Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tec. De Telecomunicaciones

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

SERVIDOR WEB PARA BUSCAR INTERPRETACIONES MUSICALES Y PROPORCIONAR DATOS DE OYENTE A PARTIR DE LA PARTITURA MUSICAL

Septiembre, 2016

ÍNDICE DE CONTENIDOS

1. RESUMEN

El propósito del presente proyecto consiste en el alineamiento de partituras digitales con el audio de la interpretación, por medio del lenguaje Matlab empleando el programa de alineamiento basado en Dynamic Time Warping, con el fin de comprobar si se trata de una interpretación de la partitura y de identificar los inicios de compases para que así queden sincronizados correctamente la reproducción de la interpretación.

A lo largo de esta memoria, se explicará el programa citado anteriormente, y el procedimiento seguido hasta visualizar en una página web el vídeo de la interpretación, la partitura digital y datos de oyente para que el usuario pueda obtener información acerca de la obra, y pueda ir añadiendo información por cada compás en un editor de comentarios.

También se verá el servicio implementado para conseguir dicho fin, basado en un sistema de colas y peticiones para que Matlab vaya procesando dicho par de audio y partitura digital y pueda ir guardando en una base de datos el vector con los inicios de compases correspondientes para cada interpretación de la que el usuario ha hecho una petición.

Se describirán toda la tecnología y lenguaje utilizado en este trabajo, todo ello realizado en el sistema operativo Linux, para el desarrollo del esquema cliente servidor, cómo se van procesando las peticiones y la lógica de control del proceso, así como las herramientas utilizadas para la visualización en el navegador del usuario.

2. INTRODUCCIÓN

A lo largo de los últimos siglos, los avances y desarrollos tecnológicos han supuesto un cambio en la vida cotidiana de las personas, ya que continuamente estamos ligados a las tecnologías de la información y la comunicación; siempre teniendo como objetivo facilitar, dar fluidez y más rapidez que el movimiento físico.

Todo ello ha afectado a múltiples tecnologías como el teléfono, la radio, la televisión, internet, y como no podía ser de otra forma, a la música. A nivel académico se van desarrollando las tecnologías de la información, que buscan estudiar cómo hacer llegar los mensajes o información a un público cada vez mayor.

Hoy en día, cualquier persona del mundo desarrollado puede acceder a múltiple información, siendo posible guardarla y compartirla en múltiples soportes y de diferentes formas como por ejemplo: textos, imágenes, sonidos y música.

Como no podía ser de otra manera, el campo musical también ha ido evolucionando, tanto para nivel académico como profesional en orquestas, obras o partituras.

A finales del siglo XX y siglo XXI se ha desarrollado un gran interés por la representación de las partituras en un formato legible para ordenador, tablets y smartphones, con el objetivo de leer partituras escaneadas de forma que los resultados puedan ser después manipulados, e incluso la reproducción MIDI (Musical Instrument Digital Interface).

Este campo continuamente está evolucionando, hasta el punto de tener métodos para coordinar la visualización de la música entre los intérpretes de una orquesta, bibliotecas virtuales para partituras digitales de dominio público, así como partituras de compositores que están dispuestos a compartir su música con el mundo de forma gratuita.

2.1. Partituras digitales en formato XML

Actualmente encontramos diversos formatos para compartir partituras digitales, tanto con otros usuarios, como para editarlas o trabajar con diferentes programas informáticos.

Este trabajo se centra en partituras musicales en formato XML, ya que MusicXML es el estándar universal de música escrita y puede ser usado por la mayoría de editores actuales, a continuación en este apartado se va a detallar en qué consiste este formato.

2.1.1. ¿Qué es el formato XML?

XML, siglas en inglés de eXtensible Markup Language (“lenguaje de marcas extensible”), es un lenguaje de marcas utilizado para almacenar datos de forma legible, que permite definir etiquetas personalizadas para la descripción y organización de datos.

A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí. Es una tecnología sencilla y fácil de estructurar, y tiene gran importancia en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura y fiable.

2.1.2. Ventajas del XML

- Extensible: Es posible extender un XML ya diseñado simplemente con la adición de nuevas etiquetas, para que se pueda continuar utilizando sin apenas dificultad.
- Si un usuario decide usar un documento ya creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones, comunicándolas entre ellas, ya que está diseñado para cualquier lenguaje.
- Separa el contenido y el formato de presentación, con mucha flexibilidad para estructurar y ver documentos, ya que no existe un visor genérico de XML.

2.1.3. Estructura de un documento XML

La tecnología XML se caracteriza por expresar información estructurada. Que la información sea estructurada se refiere a componer las partes bien definidas, y que esas partes se componen a su vez de otras partes, obteniendo una especie de árbol de trozos de información.

Los componentes esenciales de este formato son las etiquetas y los elementos:
Una etiqueta es una marca hecha en el documento, que señala una porción de éste como un elemento, trozo de información con un sentido claro y definido. Las etiquetas tienen la forma <ejemplo>, donde *ejemplo* es el nombre del elemento que se está señalando.

A continuación, en la Figura 1, se muestra un ejemplo con la estructura de un documento XML:

Figura 1. Estructura de un documento XML

El ejemplo más claro es un tema musical, una obra se compone de compases, que están formados a su vez por notas, dónde a estas partes se les llama elementos, y las señala mediante etiquetas. Mencionar que existen también archivos en formato MXL pero no es más que un formato XML pero comprimido.

2.1.4. Componentes de un documento XML

En un documento XML existen los siguientes componentes:

- Elementos: Pieza lógica del marcado, se representa con una cadena de texto encerrada entre etiquetas. Los elementos pueden contener atributos.
- Instrucciones: Ordenes especiales para ser utilizadas por la aplicación que procesa. <?xml-stylesheet type="text/css" href="estilo.css">
- Comentarios: Información que no forma parte del documento.
- Declaraciones de tipo: Especifican información acerca del documento. Comienzan por <!-- y terminan por -->
  <!DOCTYPE persona SYSTEM "persona.dtd">
- Secciones CDATA: Se trata de un conjunto de caracteres que no deben ser interpretados por el procesador. <![CDATA[Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>

Un ejemplo de un documento XML con estos componentes explicados se puede ver en la figura 2, donde se puede apreciar con claridad la estructura de estos tipos de documentos.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE peliculas SYSTEM "Peliculas.dtd">

<peliculas>
 <pelicula tipo="comedia" sistema="PG-13" ejemplares="5" año="1987">
  <titulo>Raising Arizona</titulo>
  <guionista>Ethan Coen</guionista>
  <guionista>Joel Coen</guionista>
  <productor>Ethan Coen</productor>
  <director>Joel Coen</director>
  <actor>Nicolas Cage</actor>
  <actor>Holly Hunter</actor>
  <actor>John Goodman</actor>
  <comentarios>Una pelicula clasica de Comedia.</comentarios>
 </pelicula>

 <pelicula tipo="ciencia-ficcion" sistema="PG-13" ejemplares="4" año="1989">
  <titulo>The Abyss</titulo>
  <guionista>James Cameron</guionista>
  <productor>Gale Anne Hurd</productor>
  <director>James Cameron</director>
  <actor>Ed Harris</actor>
  <actor>Mary Elizabeth Mastrantonio</actor>
  <comentarios>Una buena pelicula.</comentarios>
 </pelicula>
</peliculas>
```

Figura 2. Componentes y estructura de un documento XML

2.1.5. Representación musical en formato MusicXML

La representación musical de las partituras puede representarse en un formato MusicXML, que está basado en el lenguaje XML. En el cual se describen de una manera muy similar los aspectos de interpretación de una obra, dónde en un archivo XML las notas y medidas son escritas en formas de código, de esta manera una máquina puede almacenar toda aquella información como la armadura y características de una canción.

Toda la información relativa a la partitura se guarda dentro de la etiqueta *score-partwise* y a partir de aquí se van añadiendo los primeros hijos. A continuación, se van a detallar los primeros hijos a partir de la etiqueta anterior, los elementos más importantes y como se va estructurando la partitura en este formato.

a) *part-list*: es una lista generalizada de todos los diferentes componentes de la partitura como los instrumentos entre otros.
   - *score-part:* exponer cómo una parte (part) está contenida en la notación, por ejemplo con un id: <score-part id =”P1”>.
   - *part-name:* indica el nombre completo de un componente de la composición, por ejemplo: <part-name>Violin</part-name>.

b) *part:* es el contenedor donde se encuentran todos los compases con sus notas por instrumento, contiene la partitura en sí de cada instrumento, la notación de los instrumentos se dividen por *part,* mostrando el id de este de la siguiente manera: <part id=”P1”>.

- *measure: measure* en inglés americano es compás. Es el contenedor de la partitura, dentro de la etiqueta puede contener información adicional como el número de compás entre otros. Por ejemplo <measure number=”1”>.
- *attributes:* contiene información relativa a la musicalidad de la pieza como la armadura, la métrica y la clave. Típicamente, se encuentra solo dentro del primer compás de cada instrumento, pudiendo aparecer en otros compases si hay algún cambio en las características mencionadas.
- *divisions:* indica el número de partes en que se puede dividirse una negra. Las negras que aparezcan en la partitura tendrán la duración indicada aquí y todas las otras figuras tendrán una duración relativa a este número, por ejemplo la duración de la blanca será el doble de esta, la corchea la mitad y, así sucesivamente.
- *key:* nos informa de la armadura del instrumento. Por medio de *fifths* el número de sostenidos o bemoles que tiene la armadura, y *mode* que indica el modo de la armadura, si es menor o mayor.
- *time:* aquí se indica el tipo de compás de la partitura. Por medio de *beats,* indicando el número de beats que puede caber en un compás y *beat-type* que dice la duración de un beat, si es 1 tendrá una duración de una redonda, si es 2, una blanca, si es 4, una negra, si es 8 una corchea y así sucesivamente.
- *direction:* en esta parte se proporciona información que no está relacionada con ninguna nota en especial pero sí con la partitura. Por ejemplo, *metronome* nos dice que dentro de él tendremos información sobre el tempo de la pieza, *per-minute* expresa cuantos beats habrá en un minuto, aquí se indica el tempo en beats por minuto (BPM).
- *note:* aquí se detallan todas las características de una nota. Todas las notas que se encuentran en la partitura vienen indicadas con este elemento. Esta etiqueta puede contener información como la posición de la nota en la partitura. Se puede expresar de la siguiente manera <note defauld-x=”48”>. *Pitch o rest* dependiendo de si tenemos una nota sonora o un silencio respectivamente. En el caso de rest no se escribiría nada sólo <rest/>, en cambio, si

tenemos una nota sonora se escribe <pitch> y dentro de este elemento tendremos las características en términos del todo de la nota, como el nombre de la nota musical en notación anglosajona (C, D, E, F…), la octava en la que se encuentra la nota, la duración de la nota, etc.

En la Figura 3, se puede ver la estructura de una obra musical en formato XML tras haber explicado su estructura y elementos más importantes.

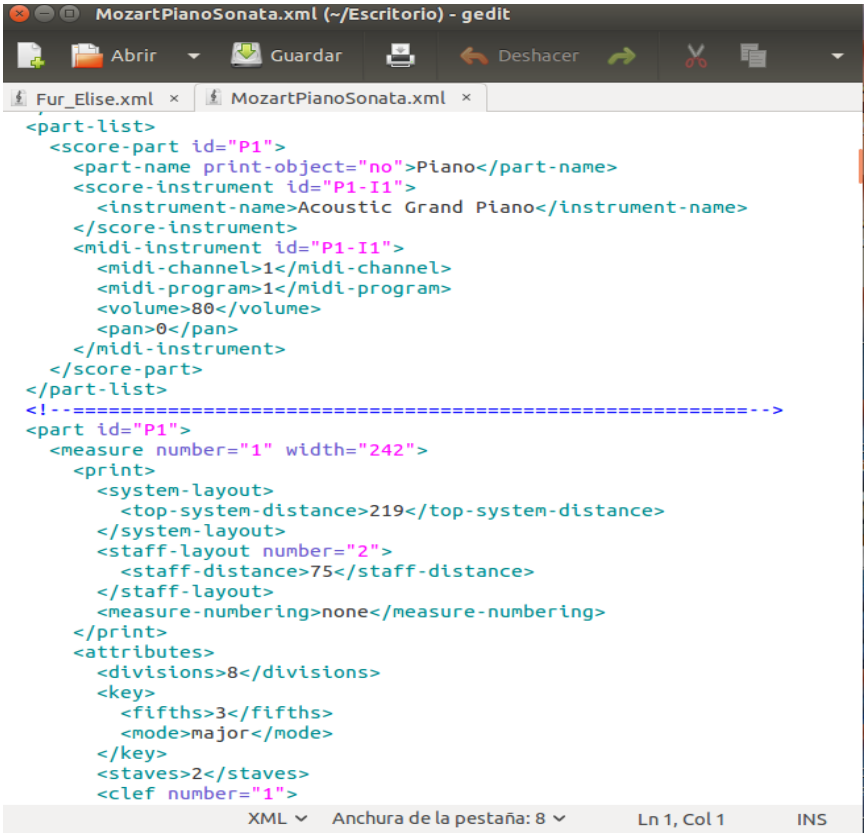


Figura 3. Estructura partitura musical en formato XML

Mencionar que estas partituras digitales se pueden descargar de librerías de partituras virtuales como “IMSLP Petrucci Music Library” [1], “Musescore Sheet Music” [2], “Open Music Score” [3] y “musicXML. Example Set” [4]. Éstas han sido los más empleados a la hora de descargar partituras musicales en formato XML utilizadas para el desarrollo del proyecto, son páginas de dominio público donde los usuarios comparten y editan una gran diversidad de partituras digitales en este tipo de formato entre otros.

2.1.6. Aplicaciones y herramientas para trabajar con XML

Algunas aplicaciones de XML es descripción de metacontenidos, formatos de mensaje para comunicación entre aplicaciones (B2B) y publicar e intercambiar contenidos de bases de datos.

Como se ha mencionado con anterioridad, la sencillez y estructura de este formato hace que pueda ser utilizado por cualquier procesador de texto, que sea capaz de producir archivos .txt, ya que por lo tanto será capaz de generar XML.

Dado a que el presente proyecto consiste en procesar partituras digitales en formato XML, mencionar algunos programas que realizan esta función, hay muchos programas para escribir y editar música. Entre las opciones comerciales más conocidas se tienen programas como *Sibelius, Finale, Musescore o Keynote Music.* Pero la ventaja que presenta Musescore para cualquier usuario es que es gratuito para cualquier usuario que quiera editar música y está disponible para distintos sistemas operativos.

2.1.6.1. MuseScore

Este programa es el que se ha utilizado en la realización de este proyecto, a la hora del procesamiento de partituras digitales, dado sus múltiples funcionalidades. Entre ellas destacar su utilización por línea de comandos, simplemente con el comando *mscore,* lo que ha sido de gran utilidad en el proceso del presente trabajo.

Como se ha mencionado anteriormente este programa está disponible en distintos sistemas operativos como Windows, OS X y Linux para su descarga [5] y funcionamiento. Puede importar y exportar tanto archivos MIDI estándar como el formato MusicXML, también puede importar y exportar las notaciones en formato PDF o PNG y a WAV, y soporta ingreso de notas con ratón, teclado de ordenador, y teclado MIDI. La página oficial [6] ofrece un rápido tutorial basado en diez cortos vídeos que enseñan las principales virtudes de MuseScore, junto con algunos detalles de su funcionamiento. Es un programa de código abierto: crea, reproduce, e imprime partituras de forma gratuita y compartición de partituras en red.

Implementa una interfaz gráfica fácil de usar con un reproductor de partituras llamativo a vista del usuario, como se puede ver en la Figura 4.

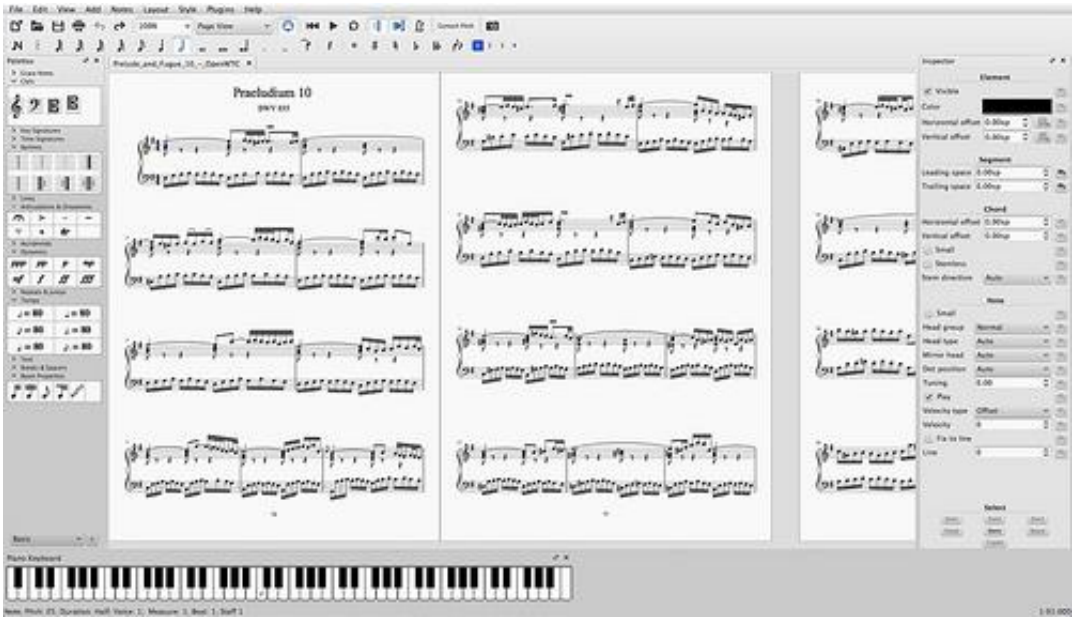


Figura 4.Interfaz gráfica programa MuseScore

2.2. Visualizador de partituras en el navegador de usuario con abcweb.

En este apartado se va a explicar la herramienta y software de abcweb, que es un programa de JavaScript que realiza un archivo de partitura de formato MusicXML o ABC, y reproduce un archivo multimedia tanto de audio, como de video, donde una vez cargado por parte del usuario se puede visualizar ambos archivos en el navegador conjuntamente sincronizados, por tanto nos permite visualizar partituras digitales en formato XML que se detalló en apartados anteriores. Este software fue desarrollado por el Profesor Wim G. Vree. [7]

Este programa presenta un cursor en la partitura que permite la sincronización con la reproducción. Todo el software es JavaScript que se ejecuta en el navegador de usuario, no hay ningún servidor involucrado ni pre-procesamiento necesario; el usuario sólo tiene que seleccionar su propio archivo de partitura con un archivo multimedia que se desee visualizar, se puede descargar, consultar y probar este software de la página oficial [8] donde aparece todo este software que se detallará posteriormente.

A continuación se va a explicar en qué consiste el formato de notación abc, características de esta herramienta y algunos ejemplos visuales donde se podrá ver lo comentado anteriormente.

2.2.1. Notación musical en formato abc

La notación musical abc es un lenguaje para escribir música que utiliza el conjunto de caracteres ASCII, es un sistema muy práctico para su uso en ordenadores e internet debido al poco espacio que ocupan los documentos y puede utilizarse desde cualquier editor de texto para editar música, aunque existen distintos paquetes de software, como el utilizado en el presente proyecto abcweb, que permiten leer y procesar música escrita en sistema abc, con la disponibilidad de utilizarse en diversos sistemas operativos. El formato abc fue desarrollado por Chris Walshaw [9]

A modo de ejemplo de cómo funciona este formato de notación musical, se puede ver en la Figura 5, como una partitura se podría escribir en formato ABC.

```
X:1
T:Speed the Plough
M:4/4
C:Trad.
K:G
|:GABc dedB|dedB dedB|c2ec B2dB|c2A2 A2BA|
  GABc dedB|dedB dedB|c2ec B2dB|A2F2 G4:|
|:g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df|
  g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|
```

Figura 5. Notación musical en formato ABC

Donde las líneas de la primera parte de la notación, que comienzan con una letra seguida por el signo de dos puntos, indican diversa información de la melodía. En el caso de que hubiese más de una melodía en un archivo (X:), el título (T:), el compás (M:) y la clave (K:). Las líneas que siguen a la indicación de la clave son la melodía propiamente dicha, el conjunto de notas, compases, silencios y distintos elementos que compongan la partitura musical.

Este ejemplo anteriormente expuesto, puede ser procesado a notación musical tradicional utilizando una de las herramientas de conversión abc existentes, obteniendo el aspecto que se puede ver en la Figura 6.

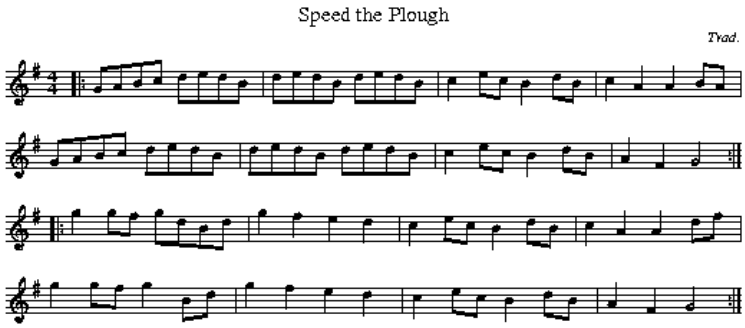


Figura 6. Resultado de procesar notación ABC

2.2.2. Características de la herramienta abcweb.

Algunas características de este software se pueden ver a continuación en este subapartado, tanto la herramienta utilizada para visualización con abc2svg, como la lectura de archivos en ABC o MusicXML y la reproducción entre otras características.

2.2.2.1. Scalable Vector Graphics con abc2svg

La partitura se representa mediante el visor SVG (Scalable Vector Graphics) con abc2svg, que permite visualizar y reproducir música desde la fuente ABC en los navegadores web. El software contiene el analizador del archivo ABC y el motor de generación, utilizándose directamente en este ABC visor/editor, permitiendo la visualización de los archivos en el navegador.

2.2.2.2. Leer archivos ABC o XML con xml2abc

Esta herramienta, se ejecuta en el navegador y convierte archivos XML seleccionados por el usuario en código ABC, o permite la lectura directamente de archivos en formato ABC, en la Figura 7 se puede cómo se cargan los archivos. También permite la opción de configurar todas las opciones soportadas y ver el efecto inmediato en vista previa, y guardar el resultado de un archivo en notación abc como local.

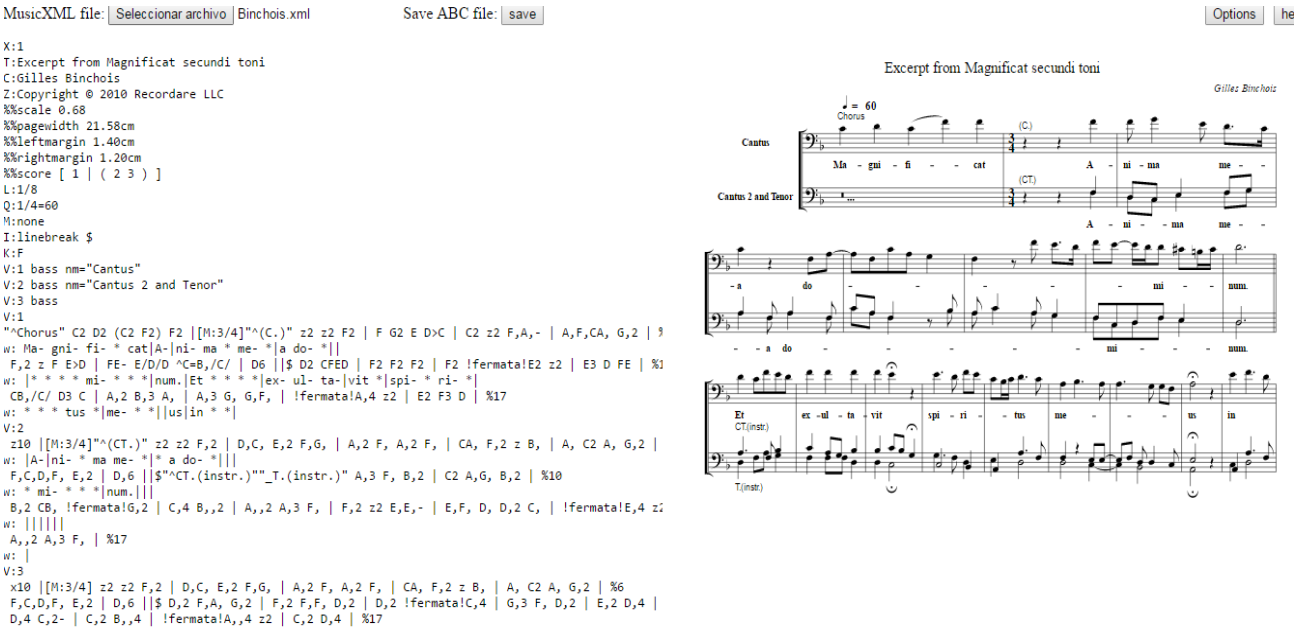


Figura 7. Lectura de partitura musical en formato MusicXML con xml2abc.js

Un requisito de esta herramienta es que el navegador del usuario deberá soportar html5 para la interfaz. Existe otro software idéntico a xml2abc.js que está escrito en JavaScrit es xml2abc.py, que permite traducir MusicXML a notación ABC pero desde línea de comandos.

2.2.2.3. Traducir MusicXML a notación ABC desde línea de comandos.

Con xml2abc se puede traducir MusicXML a notación ABC desde la línea de comandos directamente, dependiendo de si disponemos del programa Python instalado o se utiliza un ejecutable, con la posibilidad de configurar los parámetros. La línea de comandos para ambas opciones es la siguiente.

- Cuando se tiene Python instalado:

```
python xml2abc.py [-h] [-u] [-m] [-c C] [-d D] [-v V] [-n CPL] [-b BPL] [-o DIR] [-x] [-p FMT]  [..]
```

- Cuando se utiliza un ejecutable Win32:

```
xml2abc.exe [-h] [-u] [-m] [-c C] [-d D] [-v V] [-n CPL] [-b BPL] [-o DIR] [-x] [-p FMT]  file1 [file2...]
```

2.2.2.4. Otras características.

- Reproduce MP3 o OGG de audio y MP4 o webm de vídeo.
- Se pueden cargar y sincronizar vídeos directamente desde Youtube.
- Partituras y archivos multimedia pueden ser abiertos y guardados localmente o desde Dropbox.
- Permite precargar partituras, archivos multimedia y datos de sincronización.
- Muestra un cursor de línea o un medidor sombreado o ningún cursor (en este caso sólo desplazamiento de puntuación).
- Sincronización automática de audio sintetizada si la partitura tiene indicadores de tempo.
- Sincronización manual clickeando en el primer bit de un compás cuando suena.
- Puede mostrar un metrónomo unido a la medida de compás.
- Se pueden sincronizar múltiples dispositivos para la dirección de orquesta (probado con iPads).

- Existe la posibilidad de importar los mismos datos de tiempo en diferentes partes de una partitura (dirección de orquesta).

2.2.3. Demo de abcweb

En este subapartado, se va a mostrar a través de la Figura 8 lo anteriormente explicado en su conjunto, es decir, como una vez ejecutado abcweb se visualiza en el navegador del usuario, con sus distintos componentes y elementos detallados con anterioridad.

En este caso se ha cargado una partitura en formato XML y un archivo multimedia desde el navegador como local, es decir, con archivos desde el disco duro; y respecto al archivo multimedia cargado desde Youtube.

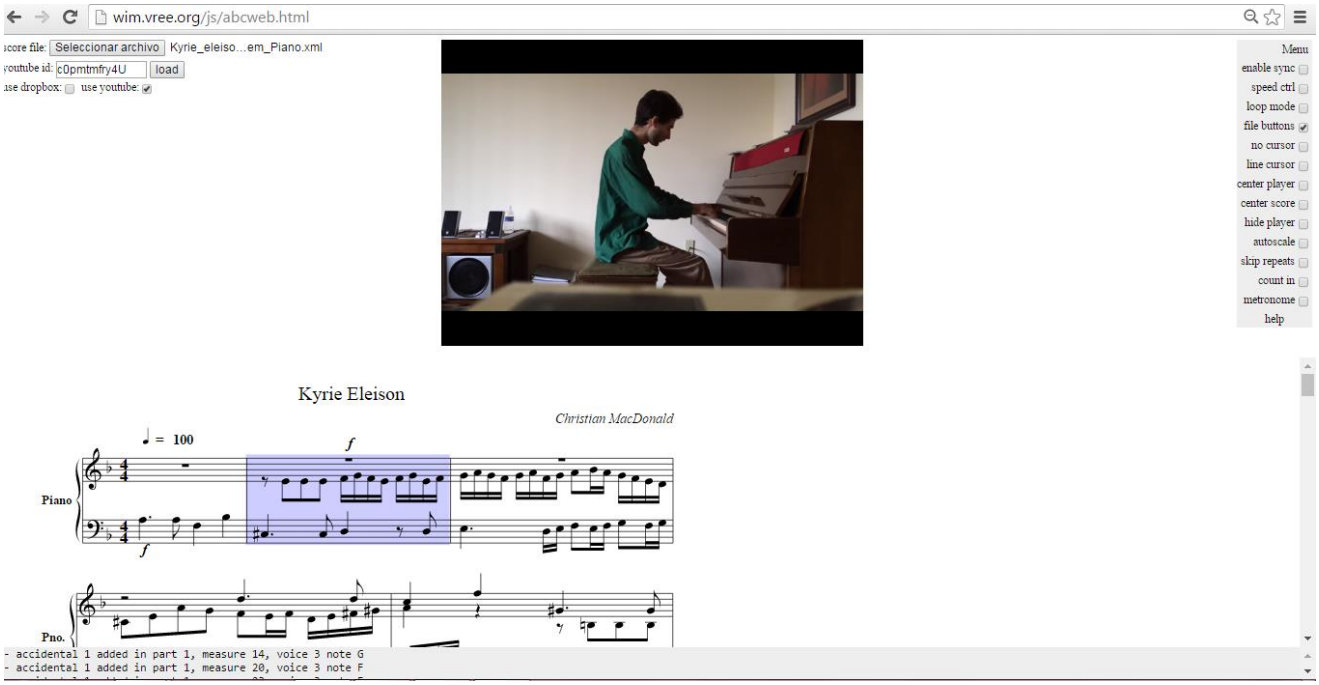


Figura 8. Ejemplo abcweb

2.3. Bases de datos

Dado que para el desarrollo del proyecto se ha realizado un servicio web junto con una base de datos para ir gestionando peticiones por parte del usuario, se vio la necesidad de indagar en tecnologías y herramientas que permitieran realizar todo este proceso, poder conectar con la base de datos a través de una aplicación web y para toda su estructura han sido necesarias tecnologías en las que se basa un esquema cliente-servidor, y que sin ellas no hubiese sido posible realizar todo el proceso creado.

Para comenzar con este apartado y entender qué es una base de datos, se podría definir como un almacén dónde se almacenan grandes volúmenes de información.

En este apartado se explicarán las diferentes alternativas y herramientas para el almacenamiento de los datos del sistema, cómo se accede a la base de datos y las herramientas usadas en el presente proyecto para llevarlo a cabo.

2.3.1 MySQL

MySQL (My Structured Query Language) es un sistema de gestión de bases de datos, un sistema que nos permite a través de una serie de sentencias, tener una información almacenada en una base de datos, recuperarla en el momento en el que la necesitemos de una forma eficiente y rápida. Está disponible para distintos sistemas operativos, como Linux, Mac OS X, Solaris, Windows y otros más y dispone de una página oficial en la que ofrece manuales e instrucciones para su implementación “MySQL Documentation” [10]. MySQL es muy popular en el desarrollo de aplicaciones web, ya que forma parte como sistema gestor de bases de datos de las plataformas LAMP, BAMP, MAMP y WAMP.

Las principales características de este gestor de bases de datos son:

- Escrito en C y C++.
- Probado en un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- Proporciona sistemas de almacenamiento transaccionales y no transaccionales.
- Un sistema de reserva de memoria muy rápido.
- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host.

2.3.2. PHP

PHP (Hipertext Preprocessor) es un lenguaje de programación diseñado para producir sitios web dinámicos y que puede ser incrustado en HTML, es decir, que se pueden combinar ambos códigos. PHP es utilizado en aplicaciones del lado del servidor aunque puede ser usado también desde una interfaz de línea de comandos o como aplicación de escritorio, se dispone de muchos manuales y tutoriales acerca de este lenguaje “Manual de PHP” [11]. Por medio de este lenguaje, se puede conectar con la

mayoría de bases de datos que se utilizan en la actualidad, por ejemplo el citado anteriormente MySQL.

- ¿Cómo trabaja PHP?

El lenguaje PHP se procesa en servidores, cuando se escribe una dirección tipo http://www.ujaen.es/index.php en un navegador web como Internet Explorer, Firefox o Chrome, se envían los datos de la solicitud al servidor que los procesa, reúne los datos (por eso se dice que es un proceso dinámico) y el servidor lo que devuelve es una página HTML como si fuera estática.

Las principales características son:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Lenguaje fácil de aprender y libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Capacidad de conexión con la mayoría de motores de bases de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Resaltar su flexibilidad, ha tenido una gran acogida como lenguaje base para aplicaciones web de manejo de contenido, y es su uso principal.
- Amplia documentación en su sitio web oficial.
- Capacidad de expandir su potencial utilizando módulos.

PHP es reemplazado a veces por lenguajes como Perl o Python, y el acrónimo se mantiene.

2.3.3 Apache HTTP Server.

El servidor Apache HTTP, también llamado Apache, es un servidor web HTTP de código abierto para la creación de páginas y servicios web. Es un software completamente gratuito y de código abierto como se puede ver en “Documentación del Servidor HTTP Apache 2.0” [12], tiene la disponibilidad de instalarlo en muchos sistemas operativos.

Algunas funcionalidades de Apache son:

- Atender de manera eficiente, debido al gran número de peticiones HTTP que puede recibir.

- Gestión de autentificaciones de usuarios o filtrado de peticiones según el origen de éstas.
- Manejar los errores por páginas no encontradas, informando al visitante y/o redirigiendo a páginas predeterminadas.
- Gestión de la información a transmitir en función de formato e informar adecuadamente al navegador que está solicitando dicho recurso.
- Gestión de logs, es decir almacenar peticiones recibidas, errores que se han producido y en general toda aquella información que puede ser registrada y analizada posteriormente para obtener las estadísticas de acceso al sitio web.

2.3.4 Tipos de servidores

En este apartado se va a explicar de manera breve y concisa, qué tipos de servidores informáticos hay [13], que incluyen todas las herramientas anteriores en el mismo paquete y para qué sirve cada uno.

- XAMPP: Es un servidor independiente de software libre que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl, proviene del acrónimo de X, (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOs X.
- MAMP: El acrónimo MAMP se refiere al conjunto de programas software comúnmente usados para desarrollar sitios web dinámicos sobre sistemas operativos Apple Macintosh, Mac OS X. Este nombre proviene de las iniciales Mac Os X, como sistema operativo. Apache, como servidor web. MySQL sistema de gestor de bases de datos y PHP, Perl o Python, lenguajes de programación usados para la creación de sitios web.
- LAMP: Es el acrónimo utilizado para describir un sistema de infraestructura de internet que usa las siguientes herramientas: LINUX como sistema operativo, Apache como servidor web, MySQL como gestor de base de datos y Perl, PHP y Python como lenguajes de programación.
- WAMPP: Es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas: Windows como sistema operativo, Apache como servidor web, MySQL, como gestor de bases de datos y PHP, Perl o Python como lenguajes de programación.

Como se ha podido observar todos estos servidores son similares, con la diferencia del sistema operativo en el que se puede utilizar, cabe mencionar que, para el desarrollo de este proyecto, al trabajar sobre el sistema operativo Linux, se ha utilizado el servidor Lamp donde toda la información para su instalación e información de configuración con todas las herramientas descritas anteriormente se puede ver en [14]

2.4 Fundamentos para el análisis de la música y su representación

Para el desarrollo adecuado del presente trabajo, se requirió unos previos conocimientos sobre las principales herramientas para el procesamiento de la señal y el tratado de la señal. En este apartado se verá de una forma sintetizada estas herramientas que ayudarán a entender mejor el desarrollo.

2.4.1. Procesamiento Digital de señales

El procesamiento digital de señales es un área de la ingeniería que ha crecido en los últimos años, por tanto, supone un crecimiento en las tecnologías, dando así muchas más facilidades en el manejo de la información.

En primer lugar, se dirá qué es una señal; está definida como cualquier cantidad física que varía con el tiempo, como se puede ver en la Figura 9. Por otro lado, matemáticamente, se puede describir como una función con una o más variables independientes (1).

$$s1(t) = 3t \qquad (1)$$

Las señales se pueden encontrar diariamente en diferentes áreas de la vida cotidiana, desde nuestra voz al hablar al producir perturbaciones en el aire, dispositivos móviles como teléfonos, la electricidad, etc.

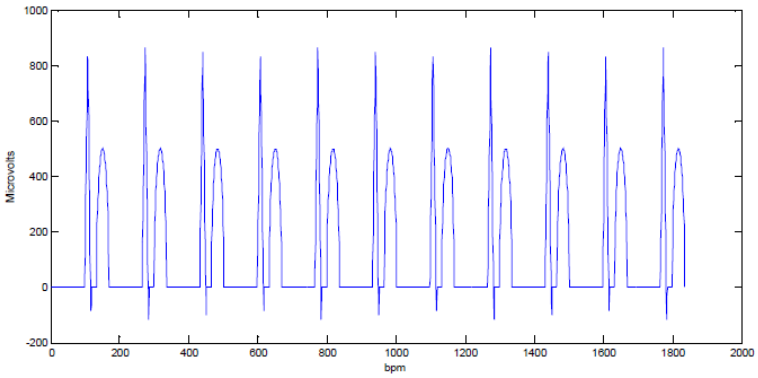


Figura 9. Señal de electrocardiograma

Para cada caso particular de una señal, la cual lleva información específica, como un mensaje o un sonido (nota musical), la señal está definida por parámetros característicos como son amplitud, frecuencias y cambios de fases que esta pueda tener, donde estos parámetros básicos son la base del procesamiento de una señal.

Una señal que es objeto de estudio será una señal de audio específicamente música. A diferencia de una señal de voz (30Hz-3000Hz) el audio en una pieza musical puede estar presente dentro del umbral de audición humana (20Hz-22010Hz).

2.4.2. Transformada de Fourier

La transformada de Fourier es una herramienta matemática que se podría considerar la más importante para el procesamiento de audio. Ya que asigna una función dependiente del tiempo a una función dependiente de la frecuencia, y esta revela el espectro de frecuencia que compone la función original; es decir, la transformada de Fourier nos muestra dos lados de la misma información.

Cuando una función f es dependiente del tiempo solo muestra la información en el tiempo, ocultando la información que podemos obtener en frecuencia. Un ejemplo claro de información que se puede obtener es cuando una grabación de un instrumento una nota es producida por un violín, el ataque de la guitarra se va a ver reflejado a menudo en la información obtenida en el tiempo, pero lo que no muestra la información es que notas han sido ejecutadas. El cálculo de la transformada de Fourier consiste en la evaluación de integrales y sumas infinitas. Un ejemplo de lo comentado, se puede ver en las gráficas de la Figura 10 con la representación en tiempo y frecuencia.

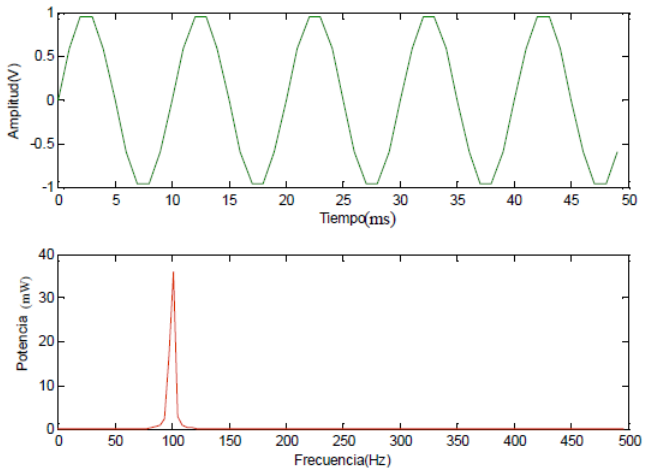


Figura 10. Representación en tiempo y frecuencia

Para realizar el cálculo en la práctica se debe realizar aproximaciones, para obtener la transformada por medio de sumas infinitas, esto puede realizarse eficientemente por la conocida transformada rápida de Fourier (FFT).

La transformada de Fourier en su forma continua puede ser definida como (2).

$$x(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t).\, e^{-j\omega t} dt \qquad (2)$$

En la práctica en el procesamiento de señales y otras áreas se utiliza ω y t, que están designadas a la frecuencia y tiempo respectivamente. Siendo la expresión que nos lleva del dominio de la frecuencia al dominio del tiempo (3).

$$x(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\omega).\, e^{j\omega t}\, d\,\omega \qquad (3)$$

La idea base de la representación de Fourier, es la de representar una señal como una superposición ponderada de las funciones de frecuencia elementales independientes. Cada una de las ponderaciones expresa la extensión que le corresponde, su función elemental que contribuye con la señal original, para revelar ciertos aspectos de la señal.

2.4.3. Representaciones musicales.

La música puede ser representada por signos los cuales pueden ser interpretados por nosotros como notas musicales y en conjunto música. Existen reglas que fueron establecidas para poder ser interpretadas, como por ejemplo cualquier lengua cuenta con sus propios signos o fonemas. En el caso de la música es muy similar, cada signo representa una letra, palabra o frase. Actualmente, las librerías musicales tienen gran diversidad de información como texto, video, audio, etc. Lo cual representa un problema dado que la música está representada en muchos formatos.

2.4.3.1. Representación Interfaz Digital de Instrumentos Musicales (MIDI)

La representación de la música en MIDI es un protocolo de información de comunicación para dispositivos electrónicos como sintetizadores, ordenadores, secuenciadores, etc. para la generación de todos musicales. Este protocolo está basado en representar la música de manera similar a una partitura.

El MIDI permite que un músico pueda controlar un instrumento musical electrónico remotamente, si consideramos un sintetizador en el cual se pulsa una tecla del instrumento este tendrá una respuesta en forma de sonido. El cual tiene las características esenciales como el tiempo que dura la nota, la tonalidad y la intensidad con la cual se toca. Destacar que el MIDI no transmite señales de audio, solo tiene los mensajes de eventos que se deben interpretar por el instrumento que los esté recibiendo.

El MIDI, es formato que contiene en valores numéricos de 0 – 127 asignadas las frecuencias de las notas (pitch), que serán generadas por instrumentos. Esto es muy similar a la asignación de notas que tenemos en un piano acústico. El cual tiene 88 teclas, en la Tabla 1 se relacionan las frecuencias en hercios a cada nota MIDI para todas las notas musicales de las nueve octavas. EL MIDI codifica estas notas asignando un número, por ejemplo, si escogemos el número 45 este número corresponde a A2. Así de esta manera los tonos son asignados a un número específico. La escala del MIDI comienza desde el C0 a G#9, en cuanto a la velocidad de la nota, de manera similar está definida por un número entero entre 0 y 127. Se puede ver que cubren un rango de 7900 Hz. Cada columna representa una escala musical y cada fila es un evento de la escala. En cada celda se muestran, en primer lugar la frecuencia fundamental (Hz) y en segundo lugar, entre paréntesis el número de nota MIDI correspondiente.

Notas musicales (Hz - MIDI)									
Nota / Octava	0	1	2	3	4	5	6	7	8
C	16.35 (12)	32.70 (24)	65.41 (36)	130.8 (48)	261.6 (60)	523.3 (72)	1047.0 (84)	2093.0 (96)	4186.0 (108)
C#	17.32 (13)	34.65 (25)	69.30 (37)	138.6 (49)	277.2 (61)	554.4 (73)	1109.0 (85)	2217.0 (97)	4435.0 (109)
D	18.35 (14)	36.71 (26)	73.42 (38)	146.8 (50)	293.7 (62)	587.3 (74)	1175.0 (86)	2349.0 (98)	4699.0 (110)
D#	19.45 (15)	38.89 (27)	77.78 (39)	155.6 (51)	311.1 (63)	622.3 (75)	1245.0 (87)	2489.0 (99)	4978.0 (111)
E	20.60 (16)	41.20 (28)	82.41 (40)	164.8 (52)	329.6 (64)	659.3 (76)	1319.0 (88)	2637.0 (100)	5274.0 (112)
F	21.83 (17)	43.65 (29)	87.31 (41)	174.6 (53)	349.2 (65)	698.5 (77)	1397.0 (89)	2794.0 (101)	5588.0 (113)
F#	23.12 (18)	46.25 (30)	92.50 (42)	185.0 (54)	370.0 (66)	740.0 (78)	1480.0 (90)	2960.0 (102)	5920.0 (114)
G	24.50 (19)	49.00 (31)	98.00 (43)	196.0 (55)	392.0 (67)	784.0 (79)	1568.0 (91)	3136.0 (103)	6272.0 (115)
G#	25.96 (20)	51.91 (32)	103.80 (44)	207.7 (56)	415.3 (68)	830.6 (80)	1661.0 (92)	3322.0 (104)	6645.0 (116)
A	27.50 (21)	55.00 (33)	110.0 (45)	220.0 (57)	440.0 (69)	880.0 (81)	1760.0 (93)	3520.0 (105)	7040.0 (117)
A#	29.14 (22)	58.27 (34)	116.50 (46)	233.1 (58)	466.2 (70)	932.3 (82)	1865.0 (94)	3729.0 (106)	7459.0 (118)
B	30.87 (23)	61.74 (35)	123.50 (47)	246.9 (59)	493.9 (71)	987.8 (83)	1976.0 (95)	3951.0 (107)	7902.0 (119)

Tabla 1. Relación entre todas las notas musicales y escalas con la nomenclatura MIDI

El Canal MIDI está dado por un valor entero de entre 0 y 15. Lo cual indica intuitivamente el instrumento que tiene el respectivo número de canal. Y por último, la marca de tiempo es representada por cuantos pulsos de reloj deben de esperarse respecto a la siguiente nota a ser ejecutada.

Una característica importante de la música MIDI es la versatilidad y la poca memoria que ocupan estos archivos. Como en la notación de una partitura, el MIDI puede medir el tiempo de duración de una nota, el tiempo siempre está especificado en la cabecera del archivo MIDI.

Generalmente, un archivo MIDI puede ser generado por un instrumento MIDI, aunque existen diversas formas de generar instrucciones. Programas de edición de audio, sintetizadores, como puede ser Timidity, Fluidsynth, que permiten crear secuencias a través de una interfaz que permite escribir de una manera gráfica las instrucciones MIDI al ejecutarse, así generando una gran variedad de sonidos sintetizados o utilizando bancos de sonido, en la siguiente Figura 11 se puede ver una señal musical representada de las diversas formas explicadas anteriormente.

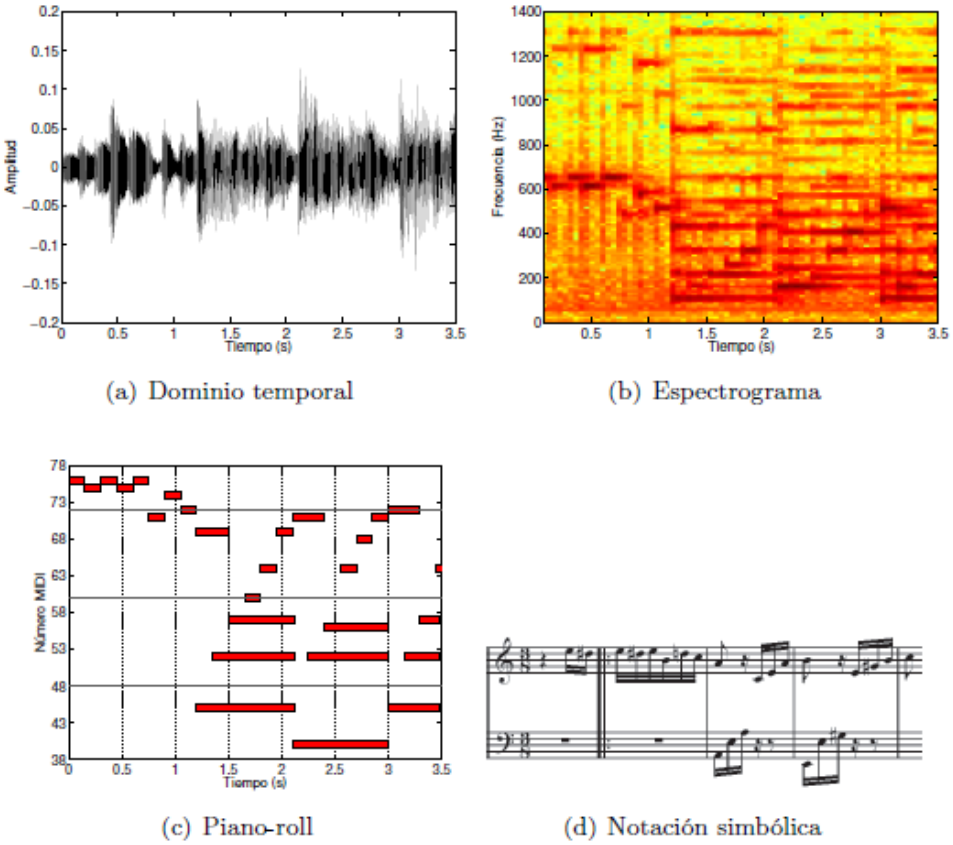


Figura 11. (a) Representación de la señal en el dominio temporal. (b) Representación tiempo-frecuencia en la que la energía de una frecuencia se representa mediante la intensidad del color (más intensidad en colores más cálidos) (c) Representación MIDI. Cada nota se indica por un rectángulo rojo. El eje horizontal representa el tiempo, el eje vertical indica el número MIDI de la nota tocada. (d) Notación musical clásica de la pieza interpretada.

Mencionar que el MIDI tiene algunas limitaciones respecto a una partitura; esta solo puede representar algunos eventos en la melodía como un interruptor encender – apagar nota. Definir tiempo de duración y el número de la nota; lo que carece de una interpretación precisa. El MIDI no tiene distinción entre bemoles y sostenidos. Para un MIDI estos son lo mismo ya que este tiene el mismo valor asignado. Por ejemplo si escogemos la nota G5# y A5b para la interpretación MIDI los dos valen 80, por este motivo se crearon herramientas como el XMLmusic para poder dar una especificación más correcta y sin pérdidas de información a la hora de interpretar música. Ya que un ser humano puede apreciar la música y un MIDI puede parecerle monótono.

2.5. Algoritmo de alineamiento

La programación en Matlab del alineamiento musical se basa en el artículo [15].

En este artículo se presenta un sistema de alineamiento de un audio y una partitura basado en la factorización espectral y online Dynamic Time Warping (Distorsión de Tiempo Dinámico DTW), con el propósito de alinear una pieza de audio con su notación musical, por ejemplo su partitura. El propósito del sistema tiene dos escenarios separados: preprocesamiento y alineamiento.

En el primero, se convierte la partitura, su señal de audio de referencia usando un software de sintetizador MIDI y se analiza la previamente la información para obtener los patrones espectrales (funciones básicas) asociados a cada combinación de las notas simultáneas en la partitura. Estos patrones espectrales son aprendidos de la señal sintética MIDI usando un método basado en la matriz no-negativa de factorización (NMF-Non-negative Matriz Factorization) con beta-divergente donde las ganancias son inicializadas como la transcripción real referida del MIDI.

En el segundo escenario, un método de descomposición de señal de baja latencia con patrones espectrales fijados por la combinación de notas se utiliza la magnitud del espectrograma de la señal de entrada resultando una matriz de divergencia que puede interpretarse como el costo de la coincidencia para cada combinación de notas de cada cuadro.

Finalmente, el enfoque de Dynamic Time Warping (DTW. Distorsión de Tiempo Dinámico) es usado para buscar la ruta con el mínimo coste y luego terminar la relación entre el rendimiento y los tiempos de la partitura musical.

Este sistema ha sido evaluado utilizando una base de datos de piezas de la época barroca y su comparación con otros sistemas proporciona resultados sólidos y buen rendimiento.

En este trabajo, se aborda el problema del alineamiento de audio-partitura (o identificar la partitura que corresponde), como tarea de sincronizar audio con una pieza musical.

Hay dos enfoques a este problema: alineamiento “offline” y “online”, según si la pieza está siendo procesada al mismo tiempo que se interpreta, o se reproduce, o se procesa tomándose más tiempo que la duración de la pieza. El alineamiento “offline”, que es el que se realiza en este proyecto, donde toda la actuación es accesible por el proceso de alineamiento, es decir, permite “mirar el futuro” mientras se establece la correspondencia, se encargan de alinear una partitura con un audio previamente grabado, utilizando el tiempo de proceso necesario Esto es interesante para aplicaciones que no requieren tiempo real, como sistemas de recuperación musical (MIR) y editores de audio para una obra. Con estos sistemas se busca tener un mejor resultado pudiendo realizar cálculos más complejos.

Por otro lado el alineamiento “online”, permite procesar los datos en tiempo real, es decir, se realizan el alineamiento de audio a notación mientras el músico está interpretando. Este seguimiento es muy útil para aplicaciones que necesiten el cambio de página automático, acompañamiento automatizado de un ordenador a un músico en vivo, sincronización en vivo entre instrumentos-elementos visuales o de sonido como por ejemplo, luces de un escenario o subtítulos de ópera, pero en cambio esto obliga al sistema a que sea computacionalmente rápido. Además, no puede disponer de información del futuro en audio, ya que el sistema no puede saber que va a tocar el músico si aún no lo ha tocado.

El sistema empleado y desarrollado en este proyecto, ha sido el sistema offline, se busca el alineamiento de audio y partitura pero no en tiempo real, ya que lo que se ha implementado es cargar este par para conseguir identificar los inicios de compases.

A continuación se va a explicar este sistema con sus diferentes etapas detalladas.

2.5.1. Visión conjunta del sistema

El sistema propuesto en este artículo está representado en la Figura 12, donde como se puede ver el esquema general dividido en dos etapas. En primer lugar la etapa de preprocesamiento donde solo la notación MIDI es requerida y una vez los parámetros son aprendidos, se puede calcular el alineamiento en tiempo real.

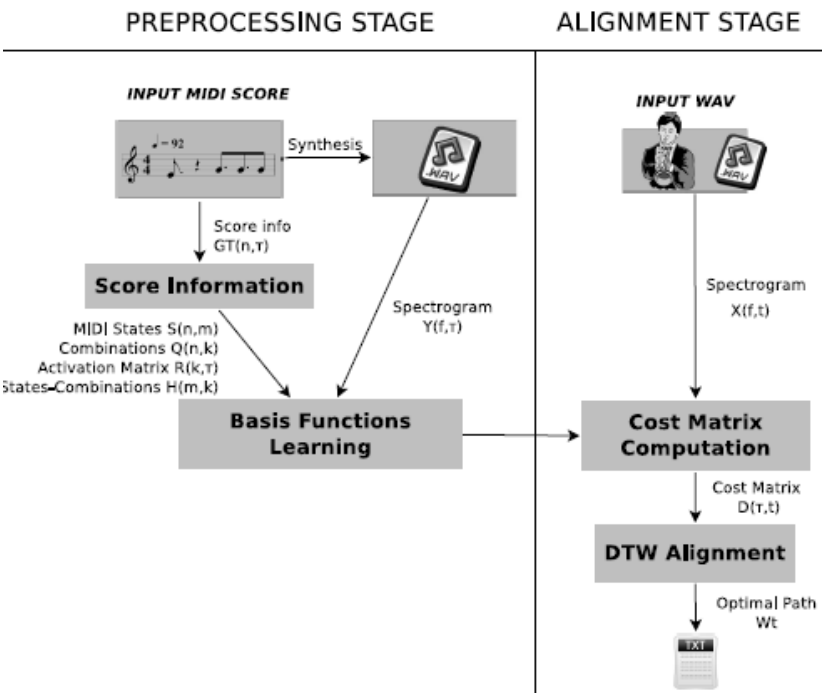


Figura 12. Diagrama de bloques del sistema propuesto

2.5.1.1. Etapa de preprocesamiento

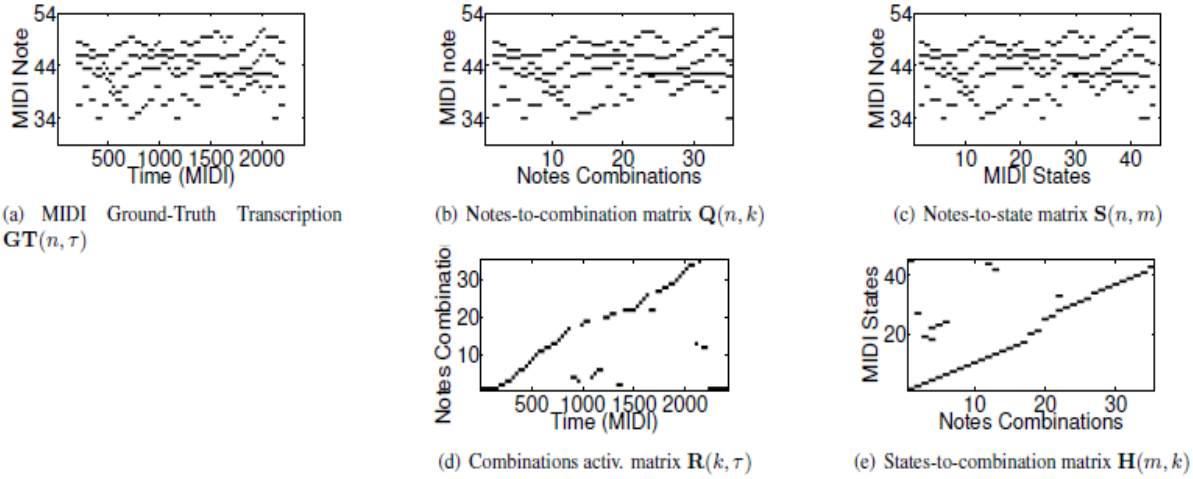
En esta etapa los parámetros para el alineamiento son aprendidos de la partitura, que debe ser proporcionada de antemano su representación MIDI. Esta etapa se realiza en dos fases: definir los estados y aprender los patrones espectrales.

En la fase de definición de los estados lo principal es organizar adecuadamente la información dada de la partitura. En primer lugar, la matriz $GT(n,\tau)$ es referida de la partitura MIDI, donde τ es el tiempo MIDI y n son las notas en escala MIDI, donde puede haber varios instrumentos definidos en la partitura. La partitura puede ser interpretada como una secuencia consecutiva de M estados y también, nos informa sobre el cambio de tiempo de cambio de un estado a otro.

De la matriz $GT(n,\tau)$ se obtiene la siguiente descomposición de matrices: $GT(n,\tau) = Q(n,k)R(k,\tau)$, donde $Q(n,k)$ es la matriz de combinación de las notas y $R(k,\tau)$ representa la activación de cada combinación en tiempo MIDI.

Con el fin de obtener la información de los estados necesarios para realizar la alineación, la matriz de combinación de notas $Q(n,k)$ es descompuesta como: $GT(n,k) = S(n,m)H(m,k)$, donde $S(n,m)$ es la matriz de estados de notas y $H(m,k)$ representa la combinación de notas activas de cada estado.

Las matrices definidas se usarán en las siguientes etapas para realizar el alineamiento y son calculadas de la partitura MIDI. En la Figura 13 se pueden observar estas matrices descritas anteriormente.



(a) MIDI Ground-Truth Transcription $\mathbf{GT}(n, \tau)$

(b) Notes-to-combination matrix $\mathbf{Q}(n, k)$

(c) Notes-to-state matrix $\mathbf{S}(n, m)$

(d) Combinations activ. matrix $\mathbf{R}(k, \tau)$

(e) States-to-combination matrix $\mathbf{H}(m, k)$

Figura 13. Diferentes matrices de definición de estados para cierta señal de música.

Mencionar que la información en tiempo MIDI se carga en Matlab en una tabla que recibe el nombre de *nmat* (o matriz de notas) viene detallado en el "Manual de MIDI Toolbox para Matlab" [16]. Esta tabla es una matriz que representa los eventos de notas en un archivo MIDI, una vez se leen y carga un archivo MIDI, la matriz es como la que se muestra en la Figura 14.

```
» nmat

nmat =
0         0.9000    1.0000   64.0000   82.0000    0         0.5510
1.0000    0.9000    1.0000   71.0000   89.0000    0.6122    0.5510
2.0000    0.4500    1.0000   71.0000   82.0000    1.2245    0.2755
2.5000    0.4500    1.0000   69.0000   70.0000    1.5306    0.2755
3.0000    0.4528    1.0000   67.0000   72.0000    1.8367    0.2772
3.5000    0.4528    1.0000   66.0000   72.0000    2.1429    0.2772
4.0000    0.9000    1.0000   64.0000   70.0000    2.4490    0.5510
5.0000    0.9000    1.0000   66.0000   79.0000    3.0612    0.5510
6.0000    0.9000    1.0000   67.0000   85.0000    3.6735    0.5510
7.0000    1.7500    1.0000   66.0000   72.0000    4.2857    1.0714
```

Figura 14. Ejemplo de matriz nmat cargada en Matlab

Se puede ver que la variable nmat contiene una matriz 7 x 10 con distintos números. Las columnas se refieren a varios tipos de información del tono MIDI y el canal MIDI.

Las filas representan los eventos de cada nota individual (en el caso de la Figura 14 la melodía de la que se ha cargado tiene 10 notas, por tanto habrá tantas filas como notas haya, y cada una de ellas se describe en términos de tono, tiempo de inicio, duración). Las etiquetas de las columnas de la matriz nmat son las siguientes que se muestran en la Tabla 2.

ONSET (BEATS)	DURATION (BEATS)	MIDI channel	MIDI PITCH	VELOCITY	ONSET (SEC)	DURATION (SEC)

Tabla 2. Información de la matriz nmat

La primera columna indica el comienzo de la nota en beats (pulsaciones), la segunda es la duración de las notas en estos mismos valores, la tercera columna indica el canal MIDI (1-16), la siguiente el tono MIDI donde el central es 60 (C4). La quinta columna es la velocidad que describe la rapidez con la que se pulsa la nota, es decir, como de fuerte se toca la nota (0-127). Las últimas dos columnas son como las dos primeras, salvo que aquí la medida es en segundos.

Por otro lado, en la fase de aprender los patrones espectrales, cuando una trama de señal es dada, lo primero se debería hacer es calcular la similitud entre la actual trama y las diferentes combinaciones de notas definidas por la partitura. Aquí se enfoca en calcular la distancia entre la transformada en frecuencia de la señal de entrada y un patrón espectral por combinación de notas. Un patrón espectral es definido como un espectro que es aprendido de la señal con ciertas características. El uso de solo un patrón espectral por combinación permite calcular la divergencia con un método de descomposición de señal de baja complejidad, esto significa que este método debe aprender de antemano el patrón espectral asociado a cada combinación única de notas para la puntuación.

Para este fin, se utiliza la técnica basada en Non-Negative Matrix Factorizacion (NMF), que se explicará en el siguiente subapartado con Beta divergente y reglas de Multiplicative Update (MU), pero en este trabajo se propone aplicarlo en la señal sintética generada a partir de la puntuación MIDI (generada usando el programa Timidity++ con las fuentes sonoras FluidR3 GM) en lugar del audio real.

En primer lugar, se define el modelo de señal como:

$$Y(f,\tau) \approx \hat{Y}(f,\tau) = B(f,k)G(k,\tau)$$

Donde $Y(f,\tau)$ es la magnitud del espectro de la señal sintetizada, $\hat{Y}(f,\tau)$ es el espectro estimado, la matriz $G(k,\tau)$ representa la ganancia de los patrones espectrales y la matriz $B(f,k)$ representa los patrones espectrales de notas las combinaciones de notas definidos en la partitura.

Cuando los parámetros se limitan a ser no negativos, como es el caso del espectro de magnitud, una forma común para calcular la factorización es minimizar el error de reconstrucción entre el espectrograma observado y modelado. Las funciones de coste más populares son la distancia euclídea (EUC con $\beta = 2$), el generalizado Kullback-Leibler (KL con $\beta = 1$) y las divergencias Itakura-Saito (IS con $\beta = 2$).

2.5.1.2. NMF: Non Negative Matrix Factorization

NMF es un algoritmo de factorización de matrices propuesto originalmente por Lee y Seung [17] para el análisis de imágenes faciales. Esta técnica se puede aplicar al análisis exploratorio de datos como métodos de proyección para reducir la dimensión de los datos o para descubrir patrones ocultos, aunque su aplicación más extendida es para facilitar la aplicación de datos.

Mediante NMF se puede reproducir de forma aproximada una matriz de datos, $V \in R^{nxm}$, como un producto de dos matrices $W \in R^{nxk}$ y $H \in R^{kxm}$. En este contexto, W representa un conjunto reducido de k factores (*vectores base*), mientras que H contiene los coeficientes de la combinación lineal de factores necesaria para la reconstrucción de V. A este conjunto de coeficientes se les denomina *vectores codificantes*.

Adicionalmente, se suelen imponer las siguientes restricciones:

- k<<n.
- V, W y H no tienen valores negativos.
- Las columnas de W están normalizadas (la suma de los elementos de cada columna vale 1).

La mayor diferencia entre NMF y otras técnicas de factorización radican en la restricción impuesta a los vectores base (W) y a los coeficientes (H) de no utilizar valores negativos. Gracias a esta restricción, se obtiene una representación en partes o secciones, ya que sólo se permiten combinaciones aditivas, nunca sustractivas.

De esta manera, los factores pueden ser interpretados como partes de los datos o, dicho de otro modo, como elementos que tienden a aparecer juntos. Por el contrario, otras técnicas de factorización como las mencionadas anteriormente, permiten que los valores de W y H tengan cualquier signo, lo que conlleva a cancelaciones complejas de elementos positivos y negativos durante la reconstrucción original de datos. En otras palabras, NMF tiende a producir factores que permiten una interpretación contextual, relativamente sencilla, mientras que otros obtenidos mediante otras técnicas no contienen un “significado” intuitivo.

2.5.1.3. Etapa de alineamiento. Algoritmo Dynamic Time Warping

En esta etapa, se va a explicar cómo se produce el alineamiento entre la partitura y el audio usando la información de la etapa de preprocesamiento. Este alineamiento resulta importante para relacionar tiempo real con tiempo MIDI, donde el tiempo real son

las tramas de la señal de entrada y el tiempo MIDI las notas activas cada 10 milisegundos de la partitura. Para el enfoque offline de este proyecto se utiliza el alineamiento clásico usando DTW, donde el código que se ha empleado para DTW está basado en [18].

Este subapartado se centra principalmente en explicar en qué consiste y el funcionamiento del algoritmo DTW (Dynamic Time Warping), que se usa en el proyecto que se desarrolla. Ha sido de gran importancia el estudio de este algoritmo para entender el proceso de alineamiento.

DTW es un potente algoritmo que mide la similitud entre dos secuencias temporales que pueden variar en tiempo, velocidad y/o longitud permitiendo realizar un alineamiento óptimo entre dos secuencias de vectores de distinta longitud; de dicho alineamiento se obtiene una medida de distancia entre los dos patrones temporales.

Este algoritmo, corrige los problemas de la distancia euclídea en la alineación de datos, la Figura 15 ilustra el problema (parte superior) usando la distancia euclídea y cómo lo corrige DTW (parte inferior).

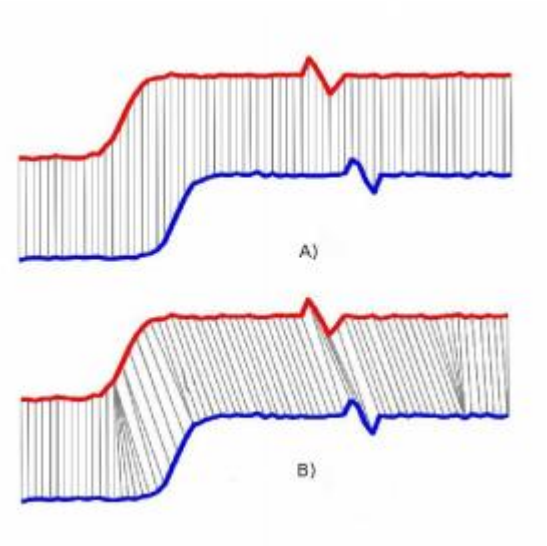


Figura 15. Comparación métodos: A) Distancia euclídea. B) DTW

El algoritmo DTW tiene el inconveniente de tener una gran complejidad. Esto hizo que al principio se utilizara más la distancia euclídea, pero posteriormente, con la mejora de la potencia de cálculo y la optimización del algoritmo, se haya extendido esta técnica.

En primer lugar con este algoritmo se construye una matriz de distancia entre el patrón y los puntos de muestreo y a continuación muestra la mejor manera de alinear la muestra con el patrón, buscando el camino óptimo.

Por ejemplo, similitudes en la trayectoria entre dos barcos pueden ser detectadas usando DTW, aunque uno de ellos se mueva más rápido que el otro.

Principalmente, este algoritmo fue usado para reconocimiento de voz, pero se puede aplicar a otras secuencias temporales de audio, datos, etc, es decir, cualquier secuencia temporal puede ser analizada por DTW.

A continuación, se va a detallar en más profundidad este algoritmo DTW, con un ejemplo.

Se parte de que el algoritmo tiene dos parámetros de entrada: Secuencia de datos Q (4) y cadena de búsqueda C (5).

$$Q = \{q_1, q_2, \ldots\ldots q_n\} \quad (4)$$

$$C = \{c_1, c_2, \ldots\ldots c_m\} \quad (5)$$

Para realizar la alineación se construye una matriz D de n x m elementos, donde cada elemento d_{ij} contiene la distancia entre los elementos q_i y c_j (6). A partir de esta matriz se define el camino de deformación ω como se muestra en la Figura 16.

$$D(q_i, c_j) = (q_i - c_j)^2 \quad (6)$$

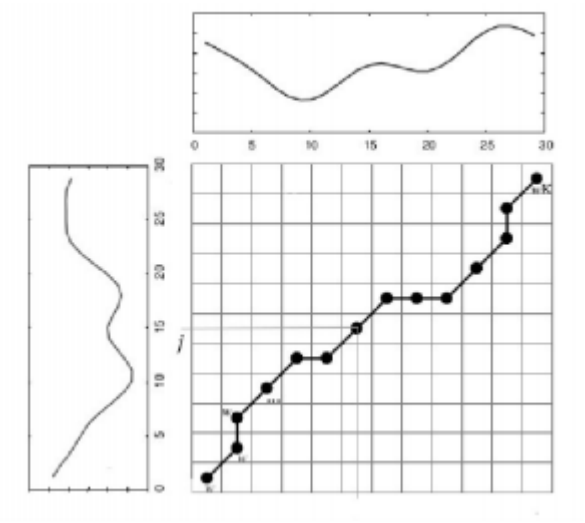


Figura 16. Un ejemplo de camino de deformación

Habrá varios caminos, pero la mejor solución va a ser la que cumpla (7), ya que se utiliza una matriz de costo para facilitar el cálculo, ya que solo interesa el camino que minimiza las distancias entre las secuencias.

$$DTW(Q, C) = min\sqrt{\sum_{k=1}^{K} w_k / K} \quad (7)$$

En el presente proyecto este alineamiento resulta importante para relacionar tiempo real con tiempo MIDI, donde el tiempo real son las tramas de la señal de entrada y el tiempo MIDI las notas activas cada 10 milisegundos de la partitura, como se puede ver en la Figura 17.

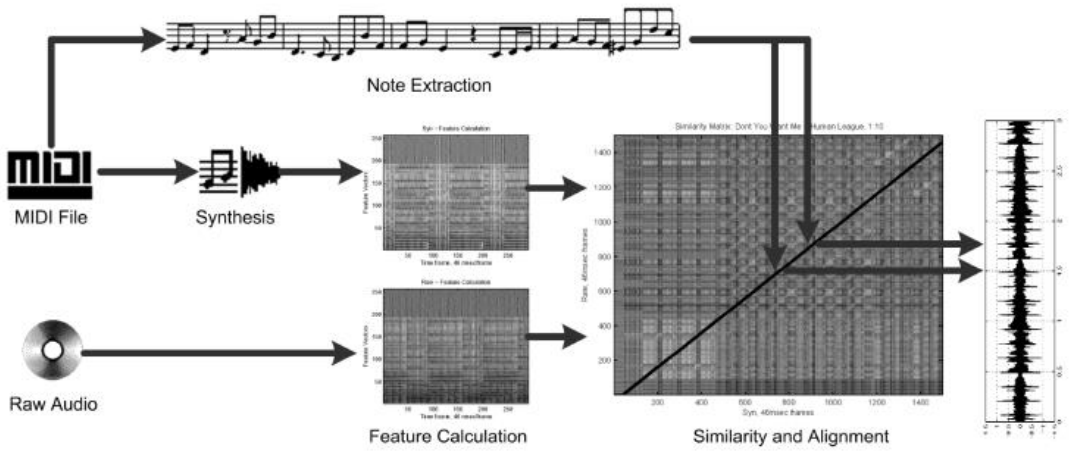


Figura 17. Proceso de alineamiento. Características calculadas de la señal original de audio son comparadas con la sintetizada MIDI en la matriz de similitud

2.5.2. Resultados

Para analizar el rendimiento de los métodos propuestos (offline y online), donde como se puede ver en la Figura 18 el propuesto sistema offline descrito obtiene los mejores resultados en términos de precisión. Como se puede observar, este método offline supera al método Ellis' offline

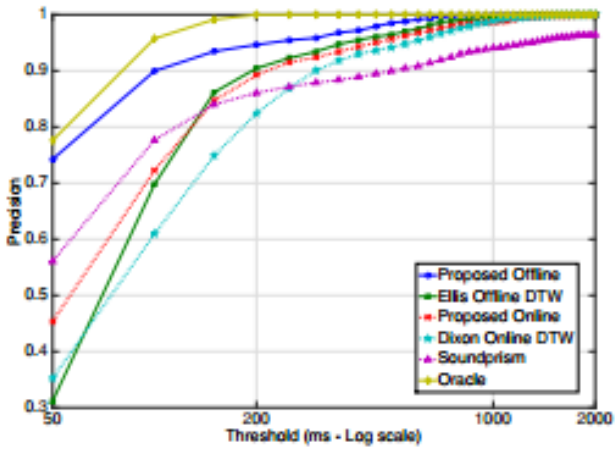


Figura 18. Valores de precisión en función de la tolerancia de la ventana

En un segundo experimento, se evalúa el rendimiento de los sistemas propuestos en función de la polifonía. Como también ocurre en el caso anterior, el rendimiento del método offline obtiene en general mejores resultados que los sistemas online como se demuestra en la Tabla 3. El sistema propuesto obtiene los mejores resultados entre los métodos comparados y ha demostrado ser robusto frente a la polifonía en el conjunto de datos analizados.

	Poly	Precision	Miss	Missalign	Av offset	Av |offset|	Std Offset
Prop. Offline	2	**94,59**	0,00	5,41	-11,27	**33,43**	44,38
	3	**94,75**	0,00	5,25	-11,78	**34,25**	44,89
	4	**94,50**	0,00	5,50	-11,09	**35,16**	46,51
Ellis Offline	2	90,57	0,00	9,42	66,90	71,28	47,30
	3	90,39	0,00	9,60	65,67	71,53	50,24
	4	89,80	0,00	10,19	66,97	72,62	49,93
Prop. Online	2	**88,44**	0,00	11,56	-41,18	57,78	56,40
	3	**90,13**	0,00	9,86	-42,96	60,56	57,65
	4	90,70	0,00	9,30	-44,15	62,97	58,58
Dixon Online	2	81,69	0,00	18,31	53,93	70,51	67,02
	3	83,17	0,00	16,83	53,88	70,65	67,02
	4	83,94	0,00	16,06	51,29	71,64	70,26
Soundprism	2	83,01	0,00	16,99	-25,90	**48,08**	55,36
	3	88,81	0,00	11,19	-21,23	**43,73**	51,54
	4	**93,50**	0,00	6,50	-22,05	**42,91**	49,13
Oracle	2	100,00	0,00	0,00	6,15	32,79	**42,47**
	3	100,00	0,00	0,00	6,09	32,67	**43,08**
	4	100,00	0,00	0,00	6,07	32,58	**43,38**

Tabla 3. Resultados de alineamiento audio-partitura en función de la polifonía en términos de precisión (%). Valores "offset en ms. El porcentaje en negrita muestra el mejor valor en cada medida"

Para ver la evolución de los resultados de años anteriores se puede ver en [20], donde la página describe el sistema de evaluación propuesto para los sistemas de seguimiento de partitura y la evolución de los resultados en los últimos años, y se puede ver cómo ha ido mejorando el sistema aumentando la precisión. Donde el archivo de salida del resultado del alineamiento va a ser en formato MIREX, donde se obtendrá un fichero .txt con cuatro columnas, como se puede ver en la Figura 19 de ejemplo:

```
1800    1800    0       75
2021    2022    187.5   73
...     ...     ...     ...
```

Figura 19. Resultados alineamiento formato MIREX

Donde la primera columna es el tiempo de comienzo estimado en el archivo de audio (en milisegundos), la segunda es el tiempo de detección relativo al archivo de audio (en milisegundos), la tercera columna es el tiempo de inicio de la nota en la partitura (en milisegundos) y la última columna es el número de la nota MIDI en la partitura (un número entero).

2.5.3. Conclusiones

En este artículo se ha presentado un seguimiento de partitura basado en la factorización espectral y DTW. La factorización espectral es usada para aprender los patrones espectrales de cada combinación de notas de la partitura MIDI. Por tanto, una matriz de coste es calculada usando la matriz de divergencia obtenida usando una descomposición de la señal no iterativa. Finalmente, una estrategia DTW es utilizada en los modos online y offline.

El sistema propuesto offline y online ha sido testado usando un conjunto de datos con diferentes polifonías (de 2 a 4) y comparados con otros métodos de referencia, obteniendo los mejores resultados en términos de precisión demostrando ser un sistema robusto.

3. OBJETIVOS

El objetivo de este proyecto es realizar un servidor web que sea capaz, a partir de una partitura digital, de encontrar en grandes bases de datos multimedia interpretaciones de esa partitura y alinear la partitura con el audio de la interpretación. Para lograr ese objetivo se realizará el seguimiento de partitura sobre todas las instancias encontradas y se determinará, a partir del coste estimado de alineamiento, si se trata de una interpretación de la partitura. Esto ha llevado a familiarizarse y usar en Matlab el programa de alineamiento basado en Dynamic Time Warping desarrollado por el grupo de investigación TIC-188 de la Escuela Politécnica Superior de Linares en su versión offline.

Además, la partitura incluirá datos de oyente asociados a cada compás que se mostrarán en el tiempo oportuno durante la reproducción de la interpretación, junto con una información básica de la partitura digital que se podrá ver en el navegador del usuario.

Por otro lado resaltar la implementación de un servicio en lugar de programación en C del alineamiento offline y del discriminador que se estableció inicialmente como metodología a desarrollar, y se pensó en su implementación para dar capacidad de atender a múltiples usuarios y detectar interpretaciones que ya han sido alineadas que van almacenándose en una base de datos a la que está conectado, y así el programa de Matlab pueda ir procesando las peticiones de manera rápida y ordenada; y con la finalidad de que no se procesen y por tanto más tiempo de espera, con solicitudes que ya han sido procesadas previamente.

Con este servicio se permite escalar el sistema, dar fluidez y establecer una lógica de control en el proceso tanto para almacenar y gestionar peticiones como almacenar los inicios de compases ya calculados para cada par de solicitud, por lo cual este servicio va conectado y enviando peticiones a una base de datos para que el programa Matlab desarrollado vaya procesándolas.

Al disponer de una base de datos, también se van almacenando el vector con el inicio de compases reales procesados con el programa para cada par de partitura digital y audio.

Por último, el usuario podrá visualizar en la página web tanto el vídeo, como la partitura de la obra para la que ha realizado la petición, por medio del visor abcweb, junto con unos datos de oyente asociados a la obra y a cada compás que serán mostrados.

4. MATERIALES Y MÉTODOS

Este capítulo se centra en explicar los elementos más importantes que se han desarrollado para el funcionamiento eficiente de todo el proceso del presente proyecto.

Se va a detallar todo el desarrollo de la aplicación, por los tres procesos por los que está compuesto: la implementación de un servicio que está conectado a una base de datos que gestiona las peticiones de usuario, el proceso Matlab que va procesando estas peticiones con la finalidad de obtener el vector de inicios de compases sincronizados y la aplicación web desarrollada donde el usuario puede visualizar la partitura, vídeo, información de la partitura y añadir comentarios por compás. Se explicará todo el desarrollo mostrando imágenes y diagramas.

4.1. Servicio desarrollado

En este apartado se va a describir la aplicación del servicio, realizado en lenguaje PHP junto con MySQL para la base de datos, que se ha implementado con el objetivo de obtener un vector con los inicios de compases reales, que permite gestionar y administrar las peticiones de un usuario compuesto por el par identificador de Youtube y partitura en formato XML por medio de un sistema de colas, que se van almacenando en la base de datos para ir realizando consultas y comprobar si hay peticiones pendientes para procesar, consiguiendo sincronizar las aplicaciones, estableciendo una lógica de control.

El programa Matlab desarrollado, que será explicado en el siguiente apartado, estará continuamente activo consultando y procesando las peticiones pendientes obteniendo el resultado. A lo largo de este apartado se va a desarrollar con detalle la estructura y funcionamiento de este servicio desarrollado.

4.1.1. Estructura

Para empezar se expondrán los elementos por los que está compuesto este servicio:

- Base de datos: La estructura de la base de datos implementada con el nombre “compases”, está compuesta por dos tablas, una llamada “peticiones”, donde se van almacenando o encolando las peticiones del usuario y otra llamada “inicios” donde se guardan el vector de tiempos de inicios de compases procesados, como se puede ver en la Figura 20.

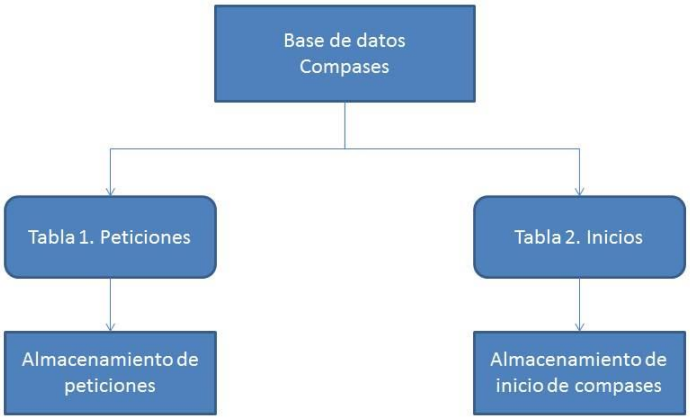


Figura 20. Estructura Base de Datos

En estas dos tablas de la base de datos se van almacenando cada par de petición, identificador de Youtube y partitura en formato XML, con la singularidad de que en el campo de la tabla *peticiones* Tabla 4, se almacenan las peticiones y la tabla de *inicios* almacena los vectores procesados y calculados Tabla 5.

Identificador Youtube	Hash contenido XML	Peticiones

Tabla 4. Campos de tabla peticiones de la base de datos

Identificador Youtube	Hash contenido XML	Vector con tiempos de inicios compases

Tabla 5. Campos de la tabla inicios de la base de datos

El método de almacenamiento y procesamiento de peticiones implementados es el sistema de colas conocido como pila, que como se ve en la Figura 21, tiene como objetivo ordenar las peticiones recibidas por parte del usuario, teniendo en cuenta que el orden de los mismos depende del tiempo que lleven "dentro" de la estructura, sólo se puede acceder al contenido de una pila mediante un extremo.

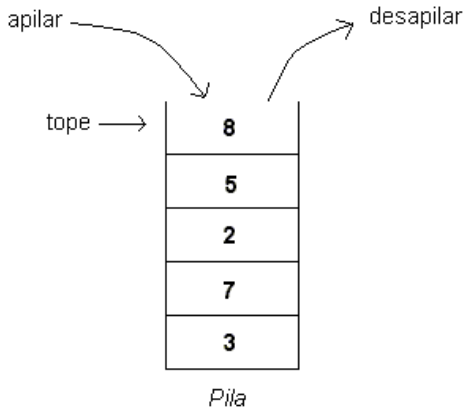


Figura 21. Sistema de colas. Pila

- Api.php: En este script se encuentra el servicio, llamado 'inicios', donde se va solicitando el par partitura XML e identificador de Youtube. Aquí una vez se tiene una partitura como petición, se realiza un hash en formato MD5 de su contenido interno, que no es más que un código asociado a un archivo permitiendo así identificarlo, una vez se obtiene se renombra la partitura digital con este código. También mediante este fichero se llama a Db.php, que se explicará a continuación, para conectar a la base de datos para realizar las consultas y comprobar si se tiene el vector de tiempos procesado, si no, se pondría en la cola de peticiones. Comprueba si está procesado y si no fuese el caso, se encolaría la petición en la base de datos.

- Db.php: Con este script se conecta a la base de datos 'compases', y consulta si para la pareja solicitada están almacenados los inicios y si no, lo almacenaría como petición.

- Rest.php: El nombre de este script deriva de "Representational State Transfer", que es "transferencia de representación de estado". Un servicio REST no tiene estado, lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos. Por tanto, no se puede llamar a un servicio REST y pasarle unos datos (por ejemplo un usuario y una contraseña) y esperar que "nos recuerde" en la siguiente petición. El estado lo mantiene el cliente y por lo tanto, es el cliente el que debe pasar el estado en cada llamada. Para que un servicio Rest recuerde, se debe pasar qué soy en cada llamada.

4.1.2. Funcionamiento

Ahora bien, descritos los componentes del servicio se va a explicar el funcionamiento y desarrollo del servicio implementado.

Imaginar que llega un usuario con un par de partitura digital en formato XML y un identificador de un vídeo de Youtube y solicita la petición al servicio para la obtención de los inicios de compases de la partitura sincronizados con el vídeo en tiempo real.

En primer lugar, el servicio realizaría un hash del contenido interno de la partitura para comprobar en la tabla *inicios* si ha sido procesado. El hash del contenido interno tiene la finalidad de identificar si existe en la base de datos, ya que el archivo XML se renombra con ese código creado, y si fuese igual el contenido del archivo, este código identificativo sería exactamente igual. Guardar la partitura digital con ese nombre es con la finalidad de identificarlo y de no volver a procesar un par que ya haya sido procesado, es decir, si un usuario vuelve a enviar una partitura en formato XML con el contenido idéntico a otra que ya haya sido procesada, gracias a este servicio se detectará y no se volverá a procesar, ahorrando tiempo y proporcionando fluidez y rapidez al proceso; no supondrá ningún retardo puesto que ya han sido calculados y directamente el servicio proporcionará esos inicios de compás.

A continuación, se va a describir los dos posibles casos que pueden ocurrir en el caso de que se tuviese ya la petición procesada o no.

En primer lugar, al consultar la tabla *inicios*, si hubiese sido procesado y se tuviera almacenado en la tabla inicios de la base de datos para el par XML e identificador de Youtube, como se puede ver en la Figura 22, obtendríamos el resultado directamente.

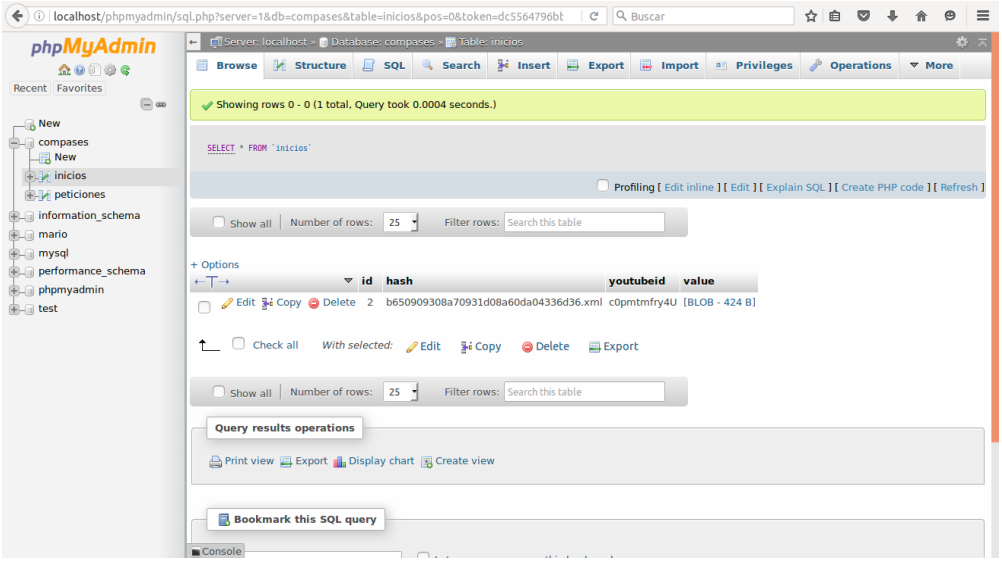


Figura 22. Base de datos. Tabla inicios.

Como se puede ver en la Figura 22, para ese par ya estaría procesado y almacenado el vector de tiempos, por tanto, el servicio nos mostraría el resultado junto con el código de respuesta de “status:200”, como se puede ver en la Figura 23.

Figura 23. Servicio con vector de inicios de compases procesados.

En cambio, si al realizar la consulta no se obtiene respuesta de que han sido procesados y calculados, la solicitud se almacena en la tabla *peticiones.* Por tanto, se van almacenando y encolando estas solicitudes, la partitura XML renombrada con su hash de contenido interno y el identificador de Youtube. Junto con un campo donde se van indicando si las peticiones han sido procesadas o no, mediante “0” si no han sido procesadas o “1” si han sido ya procesadas, como se puede ver en la Figura 24, de tal manera que el servicio al estar conectado a la base de datos va consultando las peticiones y va procesando las que estén a “0” de manera ordenada.

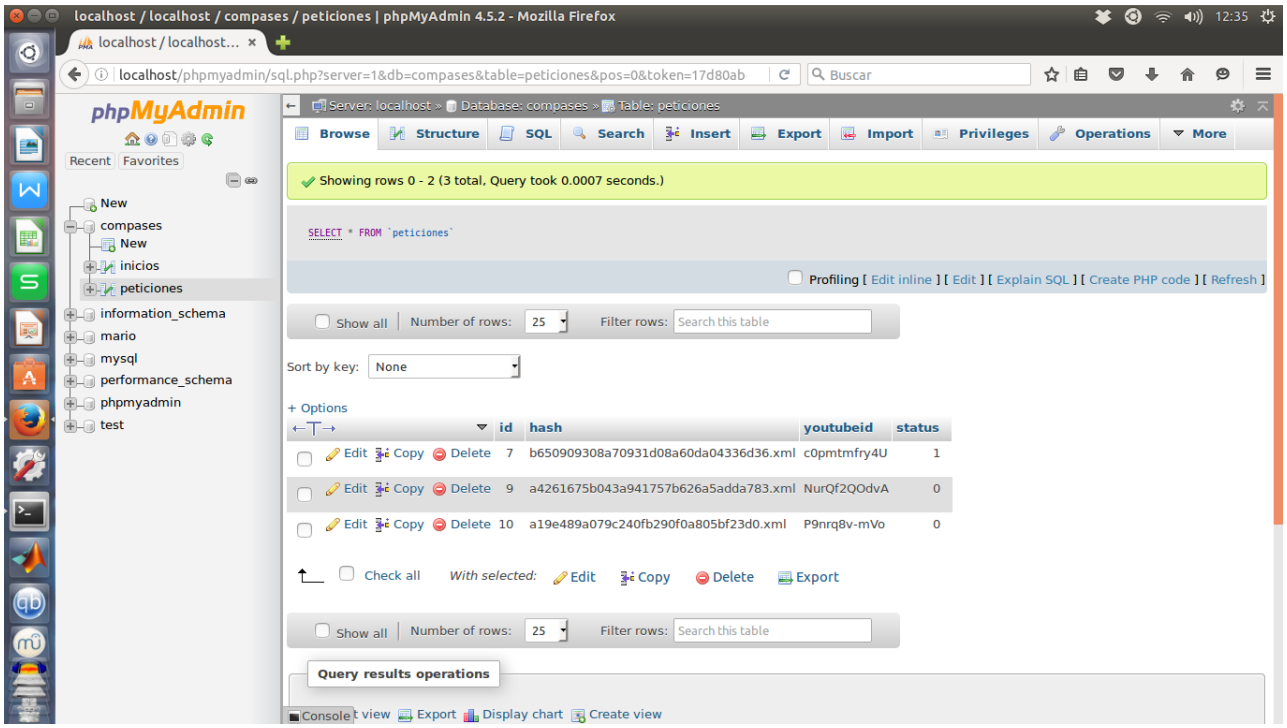


Figura 24. Base de Datos. Tabla peticiones.

En la figura mostrada, se puede ver como la petición que ya ha sido procesada está a “1”, por tanto, como se vio anteriormente al estar ya procesada se encuentra almacenado en la tabla *inicios*. En cambio, cada par que no ha sido procesado está a “0”, y estas peticiones se van procesando por el programa Matlab ordenadamente.

Para la visualización de un usuario que realiza una petición que no ha sido procesada se le devolverá una respuesta “status:201”, indicando que se está procesando

su solicitud, como se ve en la Figura 25, si al cabo de cierto tiempo vuelve a consultar y ha sido calculado ese vector de tiempos se le mostrará el resultado como se mostró anteriormente.

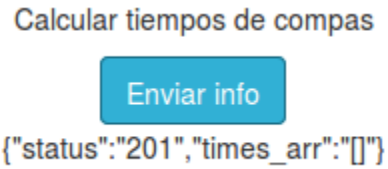


Figura 25. Servicio con vector de inicios de compases procesando.

En definitiva, este servicio está implementado con el objetivo de dar fluidez, orden y dinamismo a la aplicación, y es escalable, es decir que varios usuarios pueden ir realizando solicitudes, dónde algunas pueden haber sido ya procesadas ahorrando tiempo al proceso en general. A continuación en la Figura 26 de representa de manera visual mediante un diagrama de flujo el funcionamiento del servicio.

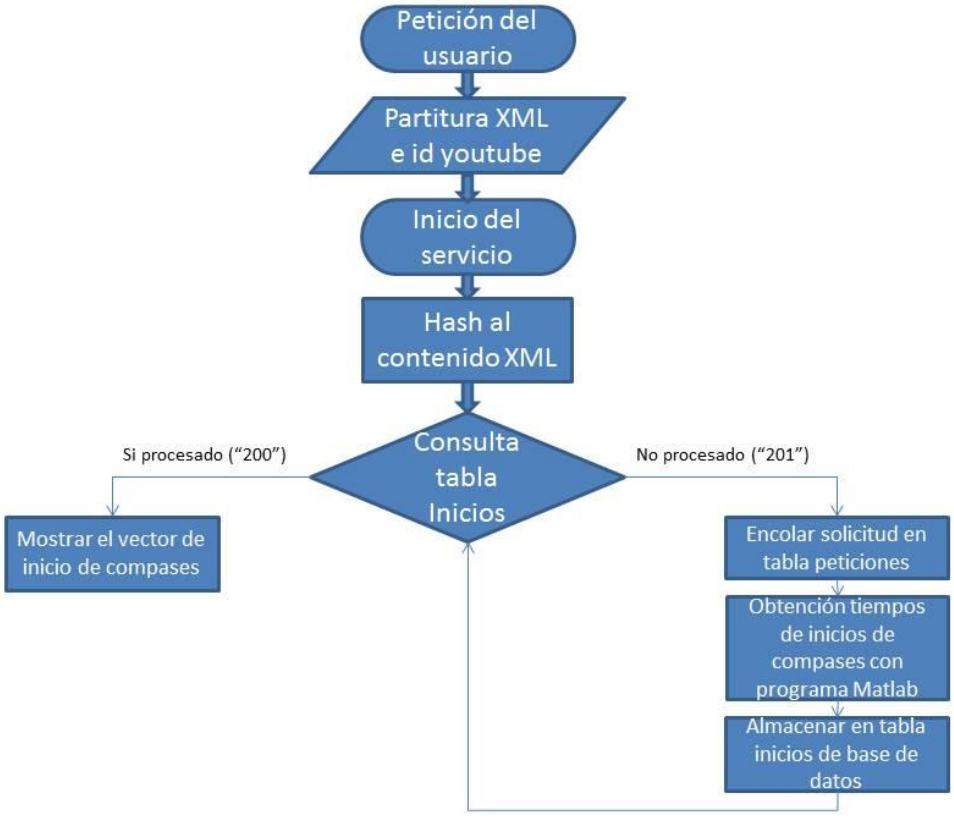


Figura 26. Diagrama de flujo del servicio

4.2. Programa Matlab desarrollado

En este apartado se va a explicar el desarrollo del programa Matlab una vez que el usuario ha solicitado una petición y queda almacenada en la base de datos, el programa realizado en Matlab que siempre estará operativo para ir comprobando si hay peticiones para procesar.

En primer lugar, se ejecuta la función PrincipalMain(), que está conectado a la base de datos y se van procesando las solicitudes pendientes por orden de preferencia y extrae los datos de identificador de Youtube para obtener el audio y partitura digital que tiene el nombre del hash de su contenido interno.

En este apartado también se va a detallar, una vez extraídos de la base de datos, cómo se preprocesan tanto el audio como la partitura digital por medio de la función Preproceso_Principal(), para que mediante la función audio2ScoreTreal_offline () se produzca el alineamiento y se identifiquen y obtengan los inicios de compases reales, estas funciones serán explicadas en el Manual Técnico (Anexo II) con sus parámetros de entrada y salida.

Resaltar el uso de la función que proporciona Matlab de system (), ya que ejecutaroperaciones de comandos del sistema en el entorno Matlab, para el proceso de los datos y devolviendo la salida; por comodidad y emplear programas que permiten llamarse desde línea de comandos como MuseScore (procesado de partitura) y Youtube-dl (extraer audio wav con identificador de Youtube).

Finalmente se irán almacenando, una vez se vayan procesando las peticiones, en la base de datos este vector de tiempos.

A continuación, en la Figura 27, se puede ver un diagrama de flujo general de este programa, en los siguientes subapartados se entrará en detalle del procesamiento tanto del audio como de partitura.

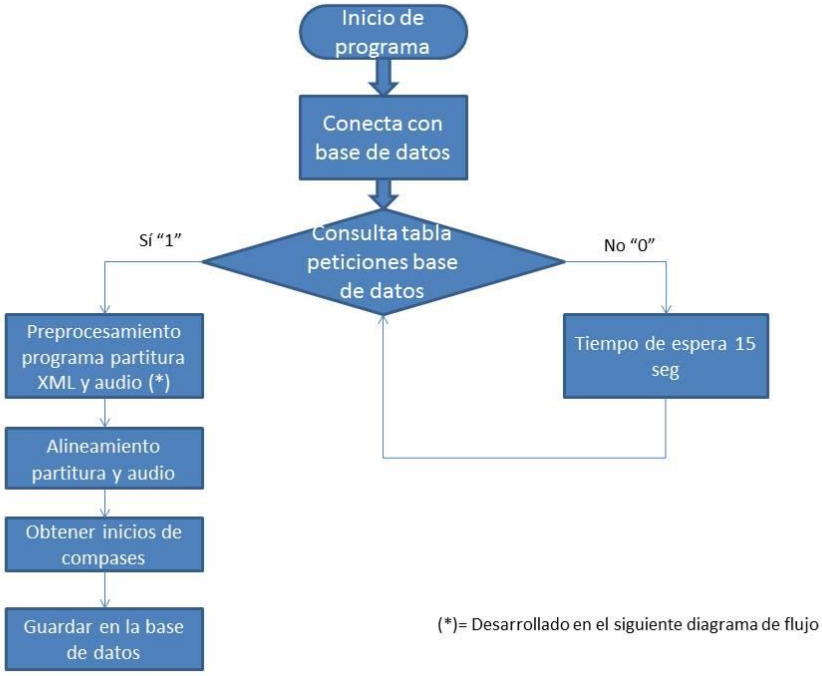


Figura 27. Diagrama de flujo general de programa

Antes de entrar en detalle del procesamiento de ambos datos, en la Figura 28, se puede ver el desarrollo de este preprocesamiento, y qué hace el programa una vez extrae una petición antes de emplear el algoritmo de alineamiento; dónde en los siguientes subapartados se va a explicar más detalladamente cómo se realiza este proceso.

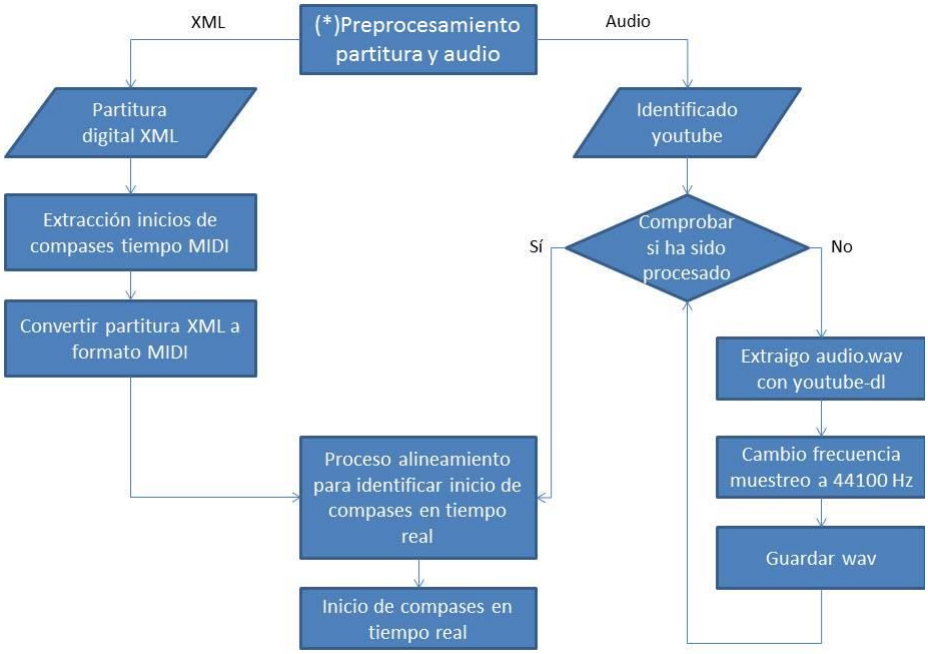


Figura 28.Diagrama de flujo de procesamiento audio y partitura digital

4.2.1. Procesamiento de audio

En este subapartado se va a explicar el desarrollo para el procesamiento de audio, una vez se extrae el identificador de Youtube.

4.2.1.1. Extracción de audio a través del programa youtube-dl

Para empezar se comentará que para la extracción del audio .wav se ha empleado el programa youtube-dl, que es un programa de línea de comandos que permite extraer vídeos y audios a partir de la url de un vídeo de youtube. Es un programa sencillo de utilizar, sólo escribir una línea en la ventana de comandos, que tiene muchas herramientas y posibilidades a la hora de realizar la descarga del fichero del audio.

En este proyecto, ya que este programa permite la siguiente utilidad, se ha empleado directamente para extraer el audio en formato wav, y guardarlo con el identificador del vídeo, para que así quede identificado y en el caso de existir ya ese wav o se haya descargado en nuestro directorio no se vuelva a procesar, ya que un usuario puede haber solicitado la petición para un mismo vídeo al que ya ha sido extraído, modificado y procesado el audio. Un ejemplo de cómo funciona o se ha empleado en el programa se puede ver en la Figura 29 mostrada a continuación.

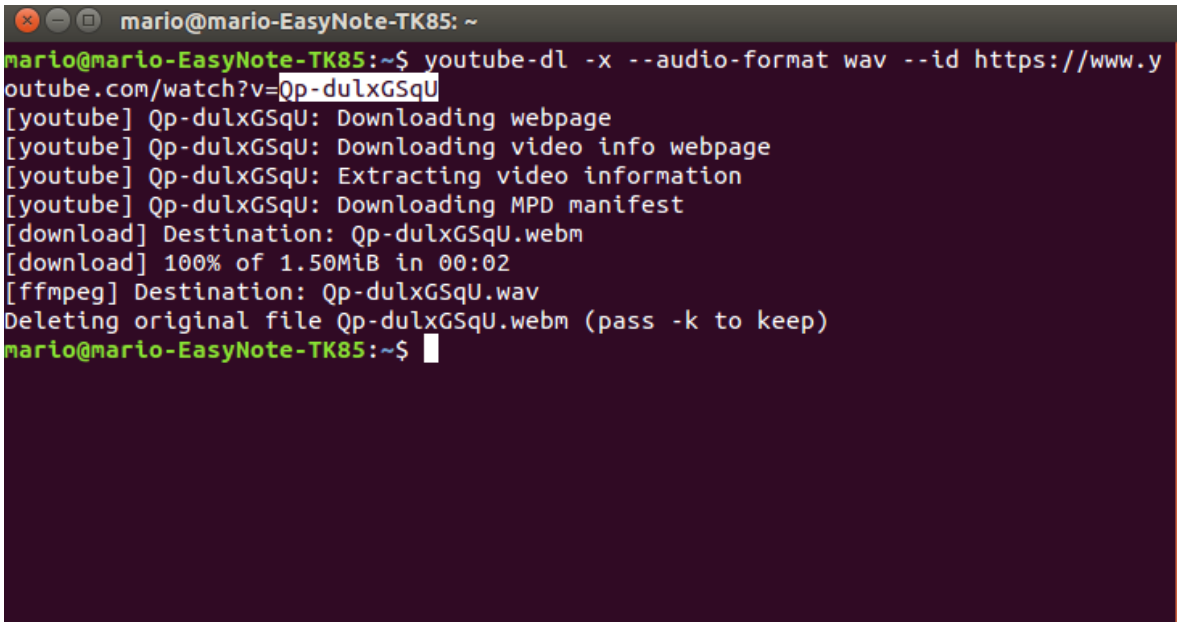


Figura 29. Ejemplo programa Youtube-dl

Hay que comentar, que el archivo de audio descargado, tiene una frecuencia de muestreo de 44.800 Hz, y que se necesita una frecuencia de muestreo de audio de 44.100 Hz, por lo que a continuación apartado se explicará su modificación.

4.2.1.2. Modificación frecuencia de muestreo mediante función resample

Como se comentó anteriormente se necesita para el procesamiento del audio en el alineamiento una frecuencia de muestreo de 44.100 Hz. Cuando nos referimos a frecuencia de muestreo, es simplemente el número de muestras por segundo, es decir, en el caso de 44.100 Hz se tienen 44.100 muestras por segundo.

Para ello, se ha empleado la función resample() que proporciona Matlab, que permite modificar la frecuencia de muestreo de una señal de audio a otra frecuencia de muestreo que nos interese, para así poderla procesar correctamente.

Una vez realizado esto, se guarda este nuevo fichero wav que será el que posteriormente se va a procesar y utilizar.

4.2.2. Procesamiento de partitura digital XML

Para el procesamiento de la partitura XML, mencionar que se ha utilizado el programa MuseScore que, como en el anterior caso también permite su utilización mediante la línea de comandos, y el empleo desde el entorno Matlab con la función system(). El empleo de MuseScore ha sido tanto para exportar a formato MIDI la partitura digital como para identificar los inicios de compases en tiempo MIDI identificados de la partitura

4.2.2.1. Formato MIDI de la partitura digital.

Ahora se va a explicar por qué se realiza el cambio de formato de musicXML a MIDI con MuSescore, ya que como se explicó en capítulos anteriores, este programa permite la exportación e importación de un formato a otro, y permite a través de una línea de comandos cambiar de formato de una partitura.

El archivo MIDI no contiene datos de audio muestreado, sino más bien una serie de instrucciones que el sintetizador, como puede ser Timidity, Fluidsynth, programas de reproducción musical u otro generador de sonido utiliza para reproducir el sonido en tiempo real. Estas instrucciones son mensajes MIDI que indican al instrumento qué sonidos hay que utilizar, qué notas hay que tocar, el volumen de cada una de ellas.

Por tanto, mediante Musescore (que permite trabajar con este formato) se pasa a formato MIDI la partitura, y en este formato es como se procesará en el algoritmo de alineamiento.

4.2.2.2. Identificación de inicios de compases en tiempo MIDI.

Otra utilidad que permite el programa MuseScore es la identificación de los inicios de compás de la partitura digital, que los emplearemos para añadir una nueva columna a la matriz nmat del archivo MIDI en la función del algoritmo de alineamiento audio2ScoreTreal_offline(), que identifica en tiempo MIDI (en milisegundos) a que compás pertenece cada nota.

Todo este proceso se realiza con la función inicioCompases(), que se explicará con detalle en el manual técnico.

Al ejecutar esta opción del Musescore se crea un archivo con la extensión .mpos que puede abrirse con cualquier editor de texto, ya que es un archivo con estructura XML, y ver cómo en el ejemplo de la Figura 30, como está estructurado por etiquetas ofreciendo la información de compás y su inicio en tiempo MIDI (en milisegundos) para la partitura que se quiera procesar.

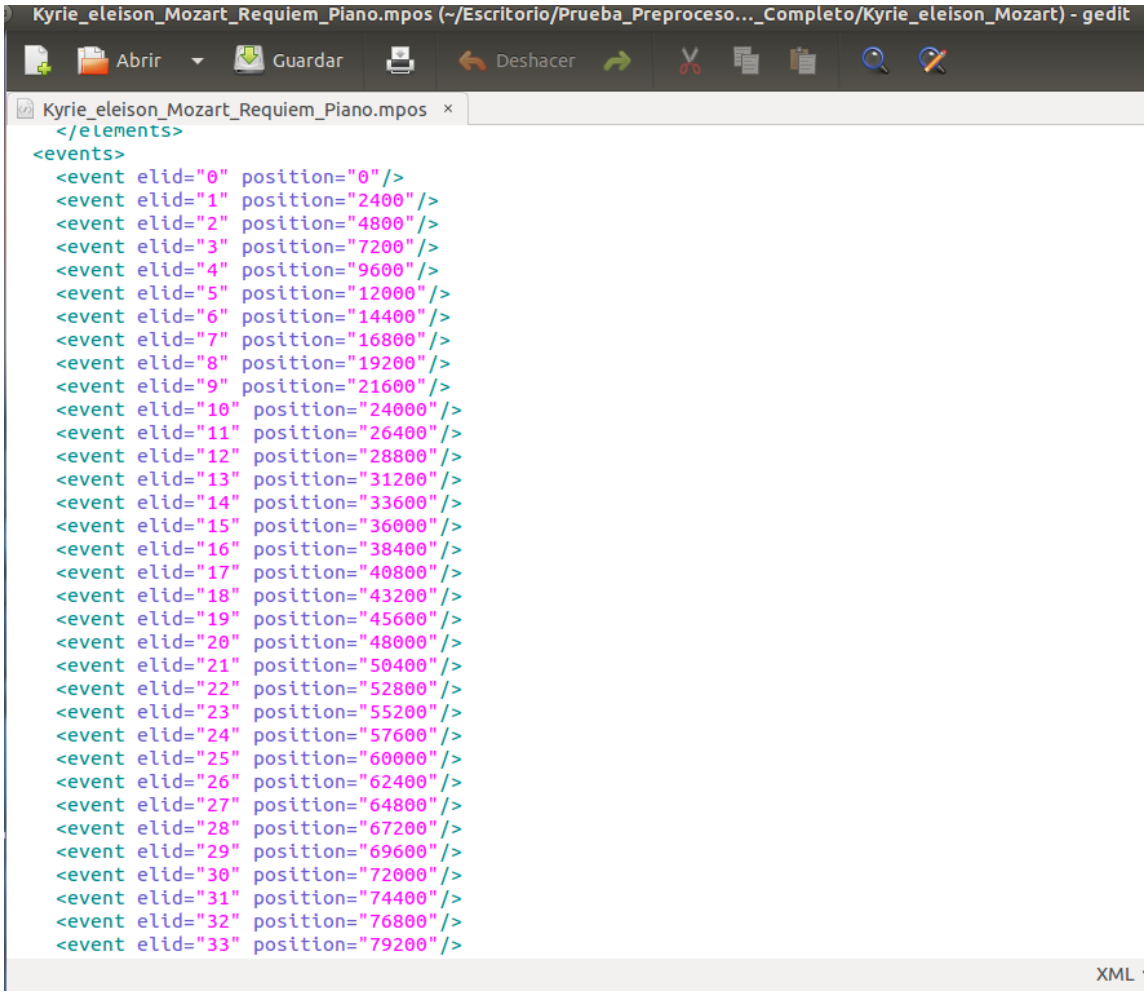


Figura 30. Identificación inicios de compases en tiempo MIDI

A continuación, se parsea o analiza sintácticamente este fichero, lo que se realiza con este archivo es leer, localizar y extraer el valor del atributo “position” que es el valor del inicio de cada compás en tiempo MIDI (en milisegundos) y el valor del atributo “elid” para cada uno de los elementos “event”. Obteniendo un vector de salida con el tiempo inicio de compás y número de compás correspondiente. Este proceso se realiza a través del script.

Finalmente, una vez se preprocesan estos tres elementos explicados anteriormente (audio en formato wav, inicios de compases en tiempo MIDI y partitura en formato MIDI) se pueden procesar mediante la función audio2ScoreTreal_offline () para sincronizar audio-partitura y obtener los inicios de compases en tiempo real en el momento que se produce o toca cada en el audio.

4.2.3. Programa algoritmo de alineamiento.

A continuación una vez se ha realizado este preproceso se tienen todos los parámetros para realizar el alineamiento con la función audio2ScoreTreal_offline(), ya que se tienen tanto el audio, como el fichero MIDI de la partitura como identificados el inicio de compases en tiempo MIDI de la partitura, y al final de este proceso se obtendrá un vector de inicios en tiempo real de la sincronización de audio y partitura, que una vez calculados se almacenarán en la tabla *peticiones.*

Para comprender mejor el proceso de esta función en este subapartado se va a seguir la explicación con un ejemplo de una partitura digital *“Bach789Sco.xml”* del compositor Johann Sebastian Bach llamada *Sinfonia III in D (BWV-783)* con el audio de su interpretación. Por tanto, para esta pareja de audio y partitura se va a describir lo que se realiza en la función, para cualquiera el procedimiento es el mismo.

Para empezar, se supone que ya se ha ejecutado la función anteriormente descrita de inicioCompases() y se han obtenido la partitura en formato MIDI y un vector con el inicio de compases en tiempo MIDI (en milisegundos), la variable o vector *res* compuesto por el tiempo MIDI donde comienza cada compás de la partitura, que se pasa a segundos para utilizarlo posteriormente y su número correspondiente de compás (en este caso se tienen 25 compases). En la Figura 31 se puede ver qué parámetros de entrada tiene la función audio2ScoreTreal_offline(), el archivo .txt es el resultado que se obtendrá del alineamiento sincronizado con el audio en tiempo real en formato MIREX que fue detallado en la subsección 2.5.4.

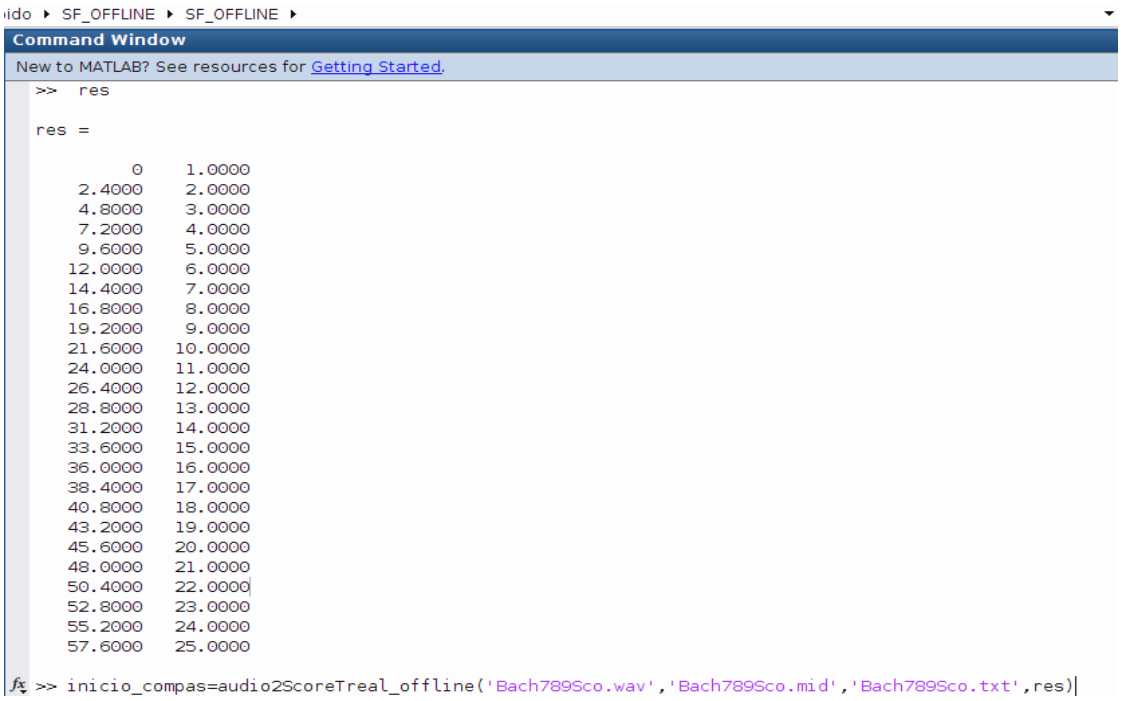


Figura 31.Llamada a función alineamiento audio2ScoreTreal_offline

Se va a realizar todo de forma manual, paso a paso, para ir viendo y detallando el proceso e ir explicando el desarrollo del código, pero todo se realiza de forma automática como se explica a lo largo de la memoria, ya que siempre está operativo este programa consultando y procesando peticiones.

En primer lugar, una vez se ejecuta la función para el alineamiento se generan los parámetros de configuración del sintetizador Timidity con las fuentes de sonido, para generar la señal sintética del archivo MIDI.

Se carga la transcripción desde el archivo MIDI con toda su información, es decir la matriz nmat que se explicó en el subapartado 2.4.1.1 y que información proporcionaba esta matriz acerca del archivo MIDI. En la Figura 32, se muestra la matriz nmat para este archivo MIDI con el que se está desarrollando el desarrollo *(“Bach789Sco.mid”),* donde se tiene toda la información de las notas de la partitura, en este caso una matriz con 684 filas y 7 columnas (en la Figura 32 se muestra parte de esta matriz).

```
nmat =

         0    0.9979    3.0000   62.0000   75.0000         0    0.5987
    0.5000    0.2479    1.0000   78.0000   75.0000    0.3000    0.1487
    0.5000    0.2479    2.0000   78.0000   75.0000    0.3000    0.1487
    0.7500    0.2479    1.0000   79.0000   75.0000    0.4500    0.1487
    0.7500    0.2479    2.0000   79.0000   75.0000    0.4500    0.1487
    1.0000    0.4979    1.0000   81.0000   75.0000    0.6000    0.2988
    1.0000    0.4979    2.0000   81.0000   75.0000    0.6000    0.2988
    1.5000    0.4979    1.0000   72.0000   75.0000    0.9000    0.2988
    1.5000    0.4979    2.0000   72.0000   75.0000    0.9000    0.2988
    1.5000    0.4979    3.0000   54.0000   75.0000    0.9000    0.2988
    2.0000    0.4979    1.0000   71.0000   75.0000    1.2000    0.2988
    2.0000    0.4979    2.0000   71.0000   75.0000    1.2000    0.2988
    2.0000    0.9979    3.0000   55.0000   75.0000    1.2000    0.5987
    2.5000    0.2479    1.0000   76.0000   75.0000    1.5000    0.1487
    2.5000    0.2479    2.0000   76.0000   75.0000    1.5000    0.1487
    2.7500    0.2479    1.0000   78.0000   75.0000    1.6500    0.1487
    2.7500    0.2479    2.0000   78.0000   75.0000    1.6500    0.1487
    3.0000    0.4979    1.0000   79.0000   75.0000    1.8000    0.2988
    3.0000    0.4979    2.0000   79.0000   75.0000    1.8000    0.2988
    3.5000    0.4979    1.0000   71.0000   75.0000    2.1000    0.2988
    3.5000    0.4979    2.0000   71.0000   75.0000    2.1000    0.2988
    3.5000    0.4979    3.0000   52.0000   75.0000    2.1000    0.2988
    4.0000    0.4979    1.0000   69.0000   75.0000    2.4000    0.2988
    4.0000    0.9979    3.0000   54.0000   75.0000    2.4000    0.5988
    4.5000    0.2479    1.0000   74.0000   75.0000    2.7000    0.1488
    4.7500    0.2479    1.0000   76.0000   75.0000    2.8500    0.1488
```

Figura 32. Matriz nmat de Bach789Sco.mid

Ahora es cuando entra en el proceso, el vector *res* calculado inicialmente que le pasamos como parámetro a esta función. Se sabe, que ese vector indica en tiempo MIDI el inicio de cada compás de la partitura, pues bien, también se conoce el momento en qué se inicia cada nota (en segundos) observando la sexta columna de la matriz nmat.

Por tanto, lo que se va a realizar es identificar a qué compás pertenecen cada nota de la matriz nmat conociendo cada valor de inicios de compases (en segundos) que se calculó con la función inicioCompases(), entonces se añade una nueva columna a la matriz nmat pasando a tener 8 columnas como se puede ver en la Figura 33 (misma matriz nmat pero con una columna nueva al final), la última columna informa de a qué compás pertenece cada nota. Para realizarlo se recorre toda la matriz nmat identificando los valores que están comprendidos entre cada compás, recorriendo la sexta columna y comparándola con el vector *res,* y así se ubica cada nota con su compás correspondiente.

```
nmat =

         0    0.9979    3.0000   62.0000   75.0000         0    0.5987    1.0000
    0.5000    0.2479    1.0000   78.0000   75.0000    0.3000    0.1487    1.0000
    0.5000    0.2479    2.0000   78.0000   75.0000    0.3000    0.1487    1.0000
    0.7500    0.2479    1.0000   79.0000   75.0000    0.4500    0.1487    1.0000
    0.7500    0.2479    2.0000   79.0000   75.0000    0.4500    0.1487    1.0000
    1.0000    0.4979    1.0000   81.0000   75.0000    0.6000    0.2988    1.0000
    1.0000    0.4979    2.0000   81.0000   75.0000    0.6000    0.2988    1.0000
    1.5000    0.4979    1.0000   72.0000   75.0000    0.9000    0.2988    1.0000
    1.5000    0.4979    2.0000   72.0000   75.0000    0.9000    0.2988    1.0000
    1.5000    0.4979    3.0000   54.0000   75.0000    0.9000    0.2988    1.0000
    2.0000    0.4979    1.0000   71.0000   75.0000    1.2000    0.2988    1.0000
    2.0000    0.4979    2.0000   71.0000   75.0000    1.2000    0.2988    1.0000
    2.0000    0.9979    3.0000   55.0000   75.0000    1.2000    0.5987    1.0000
    2.5000    0.2479    1.0000   76.0000   75.0000    1.5000    0.1487    1.0000
    2.5000    0.2479    2.0000   76.0000   75.0000    1.5000    0.1487    1.0000
    2.7500    0.2479    1.0000   78.0000   75.0000    1.6500    0.1487    1.0000
    2.7500    0.2479    2.0000   78.0000   75.0000    1.6500    0.1487    1.0000
    3.0000    0.4979    1.0000   79.0000   75.0000    1.8000    0.2988    1.0000
    3.0000    0.4979    2.0000   79.0000   75.0000    1.8000    0.2988    1.0000
    3.5000    0.4979    1.0000   71.0000   75.0000    2.1000    0.2988    1.0000
    3.5000    0.4979    2.0000   71.0000   75.0000    2.1000    0.2988    1.0000
    3.5000    0.4979    3.0000   52.0000   75.0000    2.1000    0.2988    1.0000
    4.0000    0.4979    1.0000   69.0000   75.0000    2.4000    0.2988    2.0000
    4.0000    0.9979    3.0000   54.0000   75.0000    2.4000    0.5988    2.0000
    4.5000    0.2479    1.0000   74.0000   75.0000    2.7000    0.1488    2.0000
```

Figura 33. Matriz nmat con nueva columna para indicar el compás correspondiente

Así una vez se produzca el alineamiento con el archivo de audio .wav aparezca también una nueva columna en el resultado en formato MIREX y sea posible identificar el inicio en tiempo real de cada compás con el momento que se ha producido en el audio.

A continuación se definen y se generan los diferentes estados para aprender sus bases y poder realizarse el alineamiento. El proceso hasta el alineamiento que desarrolla el algoritmo se puede ver en la Figura 34, tal y como se explicó en el apartado 2.5.

```
>> inicio_compas=audio2ScoreTreal_offline('Bach789Sco.wav','Bach789Sco.mid','Bach789Sco.txt',res)


PREPROCESSING

Generating synthetic signal from MIDI ... Playing Bach789Sco.mid
MIDI file: Bach789Sco.mid
Format: 1  Tracks: 3  Divisions: 480
Playing time: ~63 seconds
Notes cut: 0
Notes lost totally: 0

Done
Reading synthetic signal ... Done
Computing ground-truth transcription from MIDI ... Done
Preparing synthetic input data ... Done
Generating states ... Done
 Learning combination basis from synthetic data ... Done

STARTING ALIGNMENT PROCESS

Reading signal from WAV ... Done
Computing spectrogram for real signal ... Done
Estimating gains from real testing data ... Done
Peforming the alignment ... Done
Saving output data ...476       inicio_compas=compases(:,1)/1000;
```

Figura 34. Desarrollo del proceso de la función audio2ScoreTreal_offline

Una vez se ha completado el alineamiento se puede ver el resultado gráficamente en la Figura 35 donde se comprueba y verifica que funciona correctamente este algoritmo realizando el camino más corto lineal y en diagonal.

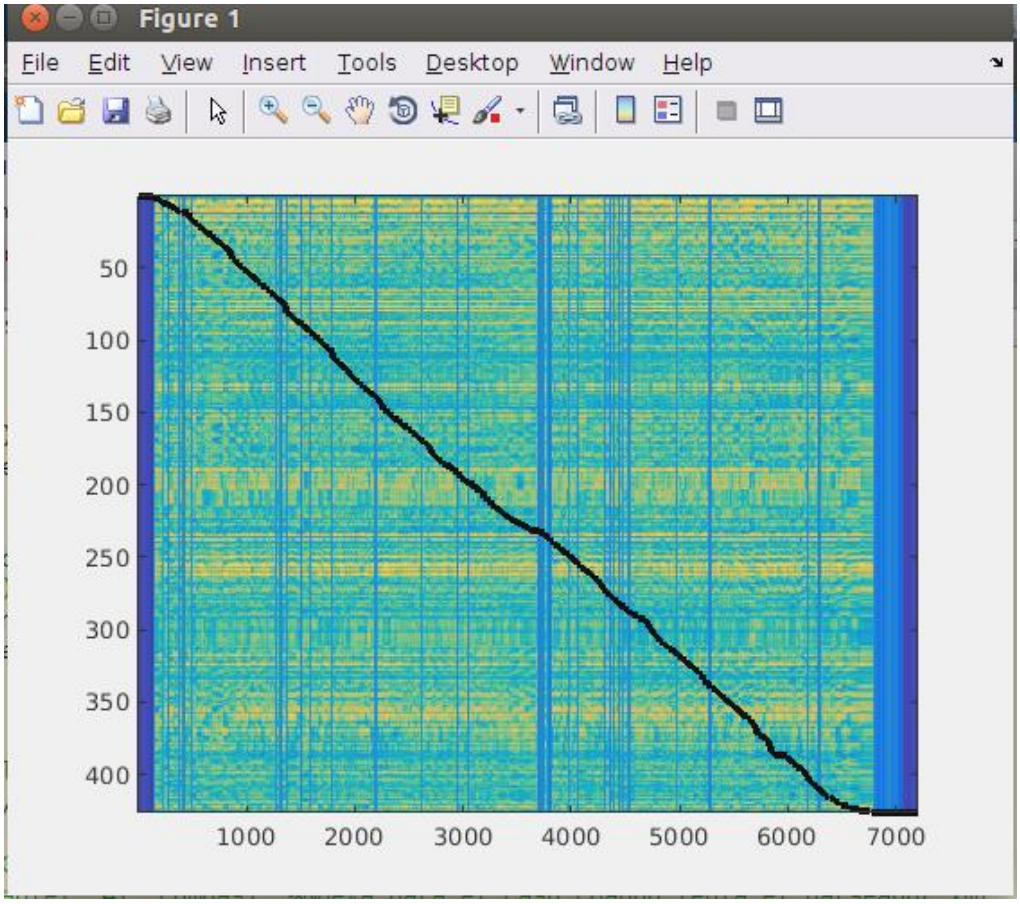


Figura 35. Resultado de alineamiento DTW para el par Bach789.mid y Bach789.wav

Y el resultado de la tabla sincronizada audio y partitura en el formato MIREX como se puede ver en la Figura 36, donde se puede comprobar que hay una columna más añadida al final, que es al compás que pertenece cada nota.

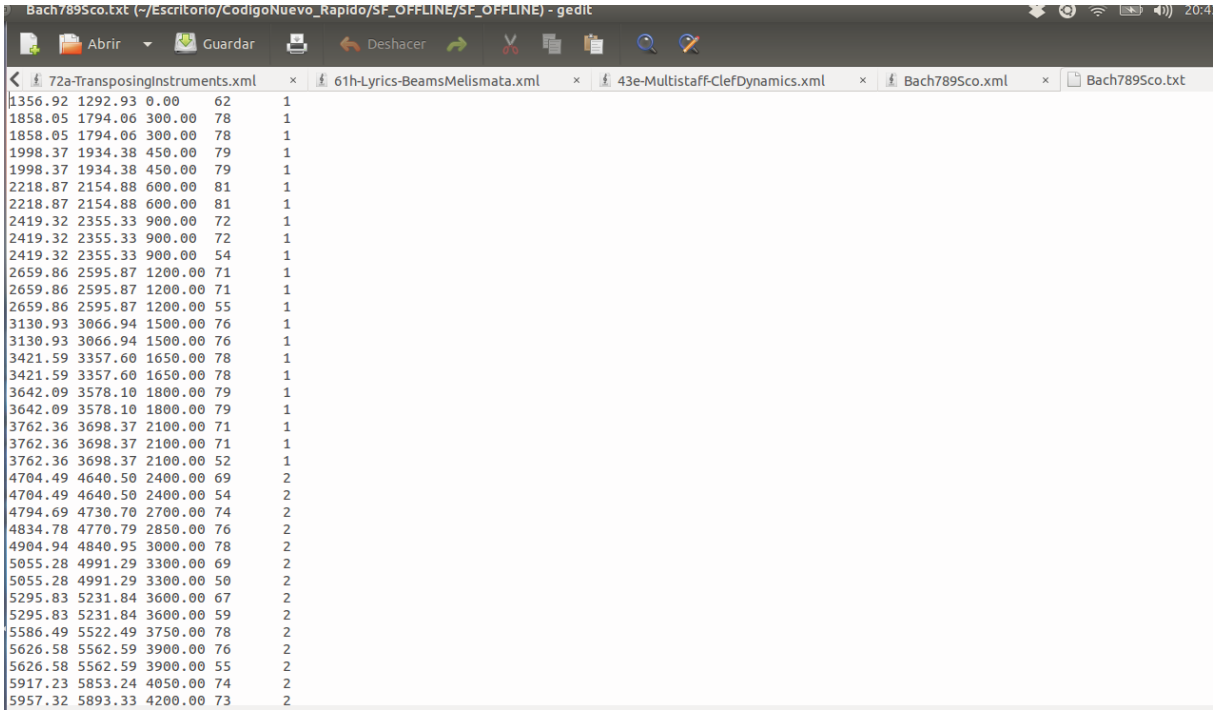


Figura 36. Resultado en formato MIREX del alineamiento

Por último, de esta Figura 36 lo que nos va a interesar es obtener el tiempo real en el que se produce el inicio de cada compás, es decir, la primera columna de esta tabla. Finalmente se obtienen esos valores de tiempo de inicio de cada compás, en el caso, en el que estamos trabajando, como se vio hay 25 compases. Por tanto, recorriendo esta matriz se identificará el tiempo inicio de cada uno de estos 25 compases (en segundos) en tiempo real sincronizados con el audio, obteniendo el resultado que se puede ver en la Figura 37,

```
inicio_compas =

    1.3569
    4.7045
    6.6188
    9.2849
   11.8406
   14.3062
   16.6215
   19.1171
   21.8433
   24.2487
   26.8546
   29.7311
   32.1065
   34.8226
   38.6513
   41.3073
   43.8531
   47.0503
   49.0849
   51.7008
   54.2265
   56.8625
   58.6566
   61.8538
   64.5399
```

Figura 37. Valores de inicios para cada compás (en segundos) para la petición Bach789.mid y Bach789.wav

Comentar que una vez calculados estos inicios de compases para las peticiones que se estén pendientes, se van almacenando en la base de datos, conforme se vayan calculando. Esto ha sido para un ejemplo en concreto, pero para todas, el desarrollo sería

exactamente el mismo, con el objetivo de proporcionar al final un vector con cada valor de inicio para cada compás.

4.3. Aplicación web desarrollada

En este apartado se va a detallar la aplicación web con más herramientas que se explicaron en el capítulo 2.2 sobre la visualización en el navegador de usuario utilizando abcweb, cómo y para qué se han añadido más utilidades a esta herramienta para el usuario, tanto el diseño para su visualización y datos de oyente como información de la partitura y comentarios para cada compás de la obra, así el usuario tendrá más herramientas y utilidades a la hora de usar la aplicación web aparte de lo que ofrece el software de abcweb. Lo que el usuario verá en su navegador una vez ejecute la aplicación web, y sean cargados vídeo y partitura, será lo que se muestra en la Figura 38.

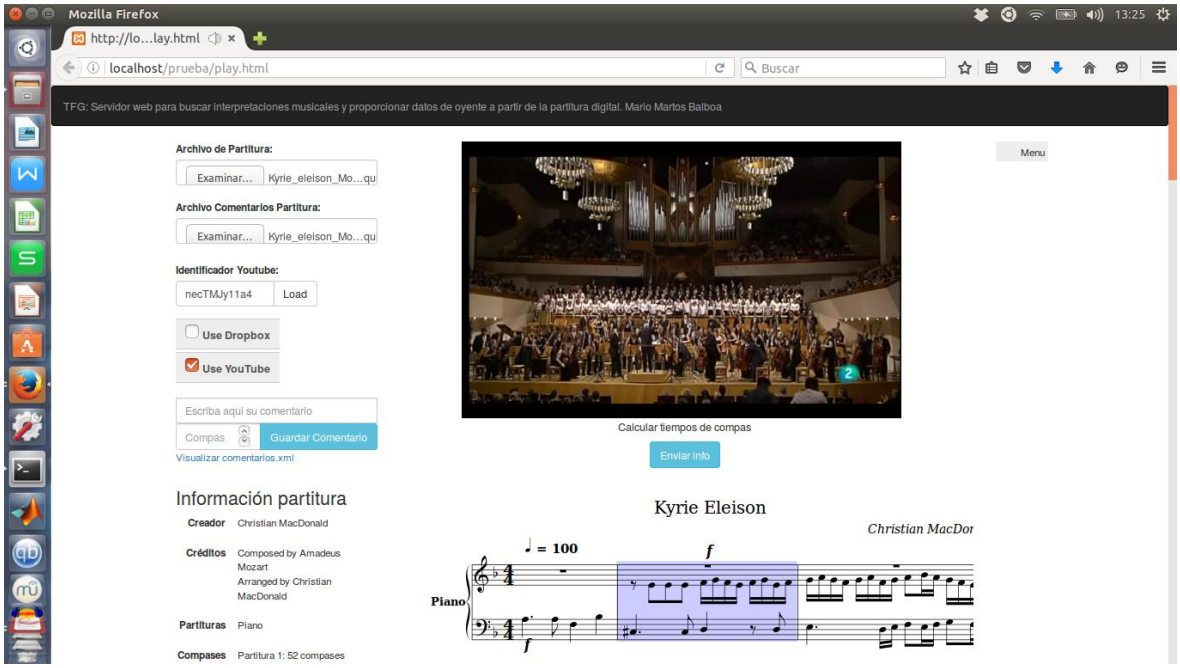


Figura 38. Aplicación web final

A continuación, se va a explicar la estructura y su funcionamiento de sus diferentes con detalle.

4.3.1. Estructura y funcionamiento

A continuación se van a detallar todos los componentes que forman la aplicación web, que permiten la visualización y herramientas que se ven en la Figura 38 y han sido añadidos al software abcweb que se explicó en la sección 2.2. Todos estos componentes se encuentran en la carpeta *mario*, donde se encuentra el archivo *play.html* que es el que se ejecuta en el navegador del usuario. Aparte de tener incluidas las herramientas

básicas de este software, como puede ser el menú de la parte derecha, visor de partitura, etc. Se suman los componentes que se van a incluir a continuación.

- Diseño:

Para el diseño de la interfaz de usuario se usó un diseño predefinido de la librería llevada a cabo por bootstrap [21] para modificar el diseño y estilo de páginas web. Se importó los CSS (Cascade Style Set) y se hizo uso de una de ellas para realizar la interfaz de usuario. Esta herramienta permite adaptarse a distintos tamaños de pantalla de cualquier dispositivo, haciendo fácil y ventajoso su utilización. Se tuvo que instalar bootstrap en el servidor Lamp empleado en el proyecto como se detalló con anterioridad y posteriormente se cambió el estilo para mejorar la interfaz de usuario y permitir visualizarse como en la Figura 38.

- Información básica de la partitura:

En segundo lugar, se puede observar en la Figura 38, en la parte inferior izquierda, que aparece un bloque con información básica de la partitura que se carga, con el objetivo de que el usuario pueda ver algunos datos acerca de la partitura como pueden ser el creador, los compases, los instrumentos, etc. Para ello, se desarrolló un script que permitiera obtener la información de las etiquetas que interesasen de la partitura musicXML, como se describieron estas etiquetas en el apartado 2.1.5, se identifican de la partitura que se procesa con formato XML para ser mostradas en ese bloque en cuestión.

- Editor de comentarios:

En la interfaz de usuario, en la parte de la izquierda, se puede ver que hay un editor de comentarios, el cual va a permitir al usuario escribir comentarios por cada compás, seleccionando el compás que le interese comentar, una vez realizado lo comentado, pulsando en guardar comentario se muestra más abajo el compás para el que ha comentado y el comentario al respecto, como se puede ver un ejemplo en la Figura 39 que se muestra a continuación un comentario para el primer y segundo compás respectivamente.

Figura 39. Editor de comentarios

Por otro lado, para ver la estructura que tienen los comentarios que se acaban de Editar, habrá que pulsar “Visualizar comentarios.xml” y se mostrará una nueva ventana en el navegador, como se puede ver en la Figura 40, donde se puede ver la estructura o formato en XML que tienen dichos comentarios.

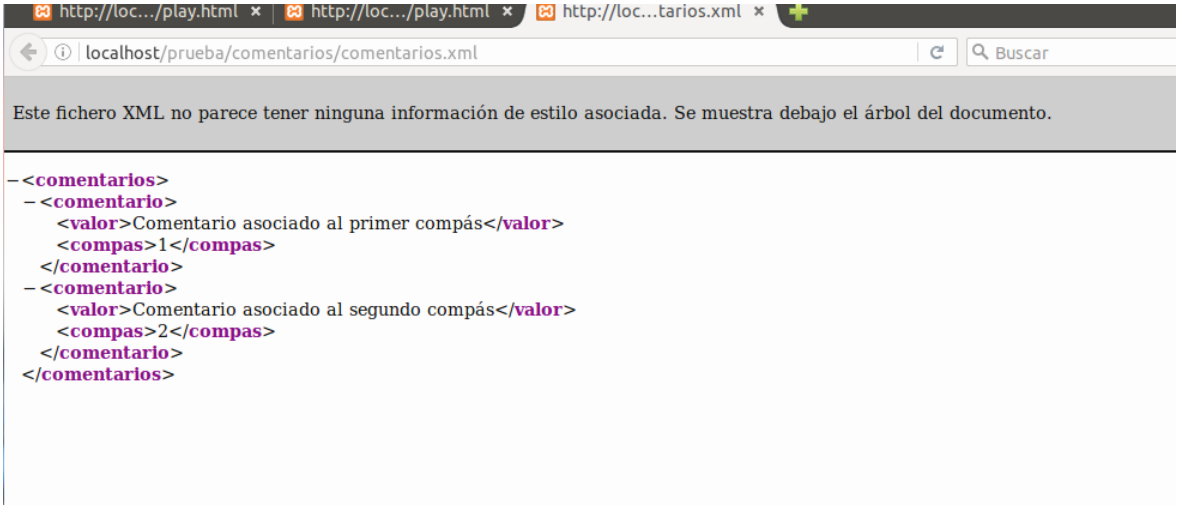


Figura 40. Estructura comentarios de usuario

También existe la posibilidad de cargar directamente unos comentarios en formato XML que se tengan en el disco duro local ya realizados, habría que seleccionarlos en el botón “Archivos Comentarios Partitura”

- Llamada al servicio desarrollado:

Como se puede ver existe un botón entre el vídeo y la partitura con el nombre “Enviar Info” para calcular los inicios de compás. Pues bien, este botón tiene la función de conectar con el servicio mediante una llamada a *ajax,* cada vez que el usuario quiera realizar una petición; tendrá que pulsar ese botón para realizar la llamada al servicio

descrito en el apartado 4.1 realizando todo el proceso. En la Figura 41 se puede ver como se realiza esta llamada al servicio para el cálculo de los inicios de compases conociendo ya toda la estructura y funcionamiento explicado anteriormente.

```
<script type="text/javascript">
var times_arr_calculados;

$("#btnSubmit").click(function(){
  var enlace=document.getElementById("enlaces");
  var fd = new FormData(document.getElementById("formu"));
  enlace.innerHTML="....Procesando.....";
    $.ajax({
    url: "http://127.0.0.1/mario/inicios",
    type: "POST",
    data: fd,
    dataType: "json",
    processData: false,
    contentType: false,
    success: function (data){
                times_arr_calculados=data.times_arr;
                if (times_arr_calculados==null){
                }
                else{
                   enlace.innerHTML=JSON.stringify(times_arr_calculados);
                }
            }
    });
```

Figura 41. Llamada al servicio en interfaz de usuario

Se puede ver como hace la llamada a *http://127.0.0.1/mario/inicios* que es la carpeta donde se encuentra la estructura del servicio y el nombre del propio servicio. El proceso que se realiza es el que se explicó en el servicio desarrollado, como se puede ver hay una variable *time_arr_calculados* que es en el caso de que esté ya procesada la petición tendremos un vector de tiempos almacenados en esa variable, que se mostrará en el caso de existir en pantalla.

Por último, una vez obtenidos los inicios de compases, el usuario puede guardar este js, a través del menú la opción de *save,* la cual ha sido localizada y modificada en el código y software abcweb. Ésta función modificada en el código de abcweb.js es para en el caso de tener el vector de tiempos ya calculado se nos guarde éste en lugar del de por defecto que nos proporciona esta herramienta y así si se requiere volver a utilizar la misma pareja partitura y vídeo de Youtube no será necesario repetir el proceso, simplemente cargar este archivo JS descargado con los valores de inicios de compases ya obtenidos. Como se puede en la variable *times_arr,* ver en la Figura 42, el archivo guardado y descargado de la izquierda (valores por defecto con los que se guardan si no son modificados todos los JS) está con unos valores distintos al de la derecha que son los que nos interesan, los calculados y procesados ya almacenados en ese vector, donde se encuentran los valores por defecto.

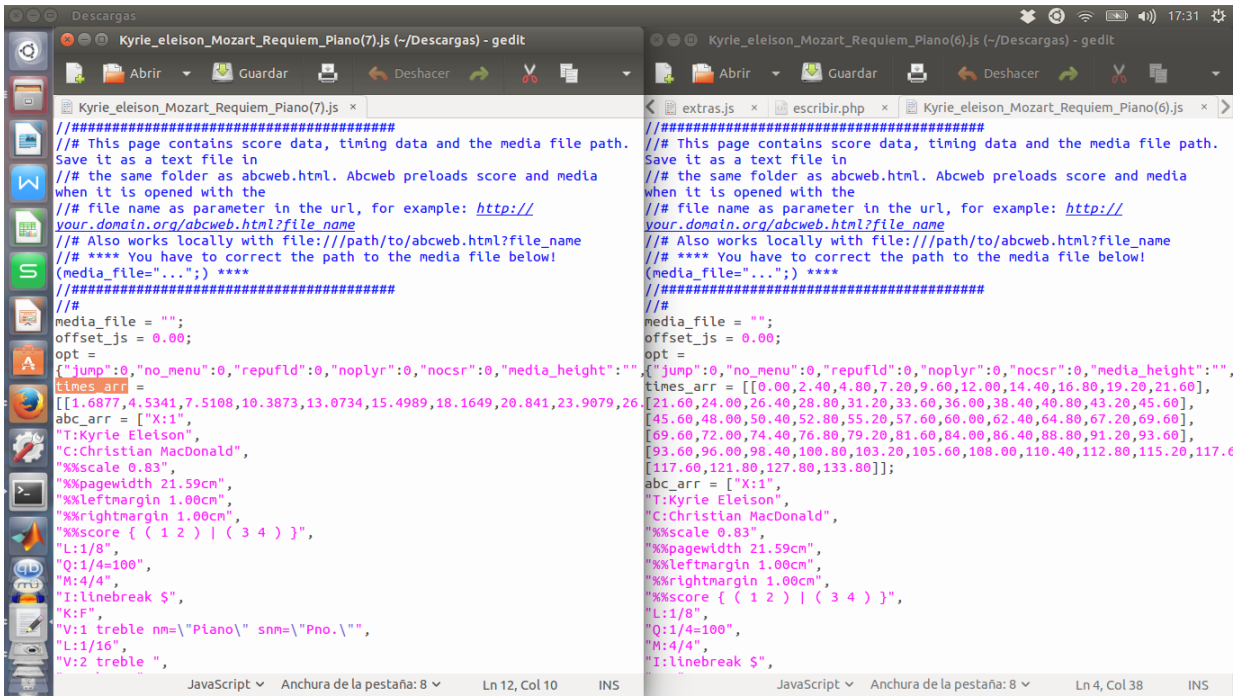


Figura 42. Almacenado JS con inicios de compases calculados

Finalmente si se quiere cargar este JS previamente guardado en nuestra computadora, simplemente hay que seleccionarlo en el botón de Archivo Partitura, cargándose para estos valores de tiempos ya calculados.

5. RESULTADOS Y DISCUSIÓN

En este apartado se van a detallar sobre el trabajo final obtenido en el presente proyecto junto con unas posibles líneas de futuro que se podrían desarrollar.

5.1. Trabajo final obtenido

El trabajo final obtenido es un programa que identifica y calcula, con una gran precisión el tiempo en el que se produce el inicio de cada compás de una obra, en qué momento la persona que toca una nota corresponde con la nota de la partitura, para de esta manera quede sincronizado en el tiempo real en el que ocurre. Mencionar que, dependiendo de la duración o extensión de la trama que se seleccione, este proceso de alineamiento explicado a lo largo de la memoria puede tener una mayor o menor duración, pero siempre con la capacidad de realizar este alineamiento con una suficiente precisión, verificando el rendimiento del algoritmo de alineamiento en su versión offline.

A continuación se van a mostrar algunos resultados a la hora de realizar el alineamiento con el programa empleado, se llega a la conclusión que se puede ver de como el alineamiento sigue un camino lineal lo más diagonal posible para obtener el camino más corto. En la Figura 43, que es el resultado de procesar la obra *Piano Sonata No.9, Mov.2 compuesta de Ludwig van Beethoven* con un audio de duración 3 minutos 37 segundos, con un tiempo de espera hasta completar el proceso de unos 5 minutos aproximadamente.

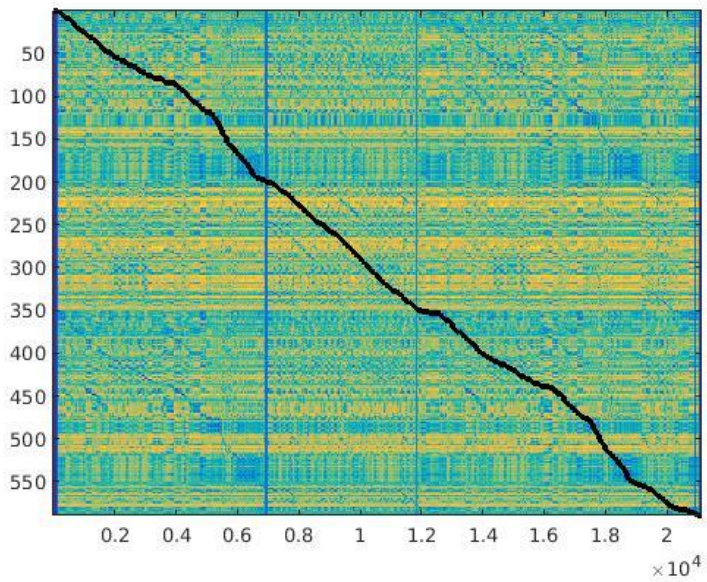


Figura 43. Resultado alineamiento Sonata No.9, Mov.2 Ludwig van Beethoven

Para otro ejemplo, este empleando un audio de más duración de 6:20 minutos de la obra *Adagio para piano compuesta por Wolfgang Amadeus Mozart* se obtiene el resultado de alineamiento de la Figura 44, empleando un tiempo de procesamiento de unos 10 minutos aproximadamente.

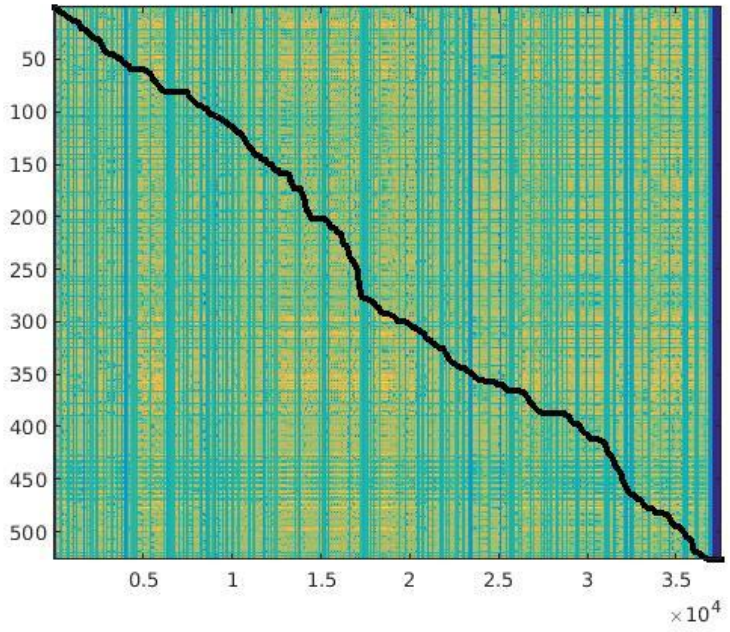


Figura 44. Resultado alineamiento Adagio para piano de Wolfgang Amadeus Mozart

Por consiguiente, se puede llegar a la conclusión de que, cuánta más duración tengan las obras que queremos procesar, más tiempo de procesamiento habrá, teniendo por tanto, un mayor tiempo de espera, y por tanto mayor gasto de consumo, memoria y batería para la computadora.

Por otro lado, al implementarse un servicio con una base de datos y un servicio, destacar que el sistema propuesto es escalable y así aumentar la envergadura del sistema, quiere decir que aunque haya sido desarrollado de forma local, está preparado para que varios usuarios se conecten a la base de datos para solicitar peticiones, ya que gracias al servicio, se establece una lógica de control para administrar estas peticiones de tantos usuarios como estén conectados al servidor. Dicho lo cual, varios ordenadores pueden estar conectados a un ordenador troncal que sea el que vaya procesando las peticiones con el fin de dar un mejor rendimiento y dinamismo, ya que para una petición ya procesada y que coincida no se vuelva a realizar el cálculo, ofreciendo por tanto fluidez en el proceso.

Finalmente, la aplicación web desarrollada de abcweb que permite visualizar la sincronización de vídeo-partitura en el navegador para los tiempos de inicios de compases reales una vez procesados. También se muestra información básica para el

usuario de la partitura junto con un editor de comentarios que permite poder realizar comentarios para cada compás de la obra.

5.2. Líneas de futuro

Este trabajo realizado se puede complementar con diversas mejoras surgidas a lo largo de su implementación, desde realizarlo en otro lenguaje de programación como C, Python o Java, que son lenguajes más ligeros que Matlab, o también, utilizando otro ordenador más potente o con más prestaciones con el que se ha realizado, ahorrando tiempo de procesamiento.

Por otro lado, comentar que se ha realizado con el fin de tener la posibilidad de tener un sistema escalable, es decir, poder atender a múltiples usuarios, que se pueda aumentar la red o estructura del sistema conectando tantos ordenadores como de los que se quieran disponer a un servidor que vaya procesando solicitudes, dado que con la lógica de control implementada lleguen tantas peticiones como usuarios soliciten, con esto se obtendrá una mayor eficiencia ya que cuántas más peticiones hayan sido procesadas en un futuro al poder un usuario solicitar el cálculo para un par ya procesado no se volvería a realizar el proceso, obteniendo una mayor rapidez. Al realizar este sistema, con muchos usuarios conectados realizando peticiones, se podría identificar a cada uno de ellos mediante por ejemplo un formulario con nombre de usuario y contraseña y así para cada uno almacenar sus comentarios, partituras u otro tipo de información.

También mencionar, que se podría emplear también para, en lugar de identificar inicios de compases, obtener en qué tiempo ocurre cada nota de una partitura digital para una mayor precisión.

6. CONCLUSIONES

En este apartado se van a detallar las conclusiones obtenidas una vez concluido todo el proceso del proyecto.

6.1. Conclusiones obtenidas.

Para empezar comentar que este proyecto ha sido desarrollado en el sistema operativo Linux, pero podría desarrollarse en cualquier otro, ya que los programas utilizados pueden instalarse en cualquier otra plataforma y funcionar adecuadamente.

Fue desarrollado en Linux ya que proporcionaba una mayor comodidad y facilidad a la hora de instalar y compilar ciertos programas necesarios en el desarrollo del proyecto. Mencionar que se optó por este sistema operativo aparte de lo expuesto anteriormente, porque durante los primeros meses de la realización se estaba utilizando sobre Windows y con parseadores de partituras XML basadas en lenguaje Matlab, pero ambos daban problemas y errores, y aumentaban la dificultad de identificar inicios de compases en tiempo MIDI de la partitura, puesto que para partituras con muchas claves, notas u instrumentos fallaba y no funcionaba correctamente; por lo que se decidió realizar este proceso a través del programa Musescore, que permitía realizar lo descrito por líneas de comandos.

Antes de realizar el presente trabajo no se poseían grandes conocimientos del sistema operativo empleado, pero a día de hoy destacar la sencillez de instalar y compilar programas o ficheros en Linux, por tanto se han adquirido conocimientos y manejo en este sistema operativo.

Además de conocimientos que ya se poseían de procesado de audio y programación en Matlab por la rama de conocimiento de telecomunicaciones y sonido e imagen, se han adquirido una mayor destreza en otros lenguajes de programación como Java, XML, PHP, HTML y bases de datos entre otras tecnologías más vinculadas a la rama de telemática que han complementado el trabajo realizado.

Finalmente, se va a mencionar algunos problemas que se han visto con abcweb a la hora de probar o testear con diferentes tipos de partituras se ha observado que el visor a veces falla ya que no todos los casos salen correctamente, vienen expuestos y han sido consultados en “Unofficial MusicXML test suite” [20], es una base de datos donde se encuentran diversos tipos de archivos MusicXML de ejemplo, con las distintas opciones que se pueden dar en una partitura. A continuación, se muestran dos ejemplos para los que ha sido probado el visor abcweb y se han detectado fallos o que no lo realiza

correctamente, obtenidos de la librería de ejemplos de partituras citada anteriormente. A continuación se van a mostrar algunos ejemplos dónde el visor de abcweb falla.

En primer lugar, cargando el archivo "*43e-Multistaff-ClefDynamics.xml*" , que es una partitura como se puede ver en la Figura 45, con diversos cambios de clave, donde los elementos se aplican a una sola voz el visor abcweb no lo carga adecuadamente, como se ve en la Figura 46, fallando en este tipos de casos.

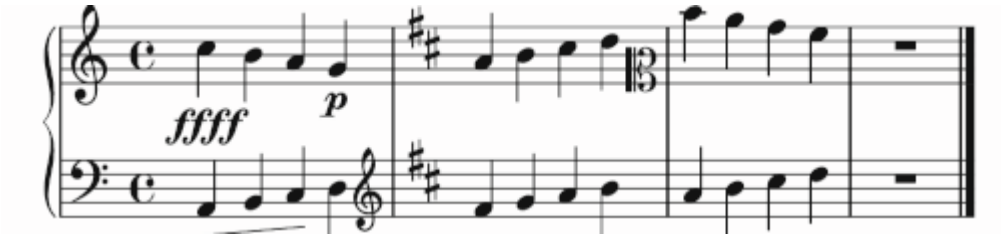


Figura 45. Ejemplo MusicXML partitura "43e-Multistaff-ClefDynamics.xml"

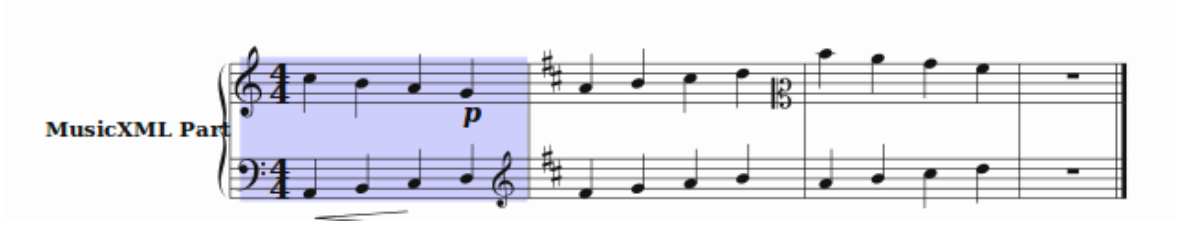


Figura 46. Ejemplo visor abcweb partitura "43e-Multistaff-ClefDynamics.xml"

Para otro ejemplo de prueba, con el archivo "*72a-TransposingInstruments.xml*" que es una partitura con varios instrumentos, como se observa en la Figura 47 y con clave y escala en C mayor, tampoco se muestra correctamente en el visor abcweb como se puede ver en la Figura 48.

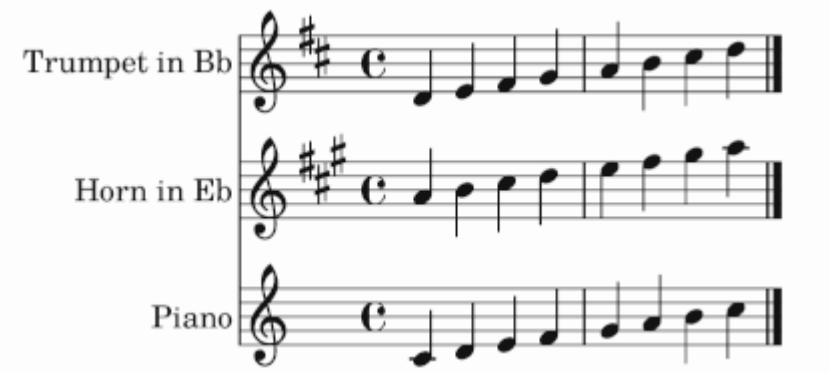


Figura 47. Ejemplo MusicXML partitura "72a-TransposingInstruments.xml"

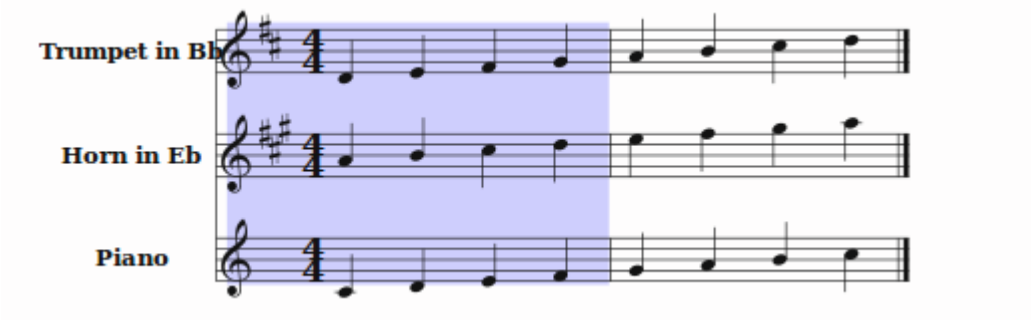


Figura 48. Ejemplo visor abcweb con "72a-TransposingInstruments.xml"

Finalmente, observando los ejemplos anteriores se puede comprobar que la herramienta abcweb para algunos casos determinados de partituras, como los anteriores, da problemas y no lo muestra adecuadamente.

7. PLIEGO DE CONDICIONES

En este apartado se van a detallar los requisitos de la aplicación, las herramientas en las que se ha desarrollado y características.

7.1. Requisitos de la aplicación

El trabajo de fin de grado que se ha desarrollado, necesita ejecutarse en un Sistema Operativo Linux, más concretamente se ha realizado en la versión Ubuntu 15.10, en la que ha sido desarrollada a lo largo de todo el proyecto. No ha sido probado para otros sistemas operativos, pero la versatilidad de los programas utilizados permiten ser instalados en Windows o IOS, al igual que en otras computadoras más potentes que en la que ha sido realizado como podría ser en un Mac, ya que el que ha sido utilizado ha sido un Packard Bell de 64-bit con procesador Intel Core i5-480M y memoria RAM de 4GB DDR3.

También mencionar los programas necesarios para las distintas tecnologías explicadas a lo largo de la memoria.

En primer lugar, para procesado y cálculos de tiempos de inicios de compases el programa Matlab, en concreto se ha implementado en la versión R2015a. Para todo el proceso realizado en Matlab del algoritmo de alineamiento, necesitan estar instalados el programa para sintetizar Timidity++ versión 2.13.2 con las fuentes de sonido FluidR3 GM, también son indispensables tener el programa MuseScore 2.0.2 para el procesamiento de partituras digitales y el programa Youtube Downloader empleado para extraer el audio en formato wav del vídeo de Youtube de la interpretación. Esta serie de programas han sido empleados para finalmente obtener el vector de tiempos y realizar el alineamiento. Una ventaja al realizarse el proyecto en el Sistema Operativo Linux es la facilidad de instalar o ejecutar a través de línea de comandos estos tipos de programas empleados.

En segundo lugar, tanto para la implementación del servicio como para el servidor donde se instala es necesario el servidor para Linux LAMPP versión 7.0.6-6, ya incluye los paquetes para poder trabajar con los lenguajes desarrollados, una vez instalado, de PHP, base de datos MySQL y Apache HTTP Server como fueron explicados cuando se detalló esta tecnología.

También es necesario la instalación del programa Python, concretamente en mi computadora se instaló la versión 2.7.10.

Por último, el navegador del usuario debe soportar HTML5 para la visualización de la aplicación, ha sido probada en el navegador Mozilla Firefox y funciona sin ningún tipo de problemas.

Por lo tanto, todos estos programas descritos han sido instalados y utilizados en mi computadora para el desarrollo del proyecto en el sistema operativo citado anteriormente.

8. ANEXOS

En este capítulo se van a presentar los anexos utilizados en este trabajo.

8.1. ANEXO I. Manual de usuario

En este manual se va a explicar al usuario como utilizar el programa y la aplicación web desarrollado. Los pasos a seguir para su funcionamiento una vez se tienen los programas citados en el apartado anterior.

Una vez se tiene instalado el servidor LAMPP y se tienen los archivos del servicio con todos sus ficheros contenidos, que es la carpeta llamada *mario* y la aplicación web *prueba* que es lo que ejecutará el usuario en el navegador, simplemente hay que instalarlo en el servidor LAMPP, más concretamente en la carpeta *htdocs* que se encuentra en el directorio donde ha sido instalado el servidor como se puede ver en la Figura 49, por lo cual se copian y pegan en ella. Mencionar que esta carpeta, al trabajar en Linux, debe tener los permisos de lectura y escritura para poder realizar en este directorio lo necesario y sin restricciones.

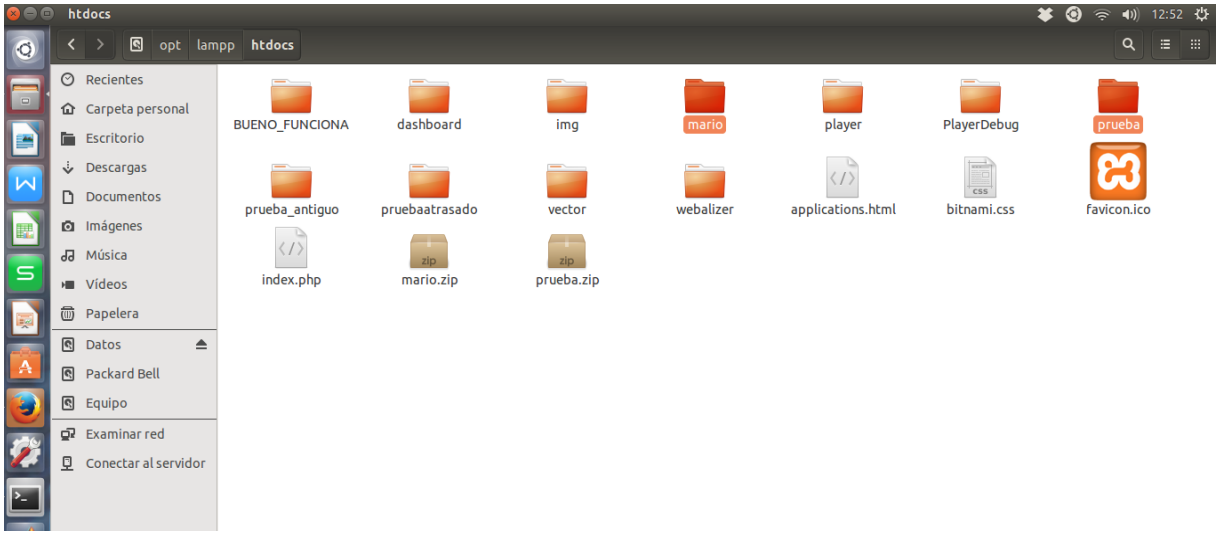


Figura 49. Carpetas servicio y aplicación web

Cuando se realiza lo anterior, hay que arrancar el servidor como se va a mostrar en la Figura 50 y ya se podrá acceder al LAMPP desde el navegador simplemente poniendo en el navegador *localhost*.

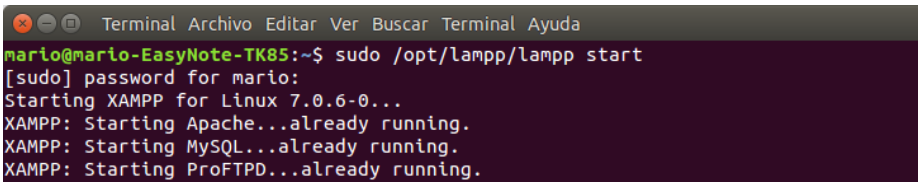


Figura 50. Arrancar servidor LAMPP

Ya puesto en marcha el servidor y accediendo a él, se importa la base de datos que es el fichero *compases.sql* como se muestra en la Figura 51¸ se cargará y ya dispondremos de la base de datos creada para el funcionamiento de la aplicación.

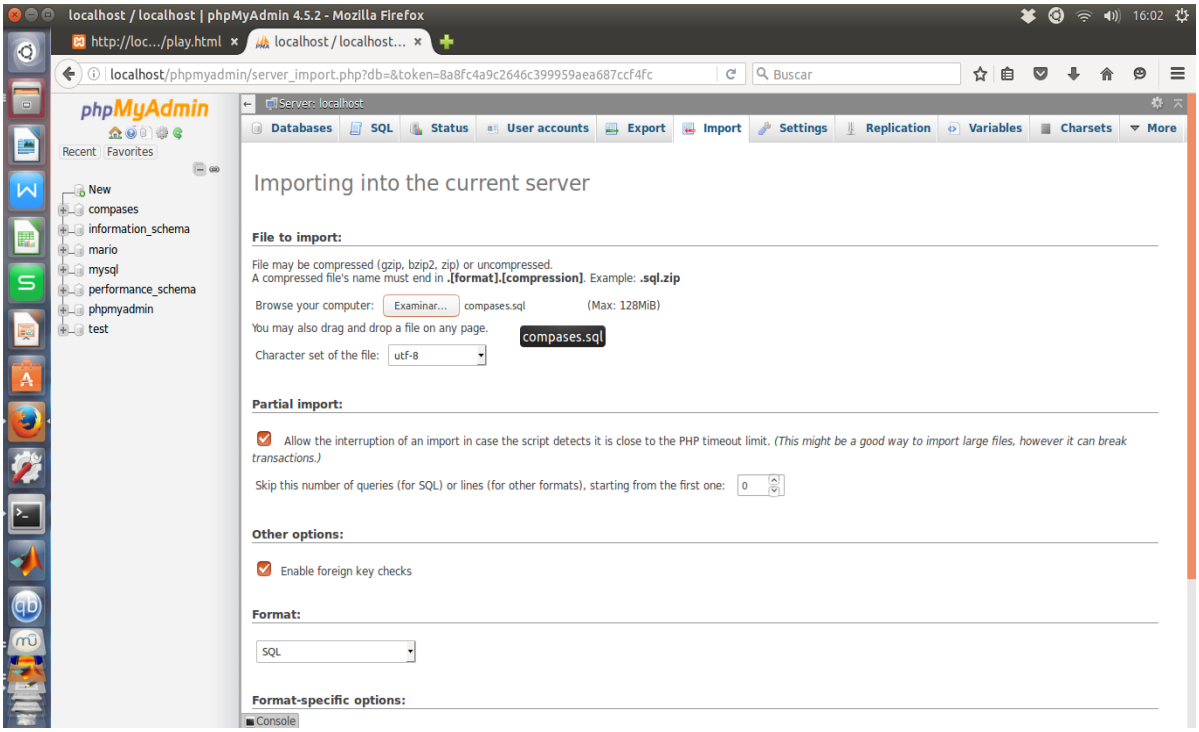


Figura 51. Importación de base de datos compases.sql

A continuación antes de poner en marcha el programa Matlab, hay que ejecutar dos scripts que se encuentran dentro de la carpeta del código Matlab, para que se pueda ejecutar el programa en el ordenador del usuario; más concretamente nos ubicamos en la carpeta DTW_Toolbox y en la ventana de comandos con el comando *mex*, que es una función que proporciona Matlab para compilar funciones en lenguaje C, se compila los ficheros dpcore.c y dpcore_var.c que crearán dos scripts respectivamente para la computadora en particular. Haría falta poner por tanto *mex dpcore.c* y una vez se complete éste, compilar el siguiente, *mex dpcore_var.c.*

Hecho esto, ya se puede iniciar el programa ejecutando en la ventana de comandos PrincipalMain(), y así se tendrá ya dicho programa funcionando en segundo plano.

Finalmente, explicado todo lo anterior ya se puede visualizar en el navegador del usuario la aplicación web, poniendo lo siguiente*: localhost/prueba/play.html* y se cargaría lo que se muestra en la Figura 52:

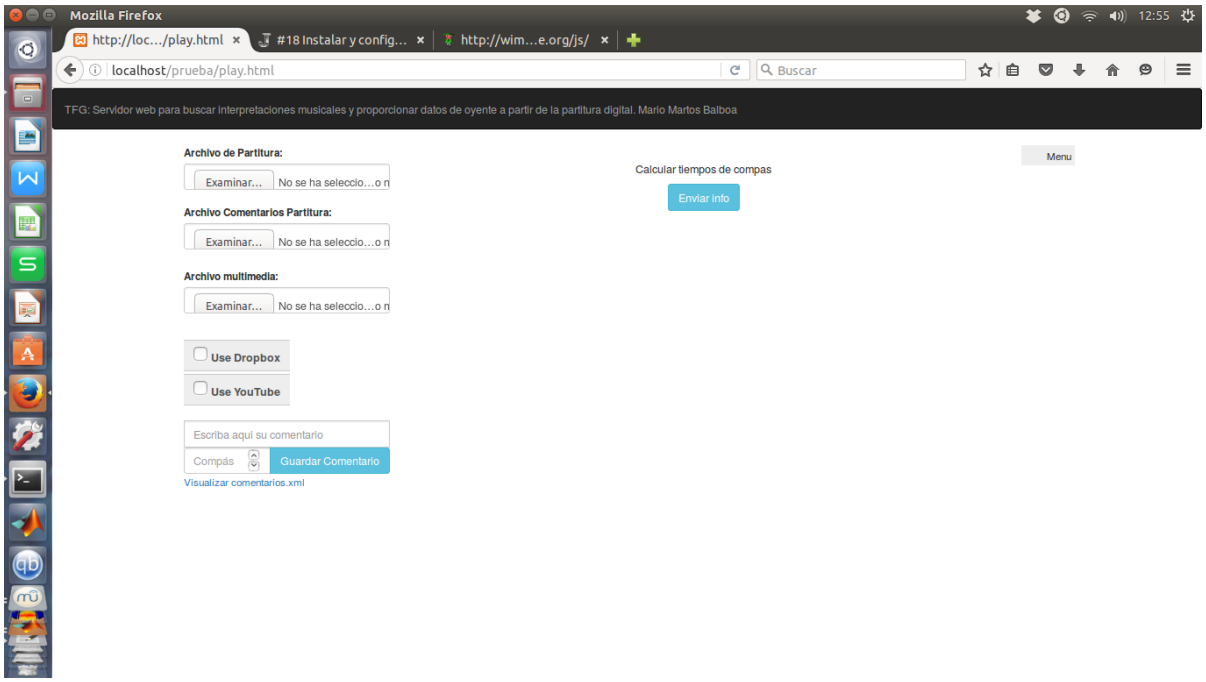


Figura 52. Carga en el navegador del usuario la aplicación web.

Ahora bien, cuando en el navegador del usuario se muestre lo anterior, ya se puede seleccionar la partitura digital que interese que se tenga en el disco duro local y así cargarla, como se puede ver un ejemplo en la Figura 53 se ha cargado la partitura digital en formato XML de *Kyrie_eleison_Mozart_Requiem_Piano.xml* una obra del compositor y pianista austriaco Wolfgang Amadeus Mozart.

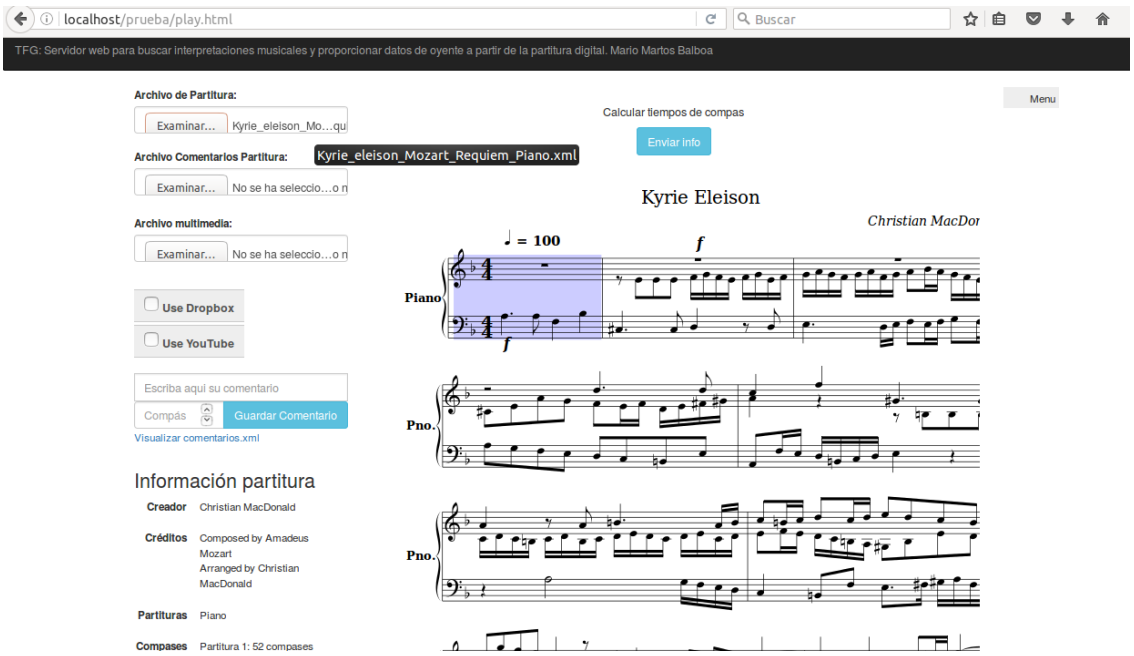


Figura 53. Cargar XML desde disco duro local

Se puede ver como se cargaría y mostraría la partitura, una vez realizado esto, se puede pasar a cargar el vídeo de la interpretación (el orden para cargar cada archivo no importa), para ello buscamos la interpretación de la obra en Youtube, y una vez encontrada seleccionar su identificador que aparece en la parte superior como se puede ver en la Figura 54.

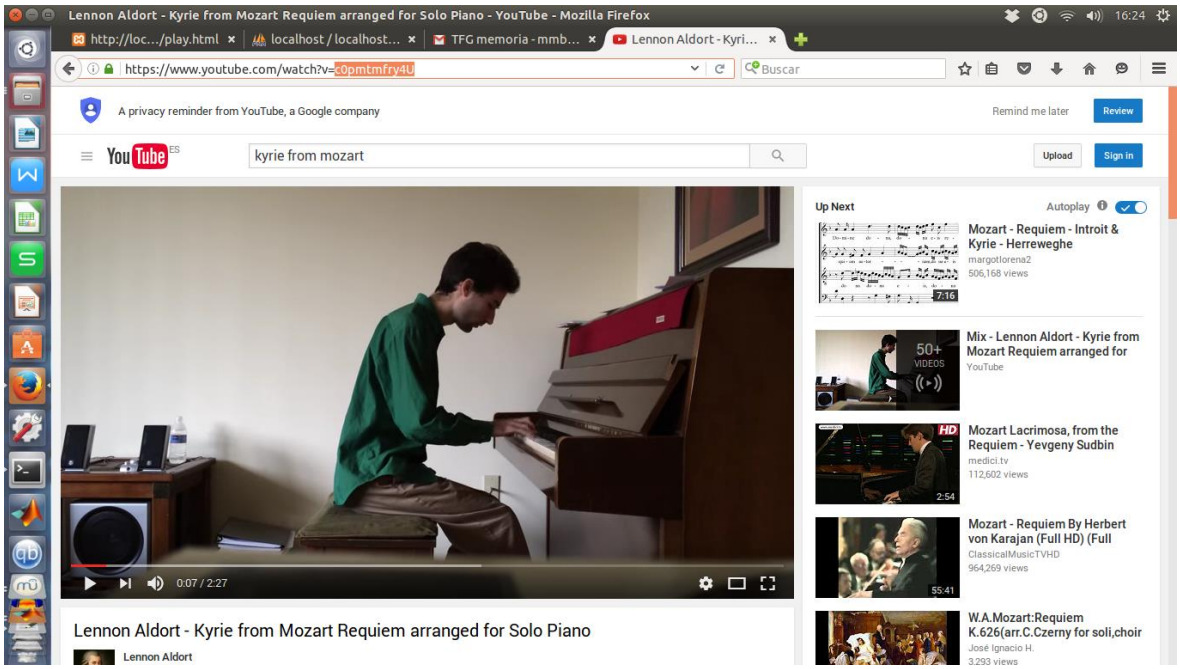


Figura 54. Identificador de video de la interpretación

A continuación es solamente copiarlo y pegarlo, seleccionando en la aplicación web *Use Youtube,* se ubica ahí el identificador del vídeo y se pulsa *Load*; de esta manera el usuario ya dispondrá de la visualización de la partitura y del vídeo que haya querido seleccionar, como se puede ver en la Figura 55.

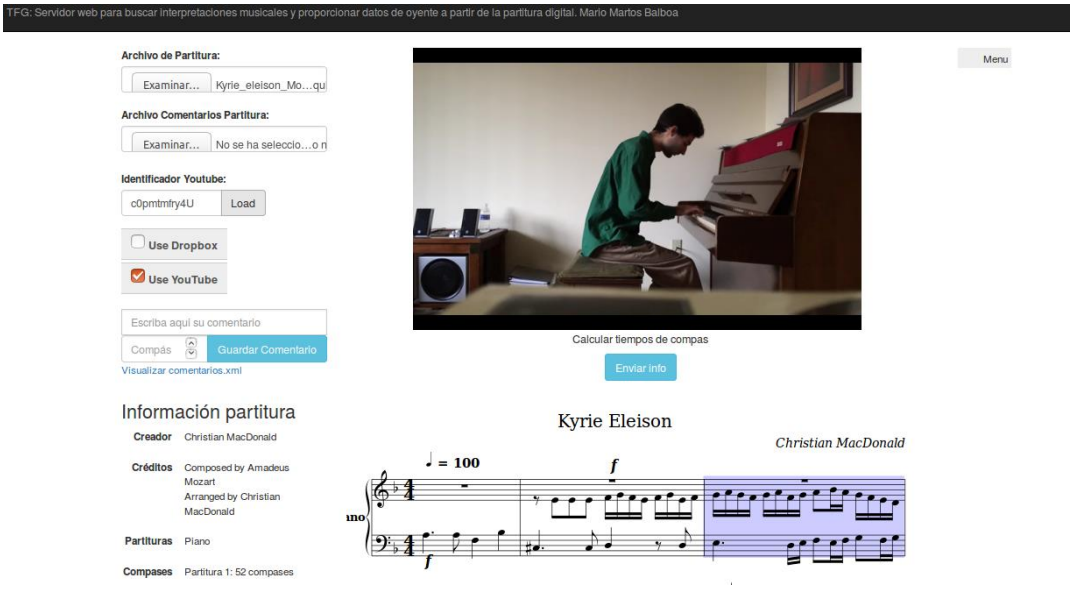


Figura 55. Cargar video Youtube de la interpretación

Por otro lado, para escribir comentarios para cada compás, el usuario puede escribirlos seleccionando el compás que quiera comentar junto con el comentario asignado al mismo en el campo *Escriba aquí su comentario* y una vez haya acabado de escribir pulsar *Guardar Comentario* y se visualizará ese comentario en la parte inferior izquierda como se puede observar en la Figura 56 donde se ha realizado un ejemplo para el primer compás de la obra cargada.

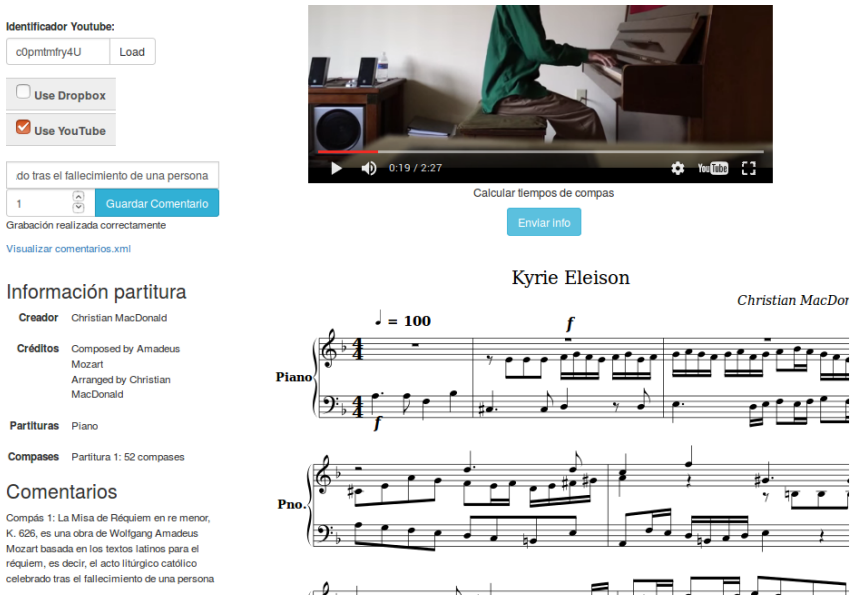


Figura 56. Editor de comentarios por compás

Finalmente, cuando el usuario quiera realizar la petición para el cálculo de los inicios de cada compás en tiempo real se tendrá que pulsar el botón *Enviar info* para calcular estos tiempos y, si ya se tienen calculados se mostrarán directamente si no, habrá que esperar a que se procesen y volver a realizar la consulta cuando se requiera.

En la Figura 57 siguiente se muestra un ejemplo con este vector ya calculado y procesado, una vez se haya pulsado el botón.

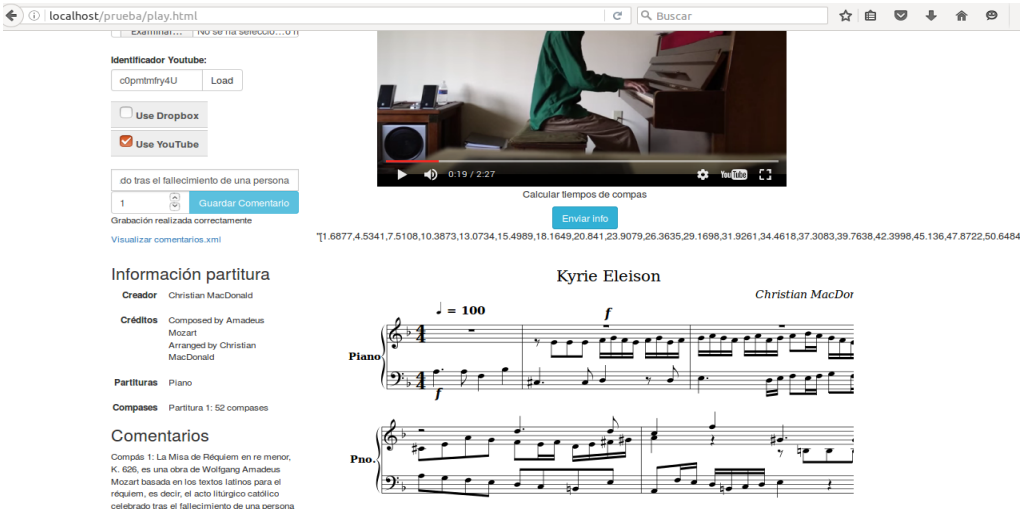


Figura 57. Petición de vector de inicios de compases

Y por tanto, una vez obtenidos, si el usuario quiere guardar el fichero con estos tiempos ya calculados puede hacerlo y así no tener que volver a cargarlo todo lo anterior ni volver a realizar la petición una vez obtenidos, simplemente en el menú de la parte superior derecha pulsando la primera opción *enable syn* y pulsando en *save*; de esta manera como se muestra en la Figura 61 se guardará el fichero y así cuando quiera cargarlo solo tendrá que seleccionar este fichero JS en lugar de la partitura digital, y directamente se mostrará y quedará sincronizado con estos valores de inicios de compases calculados.

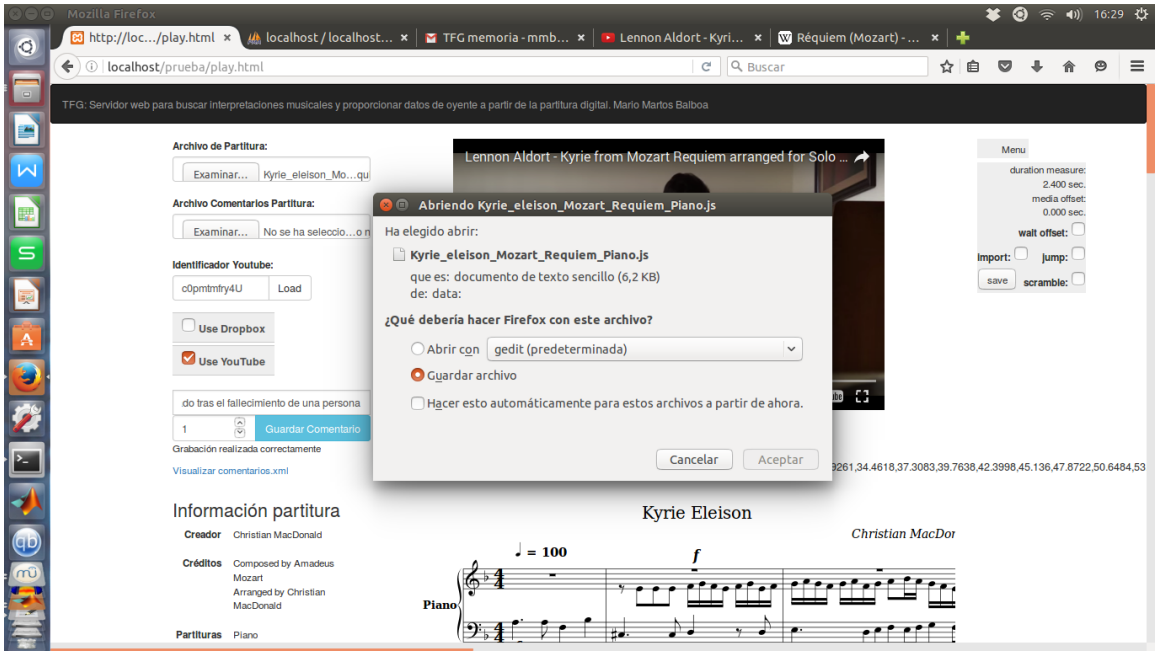


Figura 58. Guardar fichero con tiempos de inicios de compases calculados.

Como se ve y explicó en apartados anteriores, el programa Matlab actúa de forma transparente y en segundo plano, sólo habrá que ejecutarlo, ya que este programa estará continuamente funcionando.

8.2. ANEXO II. Manual técnico.

En este apartado se van a indicar las diferentes ficheros y código implementado, se van a dividir en tres partes, puesto que se tienen tres procesos tal y como se explicó en el apartado 4: El servicio desarrollado, el programa Matlab desarrollado y aplicación web desarrollada con sus respectivas funciones. Todas las funciones, scripts y código se encuentran en la carpeta que se adjunta Anexo II, donde hay tres carpetas una para cada proceso indicado.

a) Servicio desarrollado: Para su implementación se han desarrollado los siguientes ficheros (carpeta *mario*).

- **compases.sql**

En este fichero se encuentra la base de datos creada llamada “compases”, y la estructura de las dos tablas que la componen “inicios”, donde se almacenan el vector de tiempos con los valores calculados y “peticiones” como se puede ver a continuación.

```
CREATE TABLE IF NOT EXISTS `inicios` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `hash` varchar(50) NOT NULL,
  `youtubeid` varchar(30) NOT NULL,
  `value` blob NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=20 ;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `peticiones`
--

CREATE TABLE IF NOT EXISTS `peticiones` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `hash` varchar(50) NOT NULL,
  `youtubeid` varchar(30) NOT NULL,
  `status` tinyint(1) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5
```

- **Api.php**

En este script se crea para administrar las peticiones devolviendo lo que el usuario ha solicitado en el caso de que se consulte y esté procesado o

encolándolo para su procesamiento. Recibe el nombre el servicio desarrollado *"inicios",* esta función realiza un hash en formato MD5 del contenido interno de la partitura XML para la que se ha realizado la petición, que no es más que un código asociado a un archivo permitiendo así identificarlo, una vez se obtiene se renombra la partitura digital con este código. También mediante este fichero se llama a Db.php (se incluye la cabecera: include "Db.php"), que se explicará a continuación, para conectar a la base de datos para realizar las consultas y comprobar si se tiene el vector de tiempos procesado, si no, se pondría en la cola de peticiones de la tabla *peticiones*. Comprueba si está procesado y si no fuese el caso, se encolaría la petición en la base de datos (sistema de colas, pila FIFO) y muestra el estado que se encuentra la petición.

- **Db.php**

Esta función conecta con la base de datos creada en el servidor llamada "compases", para procesar las peticiones y obtener las que hayan sido procesadas seleccionándolas de la tabla inicios

b) Programa Matlab Desarrollado: Las funciones implementadas para el programa Matlab han sido las siguientes (carpeta *SF_OFFLINE*)

- **Compases.m**

Este script se ha implementado para realizar la conexión con la base de datos "compases", para que el programa ejecute consultas para una petición de identificador de Youtube y una partitura digital, que están almacenadas en la base de datos.

Lo que esta función lleva a cabo es el almacenamiento en la tabla "inicios" una vez se han calculado los inicios de compases junto con el par identificador de Youtube y partitura digital para la que se ha realizado el procesamiento.

También con este fichero ejecuta las consultas en la tabla peticiones, para ir extrayendo el primer elemento pendiente para posteriormente procesarlo y calcular el vector con los inicios de compases.

- **Function [ ]=PrincipalMain()**

Esta función es la que se ejecuta para que el programa empiece a funcionar, una vez se ejecute el programa estará continuamente funcionando ya que es un bucle infinito. En primer lugar se conecta la base de datos para

consultar la primera petición que haya que procesar, si existe petición la extrae y se llama a la función Preproceso_Principal () para el procesamiento del audio y partitura digital.

En el caso de no existir peticiones, está implementado para que se pause unos 15 segundos y vuelva a consultar pasado este tiempo a ver si ha entrado alguna petición por parte del usuario

- **Function inicio_compas=Preproceso_Principal (ytid,partitura_xml)**

-Parámetros de entrada:

ytid: identificador de youtube para extraer el audio.

partitura_xml: partitura digital XML.

-Parámetro de salida:

Inicio_compas: devuelve los inicios de cada compás (en segundos) para cada par cuando se haya completado el cálculo.

Esta función realiza todo el procesamiento tanto del audio como de la partitura, para finalmente poder obtener los tiempos reales de inicios de compases, también en esta función se realiza la conversión de XML a MIDI de la partitura. Dentro de esta función para conseguir dicho fin, se tienen dos funciones que se han desarrollado inicioCompases() y audio2ScoreTreal_offline().

i) Function res=inicioCompases(xmlFile)

-Parámetros de entrada:

xmlFile: partitura XML para identificar los inicios de compases en tiempo MIDI que luego se añadirán a la matriz nmat en la función audio2ScoreTreal_offline() para conseguir los inicios alineados con el wav.

- Parámetro de salida:

res: devuelve vector de inicios de compases identificados de la partitura (en segundos) en tiempo MIDI.

Esta función construye el comando para crear el archivo de identificación de inicios de compases de la partitura XML que se genera con la extensión .mpos. Una vez realizado esto, localiza y extrae los elementos de ese fichero que

interesan, que son el tiempo de compases identificados de la partitura (en segundos) y su compás correspondiente.

ii)Function inicio_compas=audio2ScoreTreal_offline(wav,input,output res)

-Parámetros de entrada:
wav: archivo de audio wav extraido del vídeo de Youtube.
input: archivo en formato MIDI de la partitura digital.
output:nombre con el que se generará el archivo .txt resultado del alineamiento con formato MIREX

-Parámetros de salida:
Inicio_compas: devuelve el vector con los valores de inicio de cada compás (en segundos) una vez se han alineado y sincronizado con el audio la partitura.

Esta función realiza el proceso del algoritmo de alineamiento, es decir, en ella se sincronizan video y partitura para identificar y obtener en qué momento se produce el inicio de cada compás.

Una vez concluido este proceso del programa Matlab para cada petición y obtenido este vector de tiempos, se almacenará en la base de datos y así no volver a procesarse para un par de petición idéntica.

c) Aplicación web desarrollada: los scripts desarrollados para la interfaz de usuario y su desarrollo se detallan a continuación (carpeta *prueba*).

Play.html

Este archivo HTML que junto con JavaScript invoca a los servicios de abcweb y a los servicios desarrollados, se llama desde el navegador del usuario. Es la interfaz de usuario que se visualiza en el navegador. A la estructura inicial de abcweb, se le añade un editor de comentarios, menú con información básica, cambio de estilo y diseño y servicio desarrollado para que aparte de mostrar vídeo y partitura se puedan visualizar más elementos e información. Contiene las siguientes cabeceras:

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no" />
```

```
<meta name="apple-mobile-web-app-capable" content="yes" />
<script src="js/jquery-1.11.1.min.js"></script>
<script src="js/abc2svg-1.js"></script>
<script src="js/abcweb.js"></script>
<script src="js/xml2abc-min.js"></script>
<script src="js/extras.js"></script>
<link href="css/bootstrap.min.css" rel="stylesheet">
<link href="css/extras.css" rel="stylesheet">
```

Lo que se ha implementado para conseguir la visualización de la aplicación web final son los siguientes:

i)extras.js

Mediante este JavaScript se extraen del fichero XML la información básica de la partitura, identificando las etiquetas o elementos que nos interesan para que se escriba la información en el bloque correspondiente.

También incluye la función para mostrar y cargar los comentarios, que tienen una estructura XML, y el número de comentarios que ha realizado el usuario, que se implementa con el script *escribir.php* que con la que se realiza la estructura de los comentarios.

ii) escribir.php

Mediante este fichero, se crean los comentarios siguiendo una estructura XML, por etiquetas, para poder procesarlos con el fichero *extra.js* y que se muestren en el navegador.

iii) bootstrap.css

Esta herramienta se instala en el servidor para seleccionar o cambiar el diseño de la web, para que el usuario la visualice de una forma más elegante a la que ofrecía directamente el software abcweb.

iiii) Botón llamada al servicio.

Este botón tiene el objetivo de invocar al servicio para que el usuario realice la petición mediante una llamada a ajax. Se modificó la opción de guardar en el menú de abcweb para que cuando el usuario solicite una petición y el vector de tiempos esté procesado se guarden con estos nuevos y no por los de por defecto que añade abcweb.

8.3. ANEXO III. Índice de Figuras y Tablas.

Índice de Figuras

Índice de Tablas

9. REFERENCIAS BIBLIOGRÁFICAS

[1] IMSLP. Petrucci Music Library. Librería de partituras digitales. [en línea] [última consulta: Mayo 2016]. Disponible en: http://imslp.org/

[2] Musescore Sheet Music. Librería de partituras digitales. [en línea] [última consulta Mayo 2016]. Disponible en: https://musescore.com/sheetmusic

[3] Open Music Score. Librería de partituras digitales. [en línea] [última consulta: Abril 2016]. Disponible en: http://openmusicscore.org/index.php?option=com_content&view=featured&Itemid=435

[4] musicXML. Example Set. Librería de partituras digitales. [en línea] [última consulta: Mayo 2016]. Disponible en: http://www.musicxml.com/music-in-musicxml/example-set/

[5] Descarga programa Musescore para los distintos sistemas operativos. [en línea] [última consulta: Marzo 2016]. Disponible en: https://musescore.org/es/descarga

[6] Página oficial Musescore. [en línea] [última consulta: Julio 2016]. Disponible en: https://musescore.org/es

[7] Desarrollador del software ABCWEB. [en línea] [última consulta: Junio 2016]. Disponible en: http://wim.vree.org/

[8] Página oficial para descargar el software de la herramienta abcweb. [en línea] [última consulta: Julio 2016]. Disponible en: http://wim.vree.org/svgParse/index.html

[9] Información de notación musical abcweb. [en línea] [última consulta: Mayo 2016]. Disponible en: http://abcnotation.com/

[10] MySQL Documentation. Página oficial de manuales y documentación del lenguaje MySQL. [en línea] [última consulta: Mayo 2016]. Disponible en: http://dev.mysql.com/doc/

[11] Manual de lenguaje PHP. [en línea] [última consulta: Junio 2016]. Disponible en: http://php.net/manual/es/index.php

[12] Documentación del Servidor HTTP Apache 2.0" [en línea] [última consulta: Abril 2016]. Disponible en: http://httpd.apache.org/docs/2.0/es/

[13] Información acerca de qué tipo de servidor utilizar. [en línea] [última consulta: Abril 2016]. Disponible en: https://achetemele.wordpress.com/2013/04/10/que-tipo-de-servidor-utilizar-wamp-mamp- xampp-lamp/

[14] Instalación, información y configuración del servidor Lamp empleado. [en línea] [última consulta: Abril 2016]. Disponible en: https://www.apachefriends.org/es/faq_linux.html

[15] Artículo de algoritmo de alineamiento audio-partitura. RODRIGUEZ-SERRANO, F.J., CARABIAS-ORTI, J.J, VERA-CANDEAS, P. & MARTINEZ-MUÑOZ, D. *An audio to score alignment framework using spectral factorization and dynamic time warping*. ACM Trans. Embedd, Comput. Syst. 9,4, Article 39 (March 2010), pages 742-748.

[16] Eerola, T. & Toiviainen, P. (2004). MIDI Toolbox: MATLAB Tools for Music Research. University of Jyväkylä, Finland. [en línea] [última consulta: Marzo 2016]. Disponible en: https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox/

[17] Artículo de algoritmo NMF. Lee, D., Seung, S. *Algorithms for Non-Negative Matriz Factorization*. [en línea] [última consulta: Marzo 2016]. Disponible en: http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[18] D. Ellis Dynamic Time Warp (DTW) in Matlab, Web resource. [en línea] [última consulta: Abril 2016]. Disponible en: http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/

[19] Robert J. Turetsky and Daniel P.W Ellis (2003). Ground-Truth Transcriptions of Real Music from ForceAligned MIDI. [en línea] [última consulta: Abril 2016]. Disponible en: http://www.ee.columbia.edu/~dpwe/pubs/ismir03-midi.pdf

[20] Rodriguez Serrano F.J, Vera-Candeas P. Carabias Orti Julio. 2016: Real-time Audio to Score Alignment (a.k.a Score Following Results). [en línea] [última consulta: Agosto 2016]. Disponible en: http://www.music-ir.org/mirex/wiki/2016:Real-time_Audio_to_Score_Alignment_(a.k.a._Score_Following)_Results#Summary_Results

[21] Diseño y estilos para páginas webs. Bootstrap. [en línea] [última consulta: Julio 2016]. Disponible en: http://getbootstrap.com/css/

[22] Librería virtual con diferentes tipos de partituras para testear y probar. *Unofficial MusicXML test suite.* [en línea] [última consulta: Agosto 2016]. Disponible en: http://lilypond.org/doc/v2.18/input/regression/musicxml/collated-files.html