

Flujo de eventos

1. El usuario pulsa sobre el menú superior "*play*" y dentro de este sobre la opción "*play all*".

2. Se escuchará la melodía.

Prototipo de la interfaz Reproducir partitura

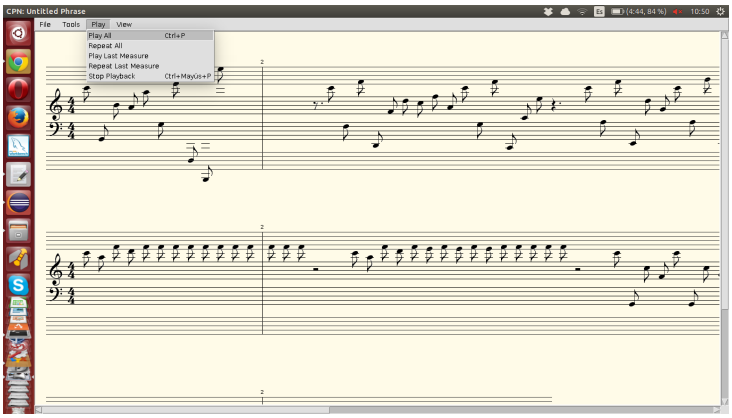


Figura 4.7: Interfaz reproducir partitura

Parar partitura

Descripción Si ya no queremos escuchar más la partitura, este botón parará todo el proceso de reproducción (Fig. 4.8).

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición Se detendrá la melodía de la partitura.

Flujo de eventos

1. El usuario pulsa sobre el menú superior "*play*" y dentro de este sobre la opción de *"stop playback"*.

2. Se detendrá la melodía.

Prototipo de la interfaz Parar partitura

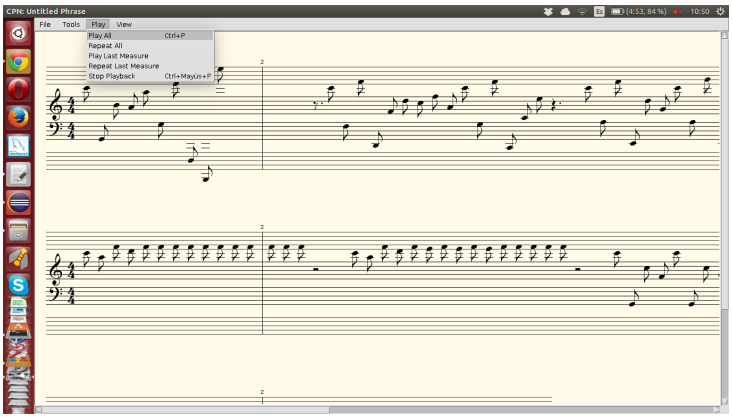


Figura 4.8: Interfaz parar partitura

Añadir elemento al pentagrama

Descripción Cuando se cargue la página crear partitura, nos aparecerá un pentagrama con un compás vacío donde con el ratón del ordenador podremos insertar un elemento ya sea una nota o un silencio. Para ello sólo debemos pulsar sobre el pentagrama y mover el ratón de izquierda a derecha. (Fig. 4.9).

Precondición Tener abierta una partitura.

Requisitos no funcionales

Postcondición Se añadirá un elemento en la partitura.

Flujo de eventos

1. El usuario pinchará y mantendrá pulsado el ratón en la posición que quiera añadir el elemento en el pentagrama
   - Si desea añadir una nota deberá mover el *mouse* mientras mantiene pulsado hacia la derecha.
   - Si desea añadir un silencio deberá mover el *mouse* mientras mantiene pulsado hacia la izquierda.
2. El pentagrama tendrá un elemento más.

Prototipo de la interfaz Añadir nota al pentagrama

Figura 4.9: Interfaz añadir nota al pentagrama

Modificar altura nota

Descripción Para subir o bajar la tonalidad de la nota pulsaremos sobre esta, y moveremos el ratón hacia arriba o hacia abajo. De esta manera podremos modificar la altura de la nota (Fig. 4.10).

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición La altura del elemento quedará modificada.

Flujo de eventos

1. El usuario pulsa sobre la nota que desea modificar.
2. Mientras mantiene pulsada la nota moverá el ratón de arriba abajo.
3. Finalmente se modificará la altura de la nota.

Prototipo de la interfaz Añadir alteración a la nota

Figura 4.10: Interfaz añadir alteración a la nota

Borrar elemento del pentagrama

Descripción Mediante esta funcionalidad se podrá eliminar un elemento añadido en el pentagrama. (Fig. 4.11).

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El pentagrama contará con un elemento menos.

Flujo de eventos

1. El usuario selecciona el elemento que quiere eliminar.
2. A continuación el usuario pinchará y mantendrá pulsado el ratón sobre el elemento en el pentagrama hasta ver la palabra "DELETE"
    - Si desea eliminar una nota deberá mover el *mouse* mientras mantiene pulsado hacia la izquierda.

- Si desea eliminar un silencio deberá mover el *mouse* mientras mantiene pulsado hacia la derecha.

3. Se borrará el elemento seleccionado.

Prototipo de la interfaz Borrar nota del pentagrama

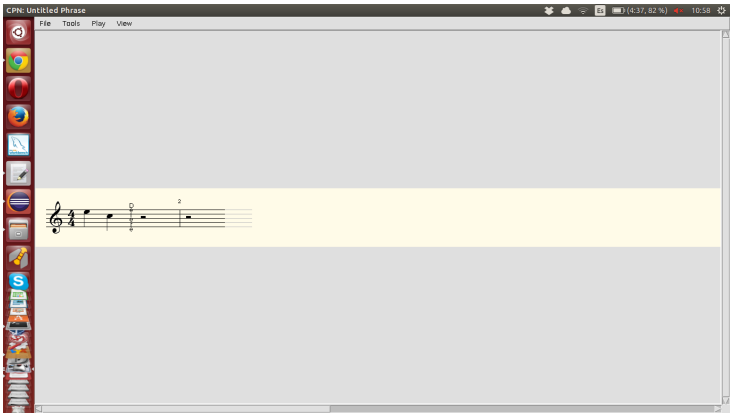


Figura 4.11: Interfaz borrar elemento del pentagrama

Borrar último elemento

Descripción Esta función borrará el último elemento añadido al pentagrama (Fig. 4.12).

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El pentagrama contará con un elemento menos.

Flujo de eventos

1. El usuario pulsará sobre "file" en el menú superior y a continuación sobre la opción de "Delete last note"
2. A continuación se borrará el último elemento del pentagrama.
3. El pentagrama tendrá un elemento menos.

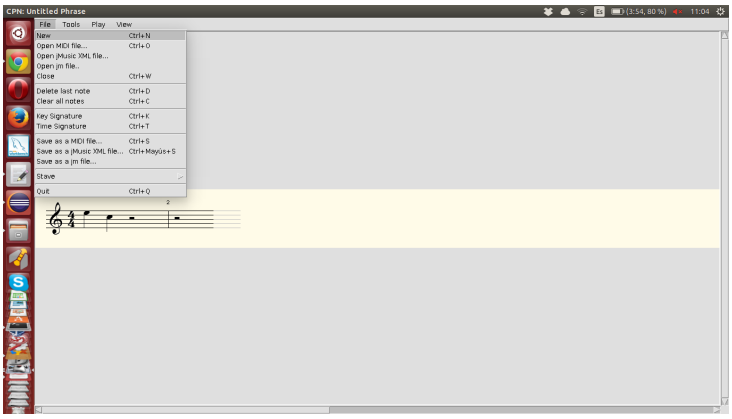


Figura 4.12: Interfaz borrar último elemento

Prototipo de la interfaz

Borrar pentagrama completo

Descripción Esta función borrará todo el pentagrama que hayamos creado (Fig. 4.13).

Precondición Tener una partitura abierta.

Requisitos no funcionales

Postcondición El pentagrama no tendrá ningún elemento.

Flujo de eventos

1. El usuario pulsará sobre la opción "file" del menú superior y a continuación sobre la opción "Clear all notes".
2. A continuación se borrará todo lo que haya en el pentagrama.
3. El pentagrama quedará vacío.

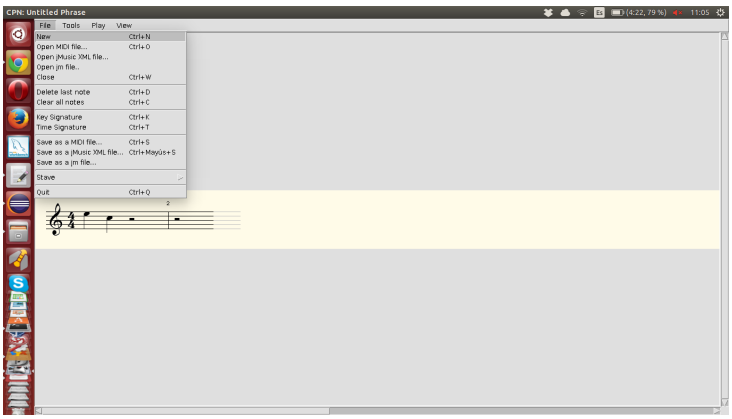


Figura 4.13: Interfaz borrar último compás

Prototipo de la interfaz

Modelo de dominio

En la figura 4.14 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.

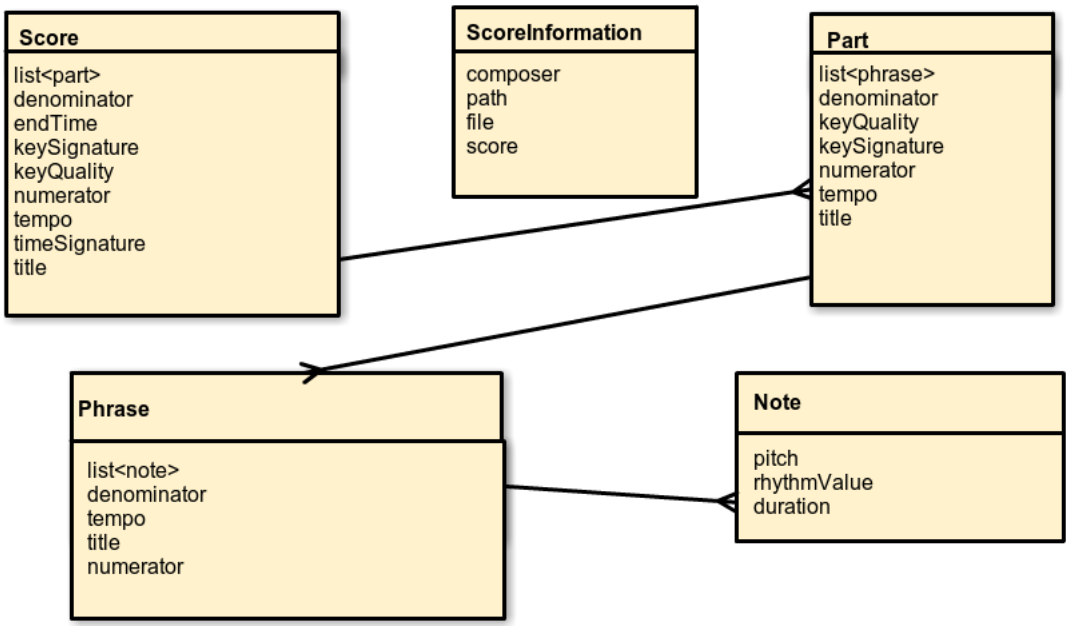


Figura 4.14: Modelo de dominio jMusic

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*: el tempo en pulsos por minuto.

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.

Versión antigua (basada en musicXML)

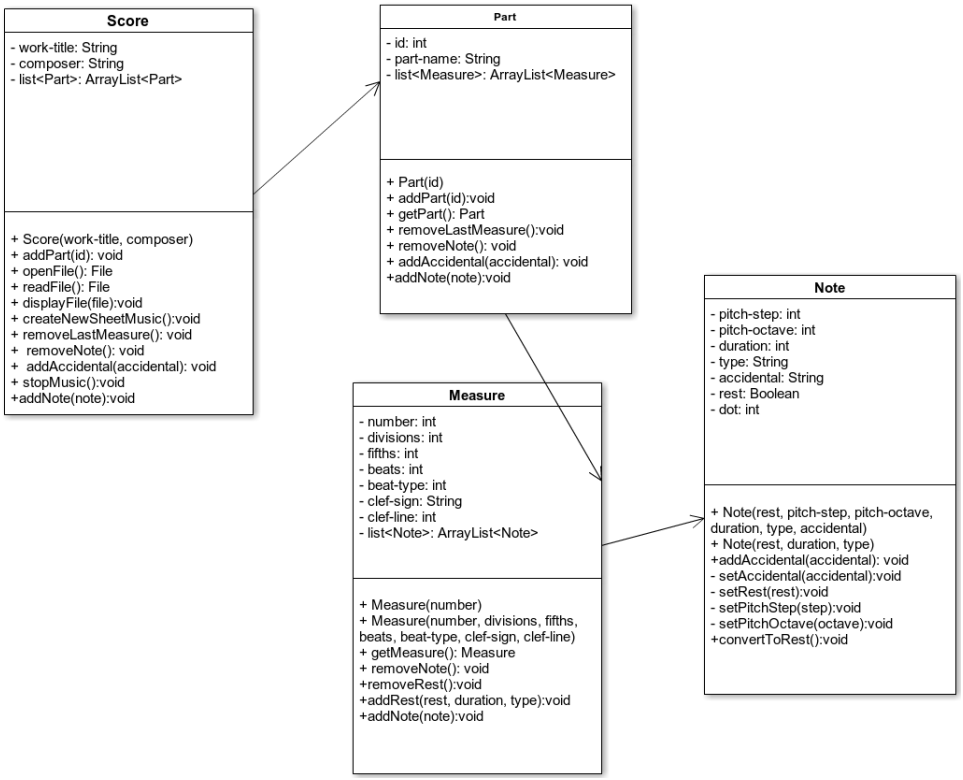


Figura 5.1: Transformación del modelo de dominio

Versión nueva (basada en jMusic)

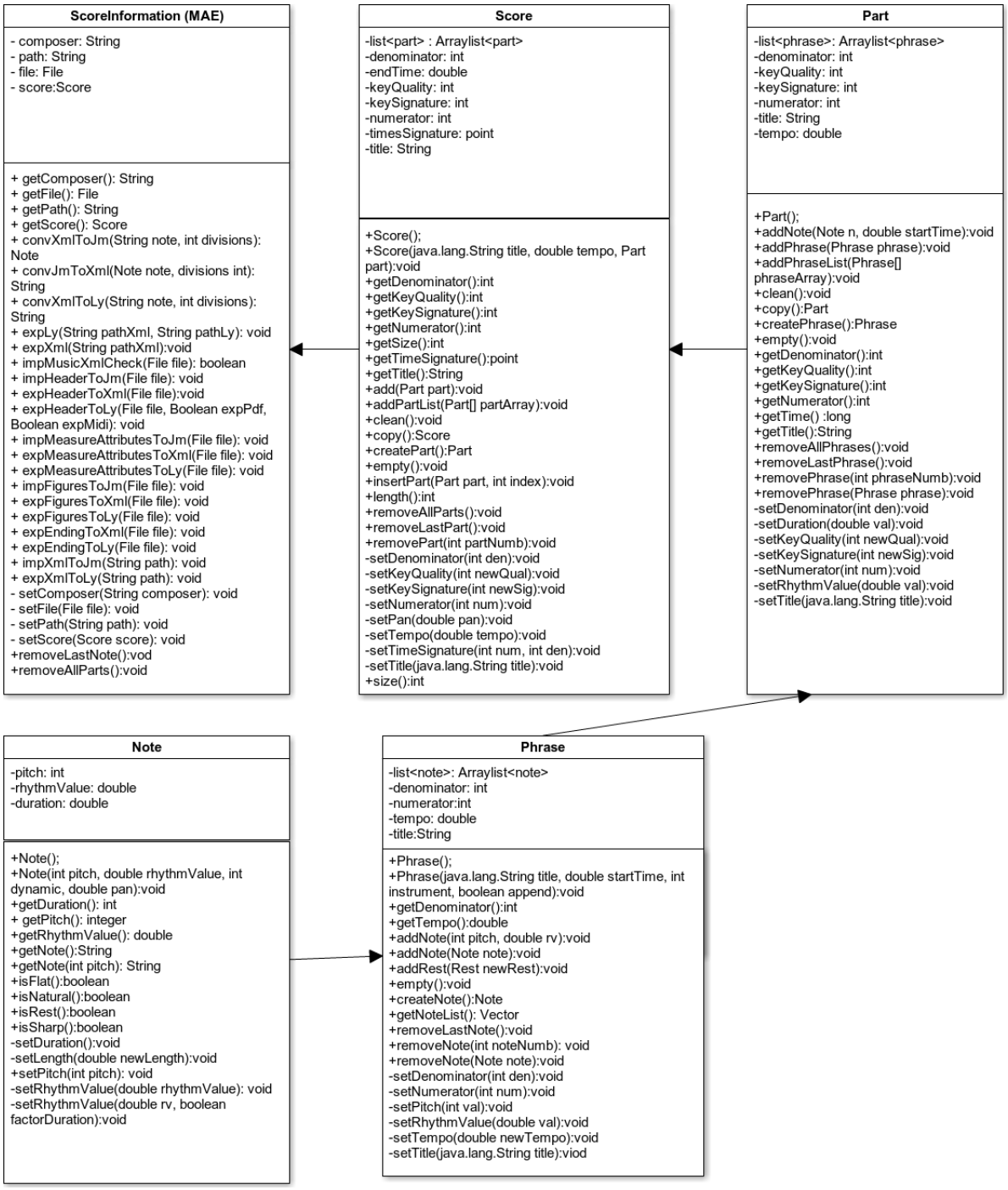


Figura 5.2: Transformación del modelo de dominio

5.2. Diagramas De Secuencia

En este apartado se visualizarán los diagramas de secuencia antiguos.

Versión antigua (basada en XML)

Aquí se detallará los diagramas de secuencia basados en MusicXml.

Crear partitura

Esta secuencia muestra cómo el usuario, en este caso el profesor, crearía una nueva partitura. Primero, se le mostrará una ventana emergente en la que elegir la estructura de la partitura (armadura, compás...).

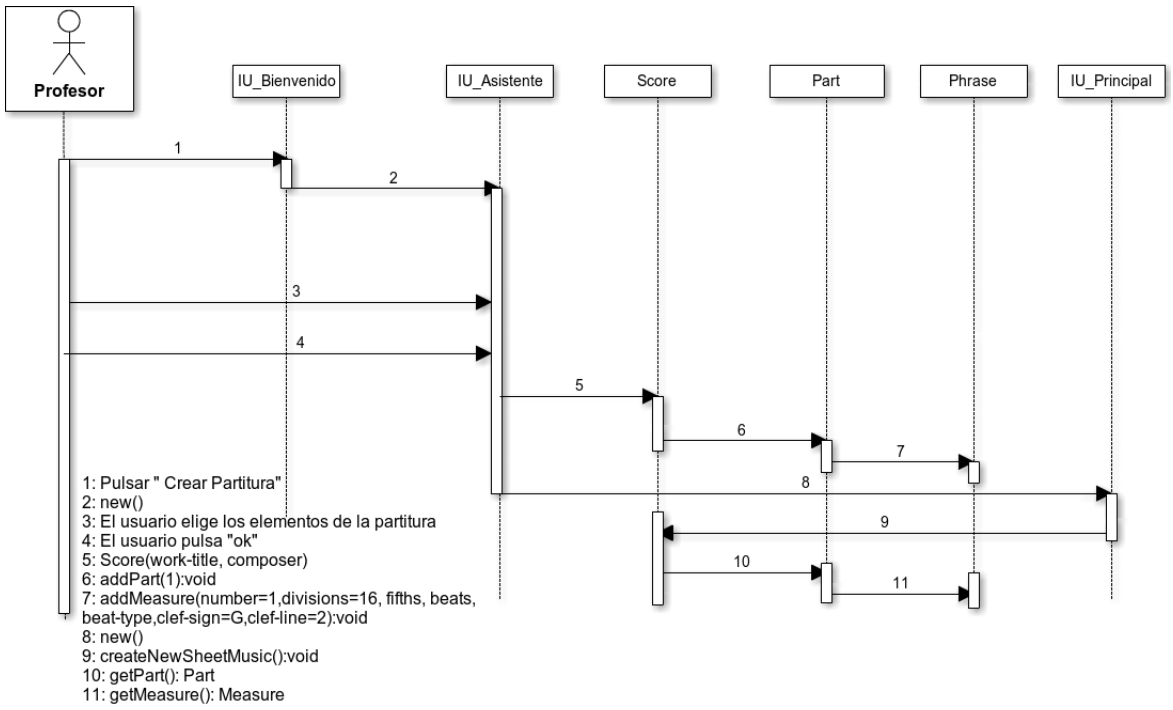


Figura 5.3: Crear Partitura

Abrir Partitura

En esta secuencia vemos como el usuario abre una partitura existente ubicada en cualquier lugar de su PC. Trás abrirla se visualizará tal y como la dejó la última vez que la modificó.

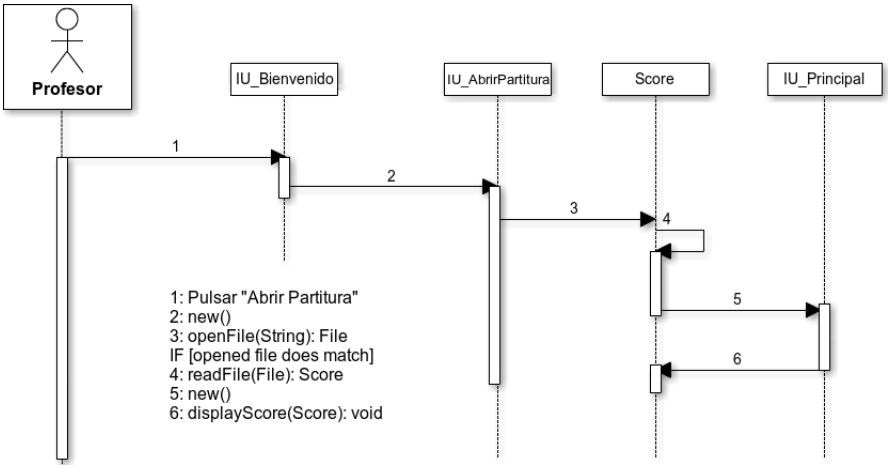


Figura 5.4: Abrir Partitura

Borrar Último Compás

Aquí vemos el progreso de cuando el usuario borra el último compás. Tras pulsar en el botón el último compás que contenga la partitura quedará eliminado.

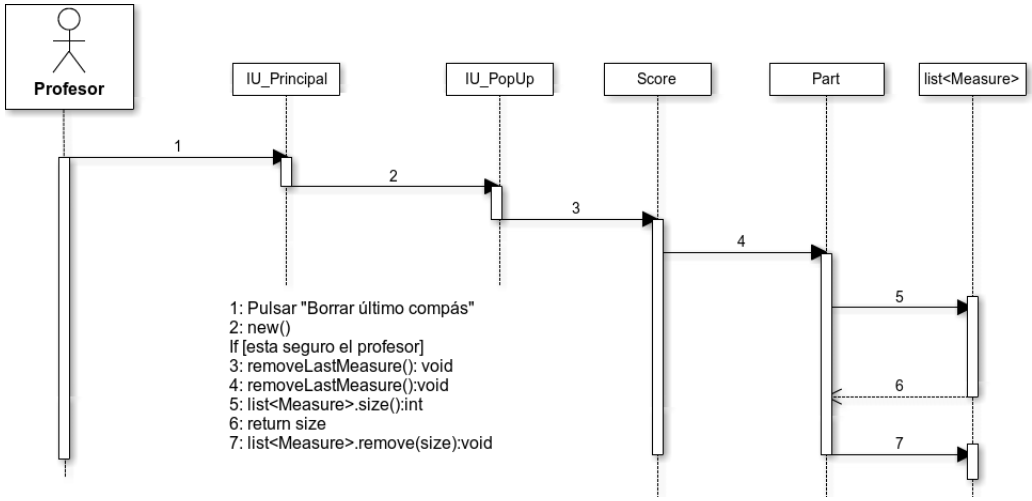


Figura 5.5: Borrar Último Compás

Añadir Último Compás

Este diagrama de secuencia muestra como se añadiría un compás al pentagrama cuando el usuario lo desee.

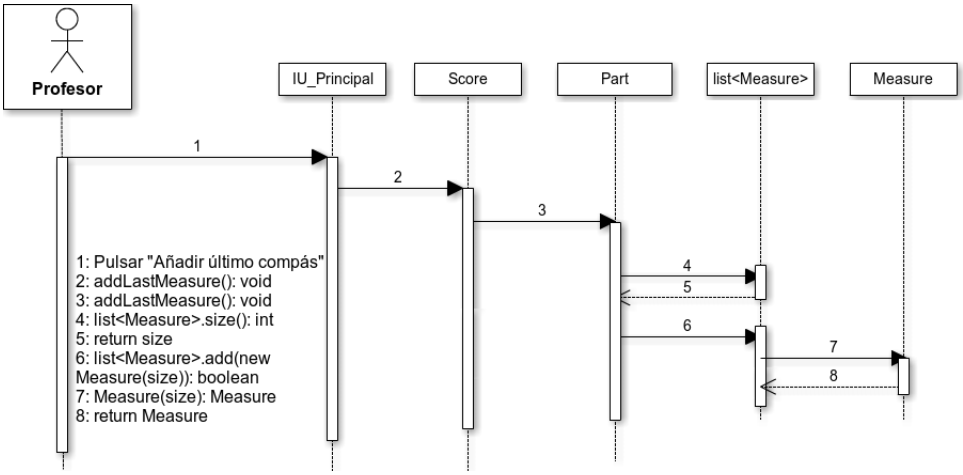


Figura 5.6: Añadir Último Compás

Borrar Alteración De La Nota

Esta secuencia muestra como podemos borrar una alteración, ya sea un bemól o un sostenido a cualquier nota del pentagrama.

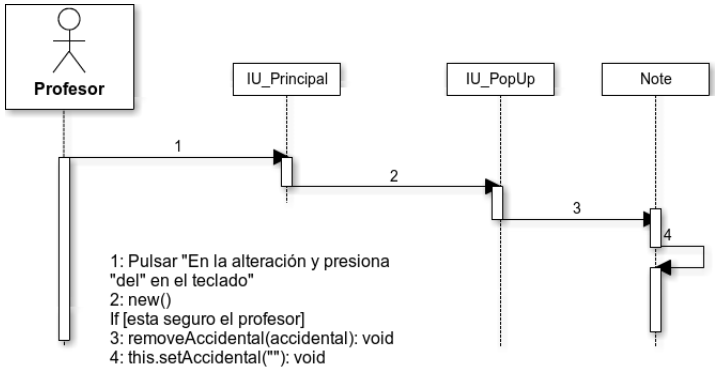


Figura 5.7: Borrar Alteración De La Nota

Borrar Nota o Silencio Del Pentagrama

Esta secuencia muestra como podemos borrar un elemento, ya sea una nota o un silencio del pentagrama.

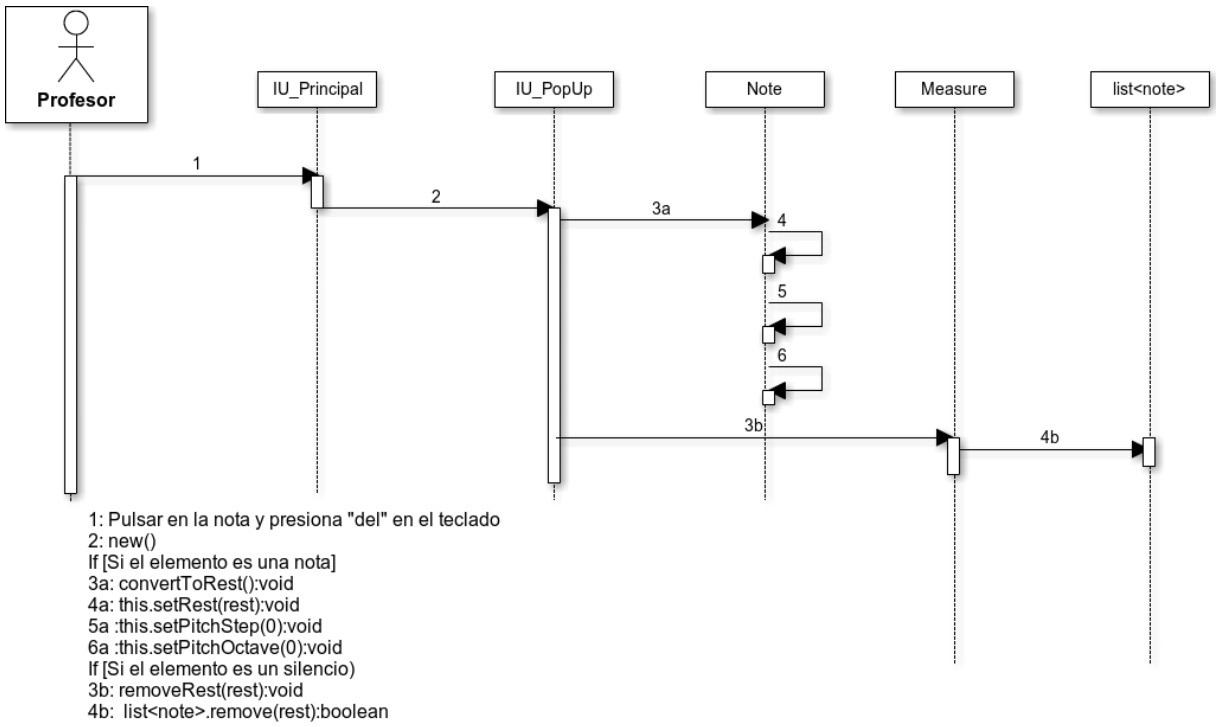


Figura 5.8: Borrar Nota o Silencio Del Pentagrama

Añadir Alteración A la Nota

Esta secuencia muestra como podemos borrar un elemento, ya sea una nota o un silencio del pentagrama.

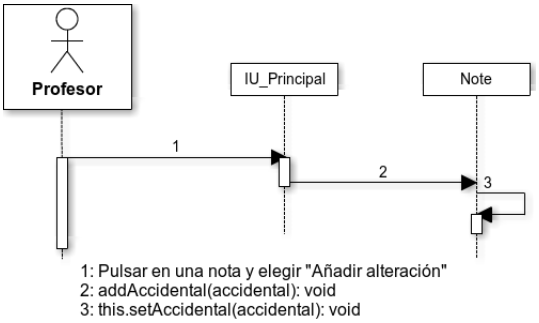


Figura 5.9: Añadir Alteración A La Nota

Añadir Silencio

En esta estructura podemos observar como añadimos un silencio a cualquier zona de la partitura.

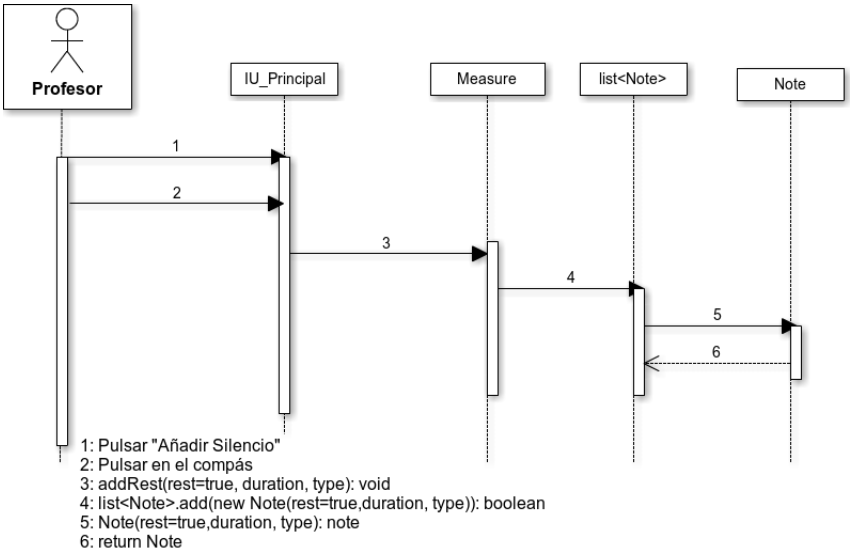


Figura 5.10: Añadir Silencio

Añadir Nota

En esta estructura podemos observar como se añade una nota al pentagrama.

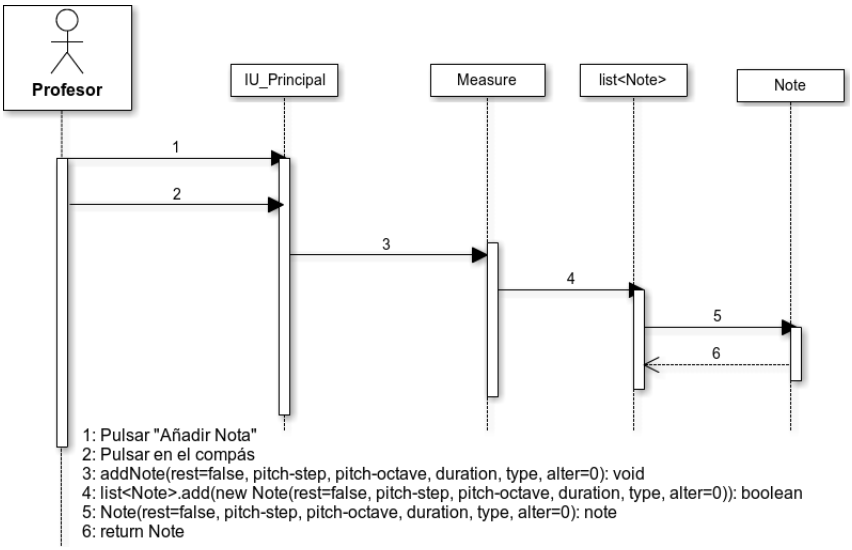


Figura 5.11: Añadir Nota

Parar Partitura

Una de las muchas funcionalidades que ofrece nuestro programa será la de poder parar una partitura cuando está este en marcha. La estructura a continuación lo explica.

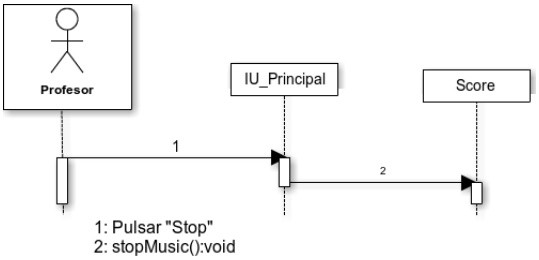


Figura 5.12: Parar Partitura

Reproducir Partitura

Ya que se puede parar la partitura tendra que estar reproduciendose antes. Este secuencia muestra los pasos.

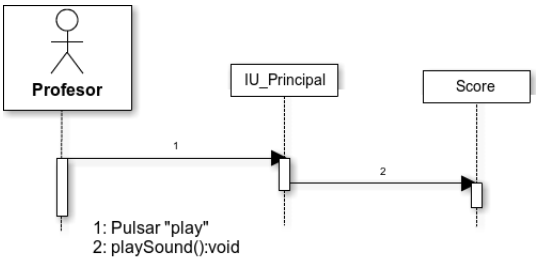


Figura 5.13: Reproducir Partitura

Versión nueva (basada en jMusic)

En este apartado se detallarán los diagramas de secuencia nuevos basados en JMusic.

Crear Partitura

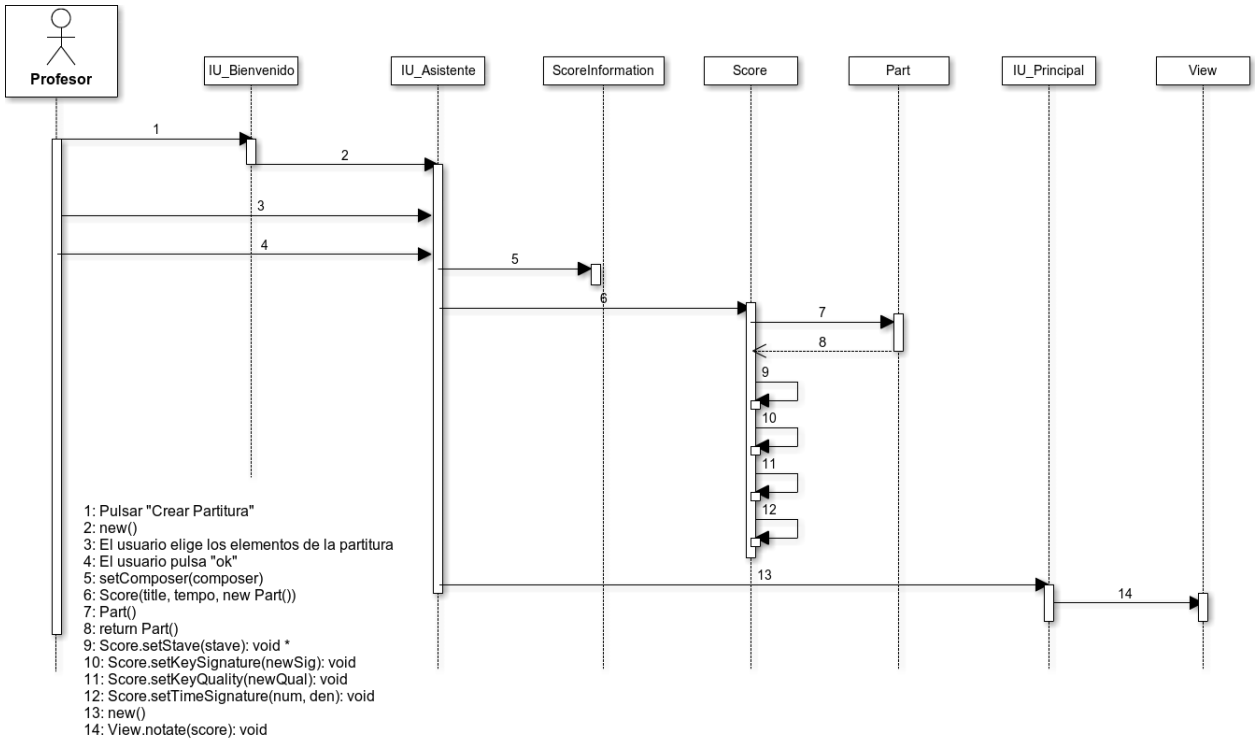


Figura 5.14: Crear Partitura

Guardar Partitura

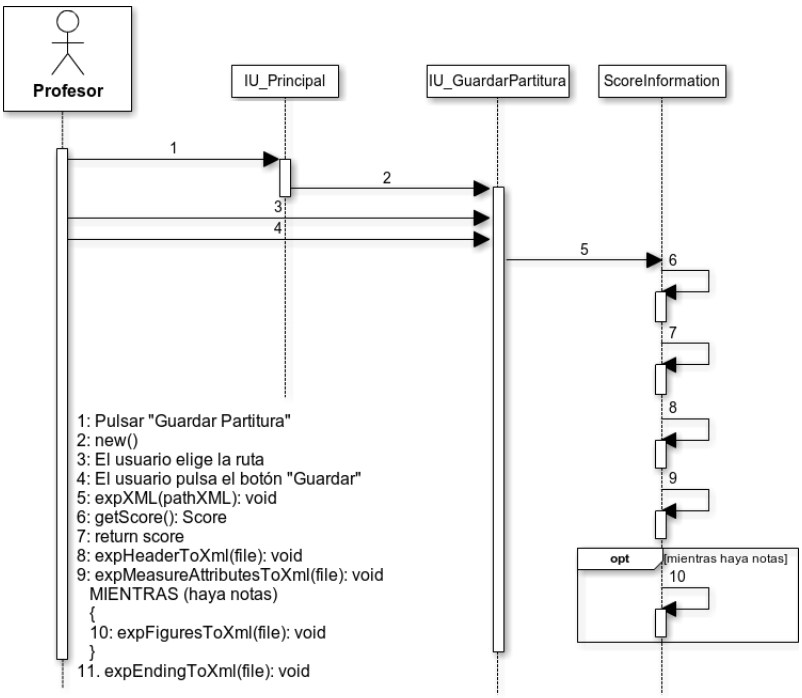


Figura 5.15: Guardar Partitura

Reproducir Partitura

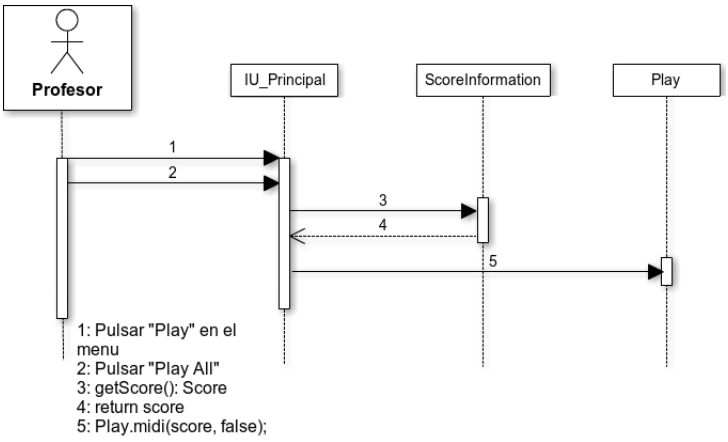


Figura 5.16: Reproducir Partitura

Parar Partitura

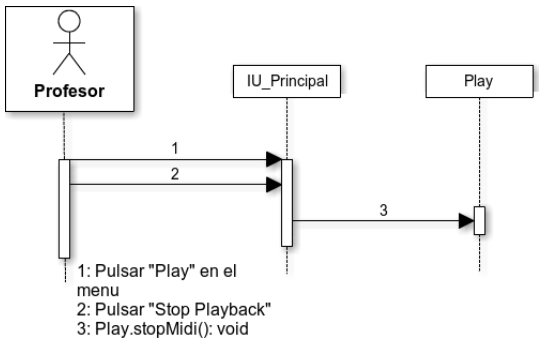


Figura 5.17: Parar Partitura

Borrar último elemento

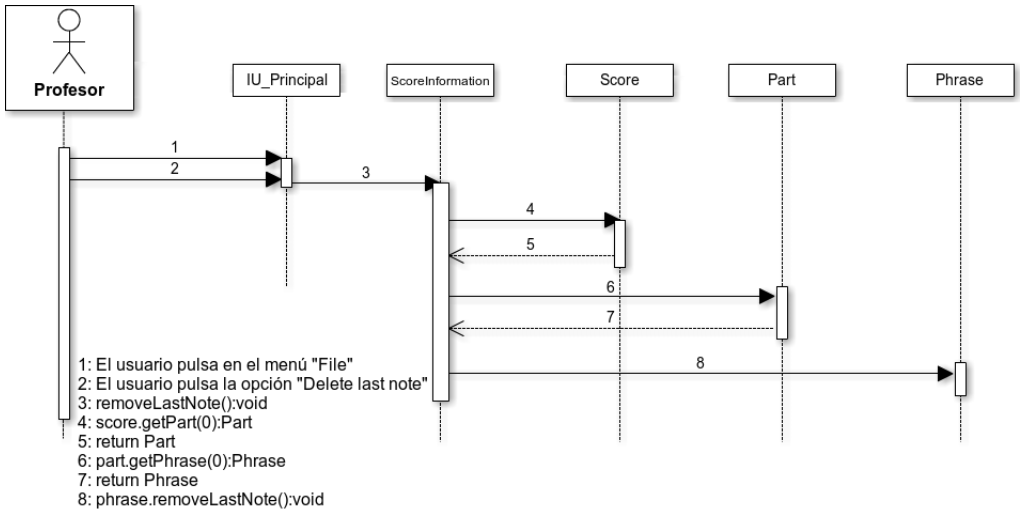


Figura 5.18: Borrar Último Elemento

Borrar pentagrama completo

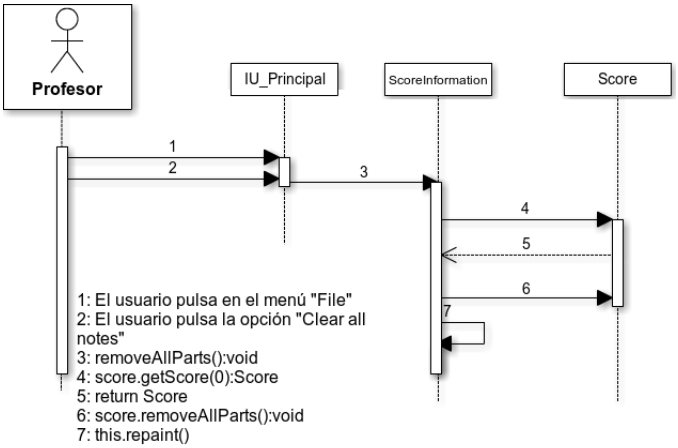


Figura 5.19: Borrar Pentagrama Completo

Añadir elemento

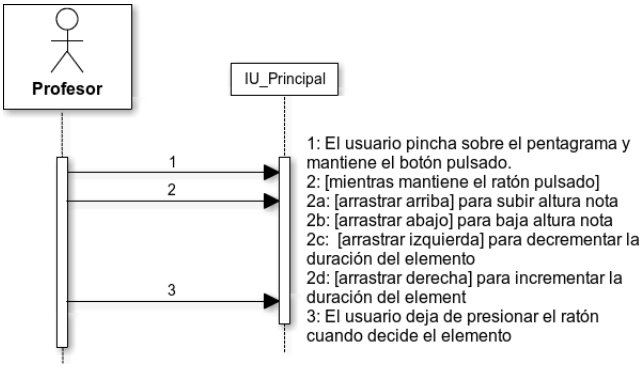


Figura 5.20: Añadir Elemento

Modificar altura nota

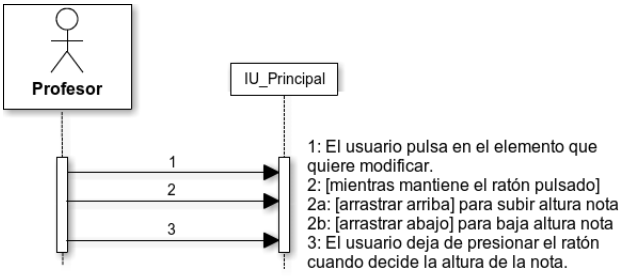


Figura 5.21: Modificar Altura Nota

Borrar elemento del pentagrama

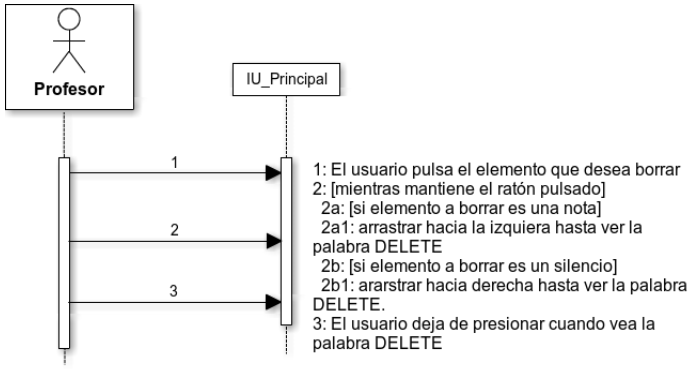


Figura 5.22: Borrar Elemento Del Pentagrama

5.3. Módulos

Los módulos son pequeñas aplicaciones en las que dividiremos cada una de las tareas que tenga programa principal. Entre ellos podremos hallar conversores entre distintos lenguajes musicales al igual que el guardado de partituras en formato MusicXML.

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.

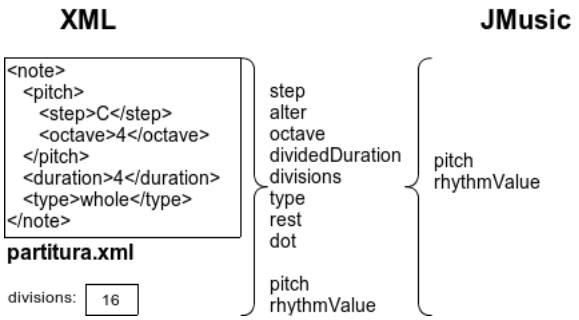


Figura 5.23: Conversor de XML a jMusic

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

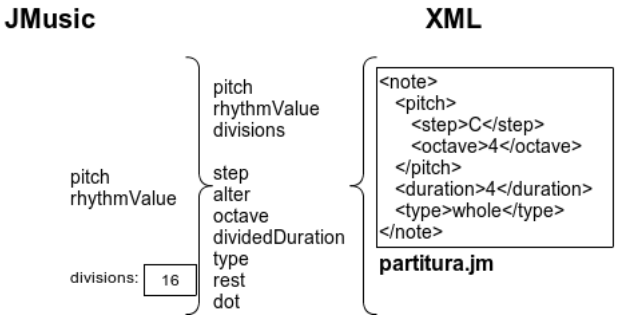


Figura 5.24: Conversor de jMusic a XML

Guardar partitura (Exportar MusicXML)

La aplicación maneja principalmente objetos de jMusic para mostrar el contenido de una partitura en pantalla. Para preservar el contenido en un archivo, la partitura se guardará en un archivo MusicXML que más tarde se podrá volver a abrir con la aplicación y otros programas que soporten dicho formato. Guardar partitura lo formarán los siguientes módulos:

expHeaderToXml

El módulo expHeaderToXml creará la cabecera del archivo MusicXML. Podemos ver un ejemplo en el cuadro 5.1. En él se anotarán:

- el título de la partitura (`<work-title>`, línea 5)
- el autor (`<creator type="composer">`, línea 8)
- el programa que lo guardó (`<software>`, línea 10)
- el nombre interno de la parte (`<score-part>`, línea 14)
- el nombre externo de la parte (`<part-name>`, línea 15)
- la parte en sí (`<part>`, línea 18)
- el comienzo 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 código de expHeaderToXml

expMeasureAttributesToXml

El módulo expMeasureAttributesToXml se encargará de guardar los atributos del compás. Podemos ver un ejemplo en el cuadro 5.2. Guardaremos:

- las divisiones de negra (`<divisions>`, línea 2)
- la armadura (`<fifths>`, línea 4)
- los pulsos (`<beats>`, línea 7)
- la subdivisión del compás (`<beat-type>`, línea 8)
- la figura de la clave (`<sign>`, línea 11)
- la línea de la clave (`<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 código de expMeasureAttributesToXml

expFiguresToXml

Con expFiguresToXml iremos creando cada una de las figuras (notas y silencios). Podemos ver un ejemplo en el cuadro 5.3. Almacenaremos:

- la altura (`<step>`, línea 3)
- la octava (`<octave>`, línea 4)
- la duración en *divisions* (`<duration>`, línea 6)
- la figura de la nota (`<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 código de expFiguresToXml

expEndingToXml

El módulo expEndingToXml irá cerrando todos las etiquetas xml al final del archivo 5.4.

```
1       </measure>
2    </part>
3 </score-partwise>
```

Tabla 5.4: Ejemplo de código de expFigures

Capítulo 6

IMPLEMENTACIÓN

En esta sección se explicarán las partes más significativas e interesantes del código del proyecto. Primero, se mostrará el código en lenguaje de programación Java y posteriormente se detallará para una mejor comprensión.

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 asignacion 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 las clases para importar la partitura en formato jMusic (ImpXmlToJm.java) y exportar la partitura en formato lilypond (ExpJmToLy.java) encontramos la siguiente:

ExpJmToXml.java

El módulo ExpJmToXml se encargará de escribir una partitura en formato jMusic y convertirla a la estructura que maneja MusicXML. Está formado por cuatro métodos principales: `expHeaderToXML()`, `expMeasureAttributesToXML()`, `expFiguresToXML()`y `expEndingToXML()` .

expHeaderToXML()

expHeaderToXML recorre la cabecera del archivo jMusic y recoge los elementos que necesite.

Su primera tarea es crear las etiquetas *<"utf-8", "1.0")> , <"!DOCTYPE score-partwise PUBLIC \"-//Recordare//DTD MusicXML 3.0 Partwise//EN\" \"http://www.musicxml.org/dtds/partwise.dtd\">, <"score-partwise"> y <"version", "3.0">*

De esta manera el archivo será un archivo musicXML válido. Después añadirá las etiquetas <work> y <work-title> donde se añadirá el nombre que ha elegido el usuario para la partitura. Lo mismo hará con las etiquetas *< "creator"> y <"type", "composer">* donde se añadirá el nombre del compositor que ha elegido el usuario.

expMeasureAttributesToXML()

expMeasureAttributesToXML creará los elementos de los atributos más importantes para la exportación de la partitura en formato .xml.

En primer lugar tendrá que crear la etiqueta <"part"> y darle un identificador <"id", partId>. De esta manera la partitura se dividirá en partes. Después se crearán las etiquetas <"measure"> y <"number", Integer.toString(measureNumber)> donde pondremos el tipo de compás con su medida. De aqui en adelante todas los elementos estarán dentro de la etiqueta <"attributes">. La primera etiqueta en crearse será <"divisions"> donde podremos saber cuantas divisiones tiene la partitura. Después se creará la etiqueta <"key"> donde contendra la etiqueta <fifths>. Más adelante se creará la etiqueta <"time"> que contendrá las etiquetas <"beats"> y <"beat-type">. Donde quedarán guardados los pulsos del compás. Para finalizar se creará la etiqueta <"clef"> que contendra las etiquetas <sign> y <line>. Aqui se guardarán las claves sol o fa con sus respectivas alturas mediante la etiqueta <line>.

expFiguresToXML()

expFiguresToXML recopilará los valores que formen una nota del pentagrama en formato jMusic y los irá añadiendo al archivo de musicXML.

Mediante el comando *Note[] noteArray = phrase.getNoteArray()*, guardamos en la lista *noteArray* todas las notas del pentagrama. Se irá recorriendo esa lista hasta que no queden más notas en el pentagrama. Para hacer el conversor de la nota del formato jMusic al formato musicXML se invocarán los métodos *convJmToXml(n) y calculateDivDur(n.getRhythmValue(), si.getDivisions())*. De esta manera se irán creando las notas en formato musicXML con sus respectivas duraciones.

expEndingToXML()

expEndingToXML() creará la fase final de una partitura en formato musicXML. Se cerrará la etiqueta *</score-partwise>* que ha sido la encargada de contener dentro todas las demás etiquetas mediante el comando *writer.writeEndElement()*. Para finalizar se hará uso del comando *writer.writeEndDocument()*, el cual dejará cerrado todo el archivo musicXML.

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. Guardar archivo jm (jMusic) a MusicXML (.xml)

En esta sección se va a hacer la prueba de exportación de una partitura creada. La partitura del programa tiene formato .jm y se quiere exportarla en formato .xml.

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
1	Exporta archivo en formato .xml	Se exporta el archivo en formato .xml	Se exporta el archivo en el formato .xml	Correcto
2	Archivo sin extensión .xml	El nombre del archivo incluye la extensión .xml	El nombre del archivo no incluye la extensión .xml	Si el usuario modifica la extensión o la elimina, el archivo mantiene ese nombre
2a			El nombre del archivo incluye la extensión .xml	Correcto. Se añade la extensión .xml 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 la escala bien	Las notas están desviadas varios semitonos	El cálculo de conversión de jMusic a MusicXML no es correcto
			Se exporta toda la escala con las alturas adecuadas	Correcto
5	Exportar todas las duraciones (corchea, negra, blanca y redonda)	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 7.5: guardar archivo jm to mxml 1

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
6	Exportar todos los silencios (de corchea, negra, blanca y redonda)	Se exporta con los silencios	No se exportan correctamente los silencios	Hacer que los silencios no exporten los tags de `<pitch>`
6a			Se exporta con los silencios	Correcto
7	Exportar una escala que incluya alteraciones de sostenido	Se exporta con las alteraciones correctas	Se exporta con las alteraciones correctas	Correcto
8	Exportar una escala que incluya alteraciones de bemol	Se exporta con las alteraciones correctas	Se exporta con las alteraciones correctas	Correcto
9	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
9a			No aparece la ligadura	No es suficiente con las etiquetas `<tie type=‘‘start’’>` y `<tie type=‘‘stop’’>`, también hay que añadir `<tied type=‘‘start’’>` y `<tied type=‘‘stop’’>` en `<notations>`
9b			Se exporta la nota con la ligadura	Correcto
10	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
10a			Se exportan las notas sin ligaduras extra	Correcto

Figura 7.1: guardar archivo jm to mxml 2

Código	*Descripción*	*Resultado esperado*	*Resultado obtenido*	*Observaciones*
11	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
11a			Las notas con puntillo se exportan bien	Correcto
12	Exportar todos los silencios con puntillo	Se exportan los silencios con sus puntillos correspondientes	Se exporta bien	Correcto
13	Exportar distintos compases (1/4, 2/4, 3/4 y 4/4)	Se exportan bien todos los compases	Se exportan bien todos los compases	Correcto
14	Exportar armadura de sostenidos	Se exportan todos los sostenidos correctamente	Se exportan todos los sostenidos correctamente	Correcto
15	Exportar armadura de bemoles	Se exportan todos los bemoles correctamente	Se exportan todos los bemoles correctamente	Correcto
16	Exportar clave de fa y de sol	Se exporta la clave elegida	Se exportan bien las claves	Correcto
17	Exportar partitura vacía	Se exporta un pentagrama vacío	Se exporta un pentagrama vacío	Correcto

Tabla 7.6: guardar archivo jm to xml 3

Capítulo 8

CONCLUSIONES

Una vez concluido el presente proyecto se pueden extraer algunas conclusiones.

En primer lugar, quiero mencionar que en un principio se había decidido hacer dos subproyectos, distintos pero relacionados entre sí, orientados a la creación de obras musicales. Pronto, mi compañero actual del proyecto y yo nos dimos cuenta de que estos proyectos se implementaban de manera importante, por lo cual se decidió unir esfuerzos hacía la creación de un único trabajo con mayor complejidad y entidad propia.

Pero es sabido que aunar voluntades conlleva dificultades añadidas a las ya propias del trabajo en sí mismo. La dificultad más importante con la que me he topado es la falta de fuentes de información, tanto en formato digital como en papel impreso, y además, las que he encontrado estaban en diferentes idiomas, ajenos a los propios, y necesitaban de traducciones veraces. Éste hecho ha obstaculizado el avance del trabajo en no pocas ocasiones y me ha obligado a invertir un gran número de horas que no habían sido estimadas *a priori*. Gran número de horas que se ha utilizado en buscar información acerca de cómo programar música con el lenguaje Java. Se decidió utilizar éste lenguaje Java porque considero que es el más completo y fácil de entender que existe en el mercado para un usuario no especializado.

Aparte de ser uno de los lenguajes con más facilidades de uso para programar en música, proporciona una amplia variedad de funcionalidades musicales que otros lenguajes carecen.

El resultado final es un trabajo que ha cumplido los objetivos principales propuestos y algunos de los secundarios.

El proyecto supuso un reto para mi, amante y conocedora tímida de la música. Un desafio del que he salido francamente satisfecha, tanto por el resultado final como por lo que he aprendido en el camino, que no es poco, por no mencionar lo que he disfrutado. Me ha supuesto un reto crear interfaces sencillas pero a la vez útiles para cualquier usuario novel, pero por el contrario, me ha divertido mucho jugar con estos elementos, ya sea con botones, colores, acciones entre otros que evidentemente son claves fundamentales para el éxito del programa KOSMOS.

En consecuencia, creo que el resultado supera las dificultades encontradas y me llevo un buen sabor de boca.

Sin embargo, no podemos obviar que existe una diversidad de programas musicales muy completos e intuitivos en el mercado pero también muy complejos de programar y de difícil acceso para el usuario aficionado que se sentirá más cómodo con programas como el mio.

Para finalizar, quiero incidir en la dificultad para conseguir algunos de los objetivos propuestos, como poder guardar o exportar las composiciones creadas, o el poder visualizar las notas en un pentagrama, debido a la falta de fuentes en las que apoyarnos. Por ello, se deja abierto el camino para que personas más cualificadas puedan, en un futuro, mejorar la aplicación con la introducción de diferentes funcionalidades de uso.

Apéndice A

jMusic

JMusic es una biblioteca de programación musical de código abierto (licencia GPLv2, más información en la sección G en la página 127) escrita en el lenguaje de programación Java. En caso de querer profundizar más acerca de esta herramienta véase el apartado [13] 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    }
```

A.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*.

A.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.

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.

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]

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.

A.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 A.1: Jerarquía jMusic

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 B.5 en la página 100.
- 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 B.3 en la página 99.
- 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);
```

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*.

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).

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).

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.

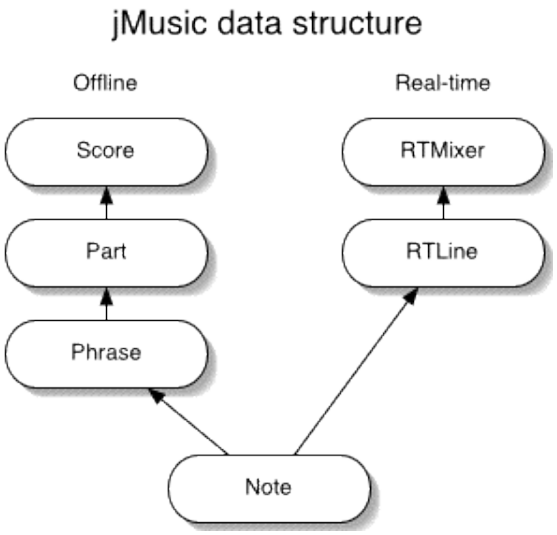


Figura A.2: jMusic data strucutre

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.

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 B

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 B.1: Código de ejemplo de MusicXML

El código superior (tabla B.1) es un ejemplo mínimo de archivo en formato MusicXML, disponible en la página oficial. [20] 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 B.1). El árbol de la tabla B.2 esquematiza estas partes de manera simple.

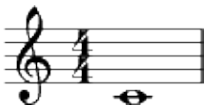


Figura B.1: Representación del ejemplo de MusicXML

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

Tabla B.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 B.3). Se usará en relación con `duration` (l. 31).

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 B.3: Notas, silencios, sus códigos y nombres

- `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 B.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 B.4: Tabla de armaduras

- `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 B.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 B.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 B.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 C

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*.

C.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.

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 con-

trario 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.

```
1  if (event.isStartElement())
2  {
3     StartElement se = event.asStartElement();
4     ...
5  }
6  else if (event.isEndElement())
7  {
8     EndElement ee = event.asEndElement();
9     ...
10 }
```

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

```
1  StartElement se = event.asStartElement();
2  localPart = se.getName().getLocalPart();
3
4  if (localPart.equals("score-part"))
5  {
6     Attribute partId = se.getAttributeByName(new QName("id"));
```

C.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).

```
1  FileOutputStream fos = new FileOutputStream(path);
2  XMLOutputFactory factory = XMLOutputFactory.newInstance();
3  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:

```
1 writer.writeStartElement("creator");
2 //<creator
3 writer.writeAttribute("type", "composer");
4 //<creator type=composer
5 writer.writeCharacters("Beethoven");
6 //<creator type=composer>Beethoven
7 writer.writeEndElement();
8 //<creator type=composer>Beethoven</creator>
```

Apéndice D

Manual de Usuario

D.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 D.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.

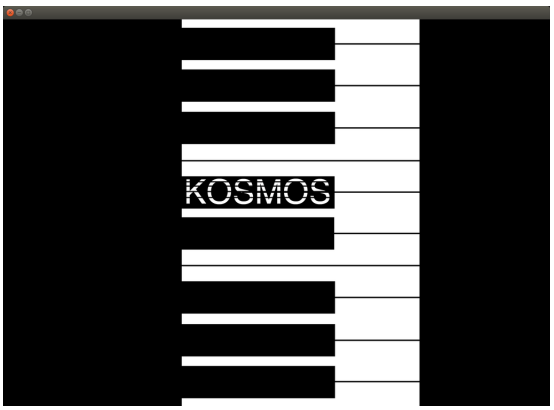


Figura D.1: Splash (Ventana al iniciar el programa)

D.2. Menu de opciones

En la siguiente ventana aparecen dos botones como se puede apreciar en la figura D.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.

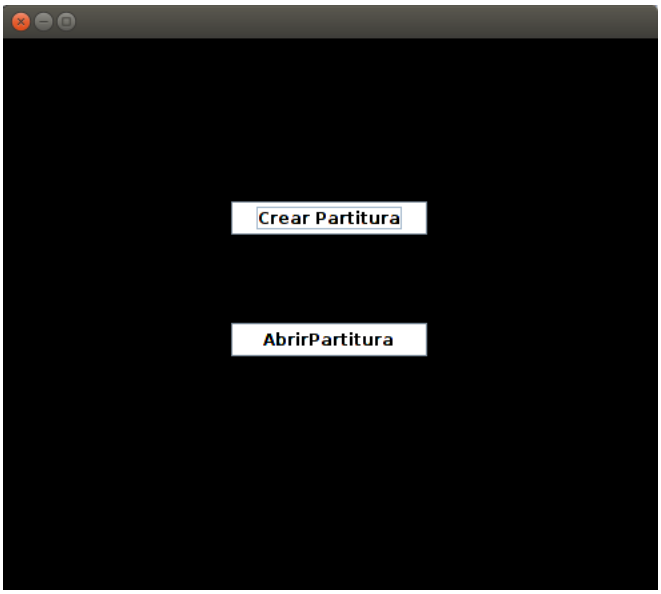


Figura D.2: Crear Partitura y Abrir Partitura

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 *D.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.

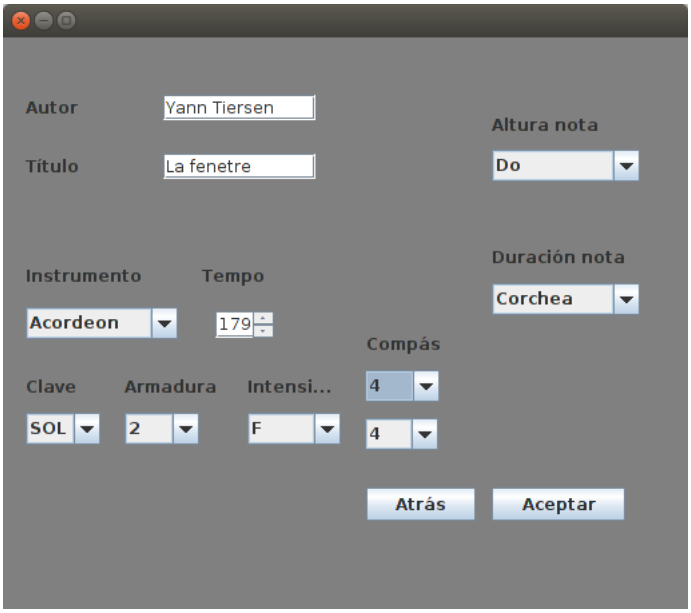


Figura D.3: Dentro de Crear Partitura

Los datos proporcionados por el usuario anteriormente se insertan en la partitura nueva como se puede apreciar en la figura D.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.

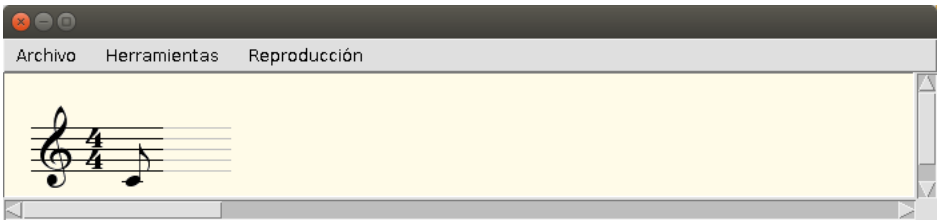


Figura D.4: Interfaz del pentagrama

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 D.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 D.2: Funcionalidades de menús Herramientas y Reproducción

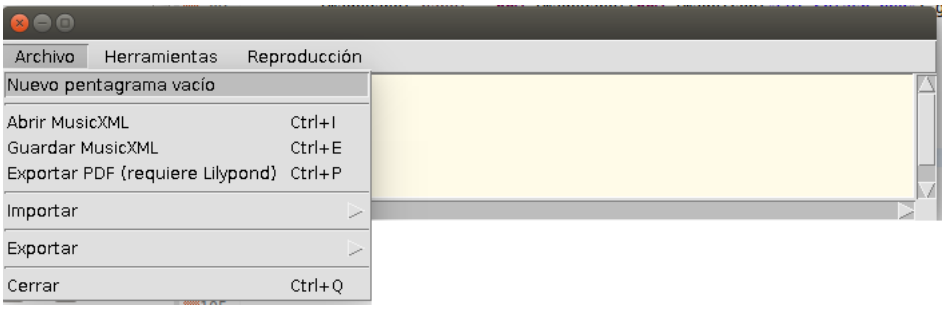


Figura D.5: Menú Archivo

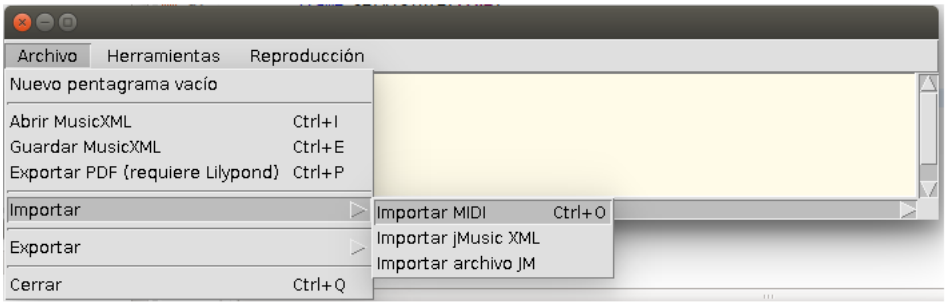


Figura D.6: Menú Archivo > Importar

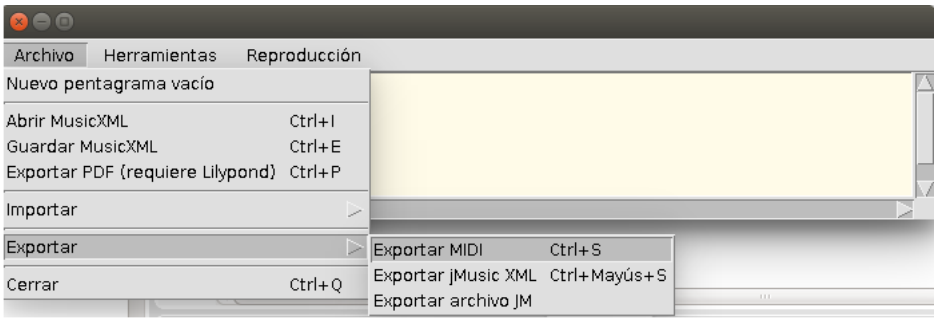


Figura D.7: Menú Archivo > Exportar

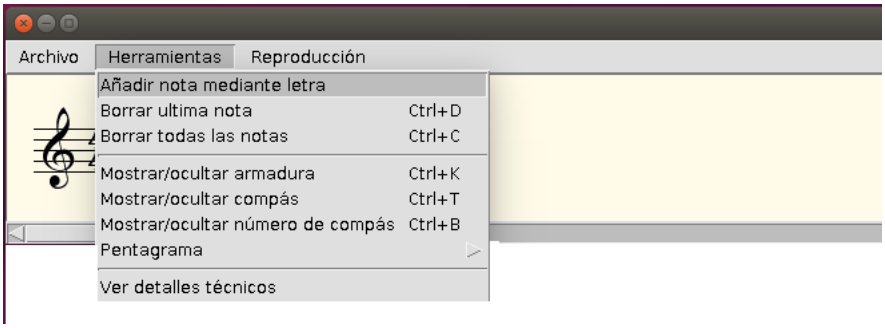


Figura D.8: Menú Herramientas

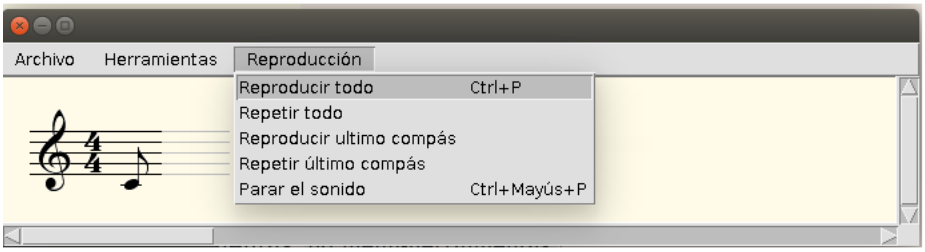


Figura D.9: Menú Reproducción

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.

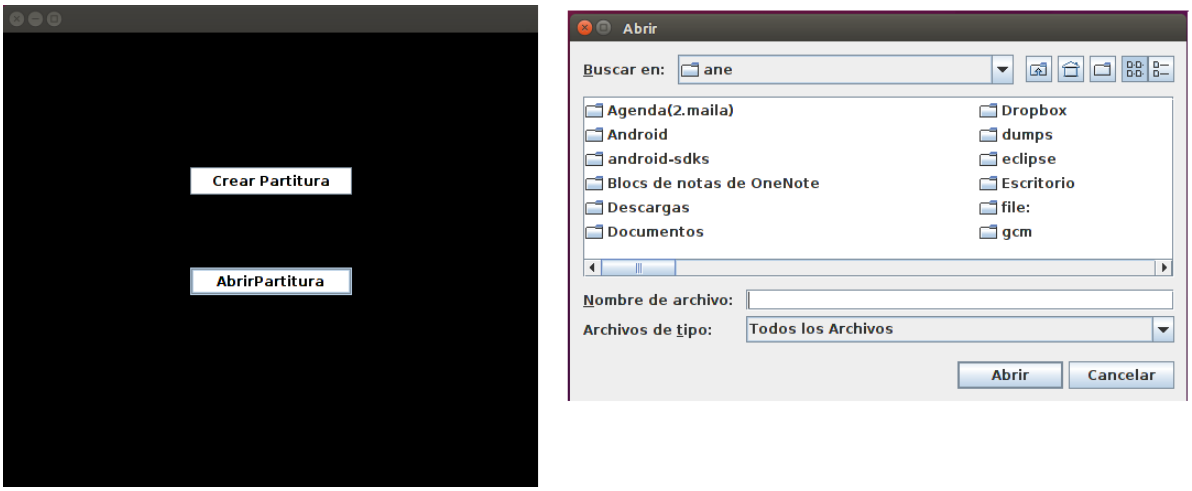


Figura D.10: Menú Abrir Partitura

Apéndice E

modules package (paquete modules)

Se explicarán ciertas partes de código de las clases que estan dentro de este paquete. Es la clase más difícil por su complejidad en el código.

E.1. *ExpJmToXml.java*

Esta clase es la que se encarga internamente de exportar la partitura creada (formato jMusic .jm) al formato .xml. De esta manera se consigue una partitura con un formato estándar para poder abrirla y modificarla en los diferentes programas musicales existentes. *Jmusic* es un lenguaje bastante nuevo y muchos programas carecen de la función de poder leer una partitura con este formato.

```
public static void main() {
        System.out.println(score);
         Part part = score.getPart(0);
         Phrase phrase = part.getPhrase(0);

         getPath();

        try {
            writer = new IndentingXMLStreamWriter(factory.createXMLStreamWriter(
                new FileOutputStream(path), "UTF-8"));

            expHeaderToXML();
            expMeasureAttributesToXML();
            expFiguresToXML(phrase);
            expEndingToXML();

            writer.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (XMLStreamException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

El método *main*() es el principal y se ejecutará nada más llamar a esta clase.

Se hace una llamada al método getPath() que se explicará más adelante. Lo importante de esta clase es su simplicidad dado que se han usado muchos subprogramas para una mejor legibilidad del código.

Se puede observar que se hace llamada a 4 métodos dentro del try{. Se explicarán más adelante, aqui solamente se les invoca.

```
private static void getPath() {
    // parent component of the dialog
            JFrame parentFrame = new JFrame();

            JFileChooser fileChooser = new JFileChooser();
            fileChooser.setDialogTitle("Specify a file to save");

            int userSelection = fileChooser.showSaveDialog(parentFrame);

            if (userSelection == JFileChooser.APPROVE_OPTION) {
                if(!fileChooser.getSelectedFile().getAbsolutePath().endsWith
                    (".xml"))
                {
                    path = new File(fileChooser.getSelectedFile() + ".xml");
                }
                else
                {
                    path = fileChooser.getSelectedFile();
                }
            }

}
```

Este método es el primero de todos pues el usuario deberá elegir donde desea exportar su partitura. Hemos utilizado el componente *JFileChooser*() para llevar a cabo esta tarea. Si el usuario al escribir el nombre del archivo no le ha puesto formato alguno (*.xml*) se guardará con la terminación *.xml*. En caso de ponerle formato se guardará con el formato que le ha puesto(*.xml*). Esto es, si el usuario guarda con el nombre *partitura* sin ningun formato se guardará como *partitura.xml* y si el usuario guarda como *partitura.xml* se guardará tal cual.

```
public static void expHeaderToXML() throws XMLStreamException{
    try {
        //<?xml version="1.0" encoding="utf-8"?>
        writer.writeStartDocument("utf-8", "1.0");


        //TODO add <!DOCTYPE
        writer.writeDTD("<!DOCTYPE score-partwise PUBLIC \"-//Recordare//DTD
             MusicXML 3.0 Partwise//EN\" \"http://www.musicxml.org/dtds/
            partwise.dtd\">");
        writer.writeCharacters("\n");

            //<score-partwise version="3.0">
            writer.writeStartElement("score-partwise");
            writer.writeAttribute("version", "3.0");

                //<work>
                writer.writeStartElement("work");

                    //<work-title>
                    writer.writeStartElement("work-title");
                    writer.writeCharacters(score.getTitle());

                    //</work-title>
                    writer.writeEndElement();
```

```
                    //</work>
                    writer.writeEndElement();

                    //<identification>
                    writer.writeStartElement("identification");

                        //<creator>
                        writer.writeStartElement("creator");
                        writer.writeAttribute("type", "composer");
                        writer.writeCharacters(si.getComposer());

                        //</creator>

                        writer.writeEndElement();

                        //<encoding>
                        writer.writeStartElement("encoding");
                        //<software>
                        writer.writeStartElement("software");
                        writer.writeCharacters("COSMOS");
                        //</software>
                        writer.writeEndElement();

                        //</encoding>
                        writer.writeEndElement();

                    //<</identification>
                    writer.writeEndElement();


                    //<part-list>
                    writer.writeStartElement("part-list");

                        //<score-part id="P1">
                        writer.writeStartElement("score-part");
                        writer.writeAttribute("id", partId);

                        //Part-name
                        writer.writeStartElement("part-name");
                        writer.writeCharacters("Music");
                        //</part-name>
                        writer.writeEndElement();


                        //</score-part>
                        writer.writeEndElement();

                    //</part-list>
                    writer.writeEndElement();

            writer.flush();

        } catch (XMLStreamException e) {
            e.printStackTrace();
        }

    }
```

Se ha pensado en dividir la partitura en 4 secciones y la primera de todas es la cabecera Header, la que insertará elementos vitales para convertirla en una partitura xml real. Se

va a hacer uso del componente writer a lo largo de esta clase porque es el encargado es escribir en formato xml.

Primero se escribe el principio del documento, que esta en utf-8 con una versión 1.0.

Después escribiremos el DTD, para hacer referencia a que es una partitura en formato xml. El comando *writer.writeCharacter("\n")* hará un salto de línea.

La primera parte de la partitura se encuentra al empezar con "score-partwise". Para empezar a escribir un elemento se usará el comando:

writer.writeStartElement() el cual creará una etiqueta principal. Si queremos que dentro de esa etiqueta tengamos más etiquetas se hará uso del comando: writer.writeAttribute() el cual se encargará de crear etiquetas con su valor. Por ejemplo:

- writer.writeStartElement("score-partwise"); –> creará la etiqueta <score-partwise>.
- writer.writeAttribute("version", "3.0"); –> creará <version, 3.0> dentro de la etiqueta score-partwise.

Si se quiere terminar con la etiqueta se hará uso del comando:

- writer.writeEndElement(); –> el cual creará por ejemplo </score-partwise> cerrando esa etiqueta.

Para acabar con el writer se hará uso del comando:

- writer.flush(); —> el cual cerrará todo.

Este método se encarga de crear el segundo nivel de la partitura, los atributos (compás, clave, divisiones, pulsos...) .

Siguiendo con el componente *writer* y habiendolo visto anteriormente es exactamente lo mismo.

- writer.writeStartElement("part"); –> Se crea la etiqueta <part>
- writer.writeAttribute("id", partId); –> Dentro de la etiqueta <part> se crea otra etiqueta con un valor <id,partId>
- writer.writeCharacters(«Music»); –> Si se quiere añadir un string solamente.
- writer.writeEndElement(); –> Para cerrar la etiquetav</part>

```
public static void expEndingToXML(){

        try {
                //</score-partwise>
                writer.writeEndElement();

            //
            writer.writeEndDocument();

            writer.flush();
            writer.close();
        } catch (XMLStreamException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }

    }
```

El último nivel de la partitura en formato xml. Siguiendo con el componente writer. Vemos que para acabar la primera etiqueta se usa *writer.EndElement()*. Para finalizar hacemos *writer.flush()* para actualizarlo y *writer.close()* para cerrarlo y acabar.

```
public static void printFigureTags(FigureJm jm) throws XMLStreamException {

        //<note>
        writer.writeStartElement("note");

            if (jm.isRest()==true){
            //</rest>
            writer.writeEmptyElement("rest");
            }

            //<pitch>
            writer.writeStartElement("pitch");

                //<step>
                writer.writeStartElement("step");
                writer.writeCharacters(jm.getStep());
                //</step>
                writer.writeEndElement();

                if (!jm.getAlter().equals("0")) {
                //<alter>
                writer.writeStartElement("alter");
                writer.writeCharacters(jm.getAlter());
                //</alter>
                writer.writeEndElement();
                }

                //<octave>
                writer.writeStartElement("octave");
                writer.writeCharacters(jm.getOctave());
                //</octave>
                writer.writeEndElement();

            //</pitch>
            writer.writeEndElement();

        //<duration>
        writer.writeStartElement("duration");

            writer.writeCharacters(Integer.toString(jm.getDividedDuration()));

        //</duration>
        writer.writeEndElement();

        //</tieStop>
        if (jm.isTieStop()){
            // tie
            writer.writeEmptyElement("tie");
            writer.writeAttribute("type","stop");
        }

        //</tieStart>
```

```
        if (jm.isTieStart()){
            writer.writeEmptyElement("tie");
            writer.writeAttribute("type","start");
        }

        //<type>
        writer.writeStartElement("type");
        writer.writeCharacters(jm.getType());
        //</type>
        writer.writeEndElement();

        //</dot>
        if (jm.isDot()==true){
            writer.writeEmptyElement("dot");
        }

        // tied notations
        if (jm.isTieStart() || jm.isTieStop())
        {
            // <notations>
            writer.writeStartElement("notations");

            if (jm.isTieStop())
            {
                //<tied type="stop"/>
                writer.writeEmptyElement("tied");
                writer.writeAttribute("type","stop");
            }
            if (jm.isTieStart())
            {
                //<tied type="start"/>
                writer.writeEmptyElement("tied");
                writer.writeAttribute("type","start");
            }
            // </notations>
            writer.writeEndElement();
        }

    //</note>
    writer.writeEndElement();
```

Es el método encargado de imprirmir o visualizar las figuras (notas y silencios). Se le pas como parámetro una de las clases (el que contenia elementos vitales de una partitura) FigureJM para obtener el elemento. El writer estará presente para escribir la etiqueta de la figura.

Todas las figuras iran dentro de la etiqueta principal <note>.

Si la nota es un silencio escribiremos *writer.writeEmptyElement(rest)* para escribirlo en la partitura.

Se jugará con la clase que se ha pasado como parametro (FigureJm jm) para ir obteniendo en cada etiqueta lo necesitado. Por ejemplo:

- *writer.writeStartElement("step"); –> Se crea la etiqueta <step>.*
- *writer.writeCharacters(jm.getStep()); –> se obtiene el step del jm y es escribe.*
- *writer.writeEndElement(); –> Se cierra la etiqueta </step>.*

Se seguirá el mismo patrón de código para todas las figuras.

```
public static String noteDuration(Double rhythmValue){

        String duration = null;
        if (rhythmValue==0.5)
        {
            duration= "eighth";
        }
        else if (rhythmValue==1.0)
        {
            duration= "quarter";
        }
        else if (rhythmValue==2.0)
        {
            duration= "half";
        }
        else if (rhythmValue==4.0)
        {
            duration= "whole";
        }
        return duration;


    }
```

Por último este pequeño método se encarga de las duraciones de las notas. Se le pasa como parametro un rhythmValue de tipo double(con decimales). Si el rhythmValue es igual a 0.5 entonces equivale a una corchea, si es 1.0 a negra, si es 2.0 a blanca y por último si es 4.0 a redonda. Asi sabremos en todo momento el valor de la nota.

Apéndice F

GNU General Public License (GPL) v2.0

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 G

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

G.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.

[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.

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.

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.

G.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.

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.

[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

[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(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.

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.

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.

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.

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.

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.

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.

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.

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] Aitor Valle Allende. Xmlscore: Representación gráfica y reproducción de partituras en formato xml. Trabajo fin de grado, Escuela Universitaria de Ingeniería Técnica Industrial de la Universidad del País Vasco, 2016.

[2] creandoPartituras. Definiciones - Glosario. http://www.creandopartituras.com/.

[3] danielMartin. Definiciones - Glosario. http://danielmartin-mallets.com/blog-percusion/es/.

[4] definicionABC. Definiciones - Glosario. http://www.definicionabc.com/tecnologia.

[5] definicionABC. Definiciones - Glosario. http://www.definicionabc.com/tecnologia.

[6] Definiciones.de. Definiciones - Glosario. http://definicion.de/.

[7] despertarMusical. Definiciones - Glosario. http://despertarmusical.blogspot.com.es/.

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

[9] fakiro. Definiciones - Glosario. http://musica.fakiro.com/.

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

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

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

[13] jMusic. jMusic tutorials and lessons. http://explodingart.com/jmusic/jmtutorial/t1.html.

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

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

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

[17] LyX. How to install correctly lyx on ubuntu. http://wiki.lyx.org/LyX/LyXOnUbuntu#toc1.

[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] 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.

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

[22] roble.pntic. Definiciones - Glosario. http://roble.pntic.mec.es/jprp0006/tecnologia/1eso_recursos/unidad02_componentes_ordenador/teoria/teoria1.htm.

[23] tocarPiano. Definiciones - Glosario. http://tocarpiano.about.com/od/musicaltermsa1/g/.

[24] wikiBros. Definiciones - Glosario. https://es.wikibooks.org/wiki/Portada.

[25] wikipedia. Definiciones - Glosario. https://es.wikipedia.org/wiki/Wikipedia:Portada.

[26] xml. jmusic: documentation, downloads, programs... http://explodingart.com/jmusic/.