ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS

DESARROLLO DE UN SOFTWARE PARA MANEJO DE ARCHIVOS MIDI Y CONTROL USB PARA ANDROID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

ANÍES SÁNCHEZ SEVERO EDUARDO
eduardoanies@gmail.com

PROAÑO FRANCO GIOVANNI DAVID
davisxdpfr@gmail.com

DIRECTOR: Ing. Carlos Montenegro
carlos.montenegro@epn.edu.ec

Quito, Agosto 2015

DECLARACIÓN

Nosotros, Aníes Sánchez Severo Eduardo y Proaño Franco Giovanni David, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que se consultó las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____ _____

Aníes Sánchez Severo Eduardo Proaño Franco Giovanni David

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Aníes Sánchez Severo Eduardo y Proaño Franco Giovanni David, bajo mi supervisión.

_____

Ing. Carlos Montenegro

DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco a mi tutor Carlos Montenegro quien nos ayudó a seleccionar este tema de tesis de entre otros y por guiarnos en el desarrollo de esta tesis, a mi profesor de aplicaciones móviles y java Santiago Mosquera por ayudarme a profundizar en mis conocimientos en estas áreas ahora aplicadas en el presente proyecto, a mi familia y amigos por el apoyo que he recibido para sacar adelante mis estudios.

Aníes Severo Eduardo

AGRADECIMIENTO

Agradezco al ingeniero Carlos Montenegro por habernos guiado de buena manera en la realización de este proyecto de titulación, a todos los profesores que de una u otra manera aportaron con sus conocimientos durante todo el transcurso de la carrera para poder aplicar estos conocimientos adquiridos en el desarrollo de este documento. Agradezco a mi familia y amigos por todo su apoyo. Finalmente, quiero agradecer a Eduardo Aníes por su apoyo y constancia para poder realizar este proyecto de la forma adecuada.

Proaño Giovanni David

DEDICATORIA

Dedico esta tesis a todos mis amigos músicos por aportar a descubrir esta idea y por ayudar a probarla, a mis profesores de universidad que aportaron mucho a mis conocimientos técnicos y por enseñarme que mediante la constancia y dedicación hasta las metas que no parecen inicialmente alcanzables en realidad si lo son, y finalmente agradezco a mi familia por el apoyo que me han dado durante estos años.

Aníes Severo Eduardo

DEDICATORIA

Dedico esta tesis a los músicos en general, a las personas que el desarrollo de este software les pueda beneficiar. A las personas que con constancia cumplen sus metas y de esta manera hacen cosas novedosas y únicas.

Proaño Giovanni David

CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

INTRODUCCIÓN

En este proyecto se desarrolla una aplicación para el sistema operativo Android, que permite cargar archivos MIDI y procesar la información contenida en estos, para realizar diversas actividades como la reproducción vía USB, la visualización de líricas, la visualización de las notas entrantes y salientes en un teclado virtual y finalmente la captura de eventos MIDI de entrada para la generación de un fichero MIDI.

El capítulo 1 del documento trata sobre la conceptualización del procesamiento de archivos MIDI, lo cual engloba el estudio detallado del protocolo MIDI, de la estructura de los paquetes, de la estructura de los ficheros MIDI y la metodología de desarrollo usada en este proyecto. Se usó Extreme Programming, una metodología para desarrollo de proyectos de software en equipos pequeños, esta incentiva las prácticas como la programación por pares, la refactorización de código para garantizar un producto adecuado, la simplicidad al desarrollar las soluciones a implementarse, entre otros valores.

El capítulo 2 trata sobre la planificación, la asignación de tareas y tiempos de desarrollo para esta aplicación. Se realizó un estudio previo a los usuarios finales para reconocer las necesidades y el problema a resolver con el desarrollo de esta aplicación. Se identificaron las historias de usuario, las tareas a realizarse y según estas, cada fecha en las que se realizarían. También se observó en cuanto al desarrollo los estándares de código, como se reflejan en el desarrollo las prácticas de Extreme Programming usadas y los diagramas de clases de la aplicación.

En el capítulo 3 se detallan las pruebas realizadas, desde las pruebas unitarias realizadas con Junit, las pruebas de integración con diferentes tipos de dispositivos MIDI y ficheros MIDI de diferentes tipos. Finalmente se documentan las pruebas de aceptación donde se verifica que lo desarrollado en la aplicación cumple con lo estipulado en las historias de usuario.

El capítulo 4 consta de las conclusiones y recomendaciones del proyecto.

RESUMEN

El protocolo MIDI es un estándar para sintetizadores, secuenciadores y otros dispositivos musicales. Paralelamente, existe un estándar de ficheros MIDI que es usado por la mayoría de los programas musicales para el intercambio, almacenamiento y procesamiento de información musical.

A partir del estudio analítico de MIDI, en este trabajo se crea una aplicación para el sistema operativo Android, que permite cargar estos archivos y procesar la información para realizar diversas actividades como la visualización de líricas, la visualización de las notas entrantes y salientes en un teclado virtual y, finalmente, la captura de eventos MIDI de entrada para la generación de un fichero MIDI y su reproducción vía USB. Las dos últimas características constituyen el aporte innovador de este proyecto.

La aplicación fue desarrollada usando la metodología ágil XP, sobre la base de un estudio previo acerca de los requerimientos de los usuarios.

1 CONCEPTUALIZACIÓN SOBRE PROCESAMIENTO DE ARCHIVOS MIDI

1.1 PROCESAMIENTO DE ARCHIVOS MIDI

1.1.1 PROTOCOLO MIDI

MIDI es una abreviación de las siglas “Musical Instrument Digital Interface”, se trata de un protocolo de comunicación serial que permite a los instrumentos musicales intercambiar información entre ellos y también con computadores.

Su invención se remonta al año de 1983 en el que un pequeño grupo de técnicos diseñadores de sintetizadores de diferentes fabricantes se reunieron para establecer un protocolo que permita a un sintetizador de un fabricante controlar mediante un cable un sintetizador de otro fabricante; la primera demostración realizada fue en enero de 1983 en el NAMM Show, donde se conectó un sintetizador “Sequential Prophet-600” y “Roland's JP6” de manera exitosa [1].

La información que es transmitida por medio del protocolo MIDI no contiene información de audio específicamente, sino más bien una secuencia de instrucciones que indican a un sintetizador acciones a ser ejecutadas, por ejemplo encender nota, apagar nota, cambiar volumen, por lo que el sonido que se genera varía según el dispositivo MIDI que ejecute las instrucciones recibidas[5].

Formato de los datos de los mensajes MIDI

En el protocolo MIDI estándar un dispositivo es el transmisor y otro es el receptor; los mensajes transmitidos contienen los siguientes bytes:

- **Byte de estado:** un mensaje MIDI puede contener un solo byte de estado, el cual debe contener como primer bit siempre un 1, por lo que el valor de este byte al imprimirlo en lenguajes de programación como java será siempre negativo.

- **Formato Byte de estado:** 1nnnnnnn, donde n puede ser 0 o 1.

- **Rango:** -127: -1

- **Bytes de datos:** un mensaje MIDI puede contener varios bytes de datos, y el primer bit de un byte de datos deberá ser siempre 0, por lo que el valor de este byte al imprimirlo en lenguajes de programación como java será siempre positivo:

    - **Formato Byte de estado:** 0nnnnnnn, donde n puede ser 0 o 1
    - **Rango:** 0: 127

Mensajes de canal

Existen 16 canales lógicos para generar sonidos según define el protocolo MIDI, lo que quiere decir que un sintetizador que recepta instrucciones MIDI podrá generar como máximo 16 distintos tipos de sonidos simultáneamente. Los llamados mensajes de voz, canal o de modo son los que permiten enviar datos a través de algún canal lógico, y cumplen con la siguiente estructura [6]:

- **Byte de estado**

  Formato: mmmmcccc

El byte de estado en los mensajes de canal se subdivide en 2 grupos de 4 bits, el primer grupo de 4 bits indica el evento MIDI a ser ejecutado por el sintetizador, y los otros 4 bits indican el canal lógico por el cual debe ser ejecutado el evento MIDI.

- **Bytes de datos**

  El número de bytes de datos en los mensajes de canal es de máximo 2 bytes y mínimo 1 byte, y como se describió anteriormente el primer bit de todo byte de datos es siempre 0 por lo que los bytes de datos van de 0: 127.

Tabla de mensajes de canal

A continuación se muestran los mensajes de voz existentes dentro del protocolo MIDI:

Byte de estado	Bytes de parámetros
1000nnnn Note Off (apagado de nota)	1. 0nnnnnnn Nota musical representada como un numero de 0-127, donde 60 es el Do central 2. 0vvvvvvv Velocidad (Volumen asociado a la nota)
1001nnnn Note On (encendido de nota)	1. 0nnnnnnn Nota musical representada como un numero de 0-127 2. 0vvvvvvv Velocidad (Volumen asociado a la nota)
1010nnnn Polyphonic Key Pressure (Cambio de presión en la tecla usada para aumentar o reducir el volumen del sonido)	1. 0nnnnnnn Nota musical representada como un número de 0-127. 2. 0vvvvvvv Velocidad (Volumen asociado a la nota).
1011nnnn Control Change (Cambio de control, puede ser cambio de volumen, pan, banco de sonido, etc.)	1. 0ccccccc Número de control el cual puede ser pedal, volumen, selección de banco, etc. 2. 0vvvvvvv Valor del controlador.
1100nnnn Program Change, (Cambio de instrumento o programa)	1. 0ppppppp Número de programa o instrumento nuevo, 127 posibles valores.
1101nnnn Channel Pressure (Cambio de presión en todas las teclas de un canal para cambiar el volumen por ejemplo)	1. 0ppppppp Nuevo valor de presión en las teclas.
1110nnnn Pitch Bend (Cambio de tono, usado para subir o bajar tonos en porcentajes pequeños como cuarto de tono)	1. 0LLLLLLL 2. 0mmmmmmm Tono en un número de 14 bits que es generado a partir de los 7 bits de cada byte.

Tabla.1.1Mensajes de canal, MIDI Manufacters Association [3]

Mensaje de canal Cambio de control

El mensaje de canal “Cambio de control” permite cambiar el valor de diversos parámetros de un canal MIDI específico, como por ejemplo el volumen, el banco de sonido a ser usado, mensaje para apagar todas las notas que están sonando en un canal MIDI específico, efecto pedal, efecto panning, entre otros[2].

A continuación se muestra la estructura de un mensaje de cambio de control

- **Mensaje MSB (Most significant byte) Byte más significativo**

Byte (binario)	Byte (decimal)	Interpretación
1011nnnn	176 + canal	Los primeros 4 bits indican el evento cambio de control, los siguientes 4 bits el canal por el cual vamos a transmitir el mensaje
0ccccccc	0-127	Identificador de controlador (MSB)
0vvvvvvv	0-127	Valor del controlador (MSB)

Tabla.1.2 Mensaje de cambio de control MSB [2]

- **Mensaje LSB (Less significant byte) Byte menos significativo**

Byte (binario)	Byte (decimal)	Interpretación
1011nnnn	176 + canal	Los primeros 4 bits indican el evento cambio de control, los siguientes 4 bits el canal por el cual vamos a transmitir el mensaje
0ccccccc	0-127	Identificador de controlador (LSB)
0vvvvvvv	0-127	Valor del controlador (LSB)

Tabla 1.3 Mensaje de cambio de control LSB [2]

Un mensaje de cambio de control podría estar determinado por 2 eventos o un solo evento MIDI, en el caso de que el mensaje contenga 2 eventos el primero contendrá los 7 bits más significativos del valor del controlador, y el segundo evento los 7 bits menos significativos del valor del controlador formando de esta manera un valor de 14 bits, en el primer evento deberá ir el identificador del controlador MSB y en el segundo el identificador del controlador LSB.

Es necesario recalcar que ambos eventos MIDI deben ir juntos, primero el evento con el byte más significativo MSB, y luego el evento con el byte menos significativo LSB para que tenga efecto el mensaje; si solo va el evento con el byte más significativo se toma en cuenta este valor como los únicos 7 bits correspondientes al valor del controlador [4].

A continuación se mostrará algunos eventos de cambio de control comunes:

- **Selección de banco:**

  Como se ha visto anteriormente existen 127 sonidos o programas que se pueden asignar a un canal MIDI específico, sin embargo existen sintetizadores que incluyen muchas veces más de 127 programas, para lo cual se pueden usar los bancos de sonido, cada banco de sonido puede contener 127 programas MIDI, por lo que en total se podría decir que existen:

  127 * 127 = 16129 posible programas MIDI, lo cual es un rango bastante amplio.

	Byte (binario)	**Byte (decimal)**
Identificador MSB	00000000	0
Identificador LSB	00100000	32

Tabla.1.4 Bytes de identificación MSB y LSB de evento selección de banco [2]

- **Cambio de volumen**

  Como su nombre lo indica cambia el valor del volumen de salida de un canal específico

	Byte (binario)	**Byte (decimal)**
Identificador MSB	00000111	7
Identificador LSB	00100111	39

Tabla 1.5 Bytes de identificación MSB y LSB de evento cambio de volumen [2]

- **Panning**

  El efecto panning en audio regula el volumen de salida de 2 canales, aumentando en uno y reduciendo en otro. A continuación se mostrará los bytes que lo identifican:

	Byte (binario)	**Byte (decimal)**
Identificador MSB	00001010	10
Identificador LSB	00101010	42

Tabla 1.6 Bytes de identificación MSB y LSB de evento cambio de volumen [2]

Mensajes exclusivos de sistema

Este tipo de mensajes permite a los fabricantes de dispositivos MIDI crear sus propios mensajes; lo que primero contienen este tipo de mensajes es el ID del fabricante, un ID único asignado a cada fabricante, este ID puede ocupar 1 o 3 bytes, por ejemplo el ID de Yamaha es 43, y posteriormente viene el mensaje propiamente dicho definido por dicho fabricante. También existen mensajes exclusivos genéricos que no referencian a ningún fabricante particular. Este tipo de mensajes contiene un byte de estado para identificar su inicio el cual es constante y siempre tiene un valor de 11110000 y un byte para indicar su finalización el cual también es un valor constante igual a 11110111.

A continuación se muestra un diagrama con la estructura especificada:

Byte (binario)	**Byte (decimal)**	**Descripción**
11110000	240	Byte de estado que indica el inicio de un mensaje exclusivo de sistema
Variable	0-127	Byte para indicar el ID del fabricante, la longitud puede ser 1 o 3 bytes
	0-127	
	0-127	
Variable(Datos)	0-127	Bytes para representar los datos, pueden tener cualquier longitud, la cual depende del momento en el que llegue el byte de finalización
	0-127	
	0-127	
	……	
	0-127	
11110111	247	Byte para indicar la finalización de un mensaje exclusivo de fabricante

Tabla.1.7 Contenido de un mensaje exclusivo de sistema [4]

Mensajes de tiempo real

Este tipo de eventos tienen este nombre porque pueden ser enviados en cualquier momento incluso entre bytes de datos de un mensaje; se asume que el receptor podrá ignorar este tipo de bytes y recolectar los datos de un mensaje exitosamente, descartando y procesando este tipo de eventos.

1.1.2 ESTRUCTURA DE FICHEROS MIDI (STANDARD MIDI FILES SMF)

Los archivos MIDI hoy en día encontrados en cientos de sitios web algunos disponibles gratuitamente, otros con algún costo siguen la especificación de archivos MIDI Versión 1.1 publicada en el año de 1990 por la Asociación Internacional del MIDI. La extensión de este tipo de archivos puede ser .mid o .midi, o en ocasiones .kar para indicar que el fichero contiene letras de canciones. Los archivos MIDI están formados por una cabecera estándar y contienen varias pistas donde se puede encontrar información sobre el tempo, nombres de pista, encendido y apagado de nota, volumen, entre otros. A continuación se muestra la estructura general de los bytes contenidos dentro de los ficheros MIDI:

Interpretación	Byte (Decimal)	Byte (Binario)	Dato
Cabecera de archivo, (codificación ASCII, un byte por carácter)	77	01001101	M
	84	01010100	T
	104	01101000	h
	100	01100100	D
Tamaño de cabecera de archivo contando desde el siguiente byte	0	00000000	Variable de tipo Int de valor constante = 6
	0	00000000	
	0	00000000	
	6	00000110	
Tipo de archivo MIDI, 0 o 1 o 2, (posteriormente se explica con más detalle cada tipo)	0	00000000	Variable de tipo short constante = 0 o 1 o 2
	0,1,2	00000000, 00000001, 00000010	
Número de pistas del archivo	0 - 127	variable	Variable de tipo short positiva
	0 - 255	variable	
División de tiempo, (explicada posteriormente)	0 - 255	variable	Variable de tipo short
	0 - 255	variable	
Pista 1			
Pista 2			
.....			
Pista n			

Tabla.1.8 Contenido de un mensaje exclusivo de sistema [4]

Formato de pista MIDI

<table>
<tr><th>Interpretación</th><th>Byte (Decimal)</th><th>Byte (Binario)</th><th>Dato</th></tr>
<tr><td rowspan="4">Cabecera de pista, (codificación ASCII, un byte por carácter)</td><td>77</td><td>01001101</td><td>M</td></tr>
<tr><td>84</td><td>01010100</td><td>T</td></tr>
<tr><td>114</td><td>01110010</td><td>r</td></tr>
<tr><td>107</td><td>01101011</td><td>k</td></tr>
<tr><td rowspan="4">Tamaño de pista a partir del siguiente byte</td><td>0 - 127</td><td>Variable</td><td rowspan="4">Variable de tipo Int (4 bytes)</td></tr>
<tr><td>0 - 255</td><td>Variable</td></tr>
<tr><td>0 - 255</td><td>Variable</td></tr>
<tr><td>0 - 255</td><td>Variable</td></tr>
<tr><td colspan="4">Intervalo de tiempo (Valor de longitud variable)</td></tr>
<tr><td colspan="4">Evento1</td></tr>
<tr><td colspan="4">Intervalo de tiempo (Valor de longitud variable)</td></tr>
<tr><td colspan="4">Evento2</td></tr>
<tr><td colspan="4">Intervalo de tiempo (Valor de longitud variable)</td></tr>
<tr><td colspan="4">Evento n</td></tr>
</table>

Tabla.1.9 Estructura de pistas MIDI [2]

Codificación del tiempo

La codificación del tiempo en los archivos MIDI se lo realiza a través de intervalos de tiempo, estos intervalos de tiempo están siempre antes de cualquier evento y para ser transformados a segundos o milisegundos se requiere primeramente leer los 2 últimos bytes de la cabecera del fichero MIDI que indican las divisiones de tiempo. Se pueden presentar dos casos:

- El primer bit del primer byte de la codificación del tiempo inicia con 1:
  Si es que se cumple la condición mencionada anteriormente los siguientes 7 bits del primer byte indicarán el número de fps (frames per second) o tramas por segundo y este valor puede ser igual a 24, 25, 29, 30, el valor de 29 realmente indica 29.97 fps; los siguientes 8 bits indican los intervalos por trama. En base a lo explicado anteriormente se puede transformar fácilmente los intervalos de tiempo a milisegundos con la siguiente fórmula:

  Variables:

  T = tiempo en milisegundos

  fps = tramas por segundo

  IT = intervalos por trama

  I = intervalos

  Fórmula

$$T = \frac{(\mathrm{I} * 1000)}{(\mathrm{IT} * \mathrm{fps})}$$

- <u>El primer bit de primer byte de la codificación del tiempo inicia con 0:</u>
  Si es que se cumple la condición mencionada anteriormente los 2 bytes forman un valor de tipo short, el cual indica el número de intervalos de tiempo contenidos dentro de un cuarto de nota, por lo cual en este caso para convertir intervalos de tiempo milisegundos es necesario también tener una variable adicional la cual es el tempo. El tempo dentro de los ficheros MIDI por defecto es 120 pulsos por minuto, sin embargo existe un evento que puede alterar este valor en cualquier punto de la reproducción. La fórmula para convertir los intervalos de tiempo a milisegundos según este esquema sería el siguiente:

  Variables

  T = tiempo en milisegundos

  TQN = intervalos de tiempo contenidos dentro de un cuarto de nota

  BPM = pulsos por minuto usados actualmente (Tempo)

  I = intervalos de tiempo

  Fórmula

$$T = \frac{60000 * \mathrm{I}}{(\mathrm{TQN} * \mathrm{BPM})}$$

Valores de longitud variable:

Como se explicó anteriormente antes de cualquier evento dentro de una pista existe un valor de longitud variable el cual indica de manera codificada el tiempo que se debe esperar antes de enviar el evento.

Algunos tipos de eventos también usan valores de longitud variable para indicar el número de bytes de datos que van a contener, por lo cual es necesario estudiar que se tratan los valores de longitud variable y como se diferencian de los valores de longitud fija como son valores de tipo int, byte, short.

Como se conoce, una variable entera normalmente ocupa 4 bytes, una variable de tipo short normalmente ocupa 2 bytes, en cambio un valor de longitud variable puede ocupar 1, 2 o n bytes debido a que cada byte contiene un bit que indica si el valor tiene más bytes o no.

El primer bit de cada byte de un valor de longitud variable es el que indica si el valor contiene más bytes o no, un 1 indica continuación, y un 0 indica parada, los siguientes 7 bits de cada valor de longitud variable se deben ir acumulando para formar el valor especificado[6].

Ejemplos de valores de longitud variable

Valor	Bytes
0	00000000
17	00010001
129	10000001, 00000001
16386	10000001, 10000000, 10000010

Tabla1.1.10 Ejemplo de los bytes de un valor de longitud variable creada por los autores

Eventos meta

Los conocidos eventos meta dentro de los ficheros MIDI sirven principalmente para describir parámetros, por ejemplo nombre de canción, nombre de pista, nota de copyright, letras de canción, tempo, finalización de pista, entre otros. Los eventos meta tienen la siguiente estructura [6].

Interpretación	**Byte (Decimal)**	**Byte (Binario)**
Indicador de inicio de evento meta	255	11111111
Identificador de evento meta	0 – 255	variable
Valor de longitud variable que indica el número de bytes de datos de este evento meta	0-255	Byte1
	0-255	Byte2
	0-255 …… ……	Byte n
Bytes de datos	0-255	Byte de datos1
	0-255	Byte de datos2
	0-255 ….. …..	Byte de datos n

Tabla.1.11 Estructura de un evento meta MIDI [4]

En la tabla 1.12 se muestra una lista de los eventos más comunes:

Evento	Byte de identificación (binario)	Byte de identificación (decimal)	Número de bytes de datos	Descripción
Tempo	01010001	81	3	Este evento MIDI nos permite saber el número de microsegundos que hay en un cuarto de nota, valor que puede ser usado para calcular el número de pulsos por segundo el cual se usa comúnmente en programas de notación musical. Cuando no se ha enviado este evento MIDI una sola vez se usa el valor de 500,000 microsegundos por cuarto de nota es decir 120 pulsos por minuto.
Fin de pista	00101111	47	0	Este evento deberá existir en toda pista MIDI una sola vez y deberá ser siempre el último evento de la mista.
Letra	00000101	5	variable	Letra correspondiente a la canción, generalmente una sílaba por evento enviado en el instante correcto.
Texto	00000001	1	variable	Un texto describiendo cualquier cosa puede ser la letra de una canción, puede contener el nombre de la pista, una descripción, etc. Y se deberán enviar caracteres ASCII imprimibles, (un carácter por byte).
Nota de Copyright	00000010	2	variable	Nota de copyright en caracteres ASCII imprimibles, esta nota deberá contener un texto como por ejemplo "Todos los derechos reservados", año del copyright y propietario del copyright.

Tabla 1.12 Tabla de eventos meta de interés en esta tesis [2]

1.1.3 ESTRUCTURA DE PAQUETES DE DATOS MIDI

Los paquetes de datos que son enviados por USB a cualquier dispositivo MIDI cumplen con la norma de tener la longitud de 32 bits es decir 4 bytes, sin embargo como se indicó anteriormente los eventos MIDI no tienen necesariamente esta longitud, la mayoría de estos eventos de canal por ejemplo cuentan con una longitud de 3 bytes, y algunos 2 bytes, por lo que para convertir un evento MIDI a paquete MIDI se le deberán agregar bytes para completar los 4 bytes. Los bytes que se agregaran serán un byte al inicio, y un byte al final en caso de que no se llegue a los 4 bytes. A continuación se puede ver más detalladamente la estructura de un paquete MIDI [7]:

Byte1	4 bits indicando el número de cable 4 bits indicando un código correspondiente al evento
Byte2	Byte que indica el evento de canal
Byte3	Primer byte de datos
Byte4	Segundo byte de datos o también puede ser byte 0 de relleno

Tabla 1.13 Estructura de un paquete de datos MIDI [7]

A continuación se muestran los códigos de eventos que corresponden a los eventos de canal del protocolo MIDI:

Código de evento	Evento
8	1000cccc (Apagado de nota)
9	1001cccc (Encendido de nota)
10	1010cccc (Presionado de tecla)
11	1011cccc (Cambio de control)
12	1100cccc (Cambio de programa o sonido)
13	1101cccc (Presión de canal)
14	1110cccc (Cambio de tono)

Tabla 1.14 Identificadores de eventos de canal en paquetes de datos MIDI [7]

Descripción estándar de un “Bulk data endopint” de un dispositivo MIDI

Índice	Campo	Tamaño	Value	Description
0	bLength	1	Number	El número de bytes totales que existen en este descriptor el cual tiene un valor de: 9
1	bDescriptorType	1	Constant	Tipo de descriptor
2	bEndpointAddress	1	Endpoint	La dirección del endpoint en el dispositivo USB. La dirección es codificada de la siguiente forma: D7: Direction. 0 = OUT endpoint 1 = IN endpoint D6...4: Reserved, reseteo a cero D3...0: Número de endpoint, determinado por el diseñador.
3	bmAttributes	1	Bit Map	D7...4: Reservado D3...2: Sincronización tipo 00 = Ninguna D1...0: Tipo de transferencia
4	wMaxPacketSize	2	Number	Tamaño máximo de paquete el cual el endpoint es capaz de enviar o recibir.
6	bInterval	1	Number	Intervalo de punto final de votación para las transferencias de datos expresada en milisegundos. Este campo se ignora para los puntos finales a granel. Debe ponerse a 0.
7	bRefresh	1	Number	Reseteo a 0
8	bSynchAddress	1	Endpoint	La dirección del punto final utiliza para comunicar información de sincronización si es requerido por este punto final. Puesta a cero.

Tabla 1.15 Estructura de un endpoint de un dispositivo MIDI [7]

1.1.4 COMUNICACIÓN SERIAL USB

La comunicación serial USB permite la transferencia en serie, es decir un bit a la vez con una velocidad acordada entre el emisor y el receptor; existen 3 velocidades de transferencia que son usadas en la comunicación serial las cuales se muestran a continuación:

• Alta velocidad de transferencia (480 Mb/s).

• Velocidad de transferencia de 12 Mb/s.

• Baja velocidad de transferencia correspondiente a 1.5 Mb/s.

Existen 2 formas de transmisión de datos dentro de la comunicación serial USB, a continuación se muestran más detalles [8]:

Bulk transfer (Transferencia masiva):

Este tipo de transferencia admite grandes cantidades de datos; se asegura que el intercambio de datos se lo haga de manera confiable mediante una detección de control de errores a nivel de hardware; la velocidad de transferencia puede variar dependiendo de las actividades del bus [8].

Control transfer (Transferencia de control):

Este tipo de transferencia es usada por los sistemas de software normalmente para configurar los dispositivos USB cuando son conectados, sin embargo se admiten otros usos también para este tipo de transferencia; la entrega de datos se la realiza sin ninguna clase de pérdidas[8].

Dentro de la comunicación USB existen 2 formas de actuación de un dispositivo dentro de la comunicación:

USB Host (Anfitrión USB):

Se caracteriza por ser el responsable de las siguientes actividades:

- Detectar cuando se conecta o se desconecta un dispositivo USB.
- Manejar el flujo de datos entre el host y el dispositivo USB.
- Proporciona una fuente de energía al dispositivo añadido.
- Colectar estado y estadísticas de actividad.
- Enumeración de dispositivos y configuración.
- Manejo de la información del bus del dispositivo.

USB Accesory (Accesorio USB)

- El accesorio USB es provisto de poder por el USB Host.
- El USB Host es el que envía los datos para que el accesorio los reciba y los interprete. [9].

1.2 SELECCIÓN DE LA METODOLOGÍA

Hablando de desarrollo de software existe una variedad de metodologías de las cuales se puede hacer uso y seguir sus prácticas. En el presente caso las metodologías ágiles presentan un buen camino para el desarrollo del proyecto en cuestión.

Tomando el punto de vista de los autores [12] y [13] existen algunos puntos a considerar en cuanto a una comparación entre metodologías tradicionales y ágiles, las cuales se detallan a continuación.

Las metodologías ágiles son soluciones a las metodologías tradicionales, que muchas veces son muy rigurosas tanto en tiempo, roles, actividades, artefactos, modelados y documentación exhaustiva. Las metodologías tradicionales como tal brinda la ventaja de poder ser útiles en proyectos donde los tiempos y recursos

son muy grandes y no muy cambiantes. De esta manera se ve que este tipo de metodologías son un tanto ceremoniosas, pero son muy útiles en proyectos grandes. En proyectos más pequeños el enfoque de las metodologías ágiles es muy adecuado, ya que representan una solución rápida, centrada de mayor manera en la comunicación con el cliente y potencian así la mejora continua en el desarrollo de software.

Las metodologías tradicionales centran gran parte de su enfoque en la documentación exhaustiva, la planificación y los procesos. Presentan una gran resistencia al cambio; si se llegan a realizar cambios en el software, su costo sería bastante grande. Este tipo de metodologías no son muy útiles para entornos cambiantes, o donde los clientes no saben bien o simplemente no saben lo que desean. Estas metodologías hacen énfasis a las etapas que cada una de ellas posee. En muchos casos estas etapas son secuenciales y en otras iterativas secuenciales; a pesar de esto cabe destacar que este tipo de metodologías se apegan en gran medida a la documentación más que en la retroalimentación continua con el cliente.

En contraste, las metodologías ágiles, son especialmente adecuadas para proyectos cambiantes, con exigencia en tiempos y poca documentación. Centran todo en el individuo y sus destrezas. En la tabla 1.16 se muestran las diferencias entre metodologías ágiles y no ágiles.

Metodologías tradicionales	Metodologías Ágiles
Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo.	Se basan en heurísticas provenientes de prácticas de producción de código
Existe resistencia a los cambios	Se adaptan a los cambios durante el proyecto
Gran control en los procesos con numerosas políticas/normas	Proceso menos controlado, con pocos principios
Contrato prefijado	No hay contratos o son bastante flexibles
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Grupos grandes y posiblemente distribuidos.	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
Más artefactos	Pocos artefactos
Más roles.	Pocos roles
La arquitectura del software es esencial y se expresa mediante modelos.	Menos énfasis en la arquitectura de software

Tabla 1.16 Diferencias entre las metodologías ágiles y no ágiles [9]

1.2.1 COMPARACIÓN DE LAS METODOLOGÍAS ÁGILES

Para el desarrollo del sistema en cuestión se escogieron algunas posibles metodologías que presentan algunos puntos interesantes en el desarrollo de software.

SCRUM

Scrum es un proceso de control y gestión que se centra en construir software que se centre en las necesidades del negocio. Scrum como tal es un marco de trabajo para una efectiva colaboración en proyectos complejos de software. Esta metodología se ha diseñado para proyectos con requisitos cambiantes. [19]

Se puede resumir en que el desarrollo se lo hace de manera iterativa mediante Sprints, con una duración de 2 semanas a 1 mes. Otra característica fundamental son las reuniones a lo largo del proyecto, entre estas están las reuniones diarias de 15 minutos para coordinar e integrar. Para mayor información sobre Scrum ver la referencia [14]

Extreme Programming (XP)

Esta metodología se centra en los cambios sobre la marcha; los cambios hasta cierto punto son inevitables e incluso deseables para el desarrollo del proyecto. [13] Se centra en dos prácticas fundamentales como son la programación orientada a pruebas y la programación en parejas. [16]

Metodologías Crystal

Se centra especialmente en las personas del equipo de trabajo y la disminución de los artefactos[1] producidos. Fue desarrollada por Alistair Cockburn. El desarrollo se basa en un desarrollo cooperativo y de bastante comunicación, solo limitado por los recursos. [19]

El factor clave es el equipo de desarrollo, sus habilidades y destrezas, también las políticas de trabajo en equipo. Para mayor información revisar la referencia [26]

Dynamic Systems Development Method (DSDM)

Marco para el desarrollo de un proceso de producción de software, es metodología RAD[2] unificada. Como principales características se tiene que es un desarrollo iterativo incremental colaborativo. Consta de 5 fases: estudio de viabilidad, estudio del negocio, modelo funcional, diseño y construcción e implementación. Mantiene la retroalimentación en todo el proyecto. [19] Para mayor información visitar la referencia [27]

Adaptive Software Development (ASD)

Como principales características de esta metodología se puede destacar que es iterativo, orientado a componentes y tolerante a los cambios. Su ciclo de vida consta de 3 fases: especulación, colaboración y aprendizaje. [19] Más información en la referencia [28]

[1] resultado de un trabajo que es usado durante un proyecto. Los artefactos son usados para capturar y llevar la información del proyecto[15]

[2] RAD desarrollo rápido de aplicaciones

Feature-Driven development (FDD)

El proceso iterativo que se plantea en esta metodología consta de 5 fases. Se centra especialmente en las fases de diseño e implementación, tomando como punto de inicio la lista de características que debe tener el software. [19] Más detalles en la referencia [29]

Lean Development (LD)

Fue diseñada por Bob Charette's gracias a su experiencia en proyectos en la industria automotriz de Japón. Los cambios se consideran riesgos, pero si se los controla adecuadamente pueden convertirse en oportunidades de mejora. Su principal característica es introducir un mecanismo para implementar los cambios. Más información en la referencia [30]

Más información en la tabla 1.17.

1.2.2 JUSTIFICACIÓN DE LA SELECCIÓN DE LA METODOLOGÍA

Entre las metodologías ágiles se escogió XP ya que es una de las más flexibles en cuanto a la definición de requerimientos y funcionalidades, en muchas de las otras metodologías se ve que ya realizada una iteración y presentado un entregable, no se puede modificar más, en cambio en XP aunque ya hayan sido entregadas las funcionalidades o entregables al cliente, en cualquier momento del desarrollo del proyecto pueden ser modificadas, para mejorar la calidad del proyecto.

Aunque en casi todas las metodologías se potencia el desarrollo cooperativo, muchas hacen que dentro del equipo cada individuo desarrolle su potencial en forma individual, para proyectos pequeños XP plantea una práctica que supera el individualismo tradicional.

XP hace énfasis en que la programación se realice en pares. [13], [14] y [25].

Si el equipo desea dentro de la iteración en XP se puede decidir que funcionalidad se desea comenzar a trabajar o cambiar por otra que el equipo desee realizar primero, en metodologías como SCRUM esto es mucho más estricto y solo se puede realizar el trabajo según lo planeado para ese Sprint, . [14]

Metodologías como SCRUM no recomiendan en sí ninguna práctica de ingeniería, en cambio en XP si se recomiendan. Este hecho es muy interesante ya que existen recomendaciones según la metodología como la programación en parejas para garantizar la calidad de los entregables y que las funcionalidades implementadas realmente cumplen con los requisitos del usuario. [14]

Por todo lo planteado XP es la metodología recomendable para este proyecto.

Característica	XP	SCRUM	DSDM	FDD	ASD	Crystal
Enfoque de desarrollo	Iterativo incremental	Iterativo incremental	Iterativo	Iterativo	Iterativo	Incremental
Tiempo estimado de la iteración	1 a 6 semanas	2 a 4 semanas	80% soluciones en 20% del tiempo	2 días a dos semanas	4 a 8 semanas	Depende del método a usarse
Grupo de trabajo	menos de 20 personas	todos los tamaños	todos los tamaños grupos independientes	muchos miembros, muchos grupos de trabajo	5 a 9 personas	todos los tamaños, dependiendo del método a usar
Comunicación	Informal, reuniones diarias	Informal, reuniones diarias	basado en documentación	basado en documentación	Informal, cara a cara	Informal, cara a cara
Tamaño del proyecto	pequeños	todos los tipos de proyectos	todos los tipos de proyectos	proyectos complejos	pequeños	todos los tipos de proyectos
Involucramiento del cliente	Cliente totalmente involucrado	Cliente toma el rol del Product Owner	El cliente a través de frecuentes entregables	El cliente a través de reportes	El cliente a través de frecuentes entregables	El cliente a través de entregas incrementales
Documentación del proyecto	Documentación básica	Documentación básica	Existe documentación	Es importante la documentación	Solo la documentación básica	Solo la documentación básica
Especialidades	TDD, Historias de usuario, Refactorización	Sprints, Product and Sprint Backlog. Planning Poker, Scrum Master	Prototipado	Diagramas UML	Ciclo de aprendizaje	Métodos adaptables
Ventajas	Workspace abierto, el cliente es parte del grupo, las mejores prácticas bien definidas, retroalimentación	Alto nivel de comunicación y colaboración	Enfoque de prioridad de requisitos, gestión eficiente de los proyectos	Los reportes y la documentación permiten trabajos múltiples	Desarrollo de componentes de alto riesgo en primer lugar, importancia del ciclo de aprendizaje	Metodología que se ajusta al tipo de proyecto y tamaño de este.
Desventajas	Poca documentación, falta de disciplina, la presencia del cliente es obligatoria	Poca documentación, control pobre sobre el proyecto	Documentación compleja	Propiedad individual del código, no se aplica a proyectos pequeños	Métodos y documentación pobre	Eficiente coordinación en grupos grandes

Tabla 1.17 Comparación de las metodologías ágiles tomada de la referencia [31]

1.2.3 EXTREME PROGRAMMING (XP)

La programación extrema durante algunos años ha tenido una amplia acogida gracias al hecho de enfatizar la satisfacción del usuario en cada fase de esta metodología. Ha funcionado perfectamente en muchas empresas y entidades de todo tipo.

A diferencia de las metodologías formales XP o Extreme programming no se centra en entregar un todo, sino más bien realizar entregas periódicas de funcionalidades según como se vayan necesitando. De esta manera, permite a los desarrolladores hacer frente a los desafíos y a los cambios pedidos por el cliente, a pesar de que sean hechos en etapas tardías del proyecto.
Un equipo colaborativo se enfatiza de gran manera con XP, el trabajo en equipo es fundamental en la filosofía de esta metodología, los encargados del proyecto, los clientes y desarrolladores son todos iguales en un grupo colaborativo. Mediante el trabajo en grupo se plantea resolver los problemas en una manera más eficiente.

Al utilizar esta metodología se mejora el desarrollo de software en cinco puntos fundamentales, estos son: simplicidad, retroalimentación, respeto, coraje y comunicación.

Constantemente se tiene comunicación entre los programadores, entre los clientes y el equipo de desarrollo; los desarrolladores mantienen por filosofía el código simple y limpio. Se mantiene una retroalimentación mediante las pruebas constantes que comienzan desde el primer día. Se realizan entregas periódicas y lo más pronto posible, también se van realizando los cambios según cómo van surgiendo. El éxito de los proyectos con XP va a depender de la colaboración de cada miembro del equipo. De esta manera el equipo de desarrollo va a poder responder de buena forma a los cambios y requerimientos, ya que los cambios son parte natural en cualquier proyecto de

software; es mucho más realista adaptarse a los cambios que intentar definir estrictos requisitos al comienzo de los proyectos e invertir esfuerzos y tiempo en controlar luego estos cambios.

XP no tiene reglas complejas en contraste tiene reglas simples. Las reglas se basan en valores y principios sólidos. Adicionalmente definen un ambiente que promueve la colaboración grupal y fortalecimiento de los equipos desarrolladores.

REGLAS EN EXTREME PROGRAMMING

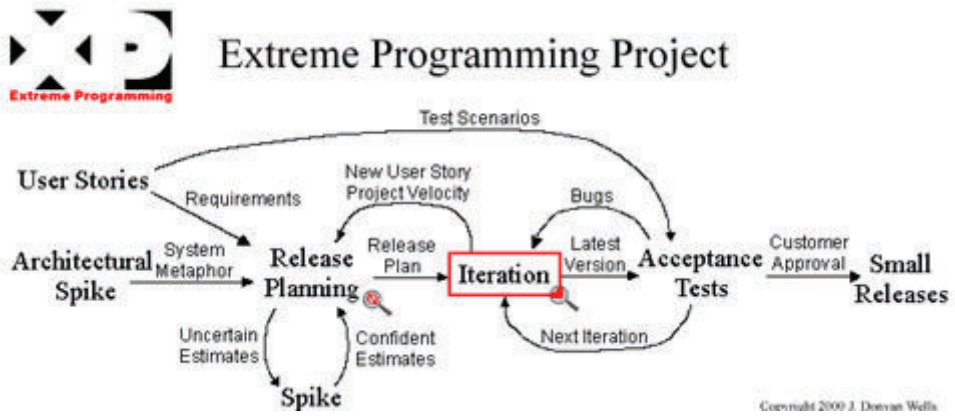


Figura 1.1 Proyecto Programación Extrema tomado de [6]

En XP se definen reglas y prácticas que se pueden agrupar de la siguiente manera:

- **Reglas y prácticas para la Planificación**

La planificación es una práctica fundamental en XP, se la plantea como un diálogo continuo entre las partes involucradas (clientes, desarrolladores, jefes de proyecto, gerentes, coordinadores, etc.).

Lo primero que se debe realizar dentro del proceso XP es la recopilación de “historias de usuario”.

- **Historias de Usuario**

La creación de historias de usuario es la técnica dentro de las metodologías ágiles que reemplaza a los casos de uso dentro de las metodologías orientadas a objetos más conocidas. El cliente escribe las características que el sistema debe o va a tener con los requerimientos funcionales y no funcionales, estas pueden ser escritas en papel, ya que el trato que se les da a estas historias es muy dinámico y flexible. Las historias de usuario deben ser fáciles de comprender y bien especificadas para que los programadores sepan exactamente qué es lo que deben hacer.

Una historia de usuario puede tener los siguientes campos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, numeración, prioridad tanto del cliente como técnica, referencia a otra historia anterior, riesgo, estimación, descripción, notas y una lista de seguimiento con la fecha, estado, partes por terminar y comentarios. En tiempo de programación una historia puede durar de una a tres semanas de desarrollo, eso sí tiene que ser menor que una iteración. Luego estas historias van a ser desglosadas en tareas de programación (tarjetas de tareas) y la asignación al responsable para ser implementada durante la iteración.

- **Roles**

Dentro de XP también se deben considerar los roles que se van a tener. Fundamentalmente se tiene:

Programador: Encargado de escribir las pruebas y de producir el código del sistema a desarrollarse.

Cliente: Realiza las historias de usuario y las pruebas funcionales para validar la implementación, prioridades a cada historia de usuario y decide cuales son las que se van a desarrollar en que iteración para aportar un mayor valor al negocio.

Encargado de las pruebas (Tester): Ayuda al cliente a escribir las pruebas funcionales, las ejecuta con regularidad y difunde los resultados en el equipo.

Encargado de Seguimiento (Tracker): Brinda la retroalimentación al equipo, verifica el grado de acierto entre la estimación realizada y el tiempo real utilizado para cada funcionalidad. Da seguimiento al progreso de cada iteración.

Entrenador (Coach): Responsable de todo el proceso, guía al equipo en cuanto a las prácticas XP y los procesos.

Consultor: Miembro externo con conocimientos necesarios en un área específica del proyecto, donde pueden darse problemas.

Gestor (Big Boss): Vínculo entre el cliente y los programadores, el gestor crea las condiciones adecuadas de trabajo para ayudar al equipo, es un coordinador principalmente.
Luego de obtener las historias de usuario, los programadores del equipo evalúan el tiempo de desarrollo de estas, también se define el riesgo de cada historia. Si en el proyecto se ven riesgos que brinden una incertidumbre de la complejidad del desarrollo de esa funcionalidad, se hacen pequeños programas de pruebas llamados spikes para mitigar estos riesgos.

Si no hay problemas o incertidumbres en los tiempos estimados se procede a realizar la planificación del proyecto con la intervención de los diversos actores

vistos previamente. Se establece un plan o cronograma de entregables. En cada entregable se van a desarrollar, probar e instalar algunas tareas.

- **Plan de iteraciones**

Se seleccionan cierto número de historias de usuario que van a ser entregadas en cada iteración estas van a ser desarrolladas y probadas en un ciclo de iteración, de acuerdo con el orden de prioridades propuesto.

Al comenzar un ciclo se tiene una reunión de la iteración, se desglosan las tareas de programación a partir de las historias de usuario y según cada historia de usuario se realizan las pruebas de aceptación. Siempre se van a ejecutar estas pruebas al final del ciclo de desarrollo, también se debe verificar que las subsiguientes iteraciones no hayan afectado las anteriores.

Si alguna prueba falla se evalúa su corrección y se evita que vuelva a ocurrir.

- **Reuniones diarias**

El objetivo principal de estas reuniones es fomentar la comunicación en el equipo de trabajo, compartir problemas y soluciones.

- **Reglas y prácticas para el Diseño**

XP resalta la importancia de un diseño simple y claro. Como conceptos fundamentales se tiene:

- **Simplicidad**

XP propone implementar el diseño más simple posible, se sugiere no adelantar funcionalidades que no estén contempladas dentro de la iteración en curso. Si se mantiene la simplicidad todo se va a implementar de manera más rápida.

- **Soluciones (Spikes)**

  Son programas de prueba que se utilizan en situaciones de difícil estimación de tiempos o donde hay problemas técnicos, su finalidad es encontrar posibles soluciones a estos problemas. Son totalmente desechables luego de su evaluación.

- **Recodificación**

  Consiste en cambiar parte del código sin cambiar su funcionalidad a efectos de hacerlo más sencillo, fácil de entender y conciso. XP sugiere recodificar el código cada vez que sea necesario, esta práctica ayuda en las iteraciones siguientes y a seguir la filosofía de mantener el código lo más simple posible.

- **Metáfora**

  Una metáfora es un concepto que se entiende sin mayores explicaciones. Esta metodología utiliza este concepto como una manera sencilla de explicar el propósito de un proyecto, guiar su estructura y arquitectura. Como ejemplo puede ser una guía sobre la nomenclatura de los métodos y clases utilizadas en el diseño del código. Las metáforas como tal deben ser fáciles de entender para el cliente y debe tener suficiente contenido para que sirva de guía a la arquitectura del proyecto.

- **Reglas y prácticas para el Desarrollo**

  - **Disponibilidad del Cliente**

    Los clientes aprecian en buena medida ser parte activa del proceso de software, a su vez los desarrolladores contribuyen sin importar el nivel de experiencia y los jefes de proyecto se concentran en la comunicación y las relaciones en el equipo.

    Es fundamental trabajar constantemente con el cliente, su apoyo es fundamental para el éxito del proyecto.
    Para comenzar el proyecto como bien se sabe el cliente debe proporcionar las historias de usuario, ya que estas son cortas y de alto nivel no tiene todos los detalles para comenzar la programación. Esos detalles se los obtiene con el cliente y se los analiza con los programadores; por lo general estos detalles se los obtiene en conversaciones con los clientes, estas formas de obtener los detalles o las tareas a programar reemplazan a las especificaciones formales que hace el cliente.
    Como el cliente va a estar continuamente en el proceso va a poder realizar correcciones y prevenir funcionalidades no deseadas.
    En otras metodologías estas correcciones son más costosas y se las realiza en fases posteriores, aumentando el costo de estas. Cabe recordar que mientras más rápido se corrige un error, va a ser menor el costo de este.

  - **Uso de estándares**

    XP promueve la utilización de estándares, para que todo el trabajo sea entendible para todo el equipo de desarrollo

Programación en pares

La metodología XP propone la programación en pareja como un medio para minimizar errores, a pesar de la duplicidad de los errores y los costos en el proyecto. Este enfoque plantea varias ventajas como la mejora en el diseño, mejor calidad en el producto final. Toda la producción del código debe realizarse en una misma máquina por dos desarrolladores. Al momento de programar una persona codifica y la otra observa, mientras se mantiene pensando en la solución del problema y todo lo que XP propone. Si el programador se atasca en algún punto del desarrollo, se pueden cambiar de papeles para resolver ese atasco de mejor manera. Así, la programación se vuelve mucho más divertida y sociable. [22]

Existen ventajas que cita el autor [8] como el descubrimiento y resolución de errores en la marcha, se minimizan los defectos encontrados en las pruebas, se tiene un mejor diseño y código más corto. Se resuelven los problemas más rápidamente. Se aprende mucho más del sistema y del desarrollo de software. Hay más personas que conocen más detalles del código. Hay mejor dinámica de grupo, la gente aprende a trabajar junta, esto genera que la información fluya rápidamente. Todos los involucrados disfrutan más del trabajo.

Integración continua o permanente

Todos trabajan siempre con la última versión. Si se hacen cambios o mejoras en una versión antigua pueden darse muchos problemas en el proyecto. Cada día se aconseja publicar una nueva versión del proyecto y solo una pareja de desarrolladores puede integrar una versión a la vez. [18]

Propiedad colectiva

Todos los involucrados en el desarrollo de un proyecto son dueños del código y cualquier par de programadores dentro del proyecto pueden cambiar partes del código si lo consideran pertinente.

La recodificación como se ha visto se realiza con el fin de mantener el código simple y de mejor calidad, además las pruebas continuas garantizan que los cambios no afecte ninguna otra funcionalidad. [18]

Ritmo sostenible

En la referencia [19] se recomienda según la metodología el trabajo máximo de 40 horas para que no se desmotive el equipo, no sería muy recomendable que el equipo trabaje horas extras en dos semanas seguidas. Más allá de las horas de trabajo.
En otra referencia [18] se muestra que se debe establecer el concepto de mantener un ritmo constante para no sobrecargar al equipo. Si el proyecto se llega a retrasar se debería renegociar el plan de entregas en una reunión tanto con el cliente, los desarrolladores y los gerentes. No siempre añadir más gente a un proyecto resuelve un problema.

- **Reglas y prácticas para las Pruebas**

En cuanto a pruebas se indican las siguientes prácticas según el autor [18]:

  - **Pruebas Unitarias**

Una parte fundamental en XP son las pruebas unitarias. Todos los módulos del sistema deben pasar las pruebas unitarias antes de ser integrados al sistema. Como ya se vió las pruebas deben ser hechas

antes de comenzar a programar. Este tipo de pruebas va a garantizar el buen funcionamiento de ese módulo.

- **Corrección de errores**

Al encontrarse un error (“bug”) debe corregirse este inmediatamente y realizar nuevamente las pruebas para verificar que el error se ha resuelto.

- **Pruebas de aceptación**

Estas pruebas son creadas en base a las historias de usuarios en cada ciclo de la iteración del desarrollo. El cliente plantea uno o varios escenarios para verificar la correcta implantación de la historia de usuario.
Se pueden considerar estas pruebas como de caja negra ya que el cliente no va a revisar el código, más bien solo el correcto funcionamiento del proceso. Si los resultados son los correctos la historia de usuario ha sido terminada, si se encuentran errores, el cliente debe indicar el orden de resolución.
Todo el equipo desarrollador debe estar al tanto de los resultados de las pruebas de aceptación ya que la responsabilidad es colectiva.

1.3 SELECCIÓN DE HERRAMIENTAS PARA CONSTRUCCIÓN DE APLICACIÓN

Como se ha especificado en puntos anteriores un dispositivo USB puede actuar como host o como accesorio dentro de la comunicación serial USB; en este caso el dispositivo Android necesariamente deberá actuar como host dentro de la comunicación debido a que se requiere que el dispositivo Android sea capaz de detectar si existe o no un dispositivo conectado, además de que como especifica la documentación oficial de Android developers, cuando el dispositivo Android

actúa como accesorio, el dispositivo USB debe estar específicamente diseñado para comunicarse con el dispositivo Android lo cual no es una característica propia de los dispositivos MIDI, en vez de ello el dispositivo Android es el que deberá contar con el software que permita establecer la comunicación y llevar el control de la transferencia y recepción de datos.

API de Android USB Host

La documentación oficial de Android indica que existe una API específicamente diseñada para establecer comunicación USB permitiendo actuar a la Tablet como HOST dentro de la comunicación. También señala que antes de iniciar la comunicación se deberá solicitar permiso al usuario para iniciar la comunicación, una vez que el usuario acepte la comunicación el dispositivo empezará con la transferencia y recepción de datos.

Los requisitos para usar esta API son los siguientes:

- Android 3.1 o superior
- El dispositivo deberá tener soporte USB Host (no todos los dispositivos Android 3.1 o superior cuentan con esta característica) [10].

Android Studio como IDE para desarrollo:

Android Studio es descrito por Google como el IDE oficial para desarrollo de aplicaciones Android, una vez que fue lanzado opacó totalmente el uso de Eclipse. Android Studio presenta diversos beneficios sobre otros IDEs de desarrollo, como por ejemplo viene con emuladores pre-configurados optimizados y manejo de dependencias con maven[3].

[3] Herramienta de software para la construcción de proyectos Java, está listo para usarse en red y puede dinámicamente descargar dependencias de un repositorio.

2 DESARROLLO DE LA APLICACIÓN

2.1 FASE 1 PLANEACIÓN.

2.1.1 REQUERIMIENTOS

2.1.1.1 Definición del mercado objetivo

El mercado objetivo o los interesados en este software son las personas que poseen algún dispositivo MIDI (baterías, pianos, controladores, guitarras, etc.) y les interesa expandir las capacidades de su dispositivo, por ejemplo: en los sintetizadores se tiene un grupo de demos que vienen pregrabados en el dispositivo, una extensión a esta funcionalidad podría ser reproducir más archivos MIDI de diferentes fuentes. Una segunda extensión podría ser guardar en formato MIDI cualquier melodía entonada para así usando cualquier software poder ver la partitura y escuchar la melodía con cualquier instrumento. Otra funcionalidad añadida sería la posibilidad de visualizar las letras de canciones estilo karaoke, es decir se distinguen las letras que ya han pasado de las que vienen.

2.1.1.2 Requerimientos iniciales

Para la toma de requisitos se realizó encuestas a músicos y usuarios de programas de música, las encuestas se las pueden encontrar en el anexo B. Las preguntas fueron acerca de temas como que es MIDI, si los usuarios disponen de dispositivos móviles, detalles de los dispositivos móviles que usan, que dispositivos MIDI conocen, que software para manejo de archivos MIDI conocen.

Las respuestas obtenidas de las encuestas permiten saber si es que existe interés hacia la aplicación y también permiten determinar que funcionalidades son prioritarias.

De lo obtenido de las encuestas los requerimientos son los siguientes:

- Reproducir archivo MIDI
- Grabar archivo y secuencia MIDI
- Guardar el archivo MIDI
- Cargar el archivo MIDI
- Visualización del teclado y Karaoke

2.1.1.2.1 Análisis

En base a los resultados de las encuestas que se presenta en el anexo B se ha determinado el siguiente orden en cuestión de los requerimientos y su importancia gracias a las respuestas obtenidas dentro de la encuesta y el orden de la implementación, se va a tener 10 como máximo valor de importancia y 1 como el valor mínimo

Requerimiento	**Importancia**
Cargar archivo MIDI	10
Guardar archivo MIDI	9
Reproducir archivo MIDI	10
Grabación de secuencia MIDI	9
Visualización de teclado virtual y karaoke	5

Tabla 2.1 Nivel de importancia de los requerimientos. Elaborado por: los autores

El mayor porcentaje de respuestas se ubicó en la reproducción y en la grabación de archivos MIDI, esto indica que estos dos requerimientos son muy importantes; en el caso de la reproducción depende de que previamente se haya cargado el archivo MIDI. En el caso de la grabación, esta tarea está relacionada con la tarea de guardar archivo MIDI por eso tienen un alto nivel de importancia ya que están asociadas a los requerimientos más importantes.

2.1.1.3 Requerimientos técnicos

Para el funcionamiento de este proyecto se tiene 3 partes, la primera el ordenador, la segunda el dispositivo móvil y por último el dispositivo MIDI.

A continuación se detalla sus características:

2.1.1.3.1 Ordenador

Los requisitos para el funcionamiento del ambiente de desarrollo Android Studio son:

- Windows 7 o posteriores.
- 2 GB RAM mínimo, recomendable 4 GB RAM.
- 1GB en disco duro mínimo.
- La resolución mínima de la pantalla es de 1280 x 800.
- Java Development Kit (JDK) mínimo en su versión 7.

Para más detalles ver la referencia [3].

2.1.1.3.2 Dispositivo Móvil

Los requerimientos para el buen funcionamiento de la aplicación son:

- Versión de Android 3.1 o superior
- Soporte USB Host

2.1.1.3.3 Dispositivo MIDI

El requisito que debe cumplir el dispositivo es tener una entrada y salida MIDI a USB.

2.1.2 ELEMENTOS EN LAS HISTORIAS DE USUARIO

Las historias de usuario presentan algunos componentes importantes los cuales son:

Número de historia: Código de identificación de la historia de usuario

Usuario dirigido: Usuario al cual va a beneficiar la historia de usuario

Nombre de la historia: Nombre descriptivo de la historia de usuario

Fecha: Fecha de realización de la historia de usuario

Tipo de actividad: Puede ser nueva, modificación de una anterior o una mejora de la historia de usuario

Prioridad en el negocio: Importancia de la historia se califica de 1 a 4 siendo 1 la prioridad más alta y 4 la más baja

Riesgo en el desarrollo: Riesgo asociado a la implementación de la historia.

Puntos estimados: estimación del tiempo realizada por el grupo de desarrollo para implementar la historia

Puntos reales: tiempo real de realización de la historia de usuario

Iteración asignada: iteración en la que se va implementar la historia de usuario

Estado: estado de la realización de historia de usuario

Descripción: texto explicativo más detallado sobre la historia de usuario

Observaciones: Información relevante para el desarrollo de la historia de usuario

2.1.3 ESPECIFICACIÓN DE LAS HISTORIAS DE USUARIO

Historia de Usuario	
Número: 1	**Usuario:** Usuario final
Nombre historia: Cargar archivo MIDI	
Fecha:22/05/2015	**Tipo de Actividad:** Nueva
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 10 días	**Puntos reales:** 10 días
Iteración asignada: 3	
Descripción: ● El usuario final podrá cargar un archivo MIDI dentro del software para su posterior uso	
Observaciones: ● El programa abrirá solamente archivos con extensión .kar .mid .midi. ● Los archivos MIDI son validados dentro del programa impidiendo que el software trate de abrir archivos dañados o archivos que no sean los adecuados	

Tabla 2.2 Historia de Usuario Carga de Archivo MIDI realizada por Autores

Historia de Usuario	
Número: 2	**Usuario: Usuario final**
Nombre historia: Guardar archivo MIDI	
Fecha: 19/06/15	**Tipo de Actividad:** Nueva
Prioridad en negocio: Alta	**Riesgo en desarrollo: Medio**
Puntos estimados: 5 días	**Puntos reales:** 4 días
Iteración asignada: 3	**Estado:** Completada
Descripción: ● El usuario final podrá guardar archivos que han sido grabados a partir de secuencias midi grabadas dentro del programa.	
Observaciones: ● El programa agregará la extensión .mid si el usuario no ha introducido una extensión en el programa.	

Tabla 2.3 Historia de Usuario Guardar Archivo MIDI realizada por Autores

Historia de Usuario	
Número: 3	**Usuario:** Usuario final
Nombre historia: Reproducir archivo MIDI	
Fecha: 05/06/2015	**Tipo de Actividad:** Nueva
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 4 días	**Puntos reales:** 3 días
Iteración asignada: 3	**Estado:** Completada
Descripción: ● El usuario podrá reproducir un archivo con las siguientes extensiones .midi .mid .kar vía USB y escucharla directamente en su dispositivo MIDI. ● También, se podrá pausar y reanudar la reproducción del archivo MIDI en cualquier momento de la canción que esté sonando. Se podrá modificar los volúmenes de los diferentes canales del archivo MIDI o la secuencia a utilizar.	
Observaciones: ● Se requiere que el dispositivo móvil corra en sistema operativo Android 3.1 o superior y que tenga soporte USB host. ● El sistema deberá enviar los eventos de apagado de nota en los 16 canales MIDI para evitar que el dispositivo MIDI permanezca sonando ● El programa nos mostrará los instrumentos que van a ser entonados en un canal MIDI específico. ● Se podrá activar o desactivar el cambio de volumen, si este no se activa se enviarán los eventos de volumen originalmente contenidos en el fichero, caso contrario se ignoraran estos eventos y el software enviará al inicio de la reproducción el volumen seleccionado, como un evento.	

Tabla 2.4 Reproducir Archivo MIDI realizada por Autores

Historia de Usuario	
Número: 4	**Usuario: Usuario Final**
Nombre historia: Grabar secuencia	
Fecha: 15/06/2015	**Tipo de Actividad:** Nueva
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 5 días	**Puntos reales:** 5 días
Iteración asignada: 3	**Estado:** Completada
Descripción: ● El usuario final podrá receptar los eventos MIDI de entrada y guardar la secuencia en formato MIDI ● El usuario final podrá controlar el tempo de grabación que aparecerá en el archivo MIDI de salida	
Observaciones: ● Se deberá crear un algoritmo que convierta la secuencia de paquetes de datos MIDI a eventos MIDI que van a estar contenidos en el archivo de salida ● El software en el que se abra el archivo MIDI de salida mostrará el tempo que se ha escogido para la grabación	

Tabla 2.5 Historia de Usuario Grabar Secuencia realizada por Autores

Historia de Usuario	
Número: 5	**Usuario:** Usuario Final
Nombre historia: Visualización de información MIDI	
Fecha: 15/06/2015	**Tipo de Actividad:** Nueva
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 5 días	**Puntos reales:** 4 días
Iteración asignada: 3	**Estado:** Completado
Descripción: ● Para visualizar de mejor manera las notas musicales que se están entonando, el usuario final podrá contar con un teclado virtual que mostrará las notas que se están entonando con un color distinto según el canal ● Se podrá controlar mediante un menú el tamaño del teclado. ● En caso de que el teclado completo no alcance en la pantalla del dispositivo Android se tendrá una barra de desplazamiento para poder cambiar el rango visible del teclado. ● Si se ha cargado un archivo MIDI que contiene información sobre la letra de canción, el software mostrará la letra de manera que se pueda diferenciar la letra que ya pasó y la que vendrá para un instante determinado de la canción	
Observaciones: ● Se tendrán 6 valores establecidos de tamaños de teclado que el usuario podrá escoger. ● La barra de desplazamiento mostrará un teclado pequeño para poder ubicar de mejor manera el rango visual del teclado actual.	

Tabla 2.6 Historia de Usuario Visualización de información MIDI realizada por Autores

2.1.4 ACTORES EN XP

Existen los siguientes roles que se especifican según lo visto en la programación extrema según el autor [34]:

Programadores: Diseñan, programan y realizan las pruebas, luego construyen el sistema y toman las decisiones técnicas.

Cliente: Determina que construir y cuando (con que prioridad), escribe tests funcionales para ver cuándo está terminado un aspecto.

Entrenador: Toma las decisiones importantes, es responsable del proceso global, guía al equipo para que se sigan de la mejor manera las prácticas de XP.

Encargado de las pruebas: Ayuda al cliente a realizar las pruebas funcionales, es responsable de las herramientas de soporte para pruebas.

Figura 2.1Roles en XP elaborada por los autores

El ingeniero Carlos Montenegro es el entrenador ya que como director del proyecto, vigila el proceso y el cumplimiento de las prácticas de XP.

Como en XP una de las prácticas más importantes es la programación en parejas, tanto Eduardo Aníes como David Proaño son programadores.

En este caso los encargados de realizar las pruebas son Eduardo Aníes y David Proaño ya que ambos van a desarrollar el software y necesariamente deberán diseñar las pruebas para validar su código, integrar las partes y validar las funcionalidades con los usuarios finales.

2.1.5 PRIORIZACIÓN Y ESTIMACIONES

2.1.5.1 Estimación de tiempos y duración del proyecto

Para la estimación de tiempo de realización del proyecto se estima la realización de la historia en base a una semana de 7 días y un día de 4 horas.

Número	Historia de Usuario	Estimaciones		
		Semanas	Días	Horas
1	Carga de archivo MIDI	2	10	40
2	Guardar archivo MIDI	2	10	40
3	Reproducir archivo MIDI	1.2	6	24
4	Grabar secuencia	0.8	4	16
5	Visualización de información musical	1.4	7	28

Tabla 2.7 Estimaciones de tiempo por historia de usuario creada por los autores

2.1.5.2 Plan de Iteraciones

De lo previsto en las historias de usuario y la recolección de los datos se han estimado 4 iteraciones en las que se muestra el tiempo que se utiliza para la realización de las mismas.

2.1.5.2.1 Planificación de tareas primera iteración

En esta fase se establecen los requerimientos funcionales y no funcionales del sistema, se definen las historias de usuario, el plan de entregas y las historias técnicas.

Las tareas a realizarse se muestran en la figura siguiente:

Nombre de tarea	Duración	Comienzo	Fin
MIDI	**88 días**	**mié 01/04/15**	**vie 31/07/15**
Primera Iteración	**10 días?**	**mié 01/04/15**	**mar 14/04/15**
Recolección de requisitos	**10 días?**	**mié 01/04/15**	**mar 14/04/15**
Creación de encuestas	1 día	mié 01/04/15	mié 01/04/15
Encuestas a usuarios finales	4 días	jue 02/04/15	mar 07/04/15
Análisis de resultados	1 día?	mar 07/04/15	mar 07/04/15
Determinación de requisitos	1 día	mié 08/04/15	mié 08/04/15
Elaboración de actividades	**4 días?**	**jue 09/04/15**	**mar 14/04/15**
Elaboración de historias de usuario	1 día?	jue 09/04/15	jue 09/04/15
Elaboración de historias técnicas	2 días?	vie 10/04/15	lun 13/04/15
Elaboración del plan de entregas	2 días?	lun 13/04/15	mar 14/04/15

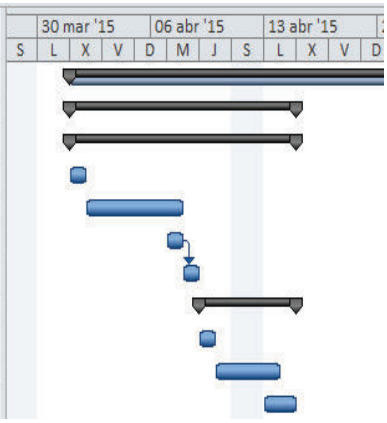
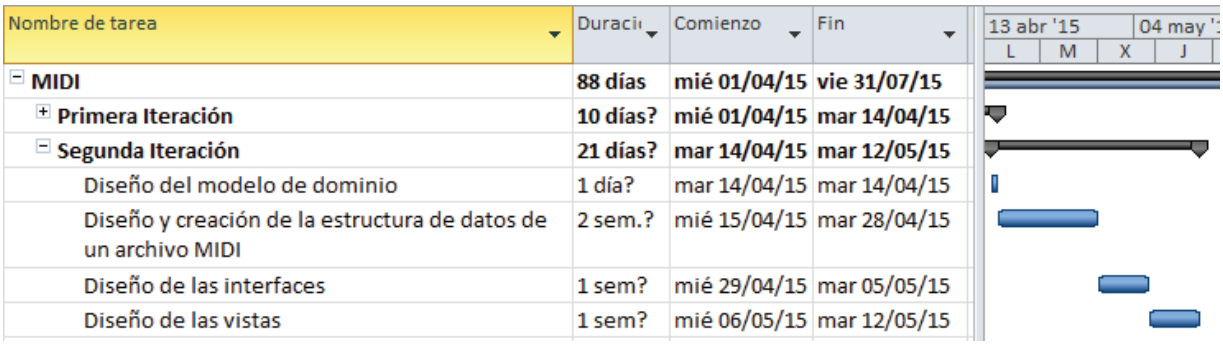


Figura 2.2 Diagrama de Gantt primera Iteración elaborada por los autores

2.1.5.2.2 Planificación de tareas segunda iteración

En la segunda iteración se tiene por objetivo la realización de un diseño tentativo para la aplicación, según este diseño y lo que se establezca en los modelos se pretende realizar la aplicación en la siguiente fase.

Las tareas a realizarse se muestran en la figura siguiente:



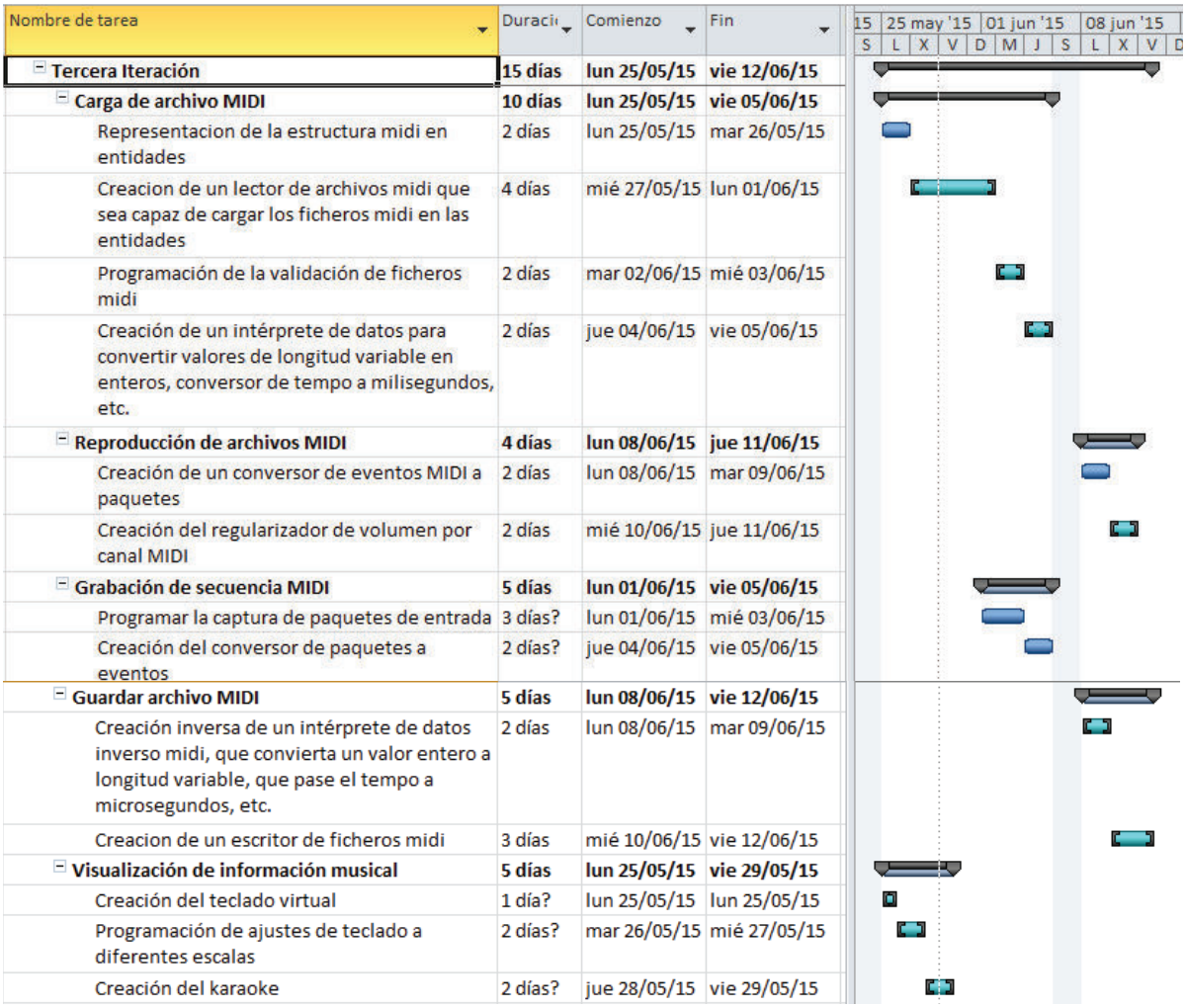
Nombre de tarea	Duraci	Comienzo	Fin
MIDI	**88 días**	**mié 01/04/15**	**vie 31/07/15**
Primera Iteración	**10 días?**	**mié 01/04/15**	**mar 14/04/15**
Segunda Iteración	**21 días?**	**mar 14/04/15**	**mar 12/05/15**
Diseño del modelo de dominio	1 día?	mar 14/04/15	mar 14/04/15
Diseño y creación de la estructura de datos de un archivo MIDI	2 sem.?	mié 15/04/15	mar 28/04/15
Diseño de las interfaces	1 sem?	mié 29/04/15	mar 05/05/15
Diseño de las vistas	1 sem?	mié 06/05/15	mar 12/05/15

Figura 2.3 Diagrama de Gantt segunda Iteración elaborada por los autores

2.1.5.2.3 Planificación de tareas tercera iteración

En esta parte se planean las tareas para la realización de la primera versión de la aplicación. Se basa en el diseño previsto en la fase anterior.

Las tareas a realizarse se muestran en las figuras siguientes:



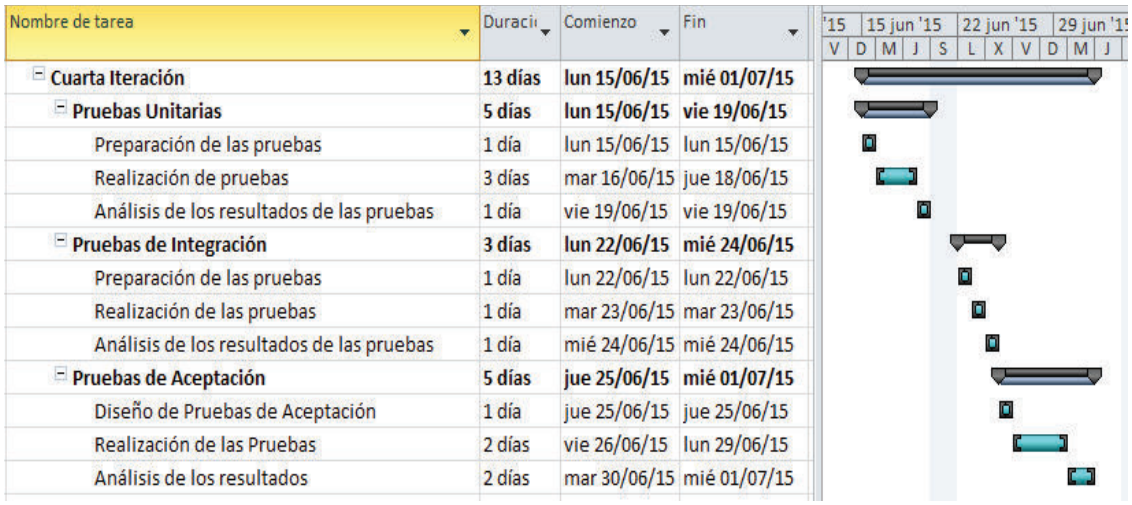
Nombre de tarea	Duración	Comienzo	Fin
Tercera Iteración	**15 días**	**lun 25/05/15**	**vie 12/06/15**
Carga de archivo MIDI	**10 días**	**lun 25/05/15**	**vie 05/06/15**
Representacion de la estructura midi en entidades	2 días	lun 25/05/15	mar 26/05/15
Creacion de un lector de archivos midi que sea capaz de cargar los ficheros midi en las entidades	4 días	mié 27/05/15	lun 01/06/15
Programación de la validación de ficheros midi	2 días	mar 02/06/15	mié 03/06/15
Creación de un intérprete de datos para convertir valores de longitud variable en enteros, conversor de tempo a milisegundos, etc.	2 días	jue 04/06/15	vie 05/06/15
Reproducción de archivos MIDI	**4 días**	**lun 08/06/15**	**jue 11/06/15**
Creación de un conversor de eventos MIDI a paquetes	2 días	lun 08/06/15	mar 09/06/15
Creación del regularizador de volumen por canal MIDI	2 días	mié 10/06/15	jue 11/06/15
Grabación de secuencia MIDI	**5 días**	**lun 01/06/15**	**vie 05/06/15**
Programar la captura de paquetes de entrada	3 días?	lun 01/06/15	mié 03/06/15
Creación del conversor de paquetes a eventos	2 días?	jue 04/06/15	vie 05/06/15
Guardar archivo MIDI	**5 días**	**lun 08/06/15**	**vie 12/06/15**
Creación inversa de un intérprete de datos inverso midi, que convierta un valor entero a longitud variable, que pase el tempo a microsegundos, etc.	2 días	lun 08/06/15	mar 09/06/15
Creacion de un escritor de ficheros midi	3 días	mié 10/06/15	vie 12/06/15
Visualización de información musical	**5 días**	**lun 25/05/15**	**vie 29/05/15**
Creación del teclado virtual	1 día?	lun 25/05/15	lun 25/05/15
Programación de ajustes de teclado a diferentes escalas	2 días?	mar 26/05/15	mié 27/05/15
Creación del karaoke	2 días?	jue 28/05/15	vie 29/05/15

Figura 2.4 Diagrama de Gantt tercera Iteración elaborada por los autores

2.1.5.2.4 Planificación de tareas cuarta iteración

En esta última iteración se planea realizar las pruebas correspondientes a la aplicación, entre las pruebas se tiene: las pruebas unitarias, las pruebas de aceptación, con las que se va a ajustar las pruebas a las necesidades del usuario según los diferentes criterios, también las pruebas de integración y al final correspondiente a cada caso la corrección de los errores encontrados.

Las tareas a realizarse se muestran en la figura siguiente:



Nombre de tarea	Duraci	Comienzo	Fin
Cuarta Iteración	**13 días**	**lun 15/06/15**	**mié 01/07/15**
Pruebas Unitarias	**5 días**	**lun 15/06/15**	**vie 19/06/15**
Preparación de las pruebas	1 día	lun 15/06/15	lun 15/06/15
Realización de pruebas	3 días	mar 16/06/15	jue 18/06/15
Análisis de los resultados de las pruebas	1 día	vie 19/06/15	vie 19/06/15
Pruebas de Integración	**3 días**	**lun 22/06/15**	**mié 24/06/15**
Preparación de las pruebas	1 día	lun 22/06/15	lun 22/06/15
Realización de las pruebas	1 día	mar 23/06/15	mar 23/06/15
Análisis de los resultados de las pruebas	1 día	mié 24/06/15	mié 24/06/15
Pruebas de Aceptación	**5 días**	**jue 25/06/15**	**mié 01/07/15**
Diseño de Pruebas de Aceptación	1 día	jue 25/06/15	jue 25/06/15
Realización de las Pruebas	2 días	vie 26/06/15	lun 29/06/15
Análisis de los resultados	2 días	mar 30/06/15	mié 01/07/15

Figura 2.5 Diagrama de Gantt cuarta Iteración elaborada por los autores

2.1.5.3 Plan de Entregas

Para cada una de las historias de usuario ya especificadas y detalladas en el plan del proyecto con sus respectivos tiempos y recursos, se va a realizar un plan de entregas de esta, tomando en cuenta el esfuerzo en el desarrollo y las fechas correspondientes.

Tabla 2.8 Plan de entregas creado por los autores

INTERFAZ	NRO	NOMBRE DE HISTORIA	ESFUERZO DE DESARROLLO			CALENDARIO ESTIMADO		ITERACION ASIGNADA				ENTREGA ASIGNADA			
			Semanas	Días	Horas	fecha inicio	fecha de fin	1	2	3	4	1	2	3	4
Explorador de archivos, Pantalla principal	1	Carga de archivo MIDI	2	10	40	25/05/2015	05/06/2015			x					x
Explorador de archivos, Pantalla principal	2	Guardar archivo MIDI	1	5	20	08/06/2015	12/06/2015			x					x
Pantalla principal	3	Reproducir archivo MIDI	0,8	4	16	08/06/2015	11/06/2015			x					x
Pantalla principal	4	Grabar secuencia	1	5	20	01/06/2015	05/06/2015			x					x
Pantalla principal	5	Visualización de información musical	1	5	20	25/05/2015	29/05/2015			x					x

2.1.5.4 Seguimiento de actividades

En esta sección se van monitorear las tareas asignadas a cada historia de usuario, estado del desarrollo, esfuerzo real de desarrollo y el esfuerzo por realizar.

Hay que tomar en cuenta todas las tareas por historia de usuario según el plan inicial del proyecto y los tiempos reales que tomó la realización de estas tareas.

La duración de una tarea se la va detallar en horas para mejor comprensión, también podrían especificarse en días, semanas o según como se lo defina en el proyecto. Para este caso, se ha decidido plantearlo en horas.

NRO	HISTORIA DE USUARIO	ITERACIÓN	TAREAS	ESTADO DE DESARROLLO	ESFUERZO ESTIMADO (HORAS)	ESFUERZO REAL INVERTIDO (HORAS)	FECHA INICIAL REAL	FECHA FINAL REAL
	Todas las historias	1	Recolección de requisitos	Completado	40	50	1-04-2015	17-04-2015
		1	Elaboración de historias de usuario	Completado	4	4	17-04-2015	17-04-2015
		1	Elaboración de historias técnicas	Completado	8	10	17-04-2015	21-04-2015
		1	Elaboración del plan de entregas	Completado	8	4	21-04-2015	21-04-2015
	Esfuerzos totales				60	68	1-04-2015	21-04-2015
		2	Diseño del modelo de dominio	Completado	4	4	22-04-2015	22-04-2015
1	Carga de archivo MIDI y Guardar archivo MIDI	2	Diseño y creación de la estructura de datos de un archivo MIDI	Completado	40	50	23-04-2015	11-05-2015
	Todas las historias	2	Diseño de las interfaces	Completado	20	25	11-05-2015	18-05-2015
5	Visualización de información musical	2	Diseño de las vistas	Completado	20	20	18-05-2015	24-05-2015
		Esfuerzos totales			84	99	22-04-2015	24-05-2015

Tabla 2.9 Historial de Seguimiento de tareas primera y segunda iteración creada por los autores

NRO	HISTORIA DE USUARIO	ITERACIÓN	TAREAS	ESTADO DE DESARROLLO	ESFUERZO ESTIMADO (HORAS)	ESFUERZO REAL INVERTIDO (HORAS)	FECHA INICIAL REAL	FECHA FINAL REAL
1	Carga de archivo MIDI	3	Representación de la estructura MIDI en entidades	Completado	8	8	25-05-2015	26-05-2015
1		3	Creación de un lector de archivos MIDI que sea capaz de cargar los datos del fichero en la estructura	Completado	16	20	27-05-2015	2-06-2015
1		3	Programación de la validación de ficheros MIDI	Completado	8	4	02-06-2015	02-06-2015
1		3	Creación de un intérprete de datos para convertir valores de longitud variable en enteros, conversor de tempo a milisegundos, etc.	Completado	8	8	03-06-2015	04-06-2015
3	Reproducción de archivos MIDI	3	Creación de un conversor de eventos MIDI a paquetes	Completado	8	8	05-06-2015	08-06-2015
		3	Creación del regularizador de volumen por canal MIDI	Completado	8	4	09-06-2015	09-06-2015
4	Grabación de secuencia MIDI	3	Programar la captura de paquetes de entrada	Completado	12	12	10-06-2015	12-06-2015
		3	Creación del conversor de paquetes a eventos	Completado	8	8	15-06-2015	16-06-2015
2	Guardar archivo MIDI	3	Creación inversa de un intérprete de datos inverso MIDI , que convierta un valor, entero a longitud variable, que pase el tempo a microsegundo, etc.	Completado	8	4	17-06-2015	17-06-2015
		3	Creación de un escritor de ficheros MIDI	Completado	12	12	18-06-2015	22-06-2015
5	Visualización de información musical	3	Creación del teclado virtual	Completado	4	4	23-06-2015	23-06-2015
		3	Programación de ajustes de teclado a diferentes escalas	Completado	8	4	24-06-2015	24-06-2015
		3	Creación del karaoke	Completado	8	8	25-06-2015	26-06-2015
		Esfuerzos totales			116	104	25-05-2015	26-06-2015

Tabla 2.10Historial de Seguimiento de tareas tercera iteración creada por los autores

NRO	HISTORIA DE USUARIO	ITERACIÓN	TAREAS	ESTADO DE DESARROLLO	ESFUERZO ESTIMADO (HORAS)	ESFUERZO REAL INVERTIDO (HORAS)	FECHA INICIAL REAL	FECHA FINAL REAL
	Todas las historias	4	Preparación de las pruebas unitarias	Completado	4	8	29-06-2015	30-06-2015
		4	Realización de pruebas unitarias	Completado	12	4	1-07-2015	1-07-2015
		4	Análisis de los resultados de las pruebas unitarias	Completado	4	4	2-07-2015	2-07-2015
		4	Preparación de las pruebas de integración	Completado	4	4	3-07-2015	3-07-2015
		4	Realización de las pruebas integración	Completado	4	2	6-07-2015	6-07-2015
		4	Análisis de los resultados de las pruebas de integración	Completado	4	4	6-07-2015	6-07-2015
		4	Diseño de Pruebas de Aceptación	Completado	4	4	7-07-2015	7-07-2015
		4	Realización de las pruebas de aceptación	Completado	8	8	8-07-2015	9-07-2015
		4	Análisis de los resultados de las pruebas de aceptación	Completado	8	8	10-07-2015	13-07-2015
		Esfuerzos totales			52	46	29-06-2015	13-07-2015

Tabla 2.11 Historial de Seguimiento de tareas cuarta iteración creada por los autores

2.2 FASE 2 DISEÑO.

2.2.1 TARJETAS CRC

Los diagramas de actividades se centran en mostrar un flujo de las actividades que se van a realizar en un sistema o el flujo de trabajo desde un punto inicial hasta un punto final.

MIDIFile	
------	------
Atributos	**Métodos**
CHUNK_ID	cargarMidi
CHUNK_SIZE	guardarMidi
Format	
trackCount	
timeDivisions	
Responsabilidades	**Colaboración**
Guarda un archivo midi convirtiendo los objetos de memoria en bytes	MIDITrack
Carga los archivos midi convirtiendo los bytes en objetos de memoria	MIDIEvent

Tabla 2.12 Tarjeta CRC MIDIFile elaborada por los autores

MIDITrack	
------	------
Atributos	**Métodos**
CHUNK_ID	setEvent
chunk_size	getCHUNK_ID
	getChunk_size
Responsabilidades	**Colaboración**
	MIDIEvent

Tabla 2.13 Tarjeta CRC MIDITrack elaborada por los autores

MIDIEvent	
------	------
Atributos	**Métodos**
deltaTime	getDeltaTime
event	setDeltaTime
Responsabilidades	**Colaboración**
Representar eventos midi	MIDIEvent
Poder leer y escribir valores de longitud variable.	

Tabla 2.14 Tarjeta CRC MIDIEvent elaborada por los autores

MIDIEventChannel	
------	MIDIEvent
Atributos	**Métodos**
parameter1	setMidiChannel
parameter2	quitChannel
event	getMidiChannel
Responsabilidades	**Colaboración**
Representar eventos de canal midi	
Extraer y asignar un canal midi	

Tabla 2.15 Tarjeta CRC MIDIEventChannel elaborada por los autores

MIDIEventMeta	
------	MIDIEvent
Atributos	**Métodos**
Type	getLength
Length	setLength
Data	setData
Event	
Responsabilidades	**Colaboración**

Tabla 2.16 Tarjeta CRC MIDIEventMeta elaborada por los autores

MIDIEventSysCommon	
------	MIDIEvent
Atributos	**Métodos**
param1	getParam1
param2	setParam1
event	getParam2
	setParam2
Responsabilidades	**Colaboración**

Tabla 2.17 Tarjeta CRC MIDIEventSysCommon elaborada por los autores

MIDIEventSysEx	
------	MIDIEvent
Atributos	**Métodos**
lenght	getLenght
message	setLenght
event	getMessage
	setMessage
Responsabilidades	**Colaboración**

Tabla 2.18 Tarjeta CRC MIDIEventSysEx elaborada por los autores

MIDIEventSysRealTime	
------	MIDIEvent
Atributos	**Métodos**
lenght	getLenght
message	setLenght
	getMessage
	setMessage
Responsabilidades	**Colaboración**

Tabla 2.19 Tarjeta CRC MIDIEventSysRealTime elaborada por los autores

MIDIPackage	
------	MIDIEvent
Atributos	**Métodos**
deltaTime	getDeltaTime
data	setDeltaTime
	getData
	setData
Responsabilidades	**Colaboración**

Tabla 2.20 Tarjeta CRC MIDIPackage elaborada por los autores

MidiUSBPackageGenerator	
------	-------
Atributos	**Métodos**
	generateMidiPackages
	generateMidiFile
	crearPaquetesSilenciarNotas
Responsabilidades	**Colaboración**
Generar paquetes midi listo a ser enviados a un dispositivo midi a partir de un archivo midi	MidiUtilities
Generar un archivo midi a partir de paquetes	MIDIPackage
Poder silenciar el dispositivo midi	

Tabla 2.21 Tarjeta CRC MIDIUSBPackageGenerator elaborada por los autores

MidiUtilities	
------	-------
Atributos	**Métodos**
	getBPMFromMidiTempo
	convertDeltaTimeToMiliSeconds
	convertMiliSecondsToDeltaTime
	getLyiricsFromMidiFile
Responsabilidades	**Colaboración**
Contiene métodos de apoyo a la generación de paquetes MIDI y a la decodificación general del tiempo	

Tabla 2.22 Tarjeta CRC MIDIUtilities elaborada por los autores

2.2.2 DIAGRAMA DE CLASES

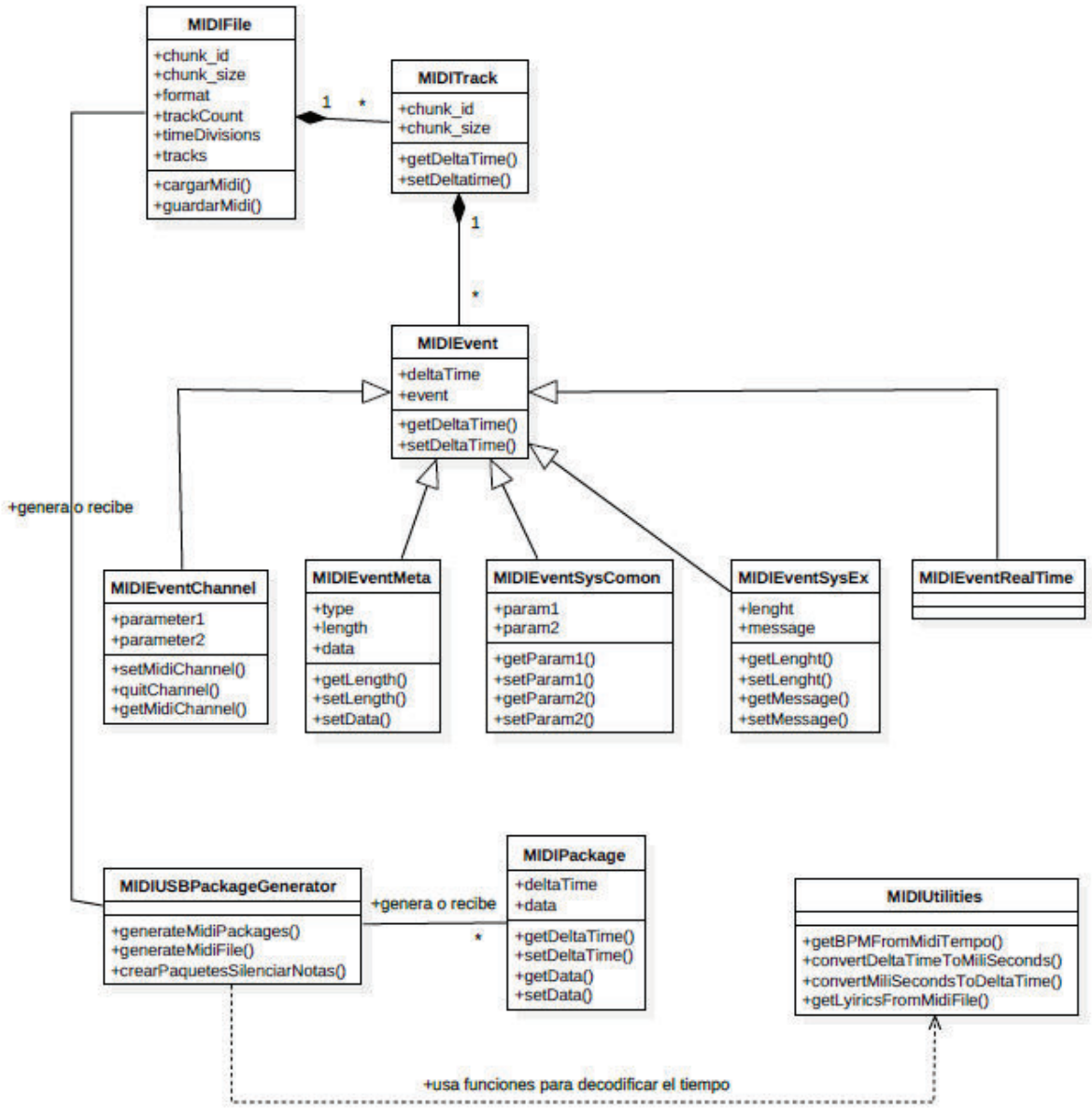


Figura 2.6 Diagrama de clases elaborado por los autores

2.2.3 DIAGRAMA DE ACTIVIDADES

Los diagramas de actividades se centran en mostrar un flujo de las actividades que se van a realizar en un sistema o el flujo de trabajo desde un punto inicial hasta un punto final.

Entre los elementos en este tipo de diagramas existen:

Nodo inicial: Es el nodo donde comienza la actividad y se representa con un punto negro.

Figura 2.7 Nodo inicial en un diagrama de actividades elaborado por los autores

Nodo final: Es el nodo don termina una actividad y se representa de la siguiente manera:

Figura 2.8 Nodo final en un diagrama de actividades elaborado por los autores

Acción: Es un solo paso dentro de una actividad, se representa con un rectángulo con puntas redondeadas.

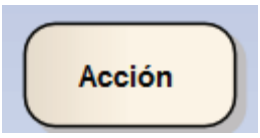


Figura 2.9 Acción en un diagrama de actividades elaborado por los autores

Flujo de control: Muestra el flujo de una acción a otra

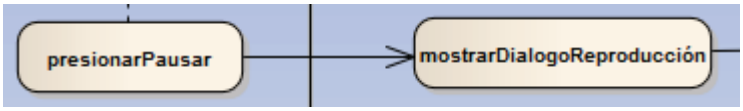


Figura 2.10 Flujo de control en un diagrama de actividades elaborado por los autores

Objetos: Un objeto se muestra como un rectángulo

Figura 2.11 Objeto en un diagrama de actividades elaborado por los autores

Restricción de acción: Se puede adjuntar a una acción, puede ser pre o post condición.

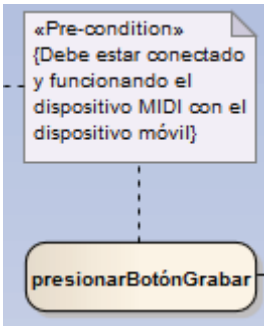


Figura 2.12 Restricción en un diagrama de actividades elaborado por los autores

Nodos de decisión: Tiene forma de diamante y nos ayuda a representar decisiones

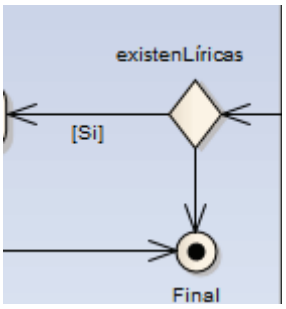


Figura 2.13 Nodo de decisión en diagrama de actividades elaborado por los autores

Nodos de unión: se representan con una barra que puede ser horizontal o vertical, indica el comienzo o final de varios hilos de control.

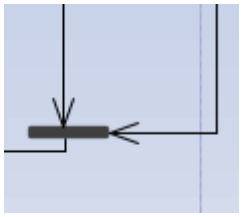


Figura 2.14 Nodo de unión en un diagrama de actividades elaborado por los autores

A continuación los diagramas de actividades de la aplicación:

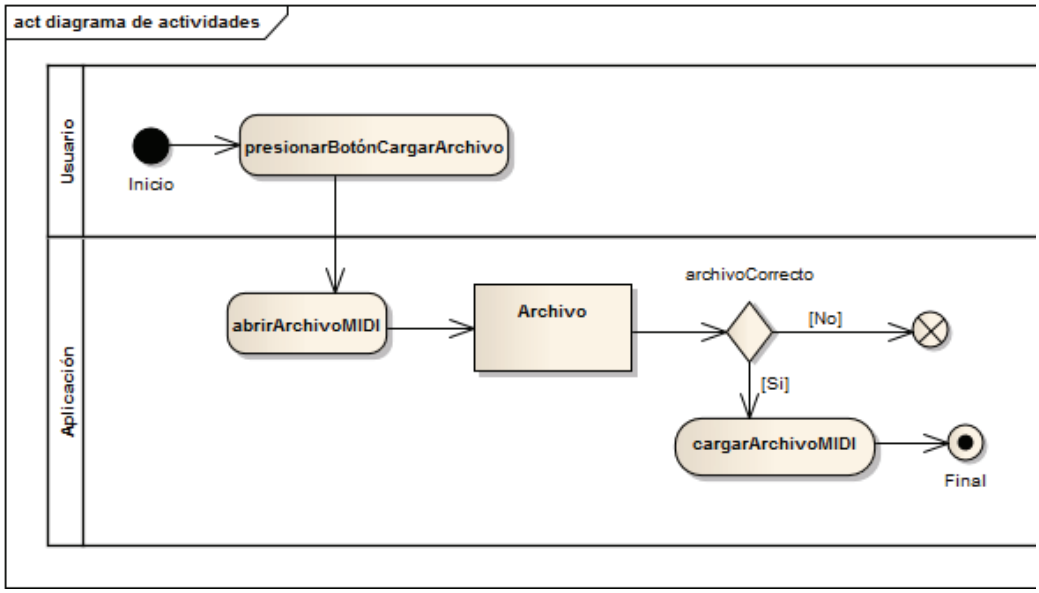


Figura 2.15 Diagrama de actividad Cargar Archivo MIDI elaborado por los autores

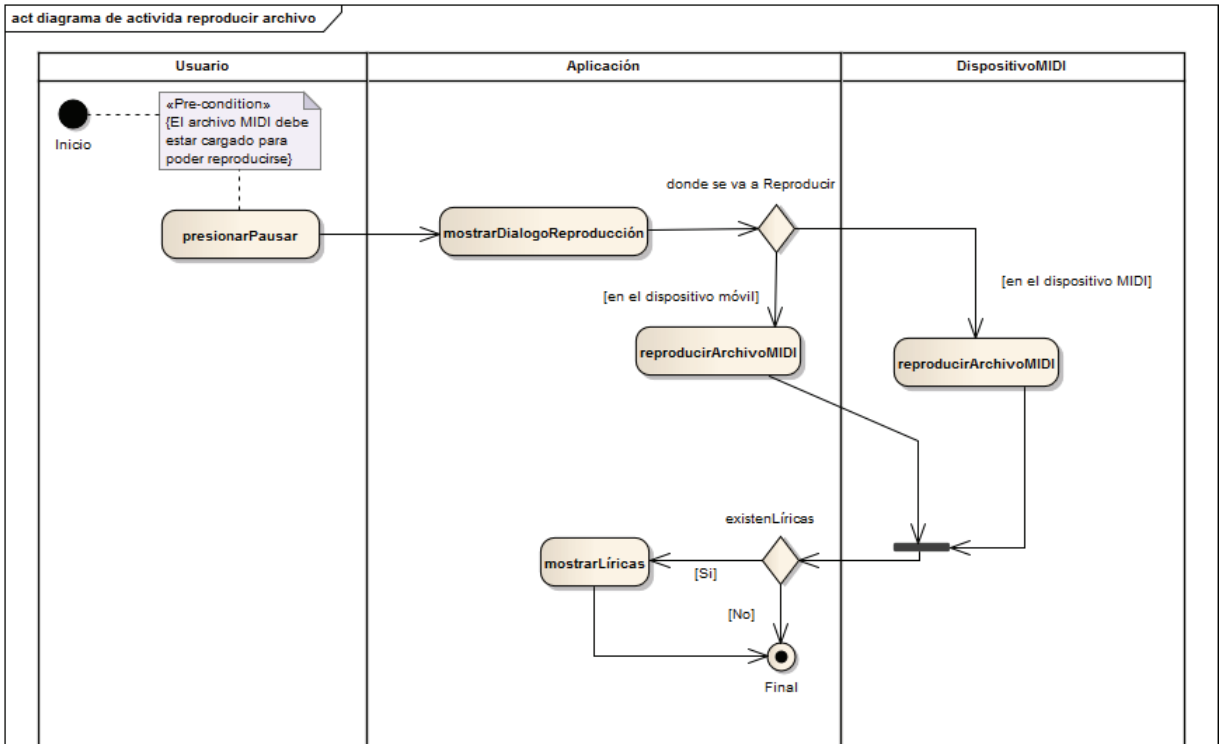


Figura 2.16 Diagrama de actividad Reproducción archivo MIDI elaborado por los autores

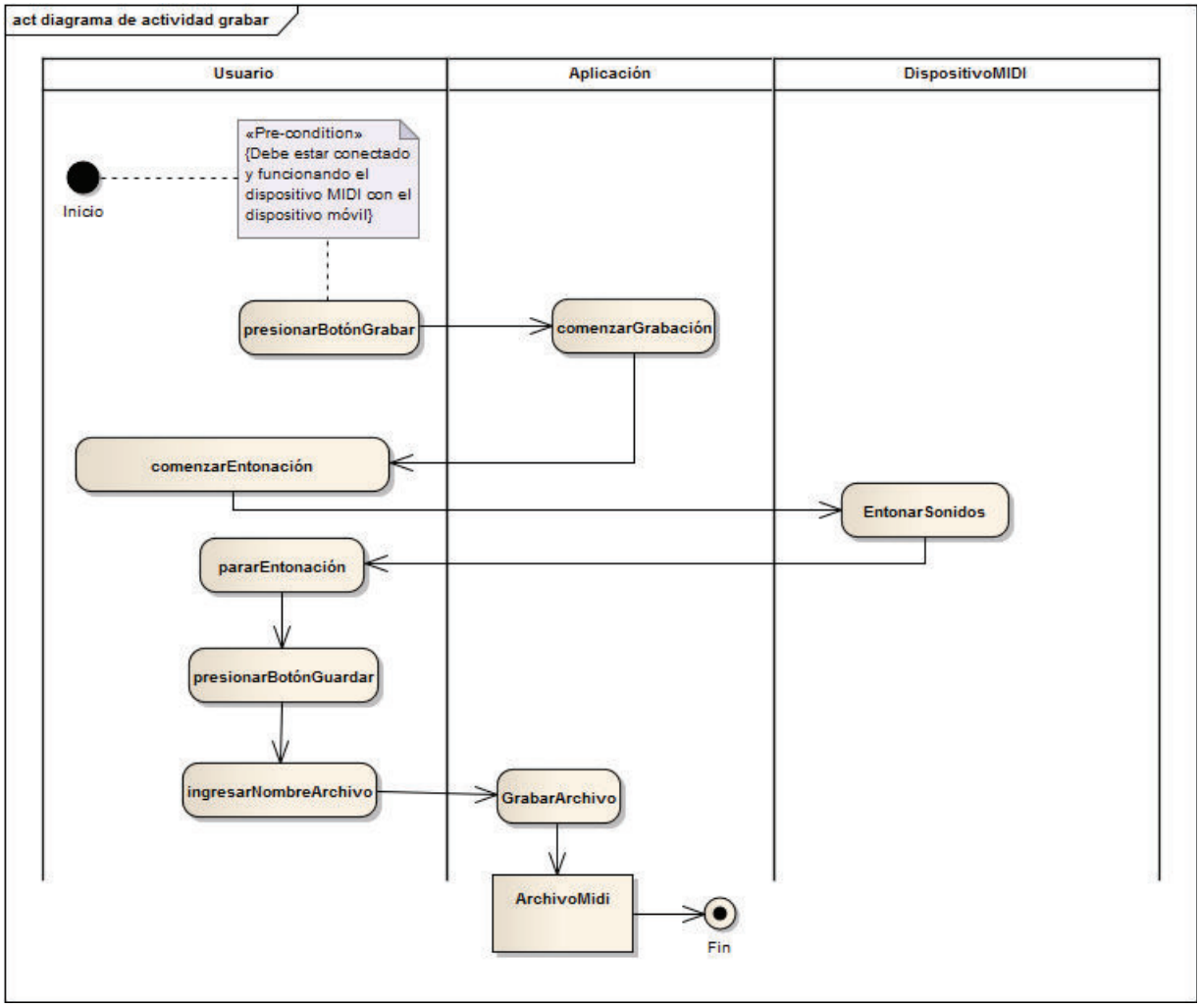


Figura 2.17 Diagrama de actividad Grabar archivo MIDI elaborado por los autores

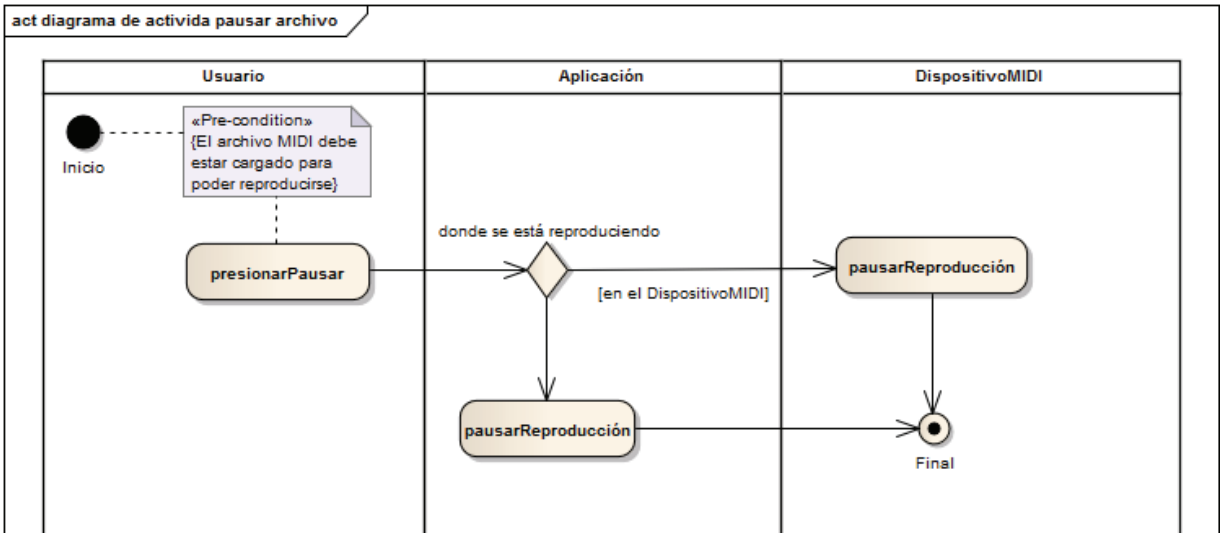


Figura 2.18 diagrama de actividad Pausar archivo MIDI elaborado por los autores

2.2.4 DIAGRAMA DE PAQUETES

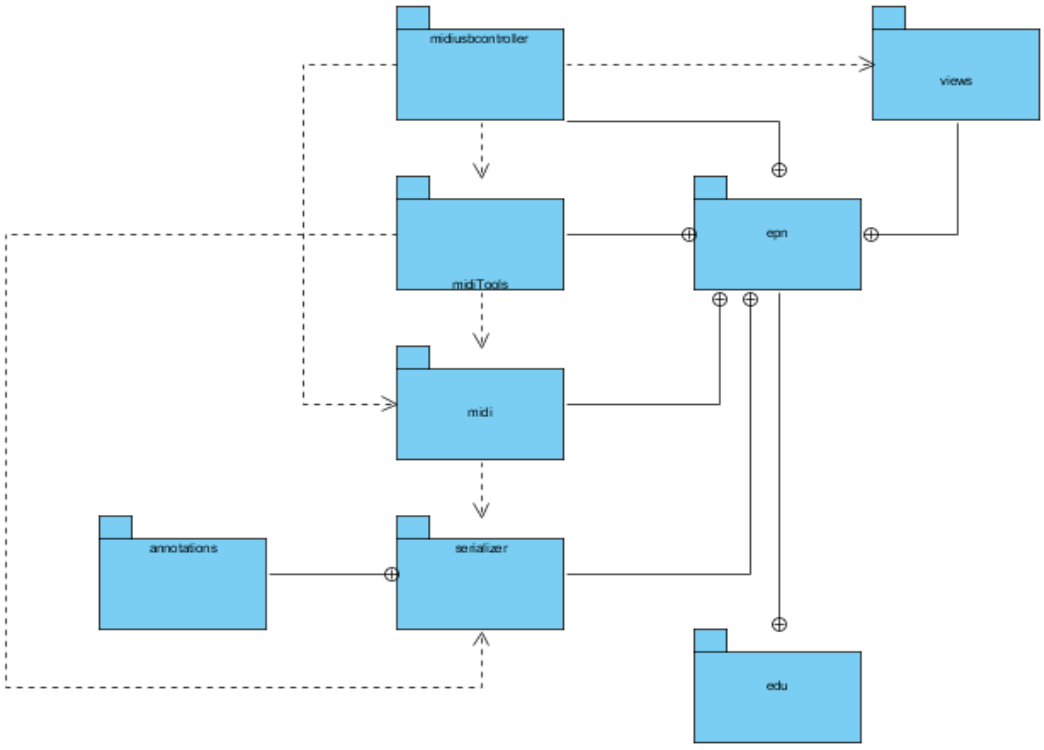


Figura 2.19 Diagramas de paquetes creado por los autores

La aplicación consta de los siguientes paquetes:

- **edu.epn.midiusbcontroller:**
  Este paquete contiene el controlador y los Activities de la aplicación.
- **edu.epn.views:**
  Este paquete contiene las vistas de la aplicación, las cuales son llamadas y manejadas por los activities.
- **edu.epn.midi:**
  Este paquete contiene las clases necesarias para almacenar un archivo midi en memoria, y para poder guardar y cargar un archivo midi.
- **edu.epn.midiTools:**
  Este paquete contiene clases que realizan operaciones sobre las clases que representan un archivo midi.
- **edu.epn.serializer:**
  Este paquete contiene clases para convertir las clases contenidas dentro del paquete midi a bytes y los bytes en clases del paquete midi.
- **edu.epn.tests:**
  Este paquete contiene las clases necesarias para realizar pruebas unitarias al programa.

2.2.5 DIAGRAMA DE COMPONENTES

Los diagramas de componentes ilustran piezas de software que conforman un sistema; representa las relaciones y dependencias entre los componentes que van a existir para dar solución a un problema. Un diagrama de componentes es de más alto nivel que un diagrama de clases y puede contener en su representación una o varias clases. [39][40] y [41]

Entre los elementos que existen en un diagrama de clases según las guías de UML 2 están:

Componente: Representa un empaquetamiento físico de una parte lógica (clases o vistas) Se representan de la siguiente manera:

Figura 2.20 Componente en diagrama de componentes de la aplicación elaborado por los autores

Interfaz: Tipo de clasificador que representa una declaración de un conjunto de funciones, especifica un contrato en este caso entre componentes.

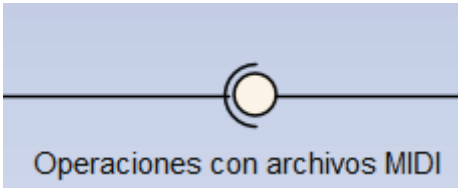


Figura 2.21 Interfaz en diagrama de componentes de la aplicación elaborado por los autores

A continuación el diagrama de componentes para la aplicación

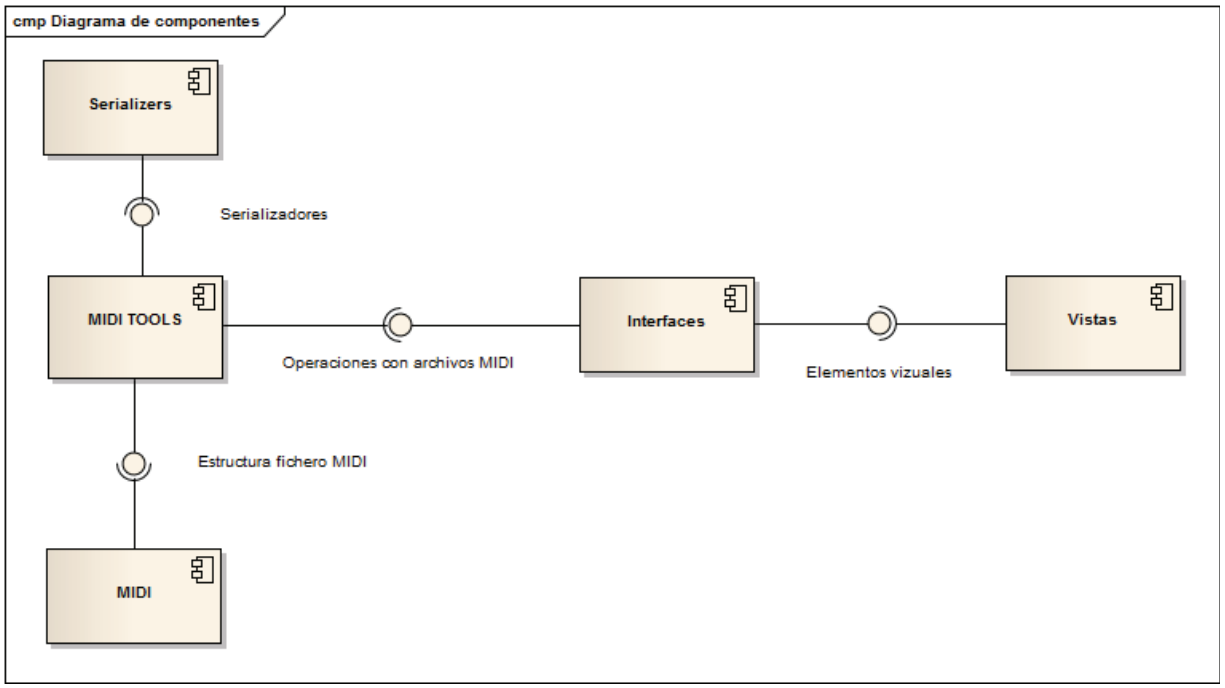


Figura 2.22 Diagrama de componentes de la aplicación elaborado por los autores

En el diagrama se muestran que los principales componentes de la aplicación. Comenzando con el componente de serializadores, éste incluye clases para transformar los objetos que conforman el archivo MIDI que se encuentran dentro del componente MIDI en bytes que van a ser transmitidos al dispositivo MIDI y viceversa. Se puede ver también el componente de las herramientas MIDI el cual contiene las clases para realizar las operaciones de las funcionalidades previstas a realizarse mediante la aplicación. Otro componente que existe es el componente de interfaces, básicamente contiene las actividades principales de la aplicación, este componente se relaciona en buena manera con las vistas que son acciones que se van a mostrar en forma de diálogos, como ejemplos hay: el dialogo para cambiar el tempo, el dialogo para modificar el volumen de cada canal del archivo MIDI, entre otros.

2.2.6 DIAGRAMA DE DESPLIEGUE

Los diagramas de despliegue representan la configuración de los nodos o el funcionamiento en tiempo real del sistema, en este diagrama se trazan los elementos de hardware y los artefactos de software.

Entre los elementos en un diagrama de despliegue están:

Nodo: Elemento de hardware o software se muestra como una caja en tres dimensiones.

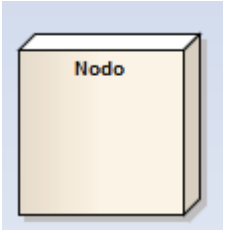


Figura 2.23 Nodo en un diagrama de despliegue elaborado por los autores

Artefacto: Es un producto de desarrollo de software, puede incluir los modelos del proceso, los casos de uso, los modelos de diseño, ejecutables, manuales y más

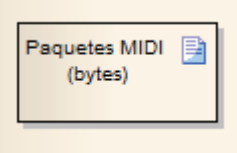


Figura 2.24 Artefacto en un diagrama de despliegue elaborado por los autores

Asociación: Representa una ruta de comunicación entre nodos.

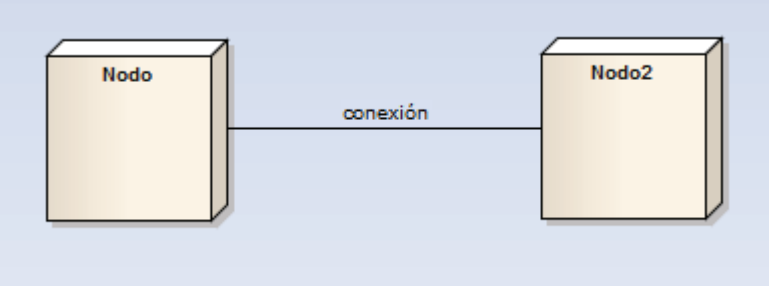


Figura 2.25 Asociación en un diagrama de despliegue elaborado por los autores

A continuación el diagrama de despliegue de la aplicación:

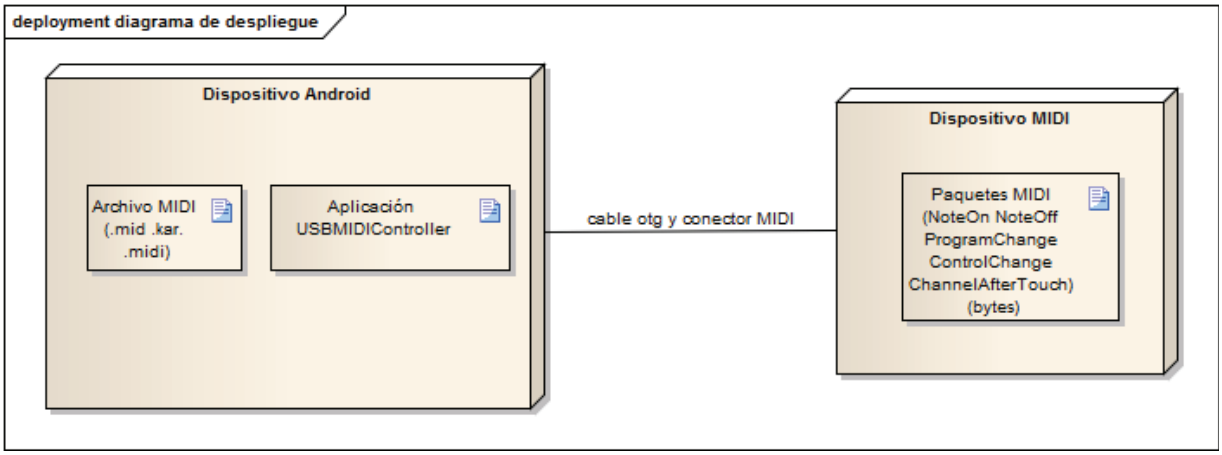


Figura 2.26 diagrama de despliegue elaborado por los autores

En el diagrama se ve que van a existir dos elementos principales o nodos, el dispositivo Android y el dispositivo MIDI, estos son los dos elementos de hardware que van a comunicarse mediante el cable otg[4] y un cable MIDI. En el dispositivo Android se van a realizar todas las operaciones mediante la aplicación USBMIDIController, ésta va a procesar los archivos MIDI para luego en la reproducción enviar como paquetes el archivo MIDI en formato de bytes que contienen la siguiente información: encendido de nota, apagado de nota, cambio de programa, cambio de control y canal después de tocar. En el caso de la grabación el dispositivo MIDI envía los paquetes del archivo MIDI, los cuales son receptados en el dispositivo Android para luego en la aplicación procesarlos y generar el archivo MIDI que se va a almacenar en el dispositivo Android.

[4] Extensión de la norma USB 2.0 que permite a los dispositivos USB tener mayor flexibilidad en la gestión de la interconexión, permite a los dispositivos actuar como host y permitir la conexión con dispositivos como memorias USB, entre otros.

2.2.1 PATRON DE DISEÑO DE LA APLICACIÓN

2.2.2 PROTOTIPOS DE LAS INTERFACES DEL SISTEMA

- Interfaz principal que permite cargar archivos MIDI, guardar archivos MIDI, reproducir vía USB de archivos MIDI y grabación USB de archivos MIDI, visualización de teclado virtual y letras.

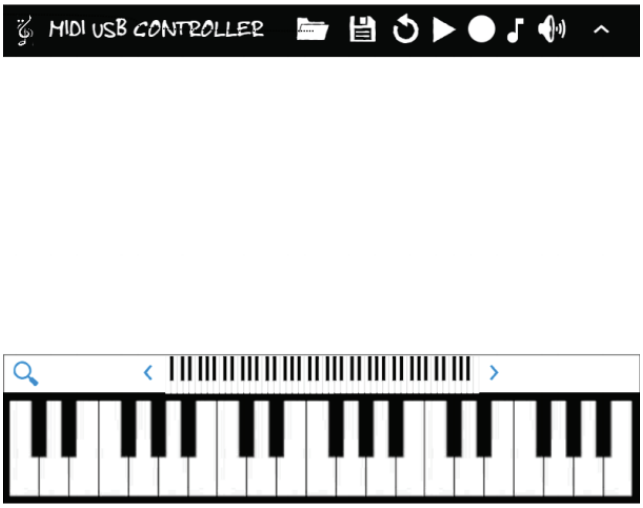


Figura 2.27 Interfaz principal del sistema creada por los autores

- Explorador de archivos MIDI

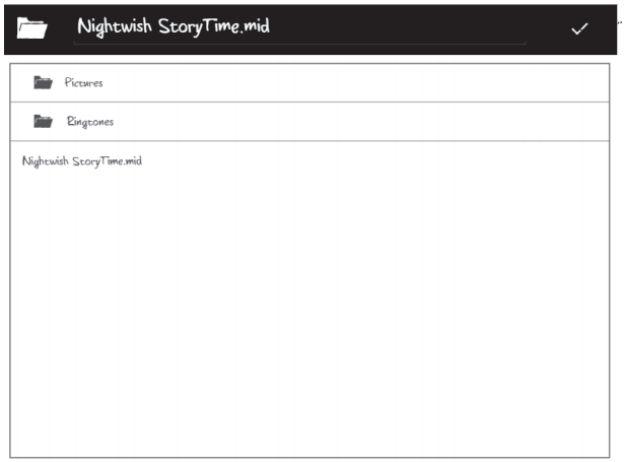


Figura 2.28 Explorador de archivos creado por los autores

- Vista control de volumen

Figura 2.29 Vista de control de volumen elaborado por los autores

- Vista tamaño de teclado

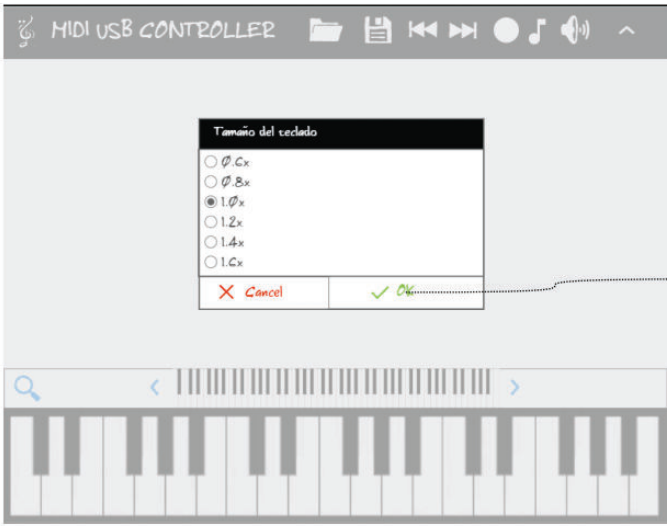


Figura 2.30 Vista tamaño del teclado elaborado por los autores

- Vista ajuste de tempo

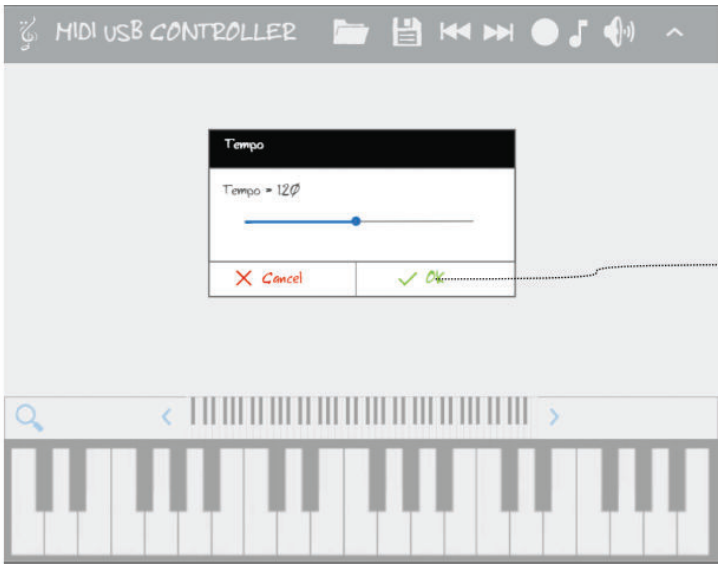


Figura 2.31 Explorador de archivos elaborado por los autores

- Visualización del karaoke

Figura 2.32 Explorador de archivos creado por los autores

2.2.3 ESTÁNDARES DE PROGRAMACIÓN

Durante el desarrollo de esta aplicación se han tomado en consideración las convenciones de código de java las cuales se consideran importantes por los beneficios que traen. Entre ellos se tiene:

- Mejorar la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo.
- Al momento de distribuir el código fuente como un producto, es necesario asegurarse de que está bien hecho y presentado como cualquier otro producto.
- Casi ningún software lo mantiene toda su vida el autor original, por lo que este debe ser legible para otros programadores.

A continuación se mostrarán las convenciones de código que cumple la aplicación con ejemplos de código:

- Cuando una expresión no entre en una línea, se deberá romper la línea después de una coma o antes de un operador [35].

```
buttonOk.setOnClickListener((arg0) -> {
        Intent resultadoIntent = new Intent();
        resultadoIntent.putExtra("directorio",
                directorioActual.getAbsolutePath()+File.separator+editTextRutaActual.getText().toString());
        setResult(RESULT_OK,resultadoIntent);
        finish();
});
```

```
else if ((eventoCanal.getParameter1() == MIDIConstants.MSBBankSelect)
        ||(eventoCanal.getParameter1() == MIDIConstants.LSBBankSelect)){
```

Figura 2.33 Ejemplo estándar de programación, tamaño de línea creada por autores

- Se recomienda declarar una variable por línea ya que facilita los comentarios:

```
private boolean playUSB = true;
private boolean grabandoUSB = false;
private boolean reproduciendoUSB = false;
private int tempo = 120;
private int indexPackage = 0;   //Indice del paquete dentro de la lista de paquetes a ser enviados via USB
```

Figura 2.34 Ejemplo estándar de programación, variable por línea, referencia: Autores

- No debe haber ningún espacio en blanco entre el nombre de un método y el paréntesis que inicia la lista de argumentos que recibe

```
private void cargarArchivoMidi(String directorio){
    try {
        File file = new File(directorio);
        if (!file.exists()){
```

Figura 2.35 Ejemplo estándar de programación, espacio entre nombre de un método y paréntesis, referencia: Autores

```
public void loadMidi(DataInputStream stream) throws IOException{
    BaseSerializer serializer = new BaseSerializer();
    readHeader(stream,serializer);
    this.setTrack(new ArrayList<MIDITrack>());
    while(stream.available() > 8){
        this.getTrack().add(readTrack(stream,serializer));
    }
}
```

Figura 2.36 Ejemplo estándar de programación, espacio entre nombre de un método y paréntesis 2, referencia: Autores

- La apertura de llaves debe estar al final de la misma línea de declaración de un método y no en una línea aparte

```
private void AlertTempo(){
    LinearLayout lyTempo = new LinearLayout(this);
    lyTempo.setOrientation(LinearLayout.VERTICAL);
    final TextView txtTempo = new TextView(this);
    txtTempo.setText("Tempo = "+tempo);
```

Figura 2.37 Ejemplo posición apertura de llaves, referencia: Autores

```
public Object getTimeDivisions() {
    if (timeDivisions >= 0){
        return timeDivisions;
    }
```

Figura 2.38 Ejemplo posición apertura de llaves 2, referencia: Autores

- El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel, actualmente com, edu, gov, mil, net, org, o uno de los códigos ingleses de dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981.

Los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada organización. Dichas convenciones pueden especificar que algunos nombres de los directorios correspondan a divisiones, departamentos, proyectos o máquinas [35].

En los paquetes que usa la aplicación desarrollada están los siguientes:

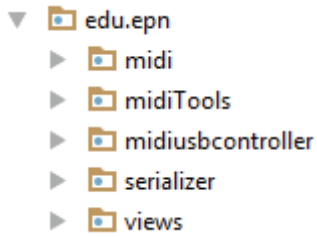


Figura 2.39 Ejemplo de paquetes, referencia: Autores

- Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL or HTML).

  Los nombres de las clases usados en el desarrollo de este software se muestran a continuación [35]:

  - FramesPerSecond
  - MIDIConstants
  - MIDIEvent
  - MIDIEventChannel
  - MIDIEventMeta
  - MIDIEventSysCommon
  - MIDIEventSysEx
  - MIDIEventSysRealTime
  - MIDIFile
  - MIDIPackage
  - MIDITrack
  - MidiUSBPackageGenerator

- MidiUtilities
- FileChooserActivity
- MainActivity
- BaseSerializer
- Cloner
- FileValidationException
- InstrumentsView
- KaraokeView
- MiniInstrumentsView

- Los nombres de las interfaces siguen la misma regla que las clases.

  Este programa tiene una sola interfaz la cual cumple con los nombramientos de las clases, la cual es la siguiente:

  - Order

- Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula[35]

```
public void loadMidi(DataInputStream stream) throws IOException{
    BaseSerializer serializer = new BaseSerializer();
    readHeader(stream,serializer);
```

Figura 2.40 Ejemplo nombre de método, referencia: Autores

```
private void readHeader(DataInputStream stream,BaseSerializer serializer) throws IOException{
    if (!serializer.readString(stream, 4).equals(chunk_ID)){
        throw new FileValidationException("Invalid chunk_ID of midi file");
```

Figura 2.41 Ejemplo nombre de método 2, referencia: Autores

- Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son

compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje [35].

A continuación algunas variables usadas dentro del programa:

```
public int idIcono;
//Constantes
private String parentDirectory;
//Componentes de interfaz
private ListView listViewDirectorios;
private EditText editTextRutaActual;
private TextView textViewExtension;
```

Figura 2.42 Ejemplo de variables, referencia: Autores

- Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas separando las palabras con un subguión ("_"). (Las constantes ANSI se deben evitar, para facilitar su depuración.)

```
public static final short MIDI_FORMAT_SINGLE_TRACK = 0;
public static final short MIDI_FORMAT_MULTIPLE_TRACK = 1;
public static final short MIDI_FORMAT_MULTIPLE_SONG = 2;
//////////////////////////////Meta Evenets////////////////////////////////////
public static final byte META_EVENT = (byte)255;
public static final byte META_EVENT_SEQUENCE_NUMBER = 0;
public static final byte META_EVENT_TEXT = 1;
public static final byte META_EVENT_COPYRIGHT_NOTICE = 2;
public static final byte META_EVENT_TRACK_NAME = 3;
public static final byte META_EVENT_INSTRUMENT_NAME = 4;
```

Figura 2.43 Ejemplo de constantes, referencia: Autores

- Todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase, información de la versión, fecha, y copyright:

```
/*
 * MIDIFile: Clase creada para cargar ficheros midi en una estructura de datos y para
 * generar ficheros midi a partir de la estructura de datos
 * 01/05/2015
 * @version 1.0
 * @authores Eduardo Anies, David Proaño
 */
@Order(fields = {"CHUNK_ID","CHUNK_SIZE","format","trackCount","timeDivisions","track"})
public class MIDIFile{
   private final String CHUNK_ID = "MThd";
   private final Integer CHUNK_SIZE = 6;
   private Short format;
```

Figura 2.44 Comentario en ficheros fuente referencia autores

- La primera línea no-comentario de los ficheros fuente Java es la sentencia package. Después de esta, pueden seguir varias sentencias import.

```
package edu.epn.midi;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.ArrayList;
```

Figura 2.45 Sentencia inicial, referencia: Autores

- Los ficheros de más de 2000 líneas son incómodos y deben ser evitados. En el presente proyecto las clases contienen el siguiente número de líneas de código:

  - FramesPerSecond: 35 líneas
  - MIDIConstants: 464 líneas
  - MIDIEvent: 71 líneas
  - MIDIEventChannel: 76 líneas
  - MIDIEventMeta: 56 líneas

- MIDIEventSysCommon: 41 líneas
- MIDIEventSysEx: 46 líneas
- MIDIEventSysRealTime: 22 líneas
- MIDIFile: 273 líneas
- MIDIPackage: 30 líneas
- MIDITrack: 37 líneas
- MidiUSBPackageGenerator: 206 líneas
- MidiUtilities: 146 líneas
- FileChooserActivity: 190 líneas
- MainActivity: 921 líneas
- BaseSerializer: 167 líneas
- Order: 21 líneas
- BaseSerializer: 167 líneas
- Cloner: 135 líneas
- FileValidationException: 20 líneas
- InstrumentsView: 130 líneas
- KaraokeView: 130 líneas
- MiniInstrumentsView: 112 líneas

Como se puede ver anteriormente ninguna clase supera ni llega a las 2000 líneas de código [35].

- La clase de sentencias if-else debe tener la siguiente forma:

```
If (condición) {
    sentencia;
} else if (condición) {
    sentencia;
} else {
    sentencia;
}
```

```
if (metaEvent.getData() != null){
    metaEvent.setLength(metaEvent.getData().length());
} else{
    metaEvent.setLength(0);
}
```

Figura 2.46 Ejemplo sentencia if else, referencia: Autores

```
if ((event == MIDIConstants.SYS_COMMON_MIDI_TIME_CODE_QUARTER_FRAME)
        || (event == MIDIConstants.SYS_COMMON_SONG_SELECT)){
    sysCommonEvent.setParam1(stream.readByte());
} else if (event == MIDIConstants.SYS_COMMON_SONG_POSITION_POINTER){
    sysCommonEvent.setParam1(stream.readByte());
    sysCommonEvent.setParam2(stream.readByte());
}
```

Figura 2.47 Ejemplo sentencia if else 2, referencia: Autores

- Una sentencia while debe tener la siguiente forma:

While (condición) {

sentencias;

}

```
while(stream.available() > 8){
    this.getTrack().add(readTrack(stream,serializer));
}
```

Figura 2.48 Ejemplo sentencia while, referencia: Autores

2.2.4 Prácticas de programación XP aplicadas a este proyecto

- **Simplicidad**

Para mantener un diseño simple primero se crearon entidades que representan la forma en la que se almacenan los datos dentro de un fichero MIDI.

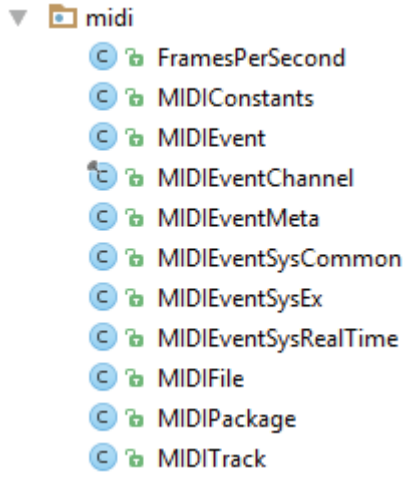


Figura 2.49 Ejemplo de simplicidad en ficheros, referencia: Autores

Se consideró que la forma más sencilla para realizar la escritura y lectura de ficheros MIDI sería mediante el uso de algoritmos que lean las variables de cada clase y las escriban, así se hizo uso de reflexión para no preocuparse por el número de variables que existan:

```
public void writeObject(Object object,DataOutputStream output) throws IllegalArgumentException,
    if (cloner.isPrimitiveType(object.getClass())) {
        writePrimitive(object, output);
    }
    else{
        List<Field> fields = getFieldsOrder(object.getClass());
        for (Field field : fields) {
            field.setAccessible(true);
            Object objField = field.get(object);
            if (objField != null) {
                if (field.getType() == List.class) {
                    List<Object> objects = (ArrayList<Object>) objField;
                    for (Object object1 : objects) {
                        writeObject(object1, output);
                    }
                } else {
                    writeObject(objField, output);
                }
            }
        }
    }
}
```

Figura 2.50 Ejemplo de reflexión en métodos referencia: Autores

Se separó las vistas de las pantallas para que el código sea más comprensible a simple vista

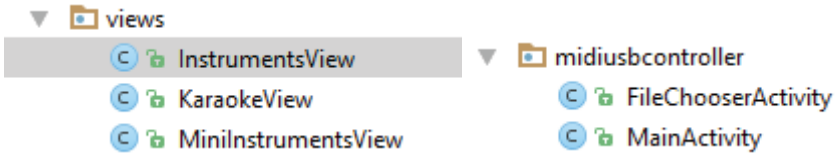


Figura 2.51 Separación de vistas y pantallas, referencia: Autores

- **Refactorización**

Se ha realizado refactorización durante todo el proyecto con el objetivo de tener un código simple, a continuación algunos ejemplos:

Para obtener el canal de un evento de canal MIDI se debe obtener el valor correspondiente a los 4 últimos bits menos significativos del evento, inicialmente se pensó lograr dicho objetivo de la siguiente manera:

```
public byte getMidiChannel(){
    return (byte)(MIDIConstants.getPositiveByteValue(this.event)%240);
}
```

Figura 2.52 Ejemplo de refactorización código antes referencia: Autores

En el código mostrado anteriormente se transforma el valor del byte de negativo a positivo, y luego se saca el residuo para 240, el cual sería un procedimiento válido para obtener el valor correspondiente a los 4 bits menos significativos, sin embargo una forma más sencilla de hacerlo es la siguiente:

```
public byte getMidiChannel(){
    return (byte)(this.event & (byte)15);
}
```

Figura 2.53 Ejemplo de refactorización código después referencia: Autores

En el código mostrado anteriormente se realiza un AND lógico con 15 cuyo valor binario es 00001111, para de esta forma obtener el valor de 0 en los 4 bits más significativos, se recuerda que:

- P & 0 = 0
- P & 1 = P

Entonces así en el caso de realizar un AND bit a bit entre 15 y un byte cualquiera el resultado fuera el siguiente:

bbbbbbbb (número cualquiera donde b es un bit cualquiera cuyo valor podría ser 0 o 1)
& 00001111 (15 en binario)

0000bbbb (Como resultado se puede ver)

Otro ejemplo de refactorización es en el siguiente código en el que inicialmente se retornaba en valor correspondiente al CIN (Code index number) según explica el documento “USB DEVICE CLASS DEFINITION FOR MIDI DEVICES”.

```
private byte getCableIndex1(byte event){
    if (event == MIDIConstants.CHANNEL_EVENT_NOTE_OFF){
        return 8;
    }
    if (event == MIDIConstants.CHANNEL_EVENT_NOTE_ON){
        return 9;
    }
    if (event == MIDIConstants.CHANNEL_EVENT_NOTE_AFTERTOUCH){
        return 10;
    }
    if (event == MIDIConstants.CHANNEL_EVENT_CONTROLLER){
        return 11;
    }
    if (event == MIDIConstants.CHANNEL_EVENT_PROGRAM_CHANGE){
        return 12;
    }
    if (event == MIDIConstants.CHANNEL_EVENT_CHANNEL_AFTERTOUCH){
        return 13;
    }
    if (event == MIDIConstants.CHANNEL_EVENT_PITCH_BEND){
        return 14;
    }
    return -1;
}
```

Figura 2.54 Ejemplo de refactorización código antes 2 referencia: Autores

Sin embargo revisando un poco el protocolo MIDI se puede observar que existe una relación directa entre el evento y el CIN (Code Index) y el evento, pues simplemente se requiere dividir el evento para 16 y se obtendrá el valor correspondiente al evento MIDI, por lo cual se cambió el código mostrado anteriormente por el siguiente.

```
private byte getCableIndex(byte event){
    return (byte)(MIDIConstants.getPositiveByteValue(event)/16);
}
```

Figura 2.55 Ejemplo de refactorización código después 2 referencia: Autores

2.3 FASE 3 DESARROLLO.

2.3.1 PATRÓN DE DISEÑO

El patrón de diseño utilizado para el desarrollo de la aplicación fue el patrón MVC.

2.3.1.1 Modelo Vista Controlador

El famoso patrón de diseño MVC fue propuesto por Trygve Reenskaug en el año de 1979 y ayuda a dividir el acceso a datos y la lógica de negocio de la forma en la que se muestra al usuario final.

MVC propone dividir en 3 elementos:

Modelo:

El modelo representa los datos y las reglas que gobiernan el acceso y actualización de dichos datos.

Vista:

Presenta el modelo y la lógica de negocio en la interfaz de usuario, por tanto si es que el modelo cambia la vista deberá actualizar la información que se requiera.

Controlador:

El controlador se encarga de traducir las interacciones del usuario con la vista en acciones que le modelo debe ejecutar.

2.3.1.2 Implementación del modelo MVC

Modelo:

El modelo en esta aplicación viene a ser los datos que representan un archivo midi contenidos dentro del paquete midi

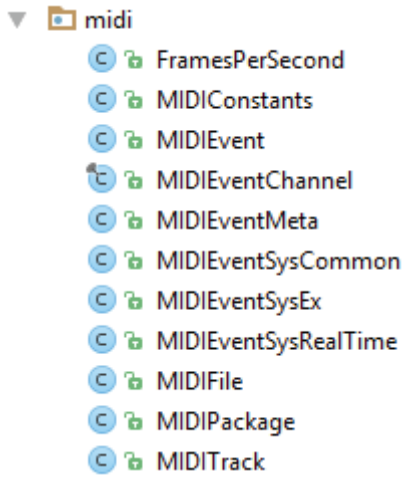


Figura 2.56 paquete MIDI elaborado por los autores

Y la lógica de negocio que se encuentra dentro de los paquetes midiTools y serializer.

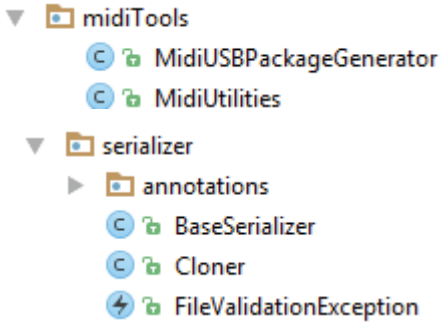


Figura 2.57 Paquetes midiTools y paquete serializer elaborado por los autores

Vista

Se tienen 3 clases que extienden de la clase View, que presentan la información midi

Figura 2.58 paquete Vistas elaborado por los autores

```
/*
 * MiniInstrumentsView: Clase creada para visualizar desde una perspecti
 * cualquier instrumento musical, ejemplo: piano
 * 24/05/2015
 * @version 1.0
 * @authores Eduardo Anies, David Proaño
 */
public class MiniInstrumentsView extends View {
    //Vista la cual va a controlar esta vista
    public InstrumentsView vistanInstruments;
    //Variables para graficar vista y capturar eventos touch
    private int incrementoScrollX;
    private float escala_width;
    private float rectanguloVisible;
    private float desp;
    private Paint paintTransparente = new Paint();
    private Paint paintFlechas = new Paint();
    //Imagenes de flechas
    private Bitmap flecha = BitmapFactory.decodeResource(getResources(),

    public MiniInstrumentsView(Context context, AttributeSet attrs) {
```

Figura 2.59 ejemplo de vista creado por los autores

Estas 3 vistas se encuentran dentro de un Activity llamado “MainActivity”, y adicionalmente se tiene otro Activity llamado “FileChooser” para navegar a través del sdcard del dispositivo Android para guardar y cargar archivos midi.

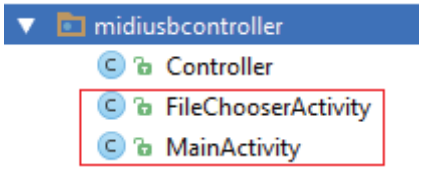


Figura 2.60 paquete midiusbcontroller elaboradopor los autores

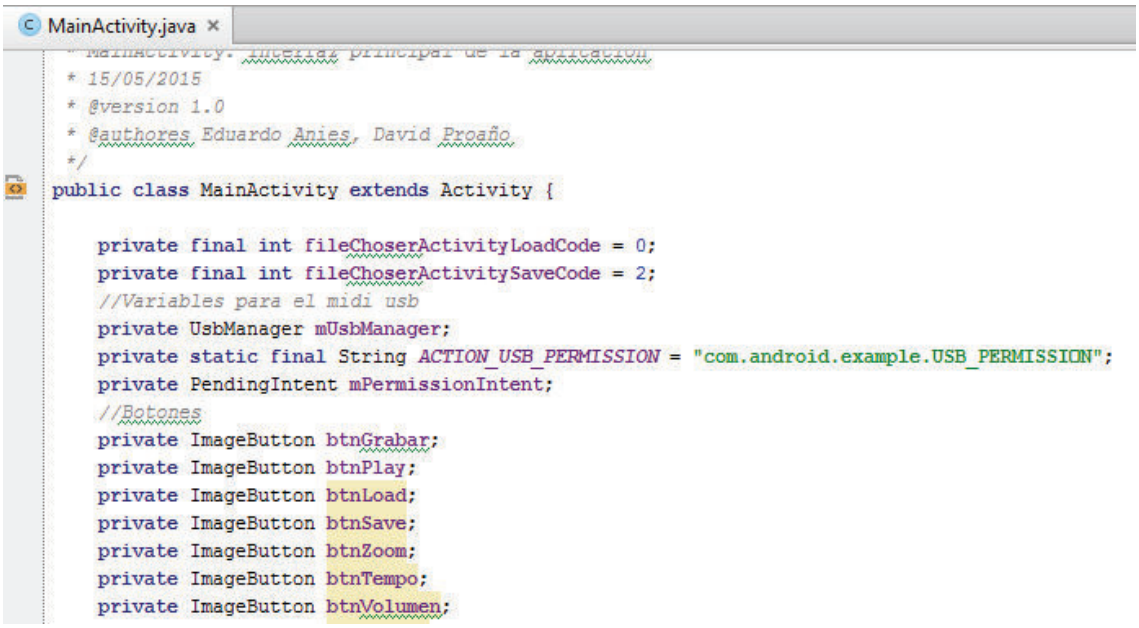


Figura 2.61 main activity elaborado por los autores

Controlador

Finalmente se tiene un solo controlador que interactúa con el modelo, es decir con las clases del paquete midi y con las clases del paquete midiUtilities mediante órdenes de la vista.

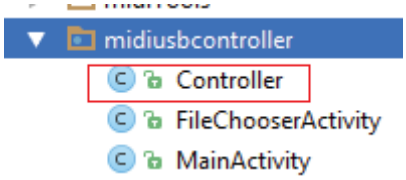


Figura 2.62 clase controler elaborado por los autores

Para hacer dicho controlador se creó una clase que extiende de la clase Application

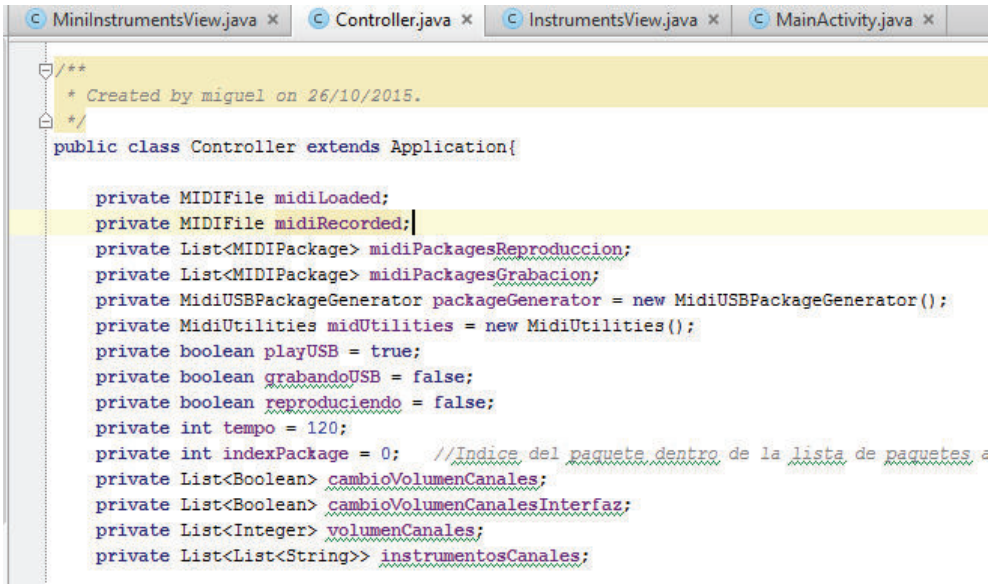


Figura 2.63 ejemplo de controlador elaborado por los autores

Se declara dicha clase dentro del archivo Manifest.xml

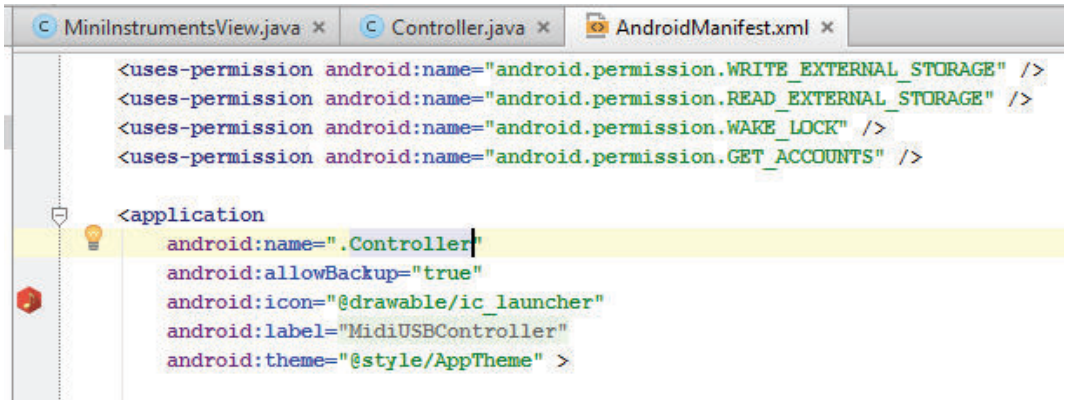


Figura 2.64 Manifest elaborado por los autores

3 PRUEBAS DE LA APLICACIÓN

3.1 PRUEBAS UNITARIAS

3.1.1 PRUEBAS UNITARIAS USANDO JUNIT

JUnit es un framework simple de código abierto que sirve para escribir y correr pruebas repetibles, las características que incluye JUnit son las siguientes:

- Aserciones para comprobar que los resultados obtenidos son los esperados.
- Ejecutores de pruebas para ver el número de pruebas exitosas y el número de pruebas fallidas.
- Accesorios de prueba para compartir data común de prueba.

JUnit fue trabaja con la licencia Eclipse Public License Versión 1.0 y se encuentra disponible para descargar en SourceForge [37].
Android incluye un framework integrado para realizar pruebas unitarias, este framework es basado en JUnit. JUnit permite organizar los métodos de pruebas en clases llamadas casos de prueba. En JUnit se usa un ejecutor de pruebas para ejecutar clases de prueba que muestra el porcentaje de pruebas unitarias llevadas a cabo con éxito.
Las clases de prueba en Android deberán extender de la clase TestCase, dentro de esta clase estarán los métodos de prueba, y dentro de ellos se deberá usar la instrucción assert para indicar que el resultado obtenido al ejecutar tal prueba deberá ser un valor especificado dentro de la misma función de assert [36].

En el presente proyecto se han realizado pruebas de las siguientes clases:

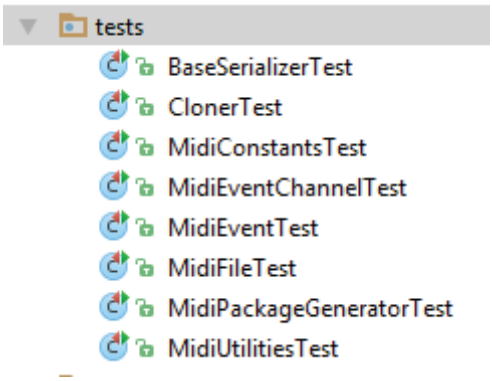


Figura 3.1 Clases para Pruebas Unitarias: Autores

Se ha hecho que todas las clases extiendan la clase TestCase, a continuación algunos ejemplos:

```
public class MidiFileTest extends TestCase {

    @Override
    protected void setUp() { super.setUp(); }

    @Override
    protected void tearDown() { super.tearDown(); }
```

Figura 3.2 Estructura básica de una clase de prueba con JUnit, referencia: Autores

Y los métodos de prueba dentro de cada clase que se vayan a probar empiezan con la palabra test, así es como JUnit diferencia de los métodos que son pruebas de los métodos que no son pruebas dentro de una clase que extiende TestCase, a continuación algunos ejemplos:

```
public void testLoadMidiFileTest() throws Exception{
    FileInputStream fin = null;
    DataInputStream din = null;
```

```
    public void testSaveMidi() throws Exception{
        //Guardar archivo midi
        FileOutputStream fout = null;
        DataOutputStream dout = null;

public void testVariableLegthValueFromByteList() throws Exception{
    MIDIEvent midiEvent = new MIDIEvent();
```

Figura 3.3 Métodos para pruebas referencia: Autores

Para saber si un método de un caso de prueba se ha ejecutado con éxito o ha fallado se deberá tener la instrucción assert dentro del método de prueba, a continuación un ejemplo:

```
public void testDeltaTimeToMiliSeconds() throws Exception {
    MidiUtilities midiUtilities = new MidiUtilities();
    int tempo = 120;
    int TQN = 480;
    int delta = 480;
    assertEquals(500, midiUtilities.convertDeltaTimeToMiliSeconds(delta,TQN,tempo));
    tempo = 120;
    TQN = 960;
    delta = 480;
    assertEquals(250, midiUtilities.convertDeltaTimeToMiliSeconds(delta,TQN,tempo));
}
```

Figura 3.4 Instrucción assert en métodos para prueba unitaria referencia: Autores

A continuación se mostrarán todos los métodos de prueba que se han creado, el número de aserciones que deberá cumplir cada método y el número de caso de prueba que se usó para probar dicho método.

Método de prueba	Aserciones	Número de casos de prueba
TempoConversion	3	3
DeltaTimeToMiliSeconds	2	2
MiliSecondsToDeltaTime	2	2
LyiricsFromMidiFile	1	1
ConvertMidiEventToMidiPackage	3	3
ConvertMIDIPackageToMIDIEvent	3	3
SaveMidi	Variable (más de 6)	1
LoadMidiFileTest	Variable (más de 5)	1
ByteListFromVariableLength	6	6
VariableLegthValueFromByteList	6	6
GetMidiChannel	3	3
GettChannel	3	3
InstrumentName	3	3
PositiveByteValue	4	4
IsChannelEvent	3	3
IsMetaEvent	3	3
isSysExEvent	3	3
IsSystemCommonMessage	3	3
IsSystemRealTimeMessage	3	3
IsEndOfTrackEvent	3	3
CloneObject	2	1
NewInstance	1	1
BytesCount	3	3
FieldOrder	11	3
GetPrimitiveSize	3	3
GetFieldIndex	5	1

Tabla 3.1 Métodos de Prueba Referencia: Autores

A continuación se muestran los resultados de la ejecución de pruebas unitarias realizado.

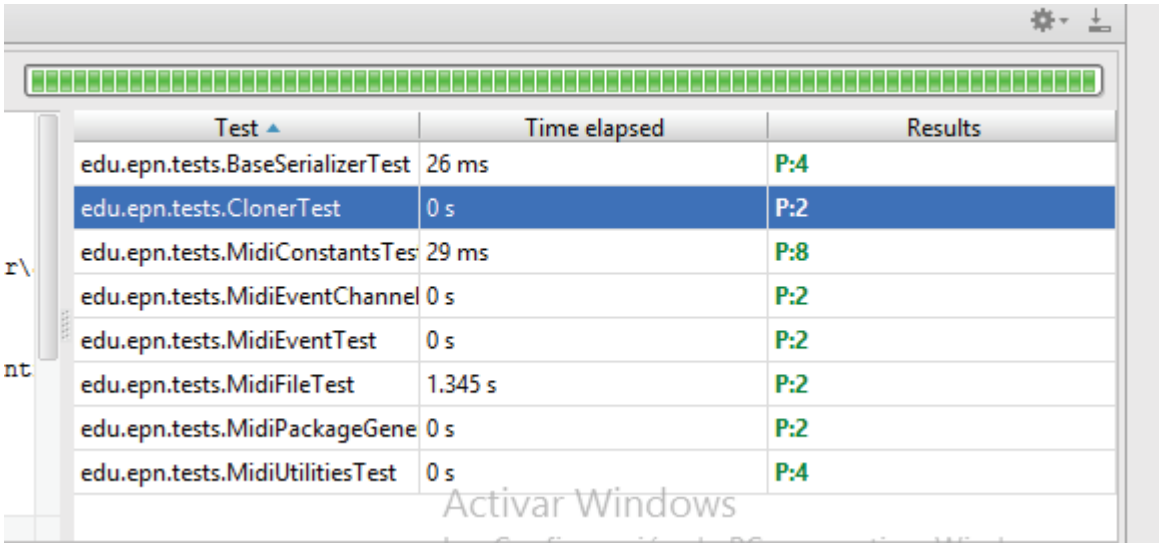


Figura 3.5 Resultados positivos de ejecución de las pruebas unitarias referencia: Autores

Como se puede ver todas las pruebas unitarias se han cumplido exitosamente, el ejecutado de pruebas de JUnit tiene la siguiente notación:

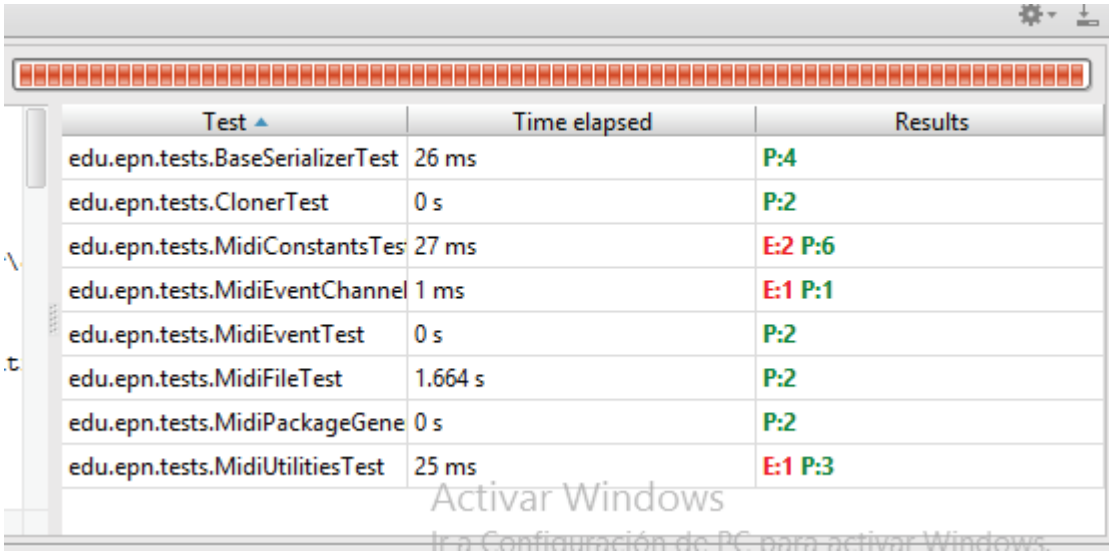
P: N

P (passed) sirve para indicar el número de métodos de prueba de un TestCase que se ejecutaron exitosamente, es decir el número de métodos en los cuales todas las aserciones internas se cumplieron. En el anterior ejemplo la N indica el número de métodos por clase.

E: N

E (Error) sirve para indicar el número de métodos internos de un Testcase que no se ejecutaron exitosamente, es decir el número de métodos en los que falló al menos una aserción interna. En el ejemplo anterior N indica el número de métodos por clase.

A continuación se mostrará un ejemplo de esta notación cuando fallan algunos métodos:



Test	Time elapsed	Results
edu.epn.tests.BaseSerializerTest	26 ms	P:4
edu.epn.tests.ClonerTest	0 s	P:2
edu.epn.tests.MidiConstantsTes	27 ms	E:2 P:6
edu.epn.tests.MidiEventChannel	1 ms	E:1 P:1
edu.epn.tests.MidiEventTest	0 s	P:2
edu.epn.tests.MidiFileTest	1.664 s	P:2
edu.epn.tests.MidiPackageGene	0 s	P:2
edu.epn.tests.MidiUtilitiesTest	25 ms	E:1 P:3

Figura 3.6 Resultados negativos de ejecución de las pruebas unitarias referencia: Autores

3.1.2 PRUEBAS UNITARIAS DE VISUALIZACIÓN DE INFORMACIÓN MIDI

3.1.2.1 Pruebas de teclado virtual

Para la visualización de información MIDI el software desarrollado cuenta con un teclado virtual el cual muestra las notas que están siendo enviadas al dispositivo MIDI y las notas que son enviadas desde el dispositivo MIDI al dispositivo Android, también se muestra con un color diferente cada nota según el canal por el que fue enviado.

Para realizar la prueba de visualización de notas en el teclado se llenará una lista de notas junto con una lista de canales los cuales sirven de argumento a una vista para saber que notas se deberán visualizar. Una vez llenas las listas se comprobarán que lo que muestra el teclado virtual sea lo esperado. A continuación se muestra un diagrama de los números que asigna el estándar MIDI a las notas musicales, se mostrarán las notas (DO, RE, MI….) y el número correspondiente (60, 61,62….).

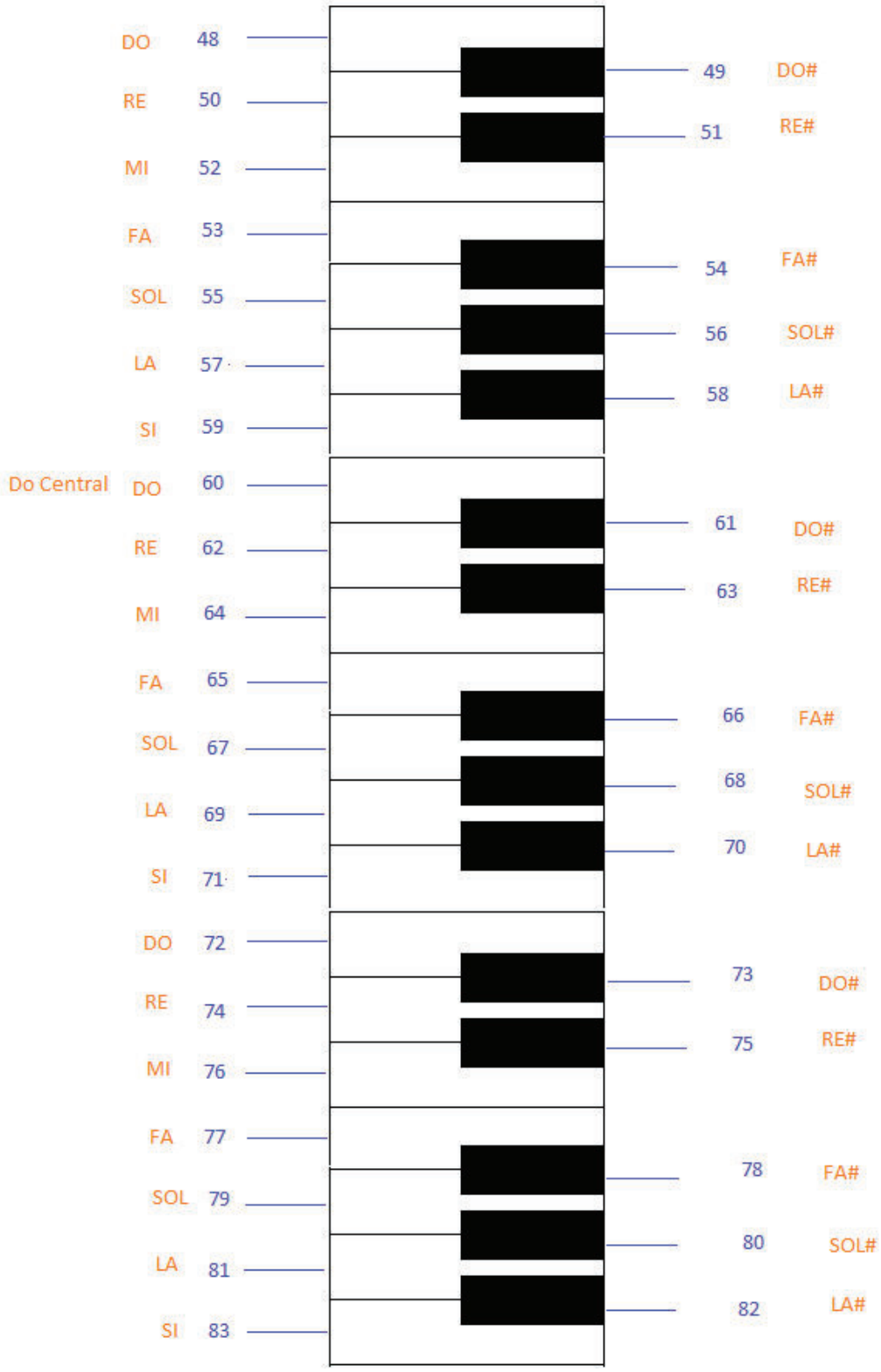


Figura 3.7 Representación numérica de notas definida por el estándar midi, referencia: Autores

A continuación se mostrará la lista de colores que se han asignado a cada canal MIDI para la visualización de notas:

Canal	Color
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Tabla 3.2 Lista de colores por canal creada por los autores

Se inicia las pruebas de visualización de notas en el teclado virtual llenando la lista de notas con los valores 60 (DO Central), 64 (MI) y se llena como datos en los canales con los valores de 0 y 0, al cual le corresponde el color rojo, a continuación el código:

```
public InstrumentsView(Context context, AttributeSet attrs) {
    super(context, attrs);
    notasTocadas.add(60);
    canalesNotas.add(0);
    notasTocadas.add(64);
    canalesNotas.add(0);
}
```

Figura 3.8 Datos de prueba para visualización de notas. Referencia: Autores

Luego se comprueba que la visualización es correcta

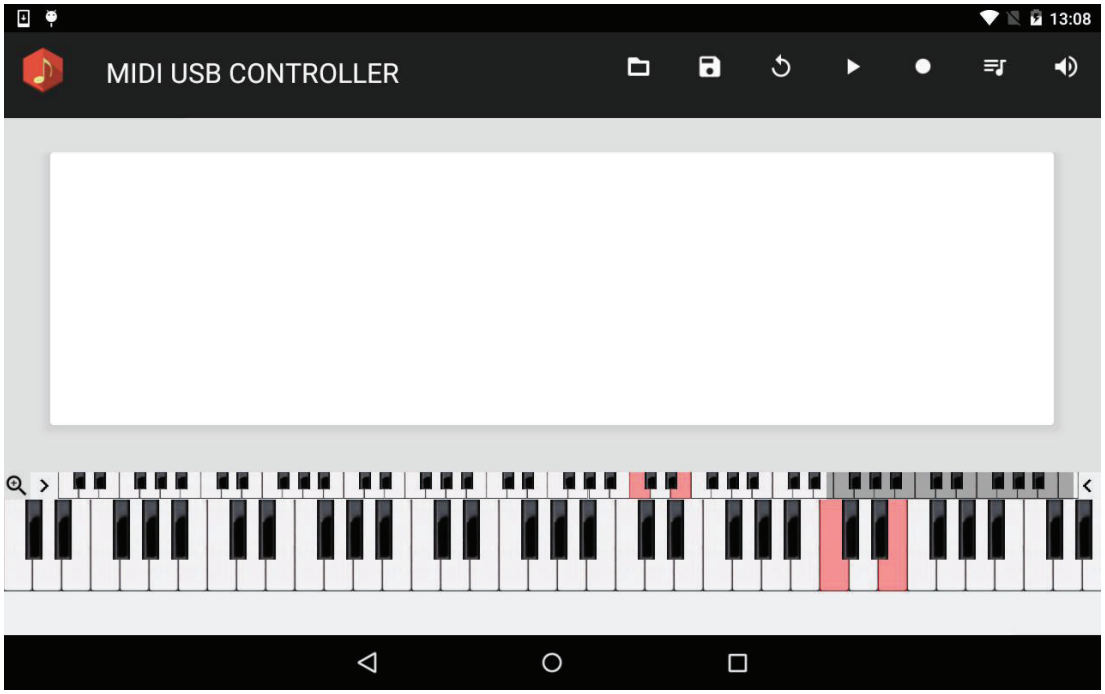


Figura 3.9 Figura 3 9 Prueba de visualización de las notas en el teclado virtual

A continuación se procede a realizar la prueba pero con todos los colores posibles usando 16 notas seguidas, y se obtiene el siguiente resultado exitoso.

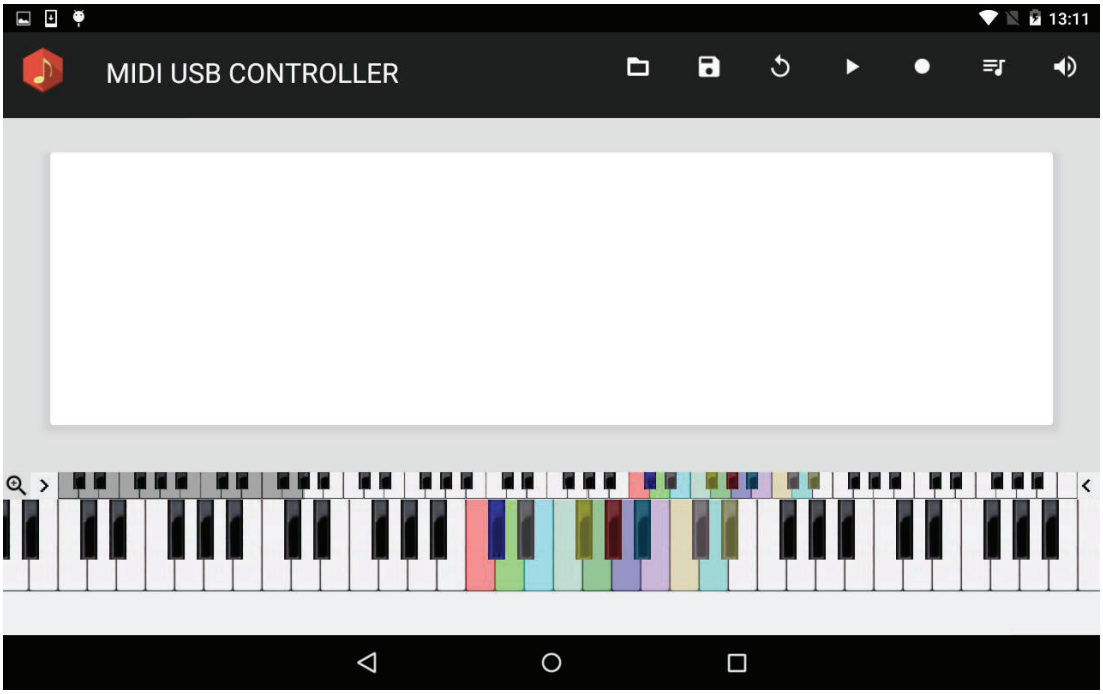


Figura 3.10 Prueba de visualización de colores según el canal de salida MIDI.

3.1.2.2 Pruebas de Karaoke

Para realizar la prueba de visualización de líricas se usa 2 listas, una que contiene las correspondientes letras normalmente en cada posición una sílaba, y en la segunda lista se tiene los tiempos correspondientes en los que deberá aparecer cada sílaba.

A continuación se muestran las sílabas de prueba que se usaron

```
//Datos de prueba
lyricsTimes = new ArrayList<>();
lyrics = new ArrayList<>();

lyricsTimes.add((long)0);
lyrics.add("Prueba"+MIDIConstants.END_OF_LINE);
lyricsTimes.add((long)2000);
lyrics.add("de"+MIDIConstants.END_OF_LINE);
lyricsTimes.add((long)3000);
lyrics.add("visualizacion\n"+MIDIConstants.END_OF_LINE);
lyricsTimes.add((long)4000);
lyrics.add("de "+MIDIConstants.END_OF_LINE);
lyricsTimes.add((long)6000);
lyrics.add("liricas "+MIDIConstants.END_OF_LINE);

currentPos = 3500;
```

Figura 3.11 Sílabas de prueba, referencia: Autores

La variable “currentPos” indica el tiempo en milisegundos que ya se ha reproducido, en el ejemplo su valor es de 3500 es decir 3 segundos y medio, por lo que las letras que tienen un indicador de tiempo menor o igual a 3500 se mostrarán en rojo, las que tengan un tiempo mayor a 3500 se deberán mostrarán en verde, además de que las letras que estén más cercanas al tiempo de reproducción actual deberán acercarse a la mitad...

Se obtiene el siguiente resultado:

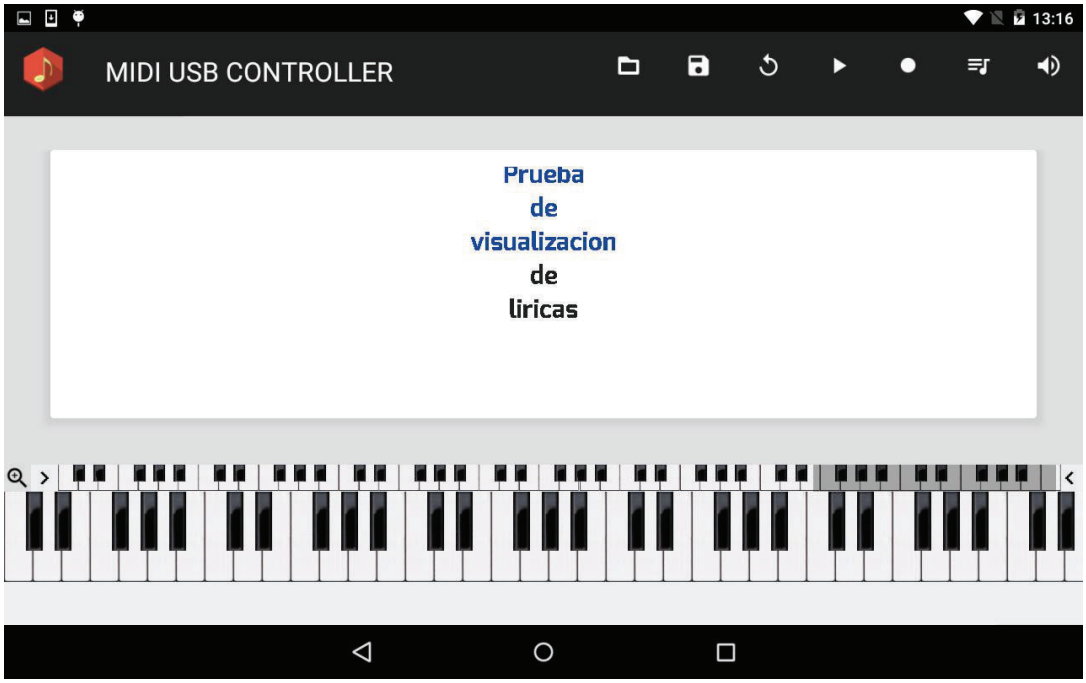


Figura 3.12 Visualización de prueba de líricas, referencia: Autores

Como se puede ver el resultado concuerda con los resultados esperados.

3.1.2.3 Pruebas de entrada y salida midi USB

Como primer dispositivo de prueba se ha usado un sintetizador Yamaha E233

Figura 3.13 sitentizador Yamaja E233 Referencia: Pulse music

El sintetizador mostrado anteriormente cuenta con la salida MIDI que se muestra a continuación la cual es bastante común

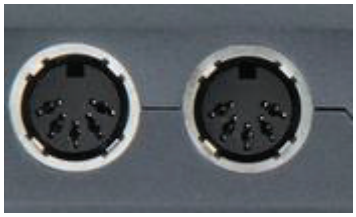


Figura 3.14 Entrada y salida midi del sintetizador Yamaja E233, Referencia: PC Midi Center

Y para realizar la correspondiente prueba se usó el cable MIDI USB clásico el cual se muestra a continuación.

Figura 3.15 Cable MIDI USB clásico Referencia: Mercado libre

Se procedió a realizar la conexión con una Tablet Nexus 7 la cual cumple con los requisitos de compatibilidad. A continuación el diagrama de conexión:

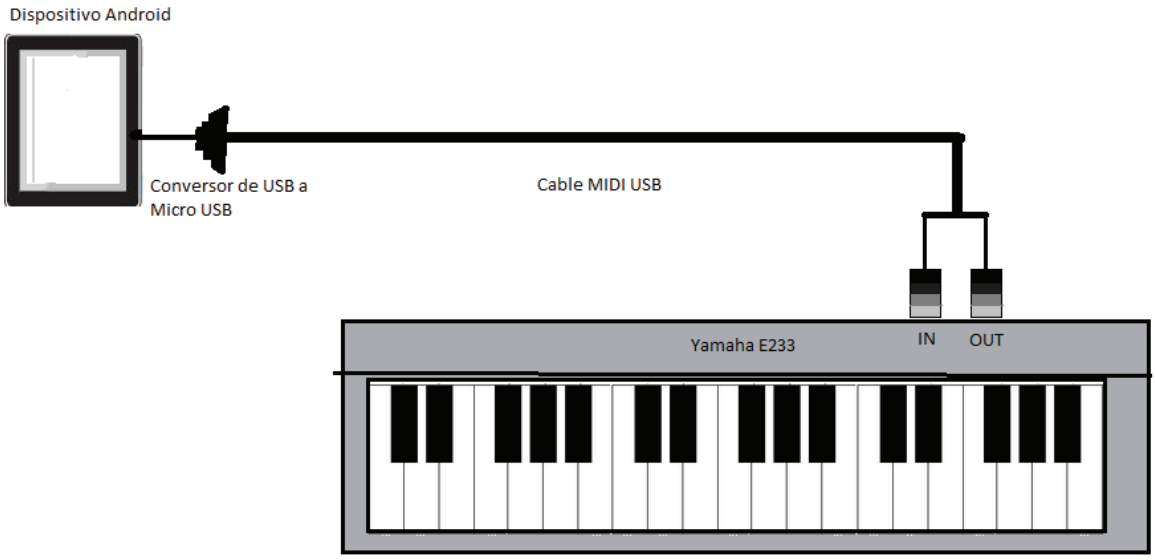


Figura 3.16 Diagrama de conexión dispositivo Android y dispositivo MIDI Referencia: Autores

Al probar la entrada efectivamente se comprobó que al presionar alguna tecla cualquiera el programa reconocía efectivamente la entrada y la mostraba en el teclado virtual de la aplicación la tecla presionada.

Yamaha E233 (Entonando la nota DO central)

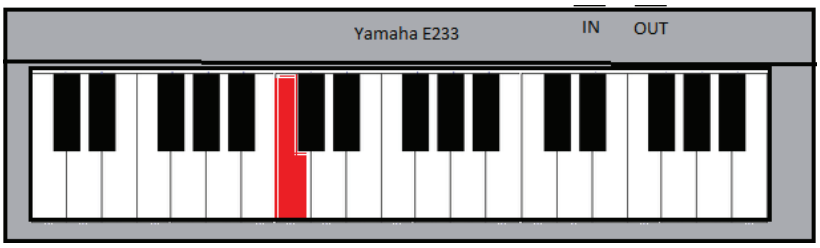


Figura 3.17 Tecla presionada en teclado Yamaja, Referencia: Autores

Visualización en la aplicación la nota correspondiente.

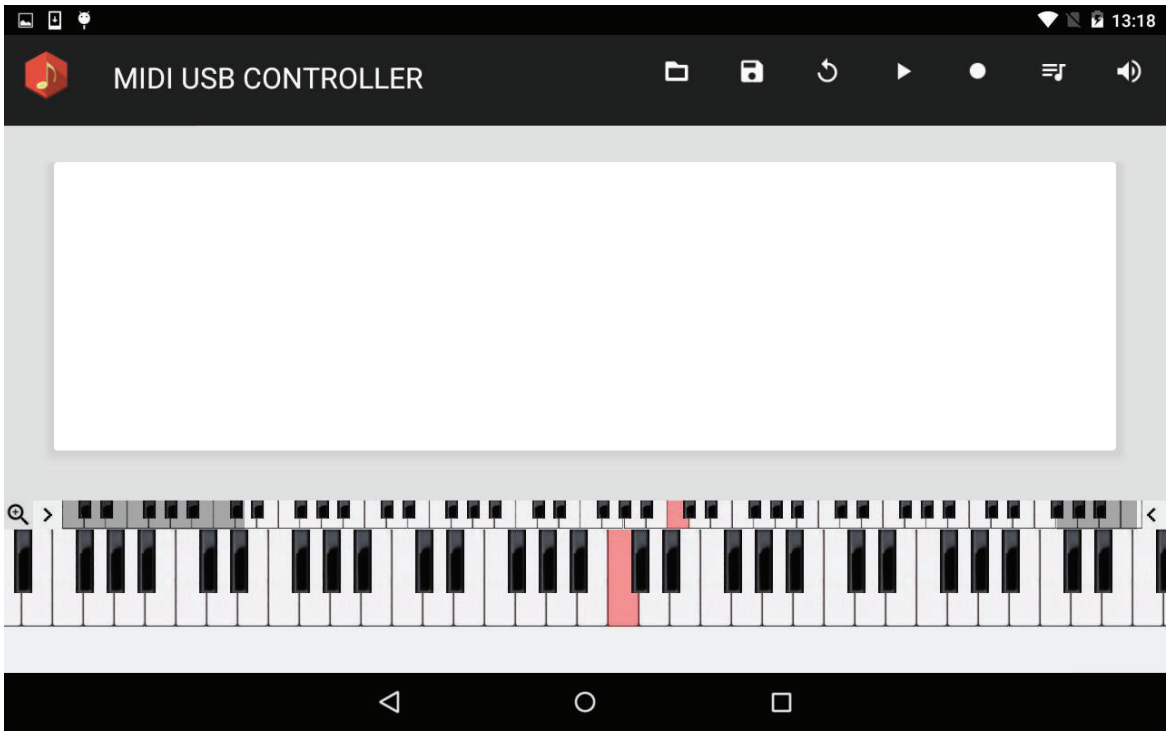


Figura 3.18 Visualización en la aplicación de la tecla presionada Referencia: Autores

Se realizó la misma prueba pero en sentido contrario enviando unos paquetes de salida vía código y el resultado fue el mismo.

Como segundo dispositivo de prueba se usó el sintetizador Yamaha PSR-E343

Figura 3.19 sintetizador Yamaha PSR-E343, Referencia: Amazon

Junto con el siguiente cable USB 2.1 tipo AB

Figura 3.20 Cable USB tipo AB, referencia: Mercado libre

Los resultados también fueron correctos también.

3.2 PRUEBAS DE INTEGRACIÓN

3.2.1 REPRODUCCIÓN DE ARCHIVOS MIDI USB

Las pruebas de integración de reproducción de archivos midi consistirán en cargar archivos MIDI que contengan aparte de información musical también líricas para que se pueda visualizar las notas a través de los diferentes canales y las líricas marcadas con rojo si ya pasaron y marcadas con verde si aún no pasan, se comprobará que la información enviada al dispositivo sea correcta al escuchar la canción y que la información visual sea correcta comparándola con la información presentada con otros programas como por ejemplo Van Basco Karaoke o Guitar Pro.

Se cargó el archivo MIDI “Billy_Joel_Honesty”, se obtuvo el siguiente resultado:

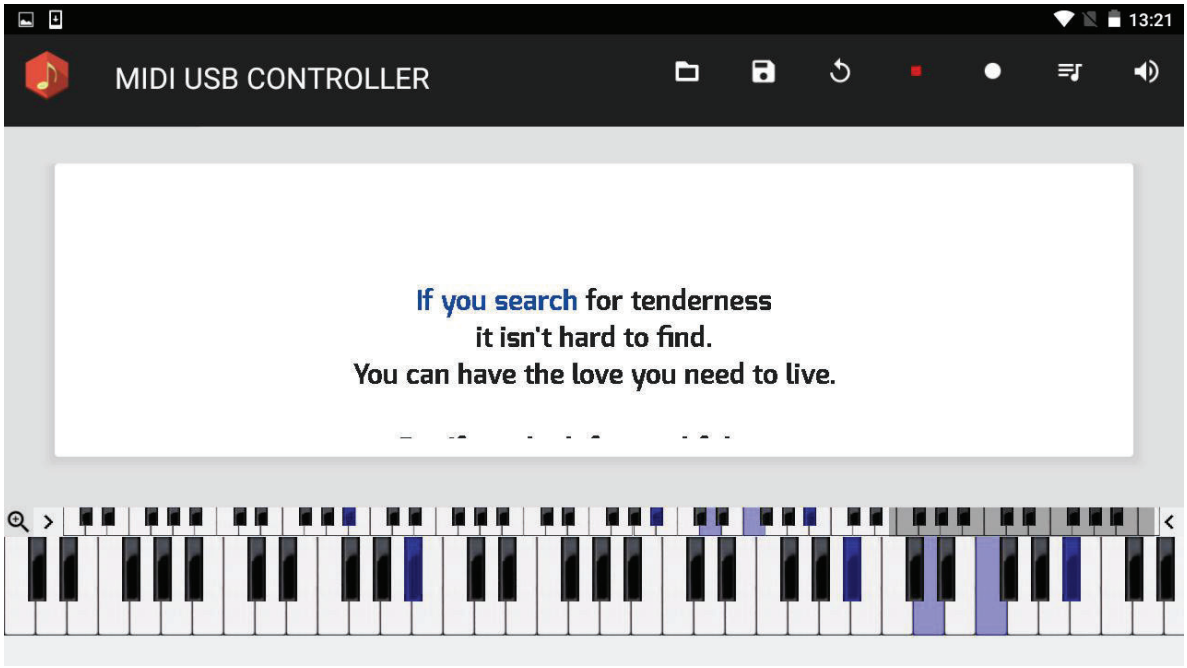


Figura 3.21 Resultado de carga en la aplicación, Referencia: autores

Se puede observar que el resultado es idéntico al abrir el mismo archivo con el software de Windows Van Basco Karaoke.

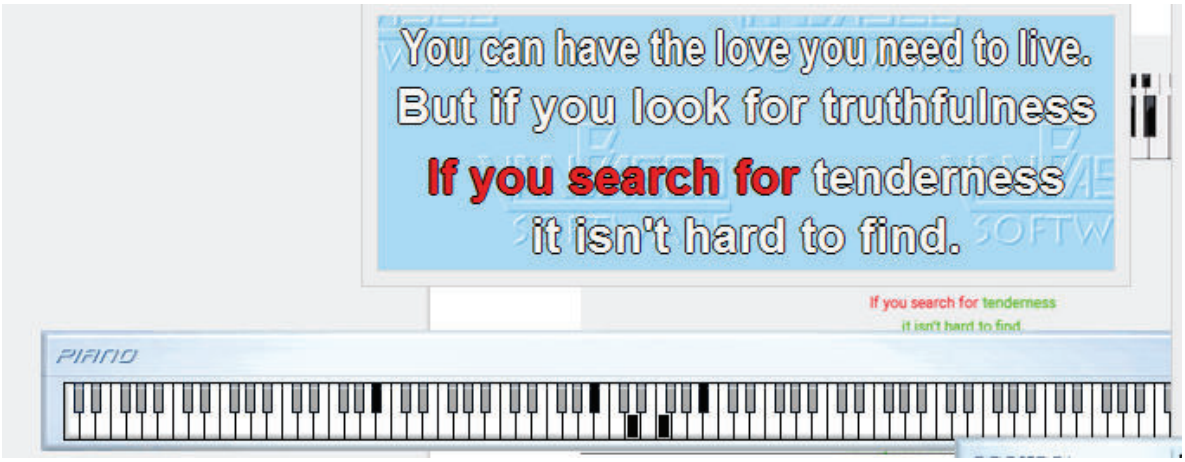


Figura 3.22 Resultado de carga de un archivo en aplicación Van Basco Karaoke Referencia: autores

Al igual que con este archivo se hizo pruebas con más archivos MIDI y se obtuvo un resultado similar en VanBasco Karaoke y la aplicación desarrollada, se notó que cuando se enviaban muchas notas a la vez se ralentizaba un poco la aplicación, pero mejorando el dispositivo Android este problema desaparecía.

3.2.2 GRABACIÓN DE ARCHIVOS MIDI

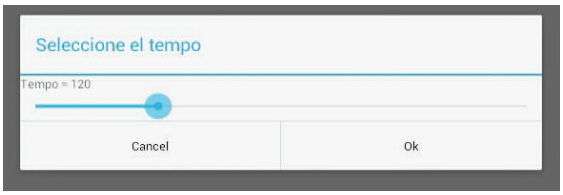


Figura 3.23 Tempo en la aplicación, Referencia: autores

Antes de iniciar la grabación se ajusta el tempo que saldrá en el archivo MIDI resultante en este caso se ha usado el valor de 120.

Se procedió a realizar la grabación tocando una melodía en el sintetizador real

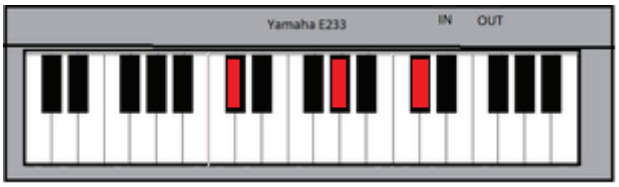


Figura 3.24 Visualización de notas de entrada durante la grabación de una melodía, Referencia: Autores

Y se pudo observar que la aplicación tomó la entrada de notas correctamente.

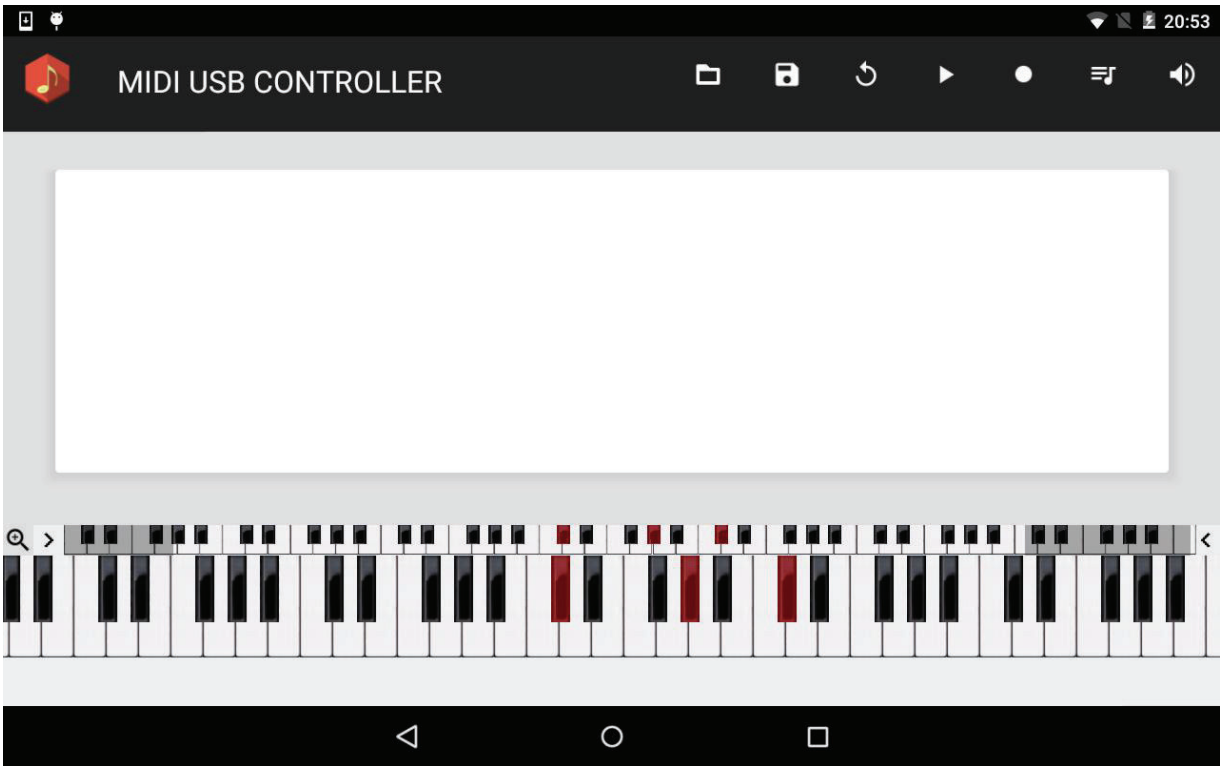


Figura 3.25 Visualización de notas correctas, Referencia: Autores

Luego por medio del explorador de archivos se guardó el archivo con un nombre especificado.

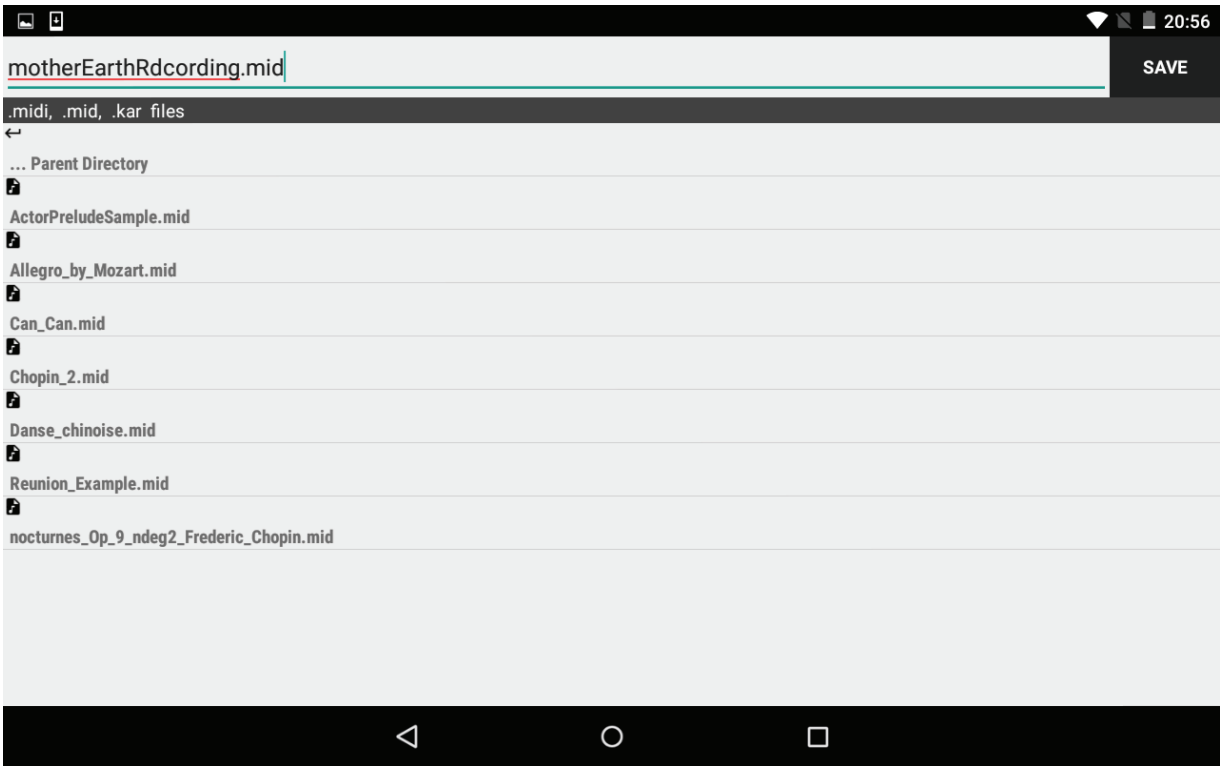


Figura 3.26 Interfaz para guardar archivo MIDI, referencia: Autores

Posteriormente se observó que el archivo MIDI fue generado con éxito

Figura 3.27 Archivo generado, referencia: Autores

Al importar el archivo MIDI en el software muy famoso para guitarristas llamado Guitar Pro se puede observar el siguiente resultado

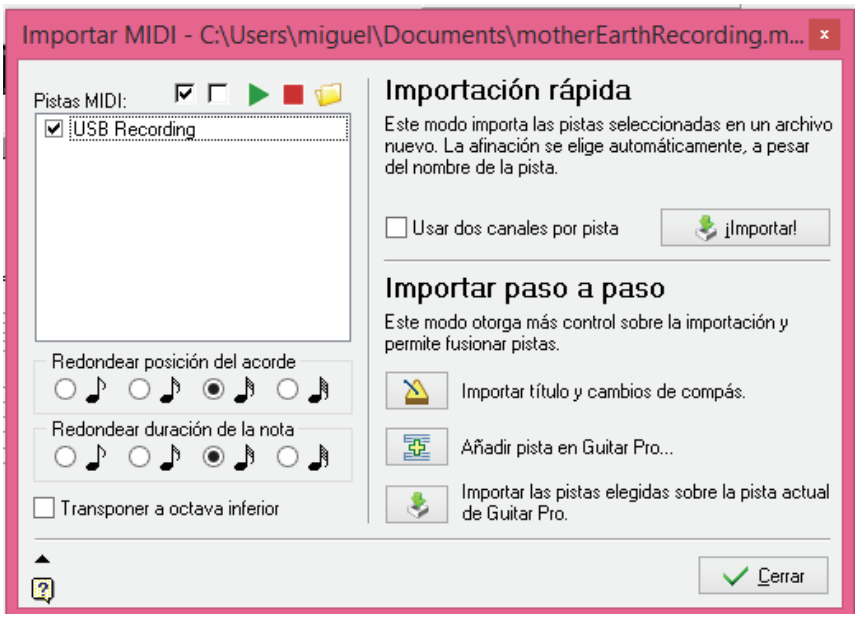


Figura 3.28 Menú de importación de archivos MIDI de software Guitar Pro, referencia: Autores

Se puede observar la partitura de la melodía entonada

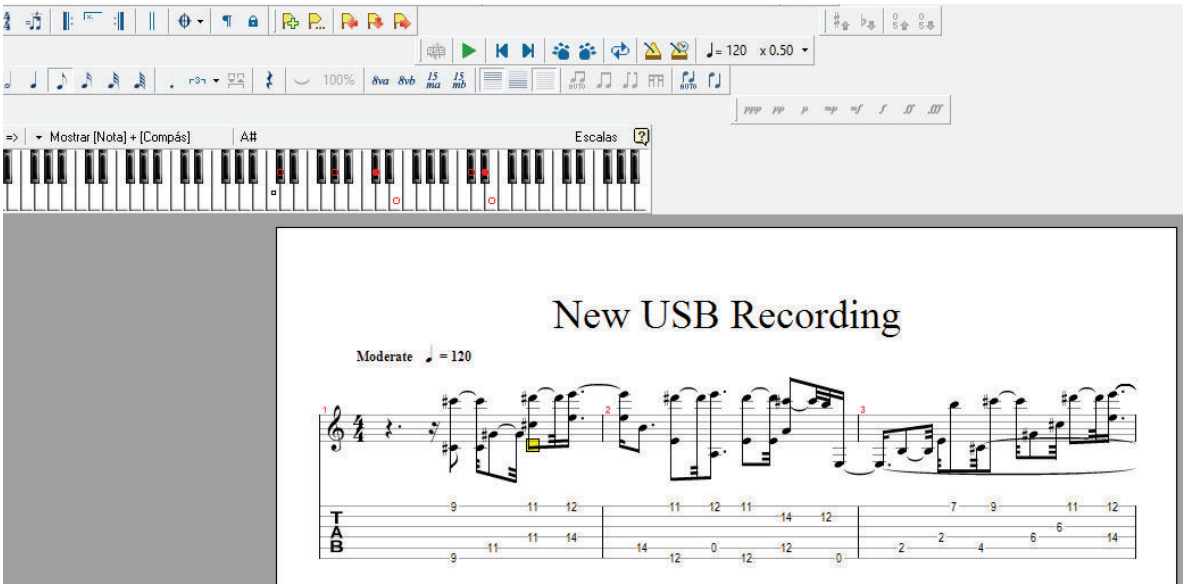


Figura 3.29 Visualización de partitura, referencia: Autores

Se puede observar también que dentro de la información de la partitura se muestra que el tempo del archivo midi es de 120 el cual es efectivamente el tempo escogido en la grabación, se puede observar también que el título de la

partitura es el de “New USB Recording” el cual es un nombre que fue establecido para aparecer en todos los archivos MIDI generados por este software.

Figura 3.30 Visualización del tempo del archivo generado dentro del software Guitar Pro, referencia: Autores

3.3 PRUEBAS DE ACEPTACIÓN

Estas pruebas se realizaron para verificar que la aplicación realizada cumple con lo estipulado en las historias de usuario descritas previamente. Se buscó un grupo de músicos con los cuales se pudo constatar el cumplimiento de las funcionalidades descritas en las historias de usuario. Se les hizo probar la aplicación y luego realizaron encuestas con preguntas relacionadas con cada funcionalidad como: la carga de archivos MIDI, la grabación de archivos MIDI, la reproducción de éstos, la visualización de las líricas, entre otras. Cada prueba se ejecutó para verificar si cumple las funcionalidades previstas y se tendrá en cuenta los diferentes escenarios posibles. Los resultados de la prueba realizada con los músicas se encuentra en el anexo C.

Las pruebas de aceptación tienen los siguientes componentes:

Caso de prueba: nombre de la prueba.
Número del caso de prueba: numeración de la prueba.
Número de la historia de usuario: Numeración correspondiente a la historia de usuario correspondiente.
Tarea: funcionalidad que va a verificarse.
Descripción: descripción de la funcionalidad.
Condiciones previas: Pre-condiciones a cumplirse antes de realizar la prueba.

Condiciones de ejecución: Condiciones que se deben cumplir dentro de las acciones de la funcionalidad

Entradas: pasos a seguir en la ejecución de las pruebas

Resultado esperado: resultado que se espera de la prueba.

Evaluación: resultado obtenido de la prueba.

Pruebas realizadas:

Prueba de Aceptación	
Caso de Prueba: Carga de archivo MIDI	
Número de Caso de Prueba: 1	**Número de Historia de Usuario:** 1
Tarea: Cargar de archivo MIDI	
Descripción: se selecciona un archivo y se lo carga a la aplicación si el formato es el correcto	
Condiciones previas: ● Debe existir un archivo en el sdcard del dispositivo	
Condiciones de ejecución: ● El archivo debe tener el formato adecuado para poder ser cargado, los archivos con extensión .kar .mid .midi son solo los que se van a cargar.	
Entradas: ● El usuario presiona el botón cargar ● El usuario va hasta el directorio donde está el archivo a cargar ● Selecciona el archivo y lo carga	
Resultados esperados: El archivo se carga correctamente	
Evaluación: Se cargó el archivo MIDI	

Tabla 3.3 Prueba de aceptación: Carga de archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Carga de archivo MIDI	
Número de Caso de Prueba: 2	**Número de Historia de Usuario:** 1
Tarea: Cargar de archivo MIDI incorrecto	
Descripción: se selecciona un archivo con el formato incorrecto	
Condiciones previas: ● Debe existir un archivo en el sdcard del dispositivo	
Condiciones de ejecución: ● Se escoge un archivo con un formato diferente de los archivos permitidos.	
Entradas: ● El usuario presiona el botón cargar ● El usuario va hasta el directorio donde está el archivo a cargar ● Selecciona el archivo y lo carga	
Resultados esperados: El archivo no se carga correctamente	
Evaluación: No se carga y se valida correctamente	

Tabla 3.4 Prueba de aceptación: Carga de archivo MIDI incorrecto realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Guardar archivo MIDI	
Número de Caso de Prueba: 3	**Número de Historia de Usuario:** 2
Tarea: Guardar archivo MIDI	
Descripción: se guarda la secuencia grabada en la aplicación.	
Condiciones previas: ● Debe existir una secuencia MIDI ya grabada previamente para guardar el archivo MIDI	
Condiciones de ejecución:	
Entradas: ● Con una secuencia MIDI el usuario procede a presionar el botón grabar ● El usuario escribe el nombre del archivo a guardar ● El usuario presiona el botón guardar	
Resultados esperados: El archivo se guarda correctamente	
Evaluación: El archivo se guarda correctamente y se abre con otro software y se comprueba que la información es correcta	

Tabla 3.5 Prueba de aceptación: Guardar archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Reproducir archivo MIDI	
Número de Caso de Prueba: 4	**Número de Historia de Usuario:** 3
Tarea: Reproducir un archivo MIDI	
Descripción: se reproduce un archivo MIDI correctamente cargado, después de cargarlo o que esté pausado.	
Condiciones previas: ● Previamente se debe tener cargado un archivo MIDI correcto. ● Si el archivo está pausado se debe reproducir desde el punto donde se pausó.	
Condiciones de ejecución: ● La conexión debe ser correcta	
Entradas: ● Con una secuencia MIDI el usuario procede a presionar el botón reproducir	
Resultados esperados: El archivo se comienza a reproducir o continúa la reproducción.	
Evaluación: El archivo se reproduce correctamente, con el sonido adecuado	

Tabla 3.6 Prueba de aceptación: Reproducir archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Pausar archivo MIDI	
Número de Caso de Prueba: 5	**Número de Historia de Usuario:** 3
Tarea: Pausar la reproducción del archivo MIDI	
Descripción: se pausa la reproducción del archivo MIDI.	
Condiciones previas: ● Previamente se debe tener cargado un archivo MIDI correcto. ● Debe estar reproduciendo un archivo MIDI para poder pausarlo.	
Condiciones de ejecución: ● El archivo MIDI debe estar en reproducción	
Entradas: ● Con un archivo MIDI pausado el usuario procede a presionar el botón pausar	
Resultados esperados: Se pausa la reproducción del archivo MIDI	
Evaluación: Se detuvo la reproducción y se detienen todos los sonidos que se hayan iniciado previamente.	

Tabla 3.7 Prueba de aceptación: Pausar la reproducción del archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Modificar el volumen de las pistas del archivo MIDI	
Número de Caso de Prueba: 6	**Número de Historia de Usuario:** 3
Tarea: Modificar el volumen de las pistas del archivo MIDI	
Descripción: se sube o baja el volumen de las pistas del archivo MIDI**.**	
Condiciones previas: ● Previamente se debe tener cargado un archivo MIDI correcto. ● Debe estar reproduciendo un archivo MIDI para poder pausarlo.	
Condiciones de ejecución: ● El archivo MIDI no debe estar en reproducción en ese momento.	
Entradas: ● Con un archivo MIDI que no esté en reproducción se selecciona el volumen individual de cada canal mediante una barra progresiva.	
Resultados esperados: Se modifica el volumen de las pistas del archivo MIDI	
Evaluación: Se modificó el volumen de las pistas del archivo MIDI	

Tabla 3.8 Prueba de aceptación: Modificar el volumen de la reproducción del archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Grabar secuencia MIDI	
Número de Caso de Prueba: 7	**Número de Historia de Usuario:** 3
Tarea: Receptar los eventos MIDI a ser guardados	
Descripción: se receptan los eventos MIDI que van a ser grabados	
Condiciones previas:	
Condiciones de ejecución: ● La conexión debe ser correcta	
Entradas: ● Conectar los dispositivos (Android y MIDI) ● Presionar el botón de grabar ● Enviar los eventos a ser almacenados.	
Resultados esperados: Se receptan los eventos MIDI a ser grabados	
Evaluación: Se receptaron los eventos en el tiempo correcto a ser grabados.	

Tabla 3.9 Prueba de aceptación: Grabación de archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Control de tempo de grabación del archivo MIDI	
Número de Caso de Prueba: 8	**Número de Historia de Usuario:** 4
Tarea: Controlar el tempo de grabación del archivo MIDI	
Descripción: controlar el tempo de grabación que aparecerá en el archivo MIDI de salida	
Condiciones previas:	
Condiciones de ejecución: ● Se debe escoger el tempo previo a realizar una grabación de secuencia	
Entradas: ● Presionar el botón tempo ● Escoger el tempo en la barra progresiva ● Presionar el botón aceptar	
Resultados esperados: se controla el tempo de grabación del archivo MIDI	
Evaluación: se controló el tempo de grabación, lo cual fue comprobado al abrir el fichero de salida del software en un software distinto y ver el tempo	

Tabla 3.10 Prueba de aceptación: Controlar el tempo de grabación del archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Visualización de información MIDI	
Número de Caso de Prueba: 9	**Número de Historia de Usuario:** 5
Tarea: Visualización de las notas en el teclado virtual	
Descripción: Para visualizar de mejor manera las notas musicales que se están entonando, el usuario final podrá contar con un teclado virtual que mostrará las notas que se están entonando con un color distinto según el canal	
Condiciones previas: ● Debe estar cargado un archivo MIDI	
Condiciones de ejecución:	
Entradas: ● Se debe presionar el botón Reproducir para poder visualizar las notas en el teclado virtual.	
Resultados esperados: se visualiza las notas en el teclado virtual.	
Evaluación: se visualizó las notas en el teclado virtual	

Tabla 3.11 Prueba de aceptación: Controlar el tempo de grabación del archivo MIDI realizada por Autores

Prueba de Aceptación	
Caso de Prueba: Visualización de información MIDI	
Número de Caso de Prueba: 10	**Número de Historia de Usuario:** 5
Tarea: Visualización de líricas del archivo MIDI	
Descripción: Si se ha cargado un archivo MIDI que contiene líricas, se muestran de manera que se pueda diferenciar entre las líricas que ya pasaron y las que vendrán.	
Condiciones previas: ● Debe estar cargado un archivo MIDI	
Condiciones de ejecución: ● El fichero MIDI escogido debe contener líricas.	
Entradas: ● Se debe presionar el botón Reproducir para poder visualizar las líricas tipo karaoke.	
Resultados esperados: se visualiza las líricas del archivo MIDI	
Evaluación: se visualizó las líricas del archivo MIDI	

Tabla 3.12 Prueba de aceptación: Visualizar letra de un MIDI realizada por Autores

4 CONCLUSIONES Y RECOMENDACIONES:

4.1 CONCLUSIONES

- El objetivo principal de esta tesis ha sido el de fusionar la tecnología MIDI conocida ya varias décadas atrás con la tecnología móvil la cual es relativamente nueva. Se puede apreciar durante el desarrollo de toda este proyecto de titulación que se ha cumplido este objetivo satisfactoriamente accediendo a información libre sobre el estándar MIDI combinándola con el estudio de las posibilidades que nos ofrece la plataforma Android.

- Durante el desarrollo de esta tesis se comprueba todo el alcance y todas las posibilidades que nos abre la tecnología MIDI, se pueden crear desde programas que enseñen a tocar piano, flauta, guitarra hasta programas avanzados de notación musical usados hoy en día por músicos profesionales para composición musical, además de que es el lenguaje de comunicación estándar de los sintetizadores actuales.

- Como revelan las encuestas realizadas a los usuarios potenciales de esta aplicación, la mayoría de músicos hoy en día se ayudan de herramientas de software para cumplir sus objetivos, sin embargo las herramientas que existen actualmente para Android en esta categoría aún les falta desarrollarse para que lleguen a igualar a la variedad de herramientas que encontramos en Windows y Mac. Esa fue la razón principal por la cual se escogió usar el sistema operativo Android.

  A pesar de que la mayoría de funcionalidades de la aplicación ya han sido implementadas anteriormente en diversas aplicaciones, esta aplicación cuenta adicionalmente con grabación midi USB, lo cual es un aporte a las herramientas de software MIDI para Android.

4.2 RECOMENDACIONES

- Se recomienda para el estudio del estándar MIDI obtener información principalmente de fuentes oficiales como es “The MIDI Manufacters Association” creadores de la especificación 1.0 del estándar, o el estándar “Universal Serial Bus Device Class Definition for MIDI Devices ” desarrollado por IBM, MICROSOFT, Roland Corporation debido a que en otros lugares la información muchas veces no está completa u organizada.

- Se recomienda que se identifiquen cuidadosamente los cables conectores y los conversores adicionales que se requieren para conectar un dispositivo MIDI a un dispositivo Android, puesto que no todos los dispositivos MIDI usan los mismos tipos de cables y conversores de USB a micro USB.

- Se recomienda usar la tecnología MIDI para el desarrollo de productos innovadores, por ejemplo la visualización de archivos MIDI en instrumentos no tan tradicionales como el piano, si no como la flauta dulce, algunos tipos de armónicas, etc., o para realizar nuevos tipos de programas que son raros verlos hoy en día.

- Se recomienda también usar sintetizadores de software en el caso que se desee que el propio dispositivo Android genere el sonido. Android nativamente ya cuenta con un sintetizador de sonido nativo, sin embargo también se pueden usar otros adicionales como por ejemplo FluidSynth el cual permite adicionalmente editar los sonidos que se usarán para generar el sonido del archivo MIDI mediante el uso de una tecnología llamada Soundfonts.

5 BIBLIOGRAFÍA

[1] Midi.org, (2015). *History of MIDI*. [online] Available at: http://www.midi.org/aboutmidi/tut_history.php

[2] MIDI Association, T. (1988). *Standard MIDI Files 1.0*. [online] csc.kth.se/. Available at: http://www.csc.kth.se/utbildning/kth/kurser/DT1174/ljud09/standard_midi_files.pdf

[3] Midi.org, (2015). *MIDI Message Table 1*. [online] Available at: http://www.midi.org/techspecs/midimessages.php

[4] MIDI Association, *Standard MIDI-File Format Spec. 1.1*, Los ángeles, 1988, pp. 2-18

[5] Midi.org, (2015). *MIDI Files*. [online] Available at: http://www.midi.org/aboutmidi/tut_midifiles.php

[6] The MIDI Manufacters Association, MIDI 1.0 Detailed Specification, Los Ángeles, 1995, pp. 9-50

[7] IBM Corporation – Microsoft – Roland - Philips, *Universal Serial Bus Device Class Definition for MIDI,* Sunnyvale-EE.UU., 1999, pp. 16-18, 27

[8] Hewlett-Packard – Compaq – Intel – Microsoft, *Universal Serial Bus Specification*, Chiell-Chih Lee, Keelung **City** (TW), 2000, pp. 49.

[9] Accessory, U. (2015). *USB Accessory | Android Developers*. [online] Developer.android.com. Available at: http://developer.android.com/guide/topics/connectivity/usb/accessory.html

[10] Host, U. (2015). *USB Host | Android Developers*. [online] Developer.android.com. Available at: http://developer.android.com/guide/topics/connectivity/usb/host.html

[11] GitHub, (2014). *LeffelMania/android-midi-lib*. [online] Available at: https://github.com/LeffelMania/android-midi-lib.

[12] V. Devi, 'Traditional and Agile Methods: An Interpretation - Scrum Alliance', 2013. [Online]. Available:

https://www.scrumalliance.org/community/articles/2013/january/traditional-and-agile-methods-an-interpretation

[13] R. Uñoja , 'Ingeniería de Software: METODOLOGIAS DE DESARROLLO DE SOFTWARE TRADICIONALES VS AGILES', *Masteringenieriasoft.blogspot.com*, 2012. [Online]. Available: http://masteringenieriasoft.blogspot.com/2012/04/metodologias-de-desarrollo-de-software.html

[14] Scrum.org, 'Scrum.org | the home of Scrum > Home', 2015. [Online]. Available: https://www.scrum.org/.

[15] Clases3gingsof.wikifoundry.com, 'Artefactos - Ingeniería Software', 2015. [Online]. Available: http://clases3gingsof.wikifoundry.com/page/Artefactos.

[16]D. Wells, 'Extreme Programming: A Gentle Introduction.' *Extremeprogramming.org*, 2015. [Online]. Available: http://www.extremeprogramming.org/

[17]D. Wells, 'Extreme Perl - Chapter 15: It's a SMOP', *Extremeperl.org*, 2015. [Online]. Available: http://www.extremeperl.org/bk/its-a-smop

[18]J. Joskowicz, *Reglas y prácticas en Extreme Programming*, 1st ed. Pontevedra, 2008, pp. 7-19.

[19]J. Canós, P. Letelier and M. Penadés, *Métodologías Ágiles en el Desarrollo de Software*, 1st ed. Valencia, 2003, pp. 1-6.

[20]S. Ambler, 'Introduction to Test Driven Development (TDD)', *Agiledata.org*, 2010. [Online]. Available: http://agiledata.org/essays/tdd.html

[22]D. Villa, '1.10.2. Programación en parejas', Arco.esi.uclm.es, 2015. [Online]. Available: http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol1/ch01s10s02.html

[23] G. Chavez, D. Macias, J. Torres and A. Videras, 'Diferencias entre scrum y xp', *Es.slideshare.net*, 2012. [Online]. Available: http://es.slideshare.net/deborahgal/diferencias-entre-scrum-y-xp-12219336.

[24] M. Cohn, 'Differences between Scrum and Extreme Programming',

Mountaingoatsoftware.com, 2007. [Online]. Available: http://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming

[25]L. Prechelt, *Agile Methods*, 1st ed. Berlin, 2015, pp. 5-7.

[26]A. Cockburn, 'Alistair.Cockburn.us | Crystal methodologies', *Alistair.cockburn.us*, 2015. [Online]. Available: http://alistair.cockburn.us/Crystal+methodologies.

[27] Dsdm.org, 'Front Page | DSDM Consortium', 2015. [Online]. Available: http://www.dsdm.org/.

[28]J. Highsmith, 'Jim Highsmith', *Adaptivesd.com*, 2015. [Online]. Available: http://www.adaptivesd.com/.

[29] Featuredrivendevelopment.com, 'Feature Driven Development | the portal for all things FDD.' 2015. [Online]. Available: http://www.featuredrivendevelopment.com/.

[30]T. Poppendieck, 'Poppendieck.LLC', *Poppendieck.com*, 2015. [Online]. Available: http://www.poppendieck.com/.

[31]E. Gaete Flores, 'Apuntes Usach Metodología ágil scrum', *Apuntes Usach*, 2015. [Online]. Available: http://apuntesusach.herokuapp.com/tutorial/scrum.

[32] E. Sanchez, P. Letelier and J. Canós, *Mejorando la gestión de historias de usuario en Extremme Programming*, 1st ed. Valencia: Universidad Técnica de Valencia, 2015, p. 4.

[33]J. Sánchez, J. Sánchez and V. perfil, 'Android Cero: Requerimientos para instalar Android Studio en Windows', *Androidcero.eledevapps.com*, 2015. [Online]. Available: http://androidcero.eledevapps.com/2015/01/requerimientos-para-instalar-android.html.

[34]]M. Adrián Anaya Villegas, 'A propósito de programación extrema XP (extreme Programming) (página 2) - Monografias.com', *Monografias.com*, 2015. [Online]. Available: http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema2.shtml.

[35]Oracle.com, (2015). *Code Conventions for the Java Programming Language: Contents*. [online] Available at: http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

[36] Fundamentals, T. (2015). *Testing Fundamentals | Android Developers*. [online]

Developer.android.com. Available at: http://developer.android.com/tools/testing/testing_android.html

[37] Junit.org, (2015). *JUnit - Frequently Asked Questions*. [online] Available at: http://junit.org/faq.html#overview_1

[38] Es.slideshare.net, 'Extreme programming (1)', 2015. [Online]. Available: http://es.slideshare.net/enriquepolo9/extreme-programming-1.

[39] Sparxsystems.com.ar, 'Sparx Systems - Tutorial UML 2 - Diagrama de Componentes', 2015. [Online]. Available: http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html

[40] *Diseño orientado a objetos con UML*. Madrid: Grupo EIDOS, 2000, pp. 19,96-97

[41] Visual-paradigm.com, 'Component Diagram - UML 2 Diagrams - UML Modeling Tool', 2015. [Online]. Available: http://www.visual-paradigm.com/VPGallery/diagrams/Component.html.

[42] Sparxsystems.com.ar, 'Sparx Systems - Tutorial UML 2 - Diagrama de Actividades', 2015. [Online]. Available: http://www.sparxsystems.com.ar/resources/tutorial/uml2_activitydiagram.html.

[43] S. Systems, 'Sparx Systems-Tutorial UML 2 - Diagrama de Despliegue', *Sparxsystems.com.ar*,2015. [Online]. Available: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html

[44] Oracle.com, 'Java SE Application Design With MVC', 2015. [Online]. Available: http://www.oracle.com/technetwork/articles/javase/index-142890.html.

ANEXOS

ANEXO A: MANUAL DE USUARIO DE LA APLICACIÓN

Al abrir la aplicación se tiene el siguiente menú principal

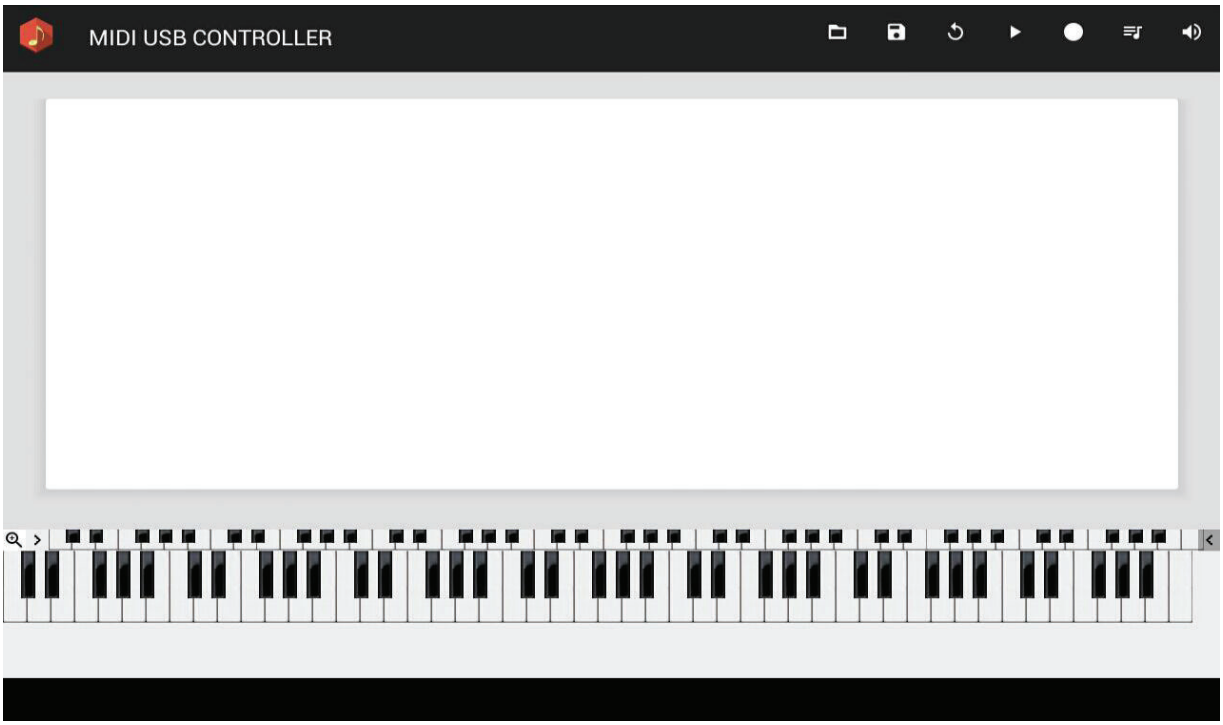


Figura A.1 Menú aplicación MIDI, referencia: Autores

Botón cargar

El primer botón como es de imaginarse es el que abre un explorador de archivos y nos permitirá abrir un archivo MIDI con todas las extensiones permitidas

Figura A.2 Botón de carga aplicación MIDI, referencia: Autores

En la imagen inferior se puede ver el explorador de archivos que contiene la aplicación, este explorador de archivos mostrará únicamente carpetas y archivos MIDI, más no archivos de ninguna otra clase.

Las extensiones que de archivos MIDI permitidas son .midi, .mid y .kar

Figura A.3 Explorador de archivos aplicación MIDI, referencia: Autores

Mientras se realice la carga del archivo MIDI se mostrará el siguiente diálogo como indicador de que debe seguir esperando

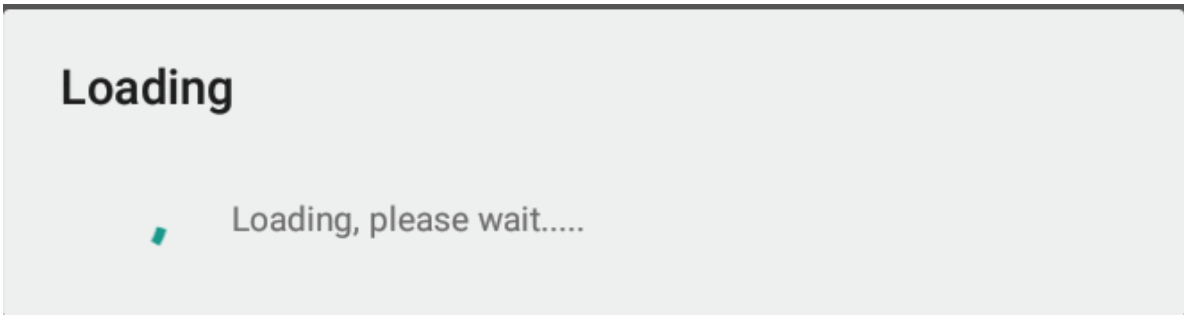


Figura A.4 Dialogo de espera mientras se carga un archivo dentro de la aplicación MIDI, referencia: Autores

Botón guardar

Figura A.5 Botón guardar aplicación MIDI, referencia: Autores

El segundo botón (Guardar) al igual que el primero abrirá un explorador de archivos en el cual se podrá escribir un nombre de archivo, si es que al archivo no se le

agrega extensión la aplicación automáticamente le guardará como archivo .mid

Botón reiniciar

Figura A.6 Botón reiniciar en la aplicación MIDI, referencia: Autores

El botón tercero (Reiniciar) cumple con la función de reiniciar la reproducción del archivo sin importar el tiempo de reproducción que llevaba previamente.

Botón reproducir

Figura A.7 Botón de reproducción dentro de la aplicación MIDI, referencia: Autores

El cuarto botón (Reproducir) inicia la reproducción del fichero MIDI si es que ya se ha cargado previamente, caso contrario indicará al usuario que debe primero cargar un archivo MIDI.

Una vez iniciada la reproducción del fichero MIDI cambiará el ícono del botón play por el ícono stop.

Figura A.8 Botón Stop aplicación MIDI, referencia: Autores

De la misma forma cuando se detenga la reproducción de un archivo MIDI cambiará el ícono stop por el ícono play.

Botón grabar

Figura A.9 Botón grabar aplicación MIDI, referencia: Autores

El quinto botón (Grabar) cumple la función de capturar los eventos MIDI de entrada y el tiempo en el que fueron receptados.

Botón tempo

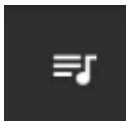


Figura A.10 Botón tempo aplicación MIDI, referencia: Autores

El sexto botón (tempo) nos permitirá ingresar el tempo en formato BPM (Beats per minute) que en español significa pulsaciones por minuto, y este valor se usará como tempo en los archivos MIDI de salida.

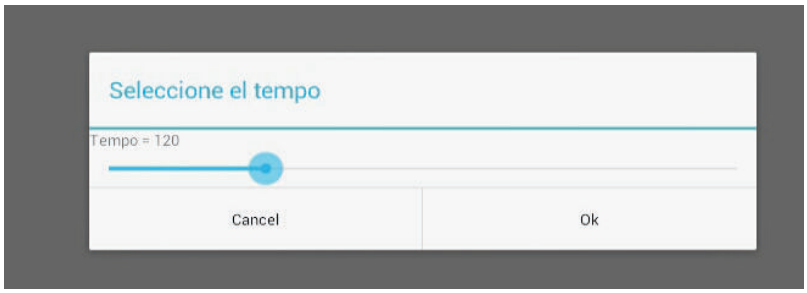


Figura A.11 Tempo en la aplicación, Referencia: autores

El gráfico anterior muestra el dialogo en el cual se puede ingresar el tempo, el ingreso se lo realiza por medio de la barra progresiva que se encuentra debajo del valor del tempo. El mínimo valor permitido es el de 60, y el máximo valor permitido es el de 300.

Botón volumen

Figura A.12 Botón volumen aplicación MIDI, referencia: Autores

El último botón del menú permite controlar los volúmenes de salida de los diversos canales MIDI, además muestra el color de salida de cada canal MIDI y los nombres de programas asociados a cada canal.

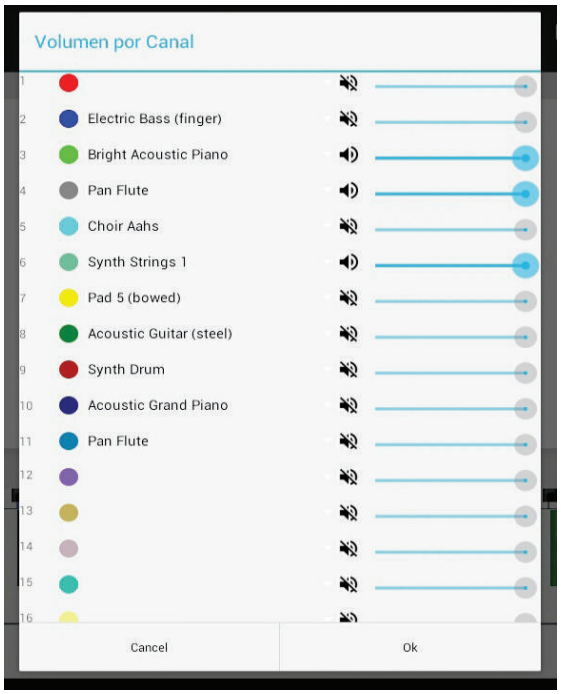


Figura A.13 Volumen de salida por canal, referencia: Autores

Conexión dispositivo MIDI con dispositivo Android

Para realizar la conexión del dispositivo MIDI al dispositivo Android se requerirá de un conector Otg USB a micro USB y un cable MIDI; la conexión se la deberá realizar de la siguiente forma:

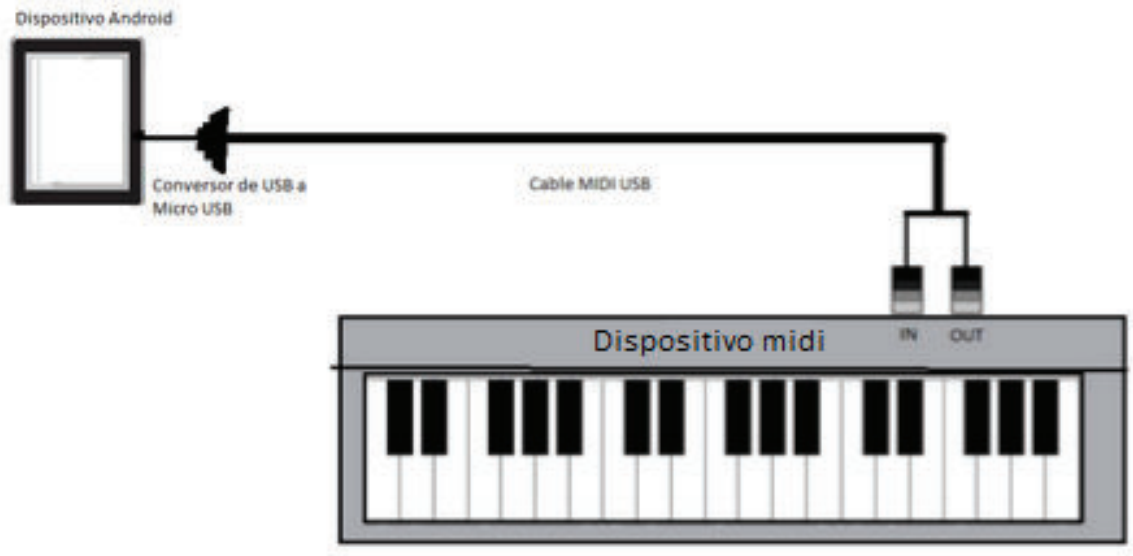


Figura A.14 Conexión dispositivo MIDI con dispositivo Android, referencia: Autores

Control del teclado virtual

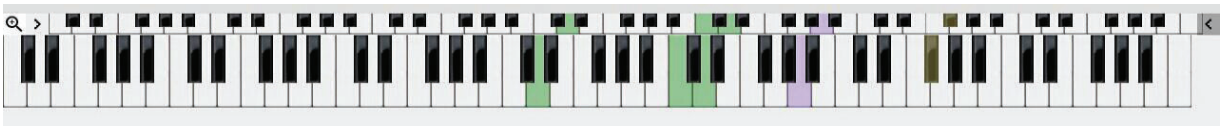


Figura A.15 Control del teclado virtual, referencia: Autores

El teclado virtual puede cambiar su tamaño con la lupa que está a la izquierda de la barra del teclado, existen ciertos tamaños establecidos de los cuales el usuario podrá escoger el que le parezca mejor.

A continuación se muestra los tamaños de teclado disponibles

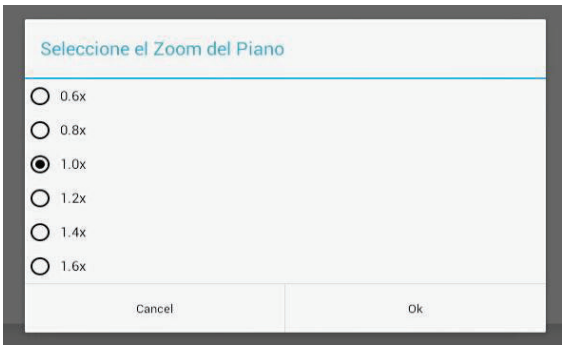


Figura A.16 Tamaño de teclado disponible, referencia: Autores

En la parte superior del teclado virtual también se puede apreciar una vista en miniatura del teclado virtual, esta vista con solo tocarla puede desplazar el rango visible del teclado, las flechas verdes de los extremos de este teclado miniatura cumplen la misma función de desplazar el rango visible del teclado pero de una manera distinta.

Karaoke

Si es que el fichero midi cargado en el software contiene líricas se mostrarán en azul las que ya pasaron según el tiempo actual de reproducción y en negro las que aún no pasan; si es que el fichero MIDI no contiene líricas simplemente el espacio para líricas se mostrará en blanco.

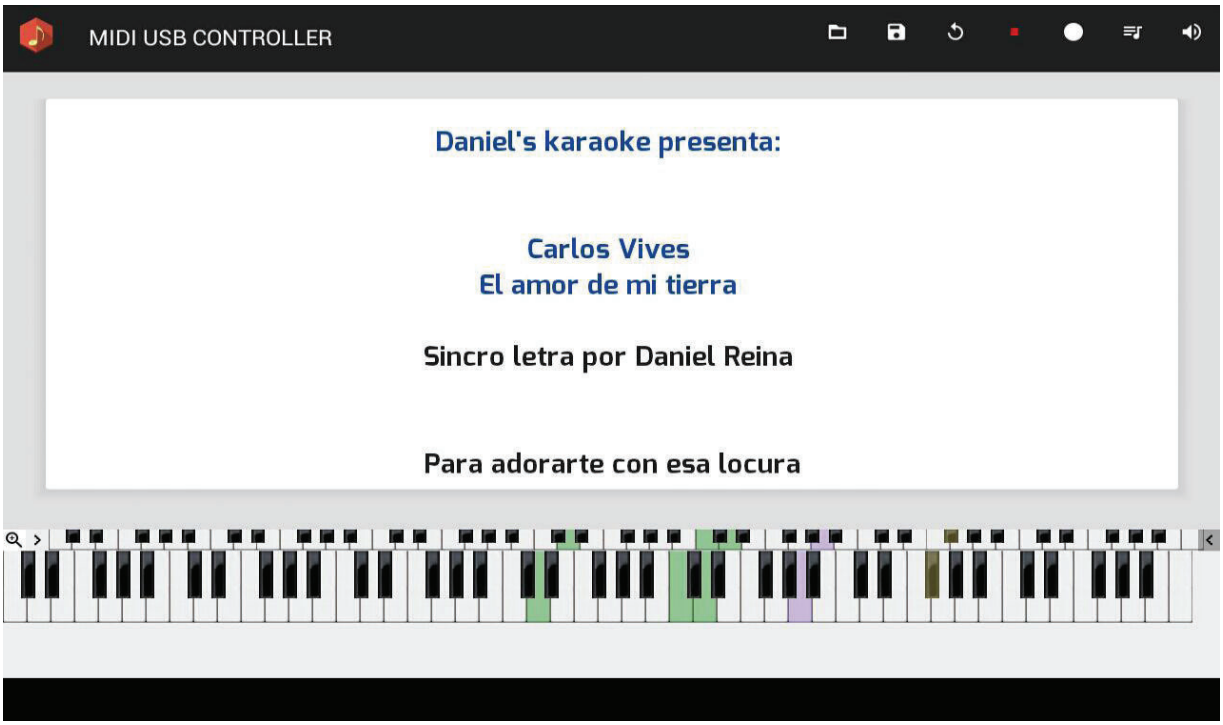


Figura A.17 Karaoke, referencia Autores

ANEXO B: ENCUESTA PARA OBTENCIÓN DE REQUISITOS

En la obtención de las respuestas a partir de las encuestas realizadas por internet para obtener los requisitos adecuados para la realización de la aplicación móvil pudimos algunos resultados muy útiles. La encuesta consta de 11 preguntas para saber el nivel de conocimiento de las personas encuestadas sobre la temática del proyecto. La mayoría de los encuestados son personas que poseen algún tipo de conocimiento sobre música y han utilizado algún software para sus necesidades. La encuesta fue realizada de manera anónima a 30 personas.

El link de la encuesta es el siguiente:

http://goo.gl/forms/w7qrNGhaAF

Las preguntas y los análisis de los datos obtenidos de estas se detallan a continuación:

1. **¿Posee usted un Smartphone o Tablet?**
    - Si
    - No

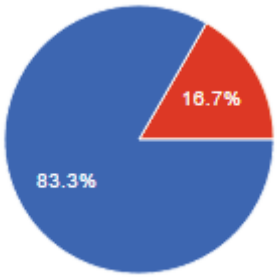


Figura B.18 resultados pregunta 1 creada por los autores

Total pregunta 1	
Sí	83.3%
No	16.7%

Tabla B.1 Resultados pregunta 1 creada por los autores

Como podemos observar la mayoría de los encuestados tiene un Smartphone o Tablet, gracias esto podemos decir que se justifica la realización de la aplicación ya que dentro del grupo de personas encuestadas es tan solo 16.7 % los que no poseen un dispositivo de este tipo.

2. **¿Con que sistema operativo trabaja su dispositivo?**
    - Android
    - iOS
    - otros

Figura B.19 resultados pregunta 2 creada por los autores

Total pregunta 2	
Android	80%
iOS	10%
otros	10%

Tabla B.2 Resultados pregunta 2 creada por los autores

Como podemos observar la mayoría de los encuestados poseen dispositivo Android, esto justifica la realización de la aplicación para dispositivos Android mayoritariamente, en cuento a los otros sistemas operativos móviles no representan un porcentaje muy grande.

3. **Si en la pregunta anterior escogió Android ¿Qué tipo de dispositivo Android posee?**
    - Tablet
    - Smartphone

Figura B.20 resultados pregunta 3 creada por los autores

Total pregunta 3	
Tablet	50%
Smartphone	50%

Tabla B.3 Resultados pregunta 3 creada por los autores

Según lo obtenido podemos ver que la mayoría de los encuestados tiene tanto un Smartphone como una Tablet lo que no muestra que la aplicación debería ser diseñada para ambos casos.

4. **¿Tiene usted conocimientos musicales, sabe tocar algún instrumento musical?**
    - Si
    - No

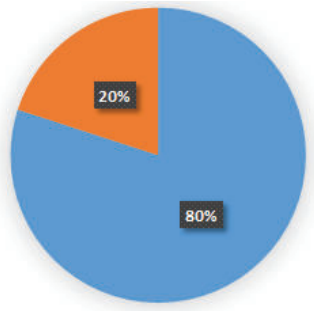


Figura B.21 resultados pregunta 4 creada por los autores

Total pregunta 4	
Sí	80%
No	20%

Tabla B.4 Resultados pregunta 4 creada por los autores

Los resultados muestran que entre los encuestados la mayoría tiene conocimientos de musicales y sabe tocar un instrumento musical, gracias a esto dentro de la encuesta podremos contar con información relevante para los usuarios objetivos de la aplicación.

5. **¿Conoce usted que es un archivo MIDI?**
   - Si
   - No

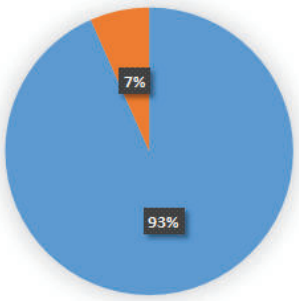


Figura B.22 resultados pregunta 5 creada por los autores

Total pregunta 5	
Sí	93%
No	7%

Tabla B.5 Resultados pregunta 5 creada por los autores

Los resultados muestran que casi la mayor parte de los encuestados tiene una idea de lo que es un archivo MIDI solo un bajo porcentaje de los encuestados no sabe que es un archivo MIDI.

6. **¿Sabe usted cómo se graba un archivo MIDI?**
   - Si
   - No

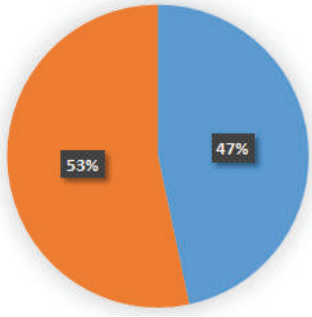


Figura B.23 resultados pregunta 6 creada por los autores

Total pregunta 6	
Sí	47%
No	53%

Tabla B.6 Resultados pregunta 6 creada por los autores

En el caso de esta pregunta pudimos observar que un buen porcentaje de los encuestados no tiene conocimiento de cómo se graba un archivo MIDI, esto es algo sorprendente ya que muchos saben lo que es un archivo MIDI pero no como se graba, por otra parte el porcentaje de personas que saben que es un archivo MIDI también es alto y eso es muy bueno.

7. **¿Tiene usted algún dispositivo MIDI (Sintetizador, guitarra MIDI, etc.)?**
   - Si
   - No

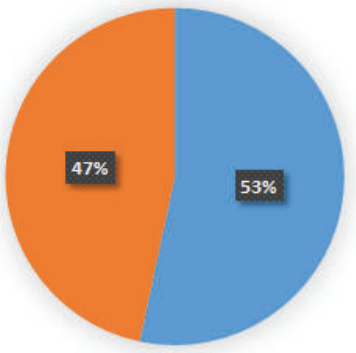


Figura B.24resultados pregunta 7 creada por los autores

Total pregunta 7	
Sí	53%
No	47%

Tabla B.7 Resultados pregunta 7 creada por los autores

Como vimos en los resultados casi son el mismo número de personas las que tienen un dispositivo MIDI como las que no lo poseen. En este caso podemos ver que un buen porcentaje de los encuestados podrá usar la aplicación y en el otro caso si llegan a tener un dispositivo MIDI van a poder llegar a usarlo.

8. **¿Le gustaría poder escuchar cualquier archivo MIDI en su dispositivo usando un dispositivo móvil Android?**
    - Si
    - No

Figura B.25 resultados pregunta 8 creada por los autores

Total pregunta 8	
Sí	100%
No	0%

Tabla B.8 Resultados pregunta 8 creada por los autores

De los resultados de las encuestas podemos ver que todos los encuestados les gustaría escuchar un archivo MIDI usando su dispositivo móvil.

9. **¿Le gustaría poder generar un archivo MIDI a partir de las notas que usted toca su dispositivo MIDI?**
    - Si
    - No

Figura B.26 resultados pregunta 9 creada por los autores

Total pregunta 9	
Sí	100%
No	0%

Tabla B.9 Resultados pregunta 10 creada por los autores

De los resultados de las encuestas podemos ver que todos los encuestados les gustaría generar un archivo MIDI a partir de las notas que hayan grabado de su dispositivo MIDI

10. **¿Conoce usted algún software que trabaje con archivos MIDI?**
    - Si
    - No

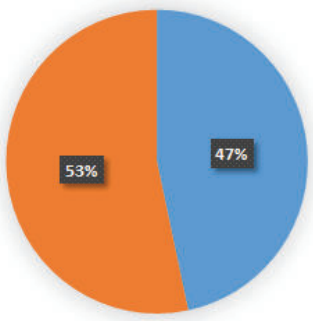


Figura B.27 resultados pregunta 10 creada por los autores

Total pregunta 10	
Sí	47%
No	53%

Tabla B.10 Resultados pregunta 10 creada por los autores

En este caso podemos ver que hay una pequeña diferencia entre los que conocen algún software que trabaje con archivos MIDI y los que no conocen pero casi están a la par.

11. **¿Le gustaría a usted tener un karaoke en su dispositivo móvil?**
    - Si
    - No

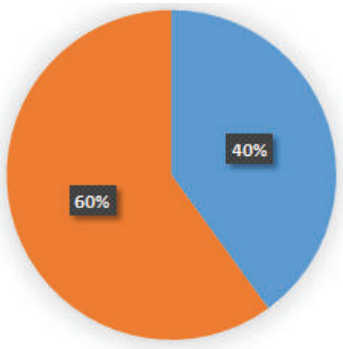


Figura B.28 resultados pregunta 11 creada por los autores

Total pregunta 11	
Sí	47%
No	53%

Tabla B.11 Resultados pregunta 11 creada por los autores

De lo que se puede ver mediante las respuestas de los encuestados podemos ver que un porcentaje grande no ve tan relevante la inclusión de un karaoke en la aplicación, pero también hay un buen porcentaje de personas que si les gustaría.

ANEXO C: ENCUESTA PARA VALIDACIÓN DE FUNCIONALIDADES

En la obtención de las respuestas a partir de las encuestas realizadas a un grupo de músicos de varios conservatorios de Quito después de haber probado la aplicación y su funcionamiento, se obtuvo los resultados para las pruebas de aceptación, validando así que la aplicación no tiene un margen de error alto y que todas las funcionalidades se han realizado según lo descrito en las historias de usuario.
La encuesta cuenta con 9 preguntas, fue realizada en total a 50 personas.

El link de la encuesta es el siguiente:

http://goo.gl/forms/rhUIKzCIc3

Las preguntas y los análisis de los datos obtenidos de estas se detallan a continuación:

1. **¿Pudo cargar un archivo MIDI sin problema?**
   - Si
   - No

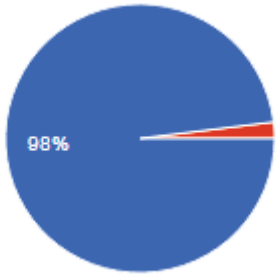


Figura C.29 resultados pregunta 1 encuesta C elaborados por autores

Total pregunta 1	
Sí	98%
No	2%

Tabla C.12 resultados pregunta 1 encuesta C realizada por los autores

De lo observado en las respuestas, se pudo ver que la aplicación pudo cargar un archivo MIDI, en el caso de las respuestas negativas como vemos son muy pocas, se debió a un error del dispositivo móvil en el cual se estaba probando.

2. **¿Se reprodujo correctamente el archivo MIDI que cargó?**

- Si
- No

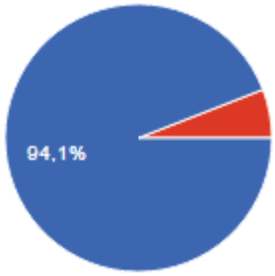


Figura C.30 resultados pregunta 2 encuesta C elaborada por los autore

Total pregunta 2	
Sí	94,1%
No	5,9%

Tabla C.13 resultados pregunta 2 encuesta C realizada por los autores

De lo observado se pudo reproducir correctamente el archivo MIDI con la aplicación, en los casos en el que no se pudo reproducir se vio que esto se debía al cable con el que se estuvo probando, no fueron muchas las respuestas negativas por eso se puede concluir que si se pudo reproducir tanto en el dispositivo móvil como en el dispositivo MIDI los archivos MIDI.

3. **¿Pudo modificar el volumen de los canales antes de la reproducción?**
   - Si
   - No

Figura C.31 resultados pregunta 3 encuesta C elaborados por autores

Total pregunta 3	
Sí	100%
No	0%

Tabla C.14 resultados pregunta 3 encuesta C realizada por los autores

Como se vio en las respuestas se pudo modificar el volumen de los diferentes canales de un archivo MIDI, no hubo respuestas negativas.

4. **¿Pudo pausar la reproducción correctamente?**
   - Si
   - No

Figura C.32 resultados pregunta 4 encuesta C elaborados por autores

Total pregunta 4	
Sí	100%
No	0%

Tabla C.15 resultados pregunta 4 encuesta C realizada por los autores

Como se vio en las respuestas se pudo pausar un archivo MIDI, se cumplió con esta funcionalidad de manera correcta.

5. **¿Se cargó un archivo que no tuviese las siguientes extensiones: .mid, .midi. .kar?**
   - Si
   - No

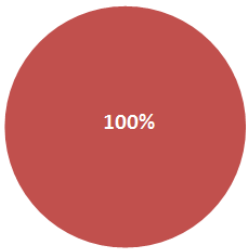


Figura C.33 resultados pregunta 5 encuesta C elaborados por autores

Total pregunta 5	
Sí	0%
No	100%

Tabla C.16 resultados pregunta 5 encuesta C realizada por los autores

De lo visto en las encuestas se pudo ver que no se pudo cargar ningún archivo que no tuviera el formato adecuado, solo los archivos con extensiones mid, midi o kar se cargaron.

6. **Después de haber grabado un archivo MIDI, ¿Se guardó correctamente?**
   - Si
   - No

Figura C.34 resultados pregunta 6 encuesta C elaborados por autores

Total pregunta 6	
Sí	100%
No	0%

Tabla C.17 resultados pregunta 6 encuesta C realizada por los autores

De lo obtenido a partir de las encuestas se pudo notar que no hubo errores en cuanto a grabar archivos MIDI.

7. **¿Pudo controlar el tempo de la grabación?**
   - Si
   - No

Figura C.35 resultados pregunta 7 encuesta C elaborados por autores

Total pregunta 7	
Sí	100%
No	0%

Tabla C.18 resultados pregunta 3 encuesta C realizada por los autores

Como se vio en las respuestas se pudo modificar el tempo de los diferentes canales de un archivo MIDI, no hubo respuestas negativas.

8. **¿Pudo ver en el piano las notas musicales al reproducirse la canción?**
   - Si
   - No

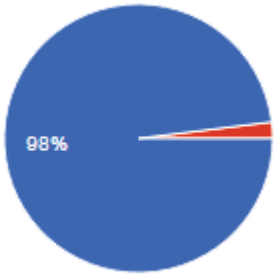


Figura C.36 resultados pregunta 8 encuesta C elaborados por autores

Total pregunta 8	
Sí	98%
No	2%

Tabla C.19 resultados pregunta 8 encuesta C realizada por los autores

De lo visto en las encuestas se vio que se cumplió con esta funcionalidad correctamente, en el caso del resultado incorrecto se puede observar que esto se dio ya que hubo un error en el dispositivo móvil.

9. **Si el archivo tenía líricas ¿Pudo visualizar éstas?**
   - Si
   - No

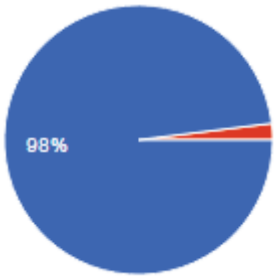


Figura C.37 resultados pregunta 9 encuesta C elaborados por autores

Total pregunta 9	
Sí	98%
No	2%

Tabla C.20 resultados pregunta 8 encuesta C realizada por los autores

De lo visto en las encuestas se vio que se cumplió con esta funcionalidad correctamente, en el caso del resultado incorrecto se dio ya que el archivo no contenía líricas pero el usuario no contemplo eso.

ANEXO D: MANUAL DE INSTALACION

1. Requisitos

Para el funcionamiento de la aplicación se tiene 3 partes, la primera el ordenador, la segunda el dispositivo móvil y por último el dispositivo MIDI.

A continuación detallamos sus características:

a. Ordenador

Los requisitos para el funcionamiento del ambiente de desarrollo Android Studio son:

- Windows 7 o posteriores.
- 2 GB RAM mínimo, recomendable 4 GB RAM.
- 1GB en disco duro mínimo.
- La resolución mínima de la pantalla es de 1280 x 800.
- Java Development Kit (JDK) mínimo en su versión 7.

Para más detalles ver la referencia [3].

b. Dispositivo Móvil

Los requerimientos para el buen funcionamiento de la aplicación son:

- Versión de Android 3.1 o superior
- Soporte USB Host

c. Dispositivo MIDI

El requisito que debe cumplir el dispositivo es tener una entrada y salida MIDI a USB.

2. Instalación

Copiar el instalador de la aplicación “MidiUSBController.apk” al sdcard de la Tablet.

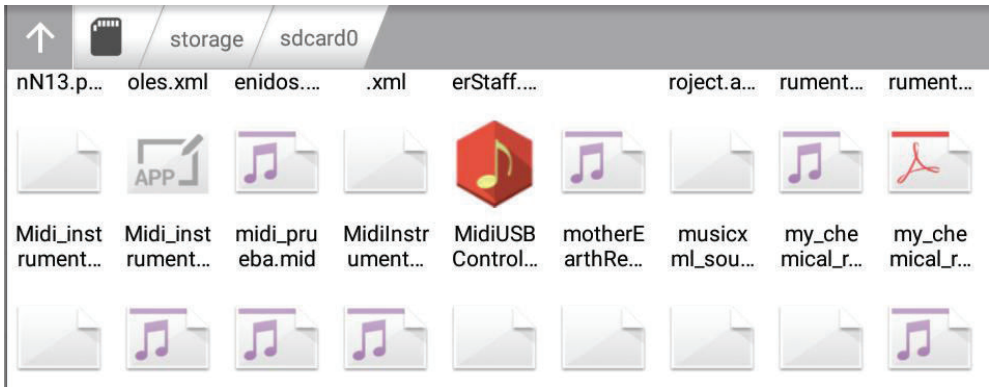


Figura D.38carpeta con archivo apk elaborado por los autores

1. Ejecutar el instalador de la aplicación y escoger la opción de instalar

Figura D.39 dialogo de instalación aplicación elaborado por los autores

2. Esperar que la aplicación se instale

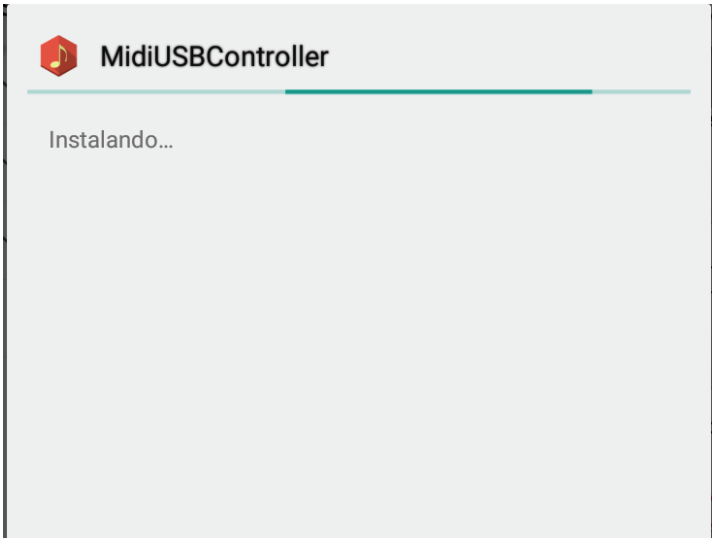


Figura D.40 Instalación aplicación elaborado por los autores

3. Abrir la aplicación presionando el botón abrir.

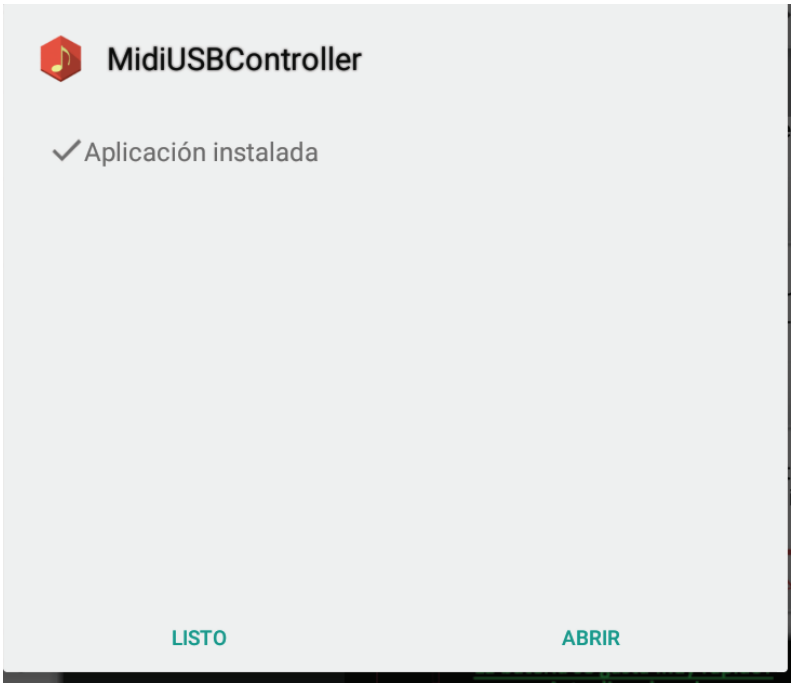


Figura D.41 Abrir aplicación creado por los autores

4. Usar la aplicación

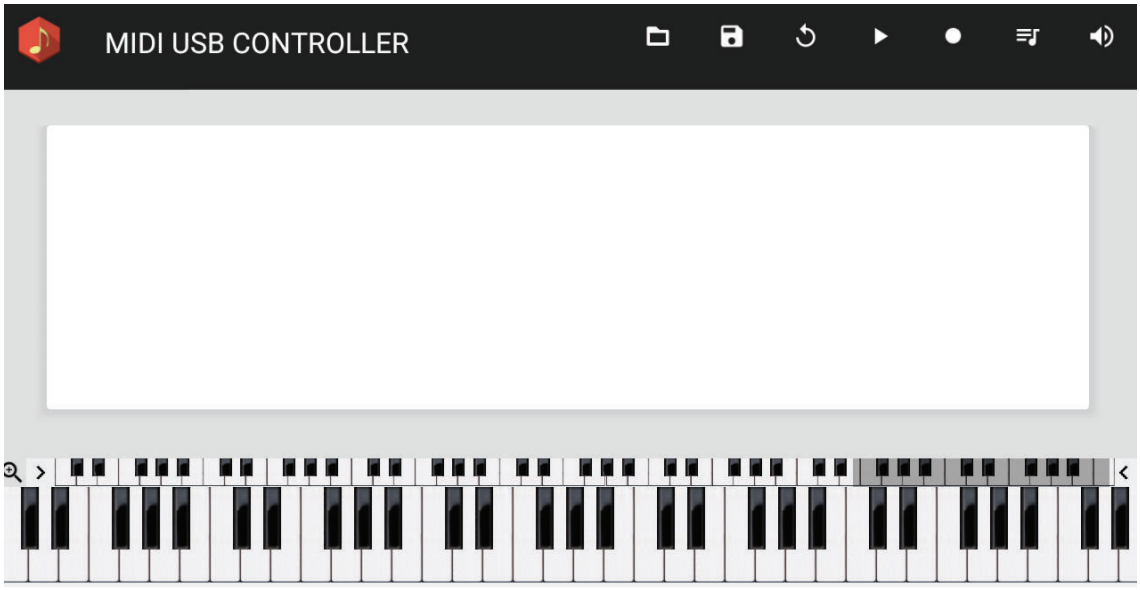


Figura D.42 Usar aplicación elaborada por los autores