UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

Corso di Laurea triennale in
Informatica Musicale

PROTOTIPO DI APPLICATIVO WEB PER LA CONVERSIONE DA **KERN A IEEE 1599

Relatore: Prof. Luca Andrea Ludovico
Correlatore: Prof. Adriano Baratè

Tesi di Laurea di:
Alessandro Di Marco
Matr. Nr. 856063

ANNO ACCADEMICO 2018-2019

Ringraziamenti

Ringrazio la mia famiglia e tutte le persone a me care che mi hanno supportato in questi anni di studi e che mi hanno permesso di raggiungere questo importante traguardo.
Un ringraziamento anche ai membri del Laboratorio di Informatica Musicale che hanno reso possibile intraprendere questo percorso di studi.

Indice

Capitolo 1

Introduzione

1.1 Obbiettivo

Lo scopo di questo elaborato è presentare il lavoro svolto per la progettazione e la realizzazione di un prototipo di applicativo web che ha come obbiettivo la conversione di file rappresentati informazioni di opere musicali codificate in Humdum a file XML validi rispetto allo standard IEEE 1599.
In particolare ci si è concentrati sull'aspetto simbolico della musica inteso come codifica dello spartito. Partendo dal formato **kern si è arrivati ad ottenere gli elementi che appartenenti ai layer General e Logic che compongono IEEE 1599, avendo come fine l'accesso alle opere rappresentate in formato **kern presenti nella libreria virtuale kernScore[1] [1]

1.2 Struttura

L'elaborato si apre con una analisi dei due formati per la rappresentazione di eventi musicali alla base del funzionamento dell'applicativo, Humdrum **kern e IEEE 1599 concentrandosi sui layer di quest'ultimo destinati alla codifica dello spartito. Segue una breve descrizione di linguaggi utilizzati per la realizzazione dell'applicativo web: PHP per la creazione della componente web dell'applicativo, Javascript utilizzato per l'implementazione degli algoritmi di conversione e XML come linguaggio alla base di IEEE 1599. Il quarto capitolo è dedicato alla descrizione degli algoritmi implementati per la conversione stessa, su loro funzionamento e sulle scelte fatte in fase di progettazione. Infine il lavoro si conclude con un analisi sui risultati ottenuti e sulle possibili migliorie sia a livello di conversione che a livello di feature offribili all'utente per migliorarne l'esperienza.

[1] `http://kern.ccarh.org/`

Capitolo 2

Stato dell'arte

Parlando di musica risulta spesso complesso trovare una rappresentazione che sia in grado di esprimere in modo soddisfacente ogni aspetto che caratterizza un'opera. Questo è dovuto alla natura della musica stessa che non si può essere limitata alla sola esecuzione ma neppure può essere completamente descritta dalla rappresentazione su di uno spartito; piuttosto si tratta di un insiemi di "livelli" ognuno dei quali è in grado di fornire informazioni diverse ed ugualmente significative tra loro.
Nel tentativo di rappresentare la musica in modo quanto più esaustivo possibile in ambito informatico si è arrivati alla creazione di formati per la rappresentazione basati sul concetto di multiple-layer ben supportati da linguaggi come XML nel caso di IEEE 1599 [2] o linguaggi originali creati su misura come nel caso di Humdrum. In tutti i casi si tratta di sintassi la cui caratteristica principale è la capacità di rappresentare in modo schematico e gerarchico i diversi gradi d'informazione. Per la realizzazione di questo prototipo si è posta l'attenzione sulla codifica che Humdrum e IEEE 1599 fanno dello spartito musicale al fine di creare uno strumento di conversione dal primo al secondo formato.

2.1 Humdrum

Humdrum è un software general-purpose pensato per assistere gli utenti che lavorano con applicazioni musicali [3]. Tra le possibilità che Humdrum è in grado di offrire vi è possibilità per l'utente di definire proprie grammatiche di rappresentazioni creabili su misura a seconda delle esigenze richieste e degli obbietti desiderati. Tra le sintassi predefinite di Humdrum per la rappresentazione delle informazioni contenute in uno spartito, la più utilizzata è **kern. [4]

2.1.1 **kern

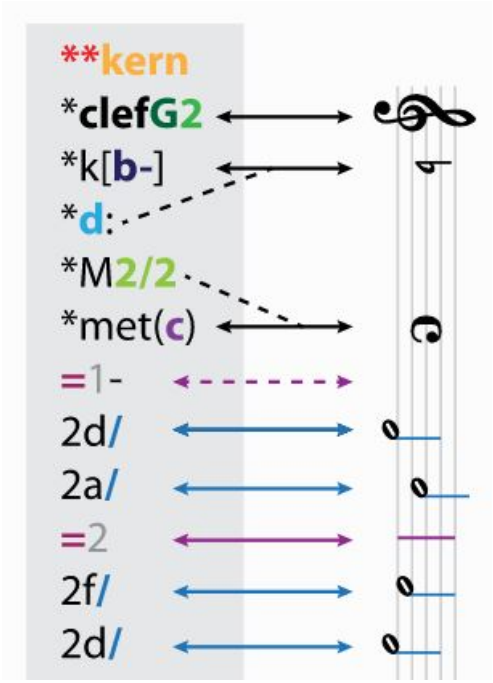
**kern è una sintassi supportata da Humdrum utilizzata per rappresentare le informazioni di uno spartito relativo ad un'opera musicale. La sintassi **kern permette di descrivere in modo completo gli eventi musicali tramite una codifica di pitch e di durata, oltre che altri elementi quali alterazioni, legature, articolazioni, e in generale tutto l'insieme delle informazioni rappresentabili graficamente in uno spartito. In aggiunta è possibile completare il file con informazioni di archiviazione che arricchiscono il file e ne facilitato l'identificazione e l'utilizzo all'interno di una libreria digitale.

2.1.2 Struttura di file formattato **kern

```
1  **kern **kern **kern **kern
2  *staff2 *staff2 *staff1 *staff1
3  *clefF4 *clefF4 *clefG2 *clefG2
4  *k[f#c#g#] *k[f#c#g#] *k[f#c#g#] *k[f#c#g#]
5  *M4/4 *M4/4 *M4/4 *M4/4
6  4AA 4c# 4a 4ee
7  =1 =1 =1 =1
8  8A 4c# 4a 4ee
9  8B . . .
10 8c# 4c# 4a 4ee
11 8A . . .
12 8D 4d 4a 4ff#
13 8E . . .
14 8F# 4d 4a 4ff#
15 8D . . .
16 *- *- *- *-
```

L'idea su cui si basa **kern è fornire una rappresentazione mirata a facilitare gli strumenti di analisi piuttosto che una visualizzazione pensata per l'utente. Per questo motivo le note sono scritte lungo colonne che se lette verticalmente rappresentano la successione nel tempo degli eventi. Devono essere sempre aperte dal termine "**kern" e chiuse dal simbolo "*-". Ogni colonna prende il nome di spine gli spine sono separati tra loro da un spazio di tabulazione. Leggendo una riga orizzontalmente invece si ottiene la rappresentazione di tutti gli eventi che avvengono contemporaneamente in un determinato punto dello spartito.
Per facilitare la comprensione del concetto alla base di **kern è possibile immaginare il file così codificato come la rappresentazione tramite codici alfanumerici di

uno spartito ruotato di 90 gradi in senso orario.



Componente fondamentale che permette la corretta rappresentazione di opere composte da più parti, ovvero con più spine, è il simbolo null ".."; questo diventa necessario perchè non tutte le parti presenteranno una nuova nota successiva alla precedente nello stesso momento e l'utilizzo del simbolo "." permette di preservare la struttura mantenendo sempre allineati gli eventi lungo tutto il file. Dopo l'apertura di uno spine ci possono essere una serie di informazioni solitamente poste all'inizio di uno spartito come la chiave, il tempo, oltre che il nome della parte e commenti aggiuntivi; questi informazioni prendono il nome di tandem interpretations e sono riconoscibili da fatto che iniziano sempre con un singolo asterisco "*" che il dato.
Il resto del file è composto dai token che codificati in successione, battuta per battuta, i singoli eventi musicali secondo la sintassi definita da **kern

2.1.3 Note, Pause, Misure

L'altezza, o pitch, di una nota viene codificata come il corrispettivo valore in notazione anglosassone in forma di lettere minuscole e maiuscole.
L'altezza è sempre rappresentata come il valore d'esecuzione. Per la rappresentazione di strumenti trasposti è necessario aprire lo spine con la tandem interpetation

"*ITr" seguita dalla descrizione dell'intervallo diatonico e cromatico necessario per ottenere il valore trasposto.
L'altezza delle note è rappresentata tramite la ripetizione di lettere maiuscole e minuscole secondo la seguente struttura:[5]

- le note della quarta ottava sono scritte con una singola lettera minuscola
- le note della terza ottava sono scritte con una singola lettera maiuscola
- le note delle ottave superiori alla quarta sono scritte con un numero crescente di lettere minuscole
- le note delle ottave inferiori alla terza sono scritte con un numero crescente di lettere maiuscole

Si avrà quindi che per esempio per rappresentare il LA centrale (A4, 440 Hz) si utilizza la singola lettera "a" minuscola; il LA della quinta ottava verrà scritto come "aa" mentre un DO della terza ottava verrà rappresentato dalla stringa "A". Per indicare la presenza di una pausa **kern utilizza la lettera "r" mentre i simboli "#", "-", e "n" sono aggiunti alla affianco alla codifica per indicare la presenza di un simbolo di alterazione come diesis, bemolle o bequadro.
Al valore di altezza viene anteposto un valore numerico corrispondente alla durata della nota o della pausa. Per esempio una semiminima sarà scritta come "4", una croma sarà scritta come "8", una semibreve sarà uguale ad "1" e così via. Tutte le note e le pause vengono elencate all'interno della propria battuta il cui inizio è rappresentato rappresentato con un riga di simboli di uguale "=", uno per ogni spine.

2.1.4 Tandem interpretations e Reference Records

Oltre agli effettivi elementi di notazione che compongono un opera musicale, **kern permette di riportare un vasto insieme di informazioni aggiuntive secondo in due forme

- Reference Records
- Tandem Interpretations

I Reference Records hanno l'obbiettivo di racchiudere tutte le informazioni bibliografiche relative all'opera come informazioni sull'autore, sull'opera stessa, sull'edizione ecc..
I Reference Records sono posti all'inizio del file e sono identificati da tre punti esclamativi "!!!" seguiti da un codice univoco di tre lettere maiuscole.Un elenco

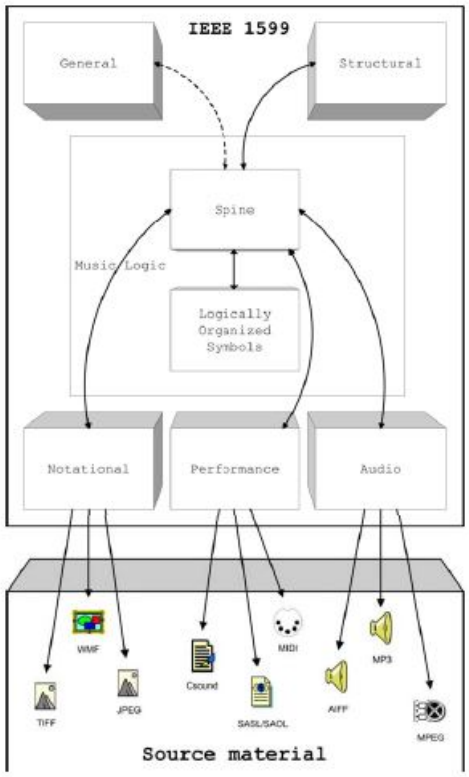
esaustivo di tutti i Reference Records definibili è disponibile tramite il sito di Humdrum. [1] Nel caso in cui si vogliano inserire altre informazioni non riconducibili a Reference Records definiti, **kern permette l'aggiunta di commenti sotto forma di linee testuali aperte da una coppia di punti esclamativi "!!".
Le Tandem Interpretations sono informazioni aggiuntive sullo spine al quale si riferiscono come per esempio lo strumento rappresentato, il tipo di chiave utilizzata, il tempo, le alterazioni in chiave e se si tratta di uno strumento trasposto e sono sempre identificabili da un singolo asterisco "*" che le precede. Di seguito l'elenco delle principali tandem interpretations dichiarabili.

clef	`*clefG2`	meter signatures	`*M6/8`
instrument	`*I`	tempo	`*MM96.3`
instrument class	`*IC`	timebase	`*tb32`
key signatures	`*k[f#c#]`	transposing instrument	`*ITr`
key	`*c#:`		

2.2 IEEE 1599

IEEE 1599 è un formato basato su XML progettato con l'obbiettivo di fornire una descrizione esaustiva dell'informazione musicale [6] sviluppato dal Laboratorio di Informatica Musicale (LIM) dell'Università di Milano e facente parte dello standard IEEE da Settembre 2008.
Sebbene, a causa della natura di **kern come descrittore dello spartito, ci si sia concentrati all'interno di questo applicativo sugli aspetti informativi del livello di rappresentazione simbolica, IEEE 1599 è stato progettato e realizzato con il preciso scopo di riunire tutti i livelli di rappresentazione dell'opera musicale: simbolici, strutturali, di notazione, d'esecuzione e rappresentazione digitale del suono. Tutto ciò è stato possibile grazie all'implementazione di una struttura a layer multipli.

[1] `http://www.humdrum.org/Humdrum/guide.append1.html`



Di particolare interesse all'interno di questo lavoro sono i primi due layer che ben si prestano a raccogliere i dati estraibili da un file **kern:

- il primo, il layer General
- il secondo, il layer Logic

2.2.1 Layer Logic

Il layer Logic svolge un ruolo importante all'interno di IEEE 1599 e ha come fine la rappresentazione simbolica ed si compone di due elementi fondamentali: lo Spine e i Logically Organized Symbols (LOS).

Spine

Lo spine è una lista di eventi musicali; la definizione di "evento" è lasciata all'utente e non deve necessariamente corrispondere ai singoli simboli che compaiono sullo spartito.
Essendo una lista unidimensionale, tutti gli eventi sono posti in ordine di successione ed è quindi richiesto un approccio che renda possibile distinguere gli eventi

che avvengono contemporaneamente nello stesso istante di tempo dagli eventi successivi a quello in analisi.
Per ogni evento sono stati definiti due attributi, hpos, una dimensione spaziale misurata in Virtual Pixel (VP), e timing, l'offset temporale dall'evento precedente misurata in Virtal Time Unit (VTU) dove il valore "0" viene utilizzato per rappresentare l'allineamento verticale e la contemporaneità temporale tra eventi. Come dice il nome, questi attributi sono valori virtuali che vengono decisi liberamente dall'utente in moda tale da mantenere i rapporti temporali e spaziali tra gli eventi con l'unica limitazione di dover essere valori interi
Lo spine ricopre il ruolo più importante all'interno del file IEEE 1599 agendo da punto di unione tra tutti i layer che di volta in volta si riferiscono agli elementi in esso elencati ed identificabili tramite ID univoci. Per esempio la descrizione della nota che verrà fatta nel LOS avrà sempre come riferimento un elemento di Spine.

LOS

Il LOS è un elemento del layer Logic che funge da contenitore di tutti i sotto elementi che descrivono gli eventi simbolici che compaiono nello spartito.[6]
L'elemento LOS raccoglie tutti i sotto elementi necessari alla descrizione simbolica degli eventi. Per garantire la validità del file IEEE 1599 l'elemento LOS deve sempre contenere almeno un elemento "staff_list" e un elemento "part".
L'elemento "part" è la rappresentazione di una parte di uno spartito e ogni parte è internamente suddivisa nelle differenti battute rappresentate dagli elementi "measure". Le battute sono organizzate voce per voce e ogni voce racchiude al proprio interno gli accordi "chord" e le pause "rest". Le note contenute che compongono gli accordi sono descritte da attributi di durata, altezza ed eventuali alterazioni.

```
<part id="piano">
 <measure number="4">
  <voice voice_item_ref="piano_1">
   <chord event_ref="piano_1_1">
    <duration num="1" den="4"/>
    <augmentation_dots number="1"/>
    <notehead>
     <pitch step="C" octave="4"/>
    </notehead>
   </chord>
  </voice>
 </measure>
</part>
```

Listing 2.1: esempio di una battuta in IEEE 1599

Capitolo 3

Tecnologie utilizzate

Avendo fissato come obbiettivo del progetto l'implementazione dell'applicativo in formato web, la scelta delle tecnologie per la sua realizzazione è ricaduta principalmente su linguaggi web come HTML e PHP per la creazione e gestione della pagina tramite cui si accede all'applicativo, e su JavaScript come linguaggio per la realizzazione degli algoritmi di conversione. È stato inoltre utilizzato XML in quanto linguaggio alla base di IEEE 1599. La progettazione e il testing del progetto sono stata invece possibile grazie all'uso della piattaforma software XAMPP.

3.1 HTML

L'Hypertext Markup Language (HTML) è un linguaggio di markup nato nel 1993 come linguaggio di formattazione per pagine ipertestuali e oggi utilizzato per la realizzazione di pagine web in combinazione con altre tecnologie per la formattazione grafica come il Cascading Style Sheets (CSS) e linguaggi di scripting come JavaScript. Attualmente alla versione HTML5 sviluppata e gestita dal World Wide Web Consortium (W3C), HTML è un linguaggio di formattazione per la descrizione strutturale di una pagina web e si compone di una serie di blocchi innestati, detti elementi HTML, gestiti tramite l'uso di tag rappresentati tra le parentesi "¡" e "¿". Ogni blocco è identificabile da un tag di apertura e uno di chiusura ed ad ognuno possono essere associati una serie di attributi che ne permettono l'identificazione in classi, la modifica o il riferimento all'interno del documento web tramite altri linguaggi.

3.1.1 DOM

La struttura delle pagine generate tramite l'uso di HTML è riferibile tramite un Document Object Model (DOM), una forma di rappresentazione sviluppata da

W3C per descrivere la struttura di un documento in modo indipendente dal linguaggio che la ha generata.
Supportato dai browser, il DOM è il principale strumento che permette visione e la modifica dinamica di documento HTML. Secondo le specifiche di W3C, il DOM è visualizzabile come un albero su diversi livelli accessibili tramite riferimento. La radice è rappresentata dal documento stesso al quale vengono appesi di volta in volta gli altri elementi della pagina formando i nodi e le foglie dell'albero che schematizza il documento.

3.2 XML

L'eXtensible Markup Language (XML) è un linguaggio di markup per la definizione di regole per documenti formattati in moda tale che siano facilmente leggibili sia da un utente sia da una macchina.
Sviluppato da W3C e basato sulla definizione di tag personalizzati, ha come obbiettivo il supporto alla creazione di altri linguaggi per documenti strutturati ed è principalmente utilizzato per l'esportazione di informazione tra DBMS.
Un documento XML è composto da elementi definiti tra tag e ai quali possono essere associati diversi attributi.

3.2.1 DTD

XML offre anche la possibilità di essere validato tramite l'utilizzo di un Document Type Definiton (DTD), un tipo di documento nel quale vengono dichiarati gli elementi e gli attributi richiesti. Per essere considerato valido, il documento XML deve rispettare la grammatica così definita all'interno del DTD

3.3 PHP

PHP: Hypertext Preprocessor (PHP)è un linguaggio di programmazione debolmente tipizzato sviluppato nel 1994 come linguaggio di scripting interpretato per le pagine web e attualmente usato per la realizzazione di applicativi web lato server.
Il codice è solitamente interpretato del web server che restituisce come output il codice interpretato e lo esegu. L'interprete esegue solamente il codice inserito all'interno dei delimitatori "¡?php" e "?¿" inseribili all'interno di una pagina HTML.
A partire da PHP 3 è stata implementata la gestione di programmazione orientata ad oggetti.
Tra i le possibilità offerte da PHP, spicca soprattutto il supporto a interfacce DBMS e l'integrazione con altri linguaggi tra cui XML.

3.4 JavaScript

JavaScript è un linguaggio si scripting orientato agli oggetti e principalmente utilizzato in ambito web per la programmazione lato client.
JavaScript è un linguaggio interpretato, non viene cioè compilato in precedenza ma viene interpretato a runtime da componenti incluse nel browser. Il codice viene quindi eseguito direttamente sul client, permettendo un alleggerimento del carico di lavoro sul server in caso di script complessi a fronte di un tempo di download del codice sorgente più elevato. Inoltre non permette l'accesso diretto a dati memorizzati sul server se non previa richiesta da parte di un altro linguaggio. Si tratta inoltre di un linguaggio debolmente tipizzato e debolmente orientato agli oggetti. L'accesso da parte di JavaScript alla pagina web tramite il DOM permette l'implementazione di script per la modifica dinamica del DOM stesso superando i limiti di HTML come linguaggio statico.

3.5 XAMPP

XAMPP è una distribuzione di Apache che comprende tutti gli strumenti necessari per la realizzazione di un applicativo web che implementi un database MariaDB e linguaggi di programmazione PHP.
XAMPP è un software open-source rilasciato attraverso GNU General Public License disponibile per sistemi Microsoft Windows e unix-like. L'uso principale per il quale è stato creato XAMPP è come strumento di sviluppo per la realizzazione e il test di pagine web dinamiche agendo in modo locale riferendosi a localhost come ad un host remoto in comunicazione tramite il protocollo FTP.

Capitolo 4

L'Applicativo

Il lavoro per la realizzazione dell'applicativo web ha previsto la realizzazione di due componenti principali: in primo luogo è stato necessario creare le pagine web, implementate tramite linguaggio PHP, che permettessero all'utente l'upload di un file .krn e che svolgessero la conversione restituendo come risultato il file IEEE 1599 visionabile e scaricabile in formato XML.
La seconda parte si è concentrata sulla creazione delle funzioni scritte in linguaggio JavaScript che si occupano delle effettive conversioni.

4.1 Implementazione dell'Applicativo

La componente dell'applicativo lato utente è stata sviluppata interamente con l'uso di HTML e PHP ed è stata suddivisa su tre diverse pagine: 1) index 2) convert 3) output.
La prima è la pagina alla quale si accede all'applicativo e che è formata principalmente da un form con un'area di testo tramite la quale l'utente inserisce il file .krn da convertire. La pagina permette sia la scrittura del testo, sia l'upload di un file testuale conforme alla sintassi di **kern che verrà letto, caricato e visualizzato all'interno della area di testo per essere controllato e caricato.
Eseguendo la conversione il testo inviato alla seconda pagina. Qui viene svolto tutto il lavoro di conversione che legge il testo caricato, lo interpreta secondo le regole dello standard di **kern e lo trasforma in un file di testo formattato come un XML valido per IEEE 1599. È importante ricordare che tutto il lavoro viene svolto lato utente tramite funzioni JavaScript comportando tempi di computazione maggiori in caso di file di grosse dimensioni.
Una volta creato, il testo risultante dalla conversione sarà visualizzabile dall'utente che potrà decidere se procedere con il download o se tornare alla pagina iniziale per eseguire una nuova conversione.

Procedendo con il download, il testo ottenuto viene inviato alla terza pagina dell'applicativo. Qui l'applicativo crea un nuovo file .xml a partire da un file mantenuto come modello di base. Il testo convertito viene scritto sul file appena creato e il risultato restituito all'utente.
A completare l'applicativo, oltre ai file che raccolgono le funzioni JavaScript di conversione suddivisi a seconda dell'elemento di IEEE 1599 creano, vi sono un header tramite il quale sono caricati tutti gli script su tutte le pagine, e un file di stile CSS che completa la componente grafica realizzata tramite l'ausilio di Bootstrap.

4.2 Implementazione della Conversione

Di seguito verranno descritti gli algoritmi implementati nell'applicativo spiegandone il funzionamento e le scelte fatte in fase di progettazione.
Gli algoritmi possono essere suddivisi in tre gruppi riportati in questo ordine:

1. algoritmi per la creazione del layer General
2. algoritmi per la creazione e il popolamento dell'elemento Spine del layer Logic
3. algoritmi per la creazione e il popolamento dell'elemento LOS del layer Logic

4.2.1 Lettura dei dati

Il primo passo è stato definire un algoritmo che leggesse i dati passati dall'utente.

```
1  function readXml(xmlDocument) {
2    var NStaves = 0;
3    var xmlDoc = xmlDocument.responseXML;
4    var arr = document.getElementById("tmp").textContent.split("\n");
5    var i;
6    for(i=0; i<arr.length; i++){
7        .
8        .
9        .
10     }
```

Sebbene **kern organizzi le informazioni in forma di colonne, ognuna delle quali racchiude tutti gli eventi musicali relativi ad un elemento "part" contenuto nel LOS di IEEE 1599, si è preferito implementare una strategia di lettura dei dati

che seguisse il flusso naturale di lettura della pagina, ovvero riga per riga. Ciò ha permesso di facilitare lo scorrimento del documento avendo come unico svantaggio quello di dover tenere traccia della colonna, o parte, che di volta in volta si sta analizzando tramite l'uso di variabili globali condivise tra le funzioni.

```
1 var posCorrection = 0;
2 var multipleVoiceCorrection = 0;
```

Per questo motivo, dopo aver analizzato tutte le informazioni di libreria contenute nei Reference Records, vengono chiamate due funzioni, "NStaves" e "instrumenList" dove la prima conta il numero di staffe che compongono lo spartito contando il numero di colonne aperte dalla parola chiave "**kern", mentre la seconda crea un array contenente il nome di ogni strumento estratto dalla tandem interpretation "*I" seguito dal nome dello strumenta, posto nella posizione dell'array corrispondente al numero della colonna a cui si riferisce.

```
1 if(tmpString.includes("**kern") == true)
2   NStaves = stavesCounter(xmlDoc, tmpString);
3 if(tmpString.includes("*I") == true)
4   instrumentsList(xmlDoc, tmpString);
```

4.2.2 La funzione ReferenceRecords

Le prime informazioni che si incontrano durante la lettura in quanto dovrebbero essere sempre poste all'inizio del documento **kern sono quelle di libreria contenute nei cosiddetti Reference Records (si noti che l'applicativo è comunque progettato per gestire anche i casi in cui il file **kern non sia strutturato secondo le convenzioni di buona formattazione del file **kern ed è quindi in grado di riconoscere Reference Records posti in posizioni non usuali a patto che essi siano riconoscibili secondo la sintassi).
Come detto in precedenza ognuno di questi records è riconoscibile da tre punti esclamativi "!!!" seguiti da un codice identificativo di tre lettere maiuscole.
La funzione legge e se riconosce il codice tra quelli definiti [1] scrive l'informazione all'interno dell'elemento corrispondente del layer General di IEEE 1599.

```
1 if(tmpString.includes("!!!") == true)
2   referenceRecords(xmlDoc, tmpString);
```

In questa fase si è notato che le informazioni di libreria codificate da IEEE 1599 si limitano ad informazioni sull'autore, sul compositore o sull'opera stessa mentre **kern offre una gamma di informazioni più ampia in particolare riguardo ad

[1] http://www.humdrum.org/Humdrum/guide.append1.html

informazioni sull'edizione e la stampa dello spartito codificato. Attualmente tutte queste informazioni vengono aggiunte come elemento "notes" di IEEE 1599 non trovando altra collocazione.

4.2.3 Da **kern allo Spine

La componente fondamentale di un file IEEE 1599 è lo Spine e le unità che lo compongono, gli elementi "event", ricoprono la funzione di punto d'unione usati da tutti i layer come riferimento univoco semptre identificabile.
La creazione dello Spine ricopre quindi un passaggio fondamentale in fase di conversione che deve garantire la creazione e l'inserimento corretti mantenendo l'ordine degli eventi.
Ogni volta che viene letta una riga che non rientra nelle casistiche descritte in precedenza, viene chiamata la funzione "createSpine".

```
1  functions.js
2      if(tmpString.includes("*clef") == true || tmpString.includes("*k
           [") == true || tmpString.includes("*M") == true || tmpString
           .includes("*met") == true)
3        createSpine(xmlDoc, tmpString, NStaves);
4
5  logicLayerFunctions.js
6  function createSpine(xmlDoc, tmpString, NStaves){
7    createLos(xmlDoc);
8    var line = tmpString.split("\t");
9    for(j=0; j<line.length; j++) {
10       .
11       .
12       .
13    }
```

Il primo passo che compie la funzione è separare la stringa (che in questo momento contiene un intera riga del testo da convertire) che riceve in input usando come separatore il segno di tabulazione identificato in precedenza come separatore definito per la distinzione tra colonne in **kern.
I singoli elementi così separati vengono a questo punto analizzati uno ad uno e, a seconda del valore letto, viene chiamata la funzione relativa al tipo di evento riconosciuto: 1) "createClefEvent" per le chiavi; 2) "createKeySignEvent" per l'armatura di chiave; 3) "createTimeSignEvent" per le indicazioni di tempo; 4) "createEventNode" per tutti gli altri eventi come note e pause. In questa sezione ci si limiterà a descrivere il funzionamento di "createEventNode" in quanto legata

alla scrittura degli elementi riguardanti ogni singola evento musicale, precisando che le altre funzioni compiono la conversione in modo analogo modificando unicamente il nome e la posizione all'interno di Spine degli eventi generati rispetto a "createEventNode".

4.2.4 La funzione createEventNode

Innanzitutto createEventNode decide il nome identificativo che verrà assegnato all'evento creato. Se all'interno dell'array di strumenti creato in precedenza dalla funzione instrumentList esiste un valore corrispondente alla posizione (numero di colonna a partire da sinistra con l'aggiunta di un eventuale offset dovuto alla presenza di colonne saltate) dell'evento, verrà utilizzato il nome dello strumento seguito da un numero incrementale.

```
1 if(IList[pos] == null)
2   instrument = "e_" + (pos+1);
3 else
4   instrument = IList[pos];
```

In caso di assenza del nome dello strumento nell'array, la funzione assegna come identificativo la lettera "e" seguita sempre da un numero incrementale. È pur sempre preferibile usare come input della conversione un file **kern che dichiari gli strumenti rappresentati in forma di tandem interpretation "*I" seguto dal nome dello strumento ai fini di creare un file IEEE 1599 in cui sia facilmente possibile distinguere le diverse parti semplicemente dal nome identificato.
Il secondo passo della conversione è quello di determinare i valori degli attribuiti "timing" e "hpos" dell'evento da creare.
Per fare ciò la funzione legge il valore rappresentante la durata della nota e lo converte in un valore che segue lo schema seguente in cui si è scelto di rappresentare ogni singola nota come un evento distanti temporalmente tra loro di un offset determinato a partire dalla definizione di una semiminima con il valore "1024" e le altre durate con le potenze di 2.

case "1":value = 4096; case "2":value = 2048; case "4":value = 1024;
case "8":value = 512; case "12":value = 384; case "16":value = 256;
case "32":value = 128; case "64":value = 64; case "96":value = 32;
case "192":value = 16; default:value = 999;

L'ultimo passaggio da compiere a questo punto è la creazione dell'evento vero e proprio (con la funzione di javaScript xmlDocument.createElement()) e inserirlo come ultimo elemento all'interno di Spine assegnando come valori degli attributi hpos e timing, i valori determinati in precedenza.

4.2.5 Il LOS: gestione delle battute

Il secondo elemento del layer Logic che richiede di essere gestito è il LOS. Quest'ultimo è suddiviso in parti, una per ogni strumento presento nella partitura, e ogni parte è suddivisa nelle singole battute identificate dagli elementi "measure". Infine ogni battuta racchiude dentro di se una serie di elementi riportanti le informazioni relative ad ogni nota o pausa della battuta e sempre riferiti ad eventi di Spine.
In **kern le battute sono rappresentate come una riga in cui per ogni colonna ceh compone il file ci deve essere un simbolo di uguale "=" seguito dal numero della battuta.
L'applicativo implementa una funzione "createMeasure" che viene chiamata ogni volta che viene lette una riga aperta dal simbolo di uguale.

```
1 if(tmpString.includes("=") == true && tmpString.includes("==") ==
     false){
2  var line = tmpString.split("\t");
3  for(j=0; j<line.length; j++){
4   if(!xmlDoc.getElementsByTagName("part")[0]) {
5    createPart(xmlDoc, "e");
6   if(xmlDoc.getElementsByTagName("part")[j])
7    createMeasure(xmlDoc, line[j],
8     xmlDoc.getElementsByTagName("part")[j].getAttribute("id"));
```

Il funzionamento è analogo a quello delle funzioni precedenti: prima viene separata la stringa usando il carattere di tabulazione come separatore, in seguito viene determinata l'elemento "part" (il numero della colonna) a cui si riferisce la battuta e a questo punto viene chiamata la funzione createMeasure che crea l'elemento e posizionato come ultimo figlio del nodo padre.
Ai fini di una corretta conversione, il file **kern deve contenere tutti i riferimenti alle battute in modo completo ed ordinato poichè tutta la conversione successiva dei singoli elementi musicali si basa sul presupposto che esistano gli elementi

"measure", delle battute, a cui appoggiarsi per posizionare le note che ne fanno parte. Ciò permette di localizzare di volta in volta la posizione corretta dove porre l'evento musicale, presupposto reso necessario dalla decisione di leggere il file riga per riga, ovvero battuta per battuta.

4.2.6 Il LOS: gestione degli eventi musicali

Ora che è stata stabilita una struttura a cui appoggiarsi per il posizionamento corretto degli eventi musicali, è possibile procedere alla definizione delle funzioni per la conversione di questi ultimi. In particolare il processo di conversione si compone di una serie di funzioni eseguite in successione:

1. createChord e createRest, due funzioni dal funzionamento analogo che distinguono se si tratta di un accordo o di una pausa
2. createDuration, funzione che determina la lunghezza di ogni singola nota o pausa convertendo il valore numerico di **kern in un valore di durata
3. createNotehead e createPitch, funzioni che creano l'evento nota e ne determinano l'altezza convertendo il valore alfabetico di **kern in notazione anglosassone

La conversione inizia all'interno di createEventNode, la funzione che una volta creato un "event" all'interno di Spine, riconosce se la stringa in analisi contiene il carattere "r" rappresentante una pausa o se si tratta di una nota musicale e chiama la funzione relativa alla creazione dell'elemento trovato (createRest per le pause, createChord per le note).

```
1 if(tmpString.includes("r"))
2   createRest(xmlDoc, tmpString, pos, eventRef);
3 else
4   createChord(xmlDoc, tmpString, pos, eventRef);
```

All'interno di queste funzioni viene dopo aver creato il nuovo elemento XML e averlo appeso come ultimo figlio all'elemento "measure" corrispondente, vengono chiamata la funzione createDuratione (sia per note che per pause) e la funzione createNotehead (solo per le note).
In createDuration avviene il primo passo di conversione (l'unico nel caso delle pause) che consiste nel leggere il valore numerico contenuto nella stringa e convertirlo nel corrispettivo valore di durata espresso come coppia numeratore e denominatore. L'applicativo supporta tutti i valori di durata tradizionali dalla semibreve (4/4) alla semifusa (1/256). In caso di valori errati non riconducibili ad un valore di durata tradizionale, la nota e la pausa vengono ugualmente create con valori nulli ai fini di mantenere la struttura del file.

```
case "1":num = 4,den = 4;        case "2":num = 2,den = 4;
case "8":num = 1,den = 8;        case "16":num = 1,den = 16;
case "32":num = 1,den = 32;      case "64":num = 1,den = 64;
case "96":num = 1,den = 128;     case "192":num = 1,den = 256;
defautl:num = ?, den = ?;
```

Se le funzioni fin qui descritte bastano per la rappresentazione delle pause, la conversione delle note richiede un ulteriore passaggio di conversione per il riconoscimento dell'altezza tramite la funzione createNotehead e createPitch.
La prima si limita alla creazione dell'elemento "notehead" utilizzato in IEEE 1599 per rappresentare ogni singola nota che compone l'accordo mentre la seconda si occupa della determinazione dei valori di altezza.
La funzione createPitch quindi analizza la stringa in input per estrarne le informazioni a partire dalla codifica alfabetica dell'altezza definita da **kern determinando il nome della nota in notazione inglese, l'ottava a cui appartiene ed individuando eventuali alterazioni.

Capitolo 5

Conclusioni

In questo capitolo viene analizzato lo stato attuale dell'applicativo valutando possibili migliore apportabili all'applicativo ed eventuali sviluppi futuri

5.1 Conclusioni

Il progetto è nato con l'obbiettivo di creare uno strumento semplice da utilizzare e in grado di produrre risultati soddisfacenti nella generazioni di file IEEE 1599. In particolare era d'interesse avere la possibilità di sfruttare la grande quantità di opere codificate in **kern liberamente accessibili come nel caso della libreria online KernScore che da sola offre oltre 100.000 spartiti registrati.[7] Per ottenere ciò si è deciso di creare un applicativo che svolge l'intera conversione via web rendendolo facilmente accessibile e compatibile con tutti i sistemi.
La scelta di operare con JavaScript ha permesso la creazione di un applicativo basato su funzioni con una struttura ben definita e orientate a lavorare sui singoli elementi XML rendendole facilmente modulabili e modificabili.
In fase di testing le conversioni effettuate su file **kern che rappresentavano spartiti con forme e strutture diverse hanno prodotto dei file IEEE 1599 con un layer Logic conforme al DTD e completo di tutte le informazioni relative agli eventi musicali codificati nel file originale.
Sebbene in questo senso i risultati siano stati effettivamente raggiunti non bisogna però dimenticare della sostanziale differenza a livello di profondità di informazione rappresentata, relativa ad un opera musicale, che esiste tra **kern e IEEE 1599.
I file prodotti dalla conversione si compongono esclusivamente dei layer General e Logic e costituiranno una solida base da ampliare in seguito con l'aggiunta dei layer successivi da parte dell'utente o di applicazioni successive.
In conclusione il lavoro svolto ha dimostrato la caratteristica di IEEE 1599 di essere basato su un linguaggio come strutturato come XML, rende possibile la creazione

di una serie di tools in grado di produrre e modificare file IEEE 1599 a partire dalle diverse forme di rappresentazione digitale della musica già esistenti.

5.2 Sviluppi futuri

Allo stato attuale l'applicativo resta ancora un prototipo che si limita a lavorare sulla traduzione delle sole notazioni di uno spartito.
Ma all'interno di uno spartito è possibile individuare una serie di informazioni aggiuntive presenti anche in **kern, sia che si tratti della rappresentazione della dinamica, sia che ci si riferisca ad altre informazioni prettamente grafiche.
Inoltre l'applicativo è aperto a migliorie rivolte a facilitarne l'uso da parte dell'utente come:

- implementazione della possibilità di caricare più di un file **kern alla volta
- strumento di analisi e di segnalazione di errori in fase di conversione dovuti ad una mal formattazione del file di partenza
- supporto di un applicativo Python lato server per conversioni con un carico di lavoro più elevato

Appendice A

Codice Sviluppato

```
1  //variabili globali
2  var flagFirstEvent = false;
3  var IList = [];
4  var dynamCheck = [];
5  var posCorrection = 0;
6  var multipleVoiceCorrection = 0;
7  //Funzioni index.php
8
9  function Resize(element) {
10     element.style.height = "auto";
11     element.style.height = (element.scrollHeight) + "px";
12 }
13
14 function ResizeDefault(element) {
15   element.style.height = "200px";
16   element.style.width = "75%";
17 }
18
19 function getFile(event) {
20   const input = event.target
21   if("files" in input && input.files.length > 0) {
22     placeFileContent(document.getElementById("content-target"),
         input.files[0])
23   }
24   document.getElementById('input-file').style.visibility="hidden"
25 }
26
```

```
27 function placeFileContent(target, file) {
28   readFileContent(file).then(content => {
29     target.value = content
30   }).catch(error => console.log(error))
31 }
32
33 function readFileContent(file) {
34   const reader = new FileReader()
35   return new Promise((resolve, reject) => {
36     reader.onload = event => resolve(event.target.result)
37     reader.onerror = error => reject(error)
38     reader.readAsText(file, "UTF-8")
39   })
40 }
41
42 //funzioni di convert.php
43
44 function enableToSend(){
45   document.getElementById("tmpOutput").disabled = false;
46 }
47
48 function disableToSend() {
49   document.getElementById("tmpOutput").disabled = true;
50 }
51
52 //funzione di lettura riga per riga del file caricato
53 function readXml(xmlDocument) {
54   var NStaves = 0;
55   var xmlDoc = xmlDocument.responseXML;
56   var arr = document.getElementById("tmp").textContent.split("\n");
57   var i;
58   for(i=0; i<arr.length; i++){
59     var tmpString = arr[i];
60     if(tmpString.includes("!!!") == true)
61       referenceRecords(xmlDoc, tmpString);
62     else if(tmpString.includes("**kern") == true)
63       NStaves = stavesCounter(xmlDoc, tmpString);
64     else if(tmpString.includes("*I") == true)
65       instrumentsList(xmlDoc, tmpString);
```

```
66      else if(tmpString.includes("*clef") == true || tmpString.
           includes("*k[") == true || tmpString.includes("*M") == true
           || tmpString.includes("*met") == true)
67        createSpine(xmlDoc, tmpString, NStaves);
68      else if(tmpString.includes("=") == true && tmpString.includes("
           ==") == false){
69        var line = tmpString.split("\t");
70        for(j=0; j<line.length; j++){
71          if(j>NStaves)
72            continue
73          if(!xmlDoc.getElementsByTagName("part")[0]) {
74            createPart(xmlDoc, "e");
75            createStaff(xmlDoc, 0);
76          }
77          if(xmlDoc.getElementsByTagName("part")[j])
78            createMeasure(xmlDoc, line[j], xmlDoc.getElementsByTagName
                ("part")[j].getAttribute("id"));
79          createVoice(xmlDoc, line[j], j);
80        }
81        posCorrection = 0;
82        multipleVoiceCorrection = 0;
83      }
84      else if((tmpString.includes("*") && !tmpString.includes("**"))
           == true || tmpString == "" || (tmpString.includes("!") ==
           true && !tmpString.includes("!!!")))
85        ;
86      else
87        createSpine(xmlDoc, tmpString, NStaves);
88    }
89    document.getElementById("tmpOutput").innerHTML = saveXml(xmlDoc);
90  }
91
92  function saveXml(xmlDocument) {
93    const serializer = new XMLSerializer();
94    return serializer.serializeToString(xmlDocument);
95  }
```

Listing A.1: function.js

```
//variabili globali

var event_length = [];
var valueFlag = false;

//funzioni per la creazione degli eventi all'interno di Spine

function stavesCounter(xmlDoc, tmpString){
  var line = tmpString.split("\t");
  var counter = 0;
  var j;
  for(j=0; j<line.length; j++){
    if(line[j].includes("**kern")){
      counter += 1;

      dynamCheck[j] = false;
    }
    else
      dynamCheck[j] = true;
  }
  return counter;
}

function instrumentsList(xmlDoc, tmpString){
  var line = tmpString.split("\t");
  var j;
  var pos = 0;
  for(j=0; j<line.length; j++){
    if(dynamCheck[j] == true)
      continue
    IList[pos] = line[j].substring(line[j].search("I") + 1)
    if(dynamCheck[j] == false)
      createStaff(xmlDoc, pos)
    createPart(xmlDoc, IList[pos]);
    IList[pos] += "_" + (pos+1);
    event_length[pos] = 1;
    pos += 1;
  }
  return 0;
```

```
40 }
41
42 function createSpine(xmlDoc, tmpString, NStaves){
43   createLos(xmlDoc);
44   var line = tmpString.split("\t");
45   var j;
46   var pos = 0;
47   for(j=0; j<line.length; j++) {
48     if(j>NStaves)
49       continue
50     if(dynamCheck[j] == true) {
51       continue;
52     }
53     if(line[j].includes("*clef") == true) {
54       createClefEvent(xmlDoc, pos);
55       createClef(xmlDoc, line[j], pos);
56     }
57     else if(line[j].includes("*k[") == true){
58       createKeySignEvent(xmlDoc, pos);
59       createKeySignature(xmlDoc, line[j], pos);
60     }
61     else if(line[j].includes("*M") == true || line[j].includes("*met
         "))
62       createTimeSignEvent(xmlDoc, line[j], pos);
63     else if(line[j].includes("==")==true)
64       ;
65     else
66       createEventNode(xmlDoc, line[j], pos, valueFlag);
67     pos += 1;
68   }
69   posCorrection = 0;
70   valueFlag = false;
71   return 0;
72 }
73
74 function createClefEvent(xmlDoc, pos){
75   var instrument;
76   if(IList[pos] == null)
77     instrument = "e_" + (pos+1);
78   else
```

```
79      instrument = IList[pos];
80    if(flagFirstEvent == false){
81      xmlDoc.getElementsByTagName("event")[0].setAttribute("id",
            instrument + "_clef");
82      xmlDoc.getElementsByTagName("event")[0].setAttribute("timing",
            0);
83      xmlDoc.getElementsByTagName("event")[0].setAttribute("hpos", 0);
84      flagFirstEvent = true;
85    }
86    else{
87      var x = xmlDoc.getElementsByTagName("event");
88      var loc = xmlDoc.getElementsByTagName("event")[x.length-1];
89      var newElem = xmlDoc.createElement("event");
90      loc.parentNode.insertBefore(newElem, loc.nextSibling);
91      xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute("
            id", instrument + "_clef");
92      xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute("
            timing", 0);
93      xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute("
            hpos", 0);
94      loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
            loc.nextSibling);
95    }
96    return 0;
97  }
98
99  function createKeySignEvent(xmlDoc, pos){
100   var instrument;
101   if(IList[pos] == null)
102     instrument = "e_" + (pos+1);
103   else
104     instrument = IList[pos];
105   if(flagFirstEvent == false){
106     xmlDoc.getElementsByTagName("event")[0].setAttribute("id",
            instrument + "_keySign");
107     xmlDoc.getElementsByTagName("event")[0].setAttribute("timing",
            0);
108     xmlDoc.getElementsByTagName("event")[0].setAttribute("hpos", 0);
109     flagFirstEvent = true;
110   }
```

```
111   else{
112     var x = xmlDoc.getElementsByTagName("event");
113     var loc = xmlDoc.getElementsByTagName("event")[x.length-1];
114     var newElem = xmlDoc.createElement("event");
115     loc.parentNode.insertBefore(newElem, loc.nextSibling);
116     xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute("
          id", instrument + "_keySign");
117     xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute("
          timing", 0);
118     xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute("
          hpos", 0);
119     loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
          loc.nextSibling);
120   }
121   return 0;
122 }
123
124 function createTimeSignEvent(xmlDoc, tmpString, pos){
125   if(tmpString.includes("*M") == true && tmpString.includes("/") ==
        true) {
126     var instrument;
127     if(IList[pos] == null)
128       instrument = "e_" + (pos+1);
129     else
130       instrument = IList[pos];
131     if(flagFirstEvent == false){
132       xmlDoc.getElementsByTagName("event")[0].setAttribute("id",
            instrument + "_timeSign");
133       xmlDoc.getElementsByTagName("event")[0].setAttribute("timing",
            0);
134       xmlDoc.getElementsByTagName("event")[0].setAttribute("hpos",
            0);
135       flagFirstEvent = true;
136     }
137     else{
138       var x = xmlDoc.getElementsByTagName("event");
139       var loc = xmlDoc.getElementsByTagName("event")[x.length-1];
140       var newElem = xmlDoc.createElement("event");
141       loc.parentNode.insertBefore(newElem, loc.nextSibling);
```

```
142       xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute(
              "id", instrument + "_timeSign");
143       xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute(
              "timing", 0);
144       xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute(
              "hpos", 0);
145       loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
               loc.nextSibling);
146     }
147     createTimeSignature(xmlDoc, tmpString, pos);
148   }
149   else {
150     if(tmpString.includes("*MM") == true)
151       createMetronomicIndication(xmlDoc, tmpString, pos);
152     else if(tmpString.includes("*met"))
153       ;
154     else
155       createTimeSignature(xmlDoc, tmpString, pos);
156   }
157   return 0;
158 }
159
160 //funzione per la creazione dei singoli eventi in Spine e in Los
161 function createEventNode(xmlDoc, tmpString, pos, valueFlag){
162   var instrument;
163   if(IList[pos] == null)
164     instrument = "e_" + (pos+1);
165   else
166     instrument = IList[pos];
167   if(!event_length[pos])
168     event_length[pos] = 1;
169   if(tmpString == ".")
170     ;
171   else {
172     var value = 0;
173     if(valueFlag == false){
174       value = setValue(tmpString);
175     }
176     else
177       value = 0;
```

```
178     if(flagFirstEvent == false){
179       xmlDoc.getElementsByTagName("event")[0].setAttribute("id",
              instrument + "_" + event_length[pos]);
180       xmlDoc.getElementsByTagName("event")[0].setAttribute("timing",
              value);
181       xmlDoc.getElementsByTagName("event")[0].setAttribute("hpos",
              value);
182       flagFirstEvent = true;
183     }
184     else{
185       var x = xmlDoc.getElementsByTagName("event");
186       var loc = xmlDoc.getElementsByTagName("event")[x.length-1];
187       var newElem = xmlDoc.createElement("event");
188       loc.parentNode.insertBefore(newElem, loc.nextSibling);
189       xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute(
              "id", instrument + "_" + event_length[pos]);
190       xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute(
              "timing", value);
191       xmlDoc.getElementsByTagName("event")[x.length-1].setAttribute(
              "hpos", value);
192       loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
              loc.nextSibling);
193     }
194     var eventRef = instrument + "_" + event_length[pos]
195     event_length[pos] += 1;
196   }
197   var levFlag;
198   if(tmpString.includes("r"))
199     createRest(xmlDoc, tmpString, pos, eventRef);
200   else
201     levFlag = createChord(xmlDoc, tmpString, pos, eventRef);
202   if(levFlag == 1)
203     createChord(xmlDoc, tmpString, pos, eventRef);
204   if(tmpString.includes("(") || tmpString.includes("[") ||
        tmpString.includes(")") || tmpString.includes("]"))
205     createSlur(xmlDoc, tmpString, eventRef);
206   return 0;
207 }
208
209 function setValue(tmpString){
```

```
210   switch(tmpString.substring(tmpString.search("[0-9]"),tmpString.
          search("[A-Za-z.]"))) {
211     case "1":
212       value = 4096;
213       break;
214     case "2":
215       value = 2048;
216       break;
217     case "4":
218       value = 1024;
219       break;
220     case "8":
221       value = 512;
222       break;
223     case "12":
224       value = 384;
225       break;
226     case "16":
227       value = 256;
228       break;
229     case "32":
230       value = 128;
231       break;
232     case "64":
233       value = 64;
234       break;
235     case "96":
236       value = 32;
237       break;
238     case "192":
239       value = 16;
240       break;
241     default:
242       value = 999;
243       break;
244   }
245   valueFlag = true;
246   return value;
247 }
```

Listing A.2: logicLayerFunctions.js

```
1  function createLos(xmlDoc){
2    if(!xmlDoc.getElementsByTagName("los")[0]) {
3      var loc = xmlDoc.getElementsByTagName("spine")[0];
4      var newElem = xmlDoc.createElement("los");
5      var newTxt = xmlDoc.createTextNode("\t\t\t");
6      newElem.appendChild(newTxt);
7      loc.parentNode.insertBefore(newElem, loc.nextSibling);
8      loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t"), loc
           .nextSibling);
9      xmlDoc.getElementsByTagName("los")[0].appendChild(xmlDoc.
           createTextNode("\n\t\t\t"));
10   }
11 }
12
13 function createAgogics(xmlDoc, tmpString){
14   if(!xmlDoc.getElementsByTagName("los")[0])
15     createLos(xmlDoc);
16   if(xmlDoc.getElementsByTagName("agogics")[0]) {
17     var loc = xmlDoc.getElementsByTagName("agogics")[0];
18     var newElem = xmlDoc.createElement("agogics");
19     var newTxt = xmlDoc.createTextNode("\n\t\t\t\t"+tmpString.
           substring(tmpString.search(":")+2)+"\n\t\t");
20     newElem.appendChild(newTxt);
21     loc.parentNode.insertBefore(newElem, loc.nextSibling);
22     loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
           loc.nextSibling);
23   }
24   else {
25     var loc = xmlDoc.getElementsByTagName("los")[0];
26     var newElem = xmlDoc.createElement("agogics");
27     loc.appendChild(xmlDoc.createTextNode("\t"));
28     var newTxt = xmlDoc.createTextNode("\n\t\t\t\t"+tmpString.
           substring(tmpString.search(":")+2)+"\n\t\t\t");
29     newElem.appendChild(newTxt);
30     loc.appendChild(newElem);
31     loc.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
32   }
33   return 0
34 }
35
```

```
36 function createMetronomicIndication(xmlDoc, tmpString, pos) {
37   if(!xmlDoc.getElementsByTagName("los")[0])
38     createLos(xmlDoc);
39   var loc = "";
40   if(xmlDoc.getElementsByTagName("metronomic_indication")[0])
41     loc = xmlDoc.getElementsByTagName("metronomic_indication")[
         xmlDoc.getElementsByTagName("metronomic_indication").length
         -1];
42   else if(xmlDoc.getElementsByTagName("text_field")[0])
43     loc = xmlDoc.getElementsByTagName("text_field")[xmlDoc.
         getElementsByTagName("text_field").length-1];
44   else if(xmlDoc.getElementsByTagName("agogics")[0])
45     loc = xmlDoc.getElementsByTagName("agogics")[xmlDoc.
         getElementsByTagName("agogics").length-1];
46   else if(xmlDoc.getElementsByTagName("staff_list")[0])
47     loc = xmlDoc.getElementsByTagName("staff_list")[xmlDoc.
         getElementsByTagName("staff_list").length-1];
48   ;
49   if(loc == "") {
50     loc = xmlDoc.getElementsByTagName("los")[xmlDoc.
         getElementsByTagName("los").length-1];
51     var newElem = xmlDoc.createElement("metronomic_indication");
52     loc.appendChild(newElem);
53     xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
         getElementsByTagName("metronomic_indication").length-1].
         setAttribute("num","1");
54     xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
         getElementsByTagName("metronomic_indication").length-1].
         setAttribute("den","4");
55     xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
         getElementsByTagName("metronomic_indication").length-1].
         setAttribute("value",tmpString.substring(tmpString.search("
         [0-9]")));
56     xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
         getElementsByTagName("metronomic_indication").length-1].
         setAttribute("event_ref",IList[pos]+"_keySign");
57     loc.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
58
59   }
60   else {
```

```
61      var newElem = xmlDoc.createElement("metronomic_indication");
62      loc.parentNode.insertBefore(newElem, loc.nextSibling);
63      xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
            getElementsByTagName("metronomic_indication").length-1].
            setAttribute("num","1");
64      xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
            getElementsByTagName("metronomic_indication").length-1].
            setAttribute("den","4");
65      xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
            getElementsByTagName("metronomic_indication").length-1].
            setAttribute("value",tmpString.substring(tmpString.search("
            [0-9]")));
66      xmlDoc.getElementsByTagName("metronomic_indication")[xmlDoc.
            getElementsByTagName("metronomic_indication").length-1].
            setAttribute("event_ref",IList[pos]+"_keySign");
67      loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
            loc.nextSibling);
68    }
69    return 0
70  }
71
72  function createStaffList(xmlDoc) {
73    if(!xmlDoc.getElementsByTagName("los")[0])
74      createLos(xmlDoc);
75    var loc = "";
76    if(xmlDoc.getElementsByTagName("staff_list")[0])
77      loc = xmlDoc.getElementsByTagName("staff_list")[xmlDoc.
            getElementsByTagName("staff_list").length-1];
78    else if(xmlDoc.getElementsByTagName("metronomic_indication")[0])
79      loc = xmlDoc.getElementsByTagName("metronomic_indication")[
            xmlDoc.getElementsByTagName("metronomic_indication").length
            -1];
80    else if(xmlDoc.getElementsByTagName("text_field")[0])
81      loc = xmlDoc.getElementsByTagName("text_field")[xmlDoc.
            getElementsByTagName("text_field").length-1];
82    else if(xmlDoc.getElementsByTagName("agogics")[0])
83      loc = xmlDoc.getElementsByTagName("agogics")[xmlDoc.
            getElementsByTagName("agogics").length-1];
84    if(loc == "") {
```

```
85      loc = xmlDoc.getElementsByTagName("los")[xmlDoc.
            getElementsByTagName("los").length-1];
86      var newElem = xmlDoc.createElement("staff_list");
87      var newTxt = xmlDoc.createTextNode("\n\t\t\t");
88      newElem.appendChild(newTxt);
89      loc.appendChild(newElem);
90      loc.appendChild(xmlDoc.createTextNode("\n\t\t"));
91    }
92    else {
93      var newElem = xmlDoc.createElement("staff_list");
94      var newTxt = xmlDoc.createTextNode("\n\t\t\t");
95      newElem.appendChild(newTxt);
96      loc.parentNode.insertBefore(newElem, loc.nextSibling);
97      loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
            loc.nextSibling);
98    }
99    return 0
100 }
101
102 function createStaff(xmlDoc, pos){
103   if(!xmlDoc.getElementsByTagName("los")[0])
104     createLos(xmlDoc);
105   if(!xmlDoc.getElementsByTagName("staff_list")[0])
106     createStaffList(xmlDoc);
107   var loc = xmlDoc.getElementsByTagName("staff_list")[xmlDoc.
        getElementsByTagName("staff_list").length-1];
108   var newElem = xmlDoc.createElement("staff");
109   loc.appendChild(xmlDoc.createTextNode("\t"));
110   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t");
111   newElem.appendChild(newTxt);
112   loc.appendChild(newElem);
113   xmlDoc.getElementsByTagName("staff")[xmlDoc.getElementsByTagName(
        "staff").length-1].setAttribute("id",IList[pos]+(pos+1)+"
        _staff");
114   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
115   return 0
116 }
117
118 function createClef(xmlDoc, tmpString, pos) {
119   if(!xmlDoc.getElementsByTagName("los")[0])
```

```
120     createLos(xmlDoc);
121   if(!xmlDoc.getElementsByTagName("staff_list")[0])
122     createStaffList(xmlDoc);
123   if(!xmlDoc.getElementsByTagName("staff")[0])
124     createStaff(xmlDoc, tmpString);
125   var loc = xmlDoc.getElementsByTagName("staff")[pos];
126   var newElem = xmlDoc.createElement("clef");
127   loc.appendChild(xmlDoc.createTextNode("\t"));
128   loc.appendChild(newElem);
129   xmlDoc.getElementsByTagName("clef")[xmlDoc.getElementsByTagName("
        clef").length-1].setAttribute("event_ref",IList[pos]+"_clef");
130   xmlDoc.getElementsByTagName("clef")[xmlDoc.getElementsByTagName("
        clef").length-1].setAttribute("shape", tmpString.substring(
        tmpString.search("[A-Z]"),tmpString.search("[0-9]")));
131   xmlDoc.getElementsByTagName("clef")[xmlDoc.getElementsByTagName("
        clef").length-1].setAttribute("staff_step", tmpString.
        substring(tmpString.search("[0-9]")));
132   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t"));
133   return 0
134 }
135
136 function createKeySignature(xmlDoc, tmpString, pos) {
137   if(!xmlDoc.getElementsByTagName("los")[0])
138     createLos(xmlDoc);
139   if(!xmlDoc.getElementsByTagName("staff_list")[0])
140     createStaffList(xmlDoc);
141   if(!xmlDoc.getElementsByTagName("staff")[0])
142     createStaff(xmlDoc, tmpString);
143   var loc = xmlDoc.getElementsByTagName("staff")[pos];
144   var newElem = xmlDoc.createElement("key_signature");
145   loc.appendChild(xmlDoc.createTextNode("\t"));
146   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t\t");
147   newElem.appendChild(newTxt);
148   loc.appendChild(newElem);
149   loc.getElementsByTagName("key_signature")[loc.
        getElementsByTagName("key_signature").length-1].setAttribute("
        event_ref",IList[pos]+"_keySign");
150   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t"));
151   createKeySignatureNumber(xmlDoc, tmpString, loc)
152   return 0
```

```
153 }
154
155 function createKeySignatureNumber(xmlDoc, tmpString, loc){
156   var type;
157   var newElem;
158   var number = 0;
159   if(tmpString.includes("#")) {
160     newElem = xmlDoc.createElement("sharp_num");
161     type = "sharp_num";
162   }
163   else {
164     var newElem = xmlDoc.createElement("flat_num");
165     type = "flat_num";
166   }
167   var loc = loc.getElementsByTagName("key_signature")[0];
168   loc.appendChild(xmlDoc.createTextNode("\t"));
169   loc.appendChild(newElem);
170   for(counter = 0; counter < tmpString.length; counter++) {
171       if(tmpString[counter] == "#" || tmpString[counter] == "-")
172         number += 1;
173   }
174   loc.getElementsByTagName(type)[0].setAttribute("number",number);
175   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t"));
176   return 0
177 }
178
179 function createTimeSignature(xmlDoc, tmpString, pos) {
180   if(!xmlDoc.getElementsByTagName("los")[0])
181     createLos(xmlDoc);
182   if(!xmlDoc.getElementsByTagName("staff_list")[0])
183     createStaffList(xmlDoc);
184   if(!xmlDoc.getElementsByTagName("staff")[0])
185     createStaff(xmlDoc, tmpString);
186   var loc = xmlDoc.getElementsByTagName("staff")[pos];
187   var newElem = xmlDoc.createElement("time_signature");
188   loc.appendChild(xmlDoc.createTextNode("\t"));
189   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t\t");
190   newElem.appendChild(newTxt);
191   loc.appendChild(newElem);
```

```
192   loc.getElementsByTagName("time_signature")[loc.
          getElementsByTagName("time_signature").length-1].setAttribute(
          "event_ref",IList[pos]+"_timeSign");
193   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t"));
194   createTimeIndication(xmlDoc, tmpString, loc)
195   return 0
196 }
197
198 function createTimeIndication(xmlDoc, tmpString, loc) {
199   var num;
200   var den;
201   loc = loc.getElementsByTagName("time_signature")[loc.
          getElementsByTagName("time_signature").length-1];
202   var newElem = xmlDoc.createElement("time_indication");
203   loc.appendChild(xmlDoc.createTextNode("\t"));
204   loc.appendChild(newElem);
205   num = tmpString.substring(tmpString.search("[0-9]"),tmpString.
          search("/"));
206   den = tmpString.substring(tmpString.search("/")+1);
207   loc.getElementsByTagName("time_indication")[0].setAttribute("num"
          ,num);
208   loc.getElementsByTagName("time_indication")[0].setAttribute("den"
          ,den);
209   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t"));
210   return 0
211 }
212
213 function createPart(xmlDoc, tmpString) {
214   for(k=0; k<xmlDoc.getElementsByTagName("part").length; k++){
215     if(xmlDoc.getElementsByTagName("part")[k].getAttribute("id") ==
            tmpString)
216       return 0
217   }
218   if(!xmlDoc.getElementsByTagName("los")[0])
219     createLos(xmlDoc);
220   var loc;
221   if(xmlDoc.getElementsByTagName("part")[0])
222     loc = xmlDoc.getElementsByTagName("part")[xmlDoc.
            getElementsByTagName("part").length-1];
223   else if(xmlDoc.getElementsByTagName("staff_list")[0])
```

```
224     loc = xmlDoc.getElementsByTagName("staff_list")[xmlDoc.
            getElementsByTagName("staff_list").length-1];
225   else if(xmlDoc.getElementsByTagName("metronomic_indication")[0])
226     loc = xmlDoc.getElementsByTagName("metronomic_indication")[
            xmlDoc.getElementsByTagName("metronomic_indication").length
            -1];
227   else if(xmlDoc.getElementsByTagName("text_field")[0])
228     loc = xmlDoc.getElementsByTagName("text_field")[xmlDoc.
            getElementsByTagName("text_field").length-1];
229   else if(xmlDoc.getElementsByTagName("agogics")[0])
230     loc = xmlDoc.getElementsByTagName("agogics")[xmlDoc.
            getElementsByTagName("agogics").length-1];
231   else
232     loc = xmlDoc.getElementsByTagName("los")[xmlDoc.
            getElementsByTagName("los").length-1];
233   var newElem = xmlDoc.createElement("part");
234   newElem.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
235   loc.appendChild(newElem);
236   loc.parentNode.insertBefore(newElem, loc.nextSibling);
237   xmlDoc.getElementsByTagName("part")[xmlDoc.getElementsByTagName("
        part").length-1].setAttribute("id",tmpString);
238   loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
        loc.nextSibling);
239   createVoiceList(xmlDoc, tmpString);
240   return 0
241 }
242
243 function createVoiceList(xmlDoc, tmpString) {
244   var loc = xmlDoc.getElementsByTagName("part")[xmlDoc.
        getElementsByTagName("part").length-1];
245   var newElem = xmlDoc.createElement("voice_list");
246   loc.appendChild(xmlDoc.createTextNode("\t"));
247   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t");
248   newElem.appendChild(newTxt);
249   loc.appendChild(newElem);
250   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
251 }
252
253 function createMeasure(xmlDoc, tmpString, partID) {
254   if(partID == "e")
```

```
255     IList[0] = "e";
256   for(k=0; k<xmlDoc.getElementById(partID).getElementsByTagName("
          measure").length; k++){
257     if(xmlDoc.getElementById(partID).getElementsByTagName("measure")
            [k].getAttribute("number") == (tmpString.substring(tmpString
            .search("=") + 1)))
258       return 0
259   }
260   if(!xmlDoc.getElementsByTagName("los")[0])
261     createLos(xmlDoc);
262   if(!xmlDoc.getElementsByTagName("part")[0])
263     createPart(xmlDoc, tmpString);
264   var loc = xmlDoc.getElementById(partID);
265   var newElem = xmlDoc.createElement("measure");
266   loc.appendChild(xmlDoc.createTextNode("\t"));
267   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t");
268   newElem.appendChild(newTxt);
269   loc.appendChild(newElem);
270   xmlDoc.getElementById(partID).getElementsByTagName("measure")[
          xmlDoc.getElementById(partID).getElementsByTagName("measure").
          length-1].setAttribute("number",(tmpString.substring(tmpString
          .search("=") + 1)));
271   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
272   return 0
273 }
274
275 function createVoice(xmlDoc, tmpString, pos) {
276   if(dynamCheck[pos] == true) {
277     posCorrection += 1;
278     return 0;
279   }
280   for(k=0; k<xmlDoc.getElementsByTagName("part").length; k++) {
281     if(k != pos && pos != IList.length && xmlDoc.
            getElementsByTagName("part")[k].getAttribute("id") == IList[
            pos].substring(0,IList[pos].search("_"))) {
282       posCorrection += 1;
283       multipleVoiceCorrection += 1;
284     }
285   }
286   pos = pos - posCorrection;
```

```
287   if(!xmlDoc.getElementsByTagName("los")[0])
288     createLos(xmlDoc);
289   if(!xmlDoc.getElementsByTagName("part")[0])
290     createPart(xmlDoc, tmpString);
291   if(!xmlDoc.getElementsByTagName("measure")[0])
292     createMeasure(xmlDoc, tmpString);
293   var loc;
294   var voiceSkipFlag = false;
295   loc = xmlDoc.getElementsByTagName("part")[pos].
          getElementsByTagName("voice_list")[0];
296   if(loc.getElementsByTagName("voice_item")[0])
297   for(voiceCounter = 0; voiceCounter<loc.getElementsByTagName("
          voice_item").length; voiceCounter++) {
298     if(loc.getElementsByTagName("voice_item")[voiceCounter].
            getAttribute("id") == IList[pos+multipleVoiceCorrection])
299       voiceSkipFlag = true;
300   }
301   if(voiceSkipFlag == false) {
302     var newElem = xmlDoc.createElement("voice_item");
303     loc.appendChild(xmlDoc.createTextNode("\t"));
304     loc.appendChild(newElem);
305     loc.getElementsByTagName("voice_item")[loc.getElementsByTagName(
          "voice_item").length-1].setAttribute("id",IList[pos+
          multipleVoiceCorrection]);
306     loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t"));
307   }
308   loc = xmlDoc.getElementsByTagName("part")[pos].
          getElementsByTagName("measure")[xmlDoc.getElementsByTagName("
          part")[pos].getElementsByTagName("measure").length-1];
309   var newElem = xmlDoc.createElement("voice");
310   loc.appendChild(xmlDoc.createTextNode("\t"));
311   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t\t");
312   newElem.appendChild(newTxt);
313   loc.appendChild(newElem);
314   loc.getElementsByTagName("voice")[loc.getElementsByTagName("voice
          ").length-1].setAttribute("voice_item_ref",IList[pos+
          multipleVoiceCorrection]);
315   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t"));
316   return 0
317 }
```

```
318
319 function createChord(xmlDoc, tmpString, pos, eventRef) {
320   for(k=0; k<xmlDoc.getElementsByTagName("part").length; k++) {
321     if(k != pos && pos != IList.length && xmlDoc.
          getElementsByTagName("part")[k].getAttribute("id") == IList[
          pos].substring(0,IList[pos].search("_")))
322       posCorrection += 1;
323   }
324   if(tmpString == ".")
325     return 0;
326
327   pos = pos - posCorrection;
328   if(!xmlDoc.getElementsByTagName("los")[0])
329     createLos(xmlDoc);
330   if(!xmlDoc.getElementsByTagName("part")[0])
331     createPart(xmlDoc, tmpString);
332   if(!xmlDoc.getElementsByTagName("part")[pos].getElementsByTagName
        ("measure")[0])
333     return 0;
334   if(!xmlDoc.getElementsByTagName("part")[pos].getElementsByTagName
        ("voice")[0])
335     createVoice(xmlDoc, tmpString, pos);
336   var loc;
337   if(xmlDoc.getElementsByTagName("part")[pos].getElementsByTagName(
        "measure")[xmlDoc.getElementsByTagName("part")[pos].
        getElementsByTagName("measure").length-1].getElementsByTagName
        ("voice")[1])
338     loc = xmlDoc.getElementsByTagName("part")[pos].
          getElementsByTagName("measure")[xmlDoc.getElementsByTagName(
          "part")[pos].getElementsByTagName("measure").length-1].
          getElementsByTagName("voice")[pos + posCorrection];
339   else
340     loc = xmlDoc.getElementsByTagName("part")[pos].
          getElementsByTagName("measure")[xmlDoc.getElementsByTagName(
          "part")[pos].getElementsByTagName("measure").length-1].
          getElementsByTagName("voice")[0];
341   var newElem = xmlDoc.createElement("chord");
342   loc.appendChild(xmlDoc.createTextNode("\t"));
343   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t\t\t\t");
344   newElem.appendChild(newTxt);
```

```
345   loc.appendChild(newElem);
346   loc.getElementsByTagName("chord")[loc.getElementsByTagName("chord
        ").length-1].setAttribute("event_ref", eventRef);
347   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t"));
348   createDuration(xmlDoc, tmpString, pos, eventRef, "chord", loc);
349   if(tmpString.indexOf(".") != -1) {
350     createAugmentationDots(xmlDoc, tmpString, "chord", loc);
351   }
352   var line = tmpString.split(" ");
353   for(c=0; c<line.length; c++){
354     createNotehead(xmlDoc, line[c], pos, eventRef, "chord", loc);
355   }
356   return 0
357 }
358
359 function createRest(xmlDoc, tmpString, pos, eventRef) {
360   for(k=0; k<xmlDoc.getElementsByTagName("part").length; k++) {
361     if(k != pos && pos != IList.length && xmlDoc.
          getElementsByTagName("part")[k].getAttribute("id") == IList[
          pos].substring(0,IList[pos].search("_")))
362       posCorrection += 1;
363   }
364   pos = pos - posCorrection;
365   if(!xmlDoc.getElementsByTagName("los")[0])
366     createLos(xmlDoc);
367   if(!xmlDoc.getElementsByTagName("part")[0])
368     createPart(xmlDoc, tmpString);
369   if(!xmlDoc.getElementsByTagName("part")[pos].getElementsByTagName
        ("measure")[0])
370     return 0;
371   if(!xmlDoc.getElementsByTagName("part")[pos].getElementsByTagName
        ("voice")[0])
372     createVoice(xmlDoc, tmpString, pos);
373   var loc = "";
374   if(xmlDoc.getElementsByTagName("part")[pos].getElementsByTagName(
        "measure")[xmlDoc.getElementsByTagName("part")[pos].
        getElementsByTagName("measure").length-1].getElementsByTagName
        ("voice")[1])
```

```
375     loc = xmlDoc.getElementsByTagName("part")[pos].
            getElementsByTagName("measure")[xmlDoc.getElementsByTagName(
            "part")[pos].getElementsByTagName("measure").length-1].
            getElementsByTagName("voice")[pos + posCorrection];
376   else
377     loc = xmlDoc.getElementsByTagName("part")[pos].
            getElementsByTagName("measure")[xmlDoc.getElementsByTagName(
            "part")[pos].getElementsByTagName("measure").length-1].
            getElementsByTagName("voice")[0];
378   var newElem = xmlDoc.createElement("rest");
379   loc.appendChild(xmlDoc.createTextNode("\t"));
380   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t\t\t\t\t");
381   newElem.appendChild(newTxt);
382   loc.appendChild(newElem);
383   loc.getElementsByTagName("rest")[loc.getElementsByTagName("rest")
        .length-1].setAttribute("event_ref", eventRef);
384   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t"));
385   createDuration(xmlDoc, tmpString, pos, eventRef, "rest", loc);
386   return 0
387 }
388
389 function createDuration(xmlDoc, tmpString, pos, eventRef, type, loc
        ){
390   if(!xmlDoc.getElementsByTagName("los")[0])
391     createLos(xmlDoc);
392   if(!xmlDoc.getElementsByTagName("part")[0])
393     createPart(xmlDoc, tmpString);
394   if(!xmlDoc.getElementsByTagName("measure")[0])
395     createMeasure(xmlDoc, tmpString);
396   if(!xmlDoc.getElementsByTagName("voice")[0])
397     createVoice(xmlDoc, tmpString, pos);
398   if(!xmlDoc.getElementsByTagName(type)[0])
399     createChord(xmlDoc, tmpString, pos, eventRef);
400   var loc;
401   loc = loc.getElementsByTagName(type)[loc.getElementsByTagName(
        type).length-1];
402   var newElem = xmlDoc.createElement("duration");
403   loc.appendChild(newElem);
404   var num;
405   var den;
```

```
406   switch(tmpString.substring(tmpString.search("[0-9]"),tmpString.
          search("[A-Za-z.]"))) {
407     case "1":
408       num = 4;
409       den = 4;
410       break;
411     case "2":
412       num = 2;
413       den = 4;
414       break;
415     case "4":
416       num = 1;
417       den = 4;
418       break;
419     case "8":
420       num = 1;
421       den = 8;
422       break;
423     case "16":
424       num = 1;
425       den = 16;
426       break;
427     case "32":
428       num = 1;
429       den = 32;
430       break;
431     case "64":
432       num = 1;
433       den = 64;
434       break;
435     case "96":
436       num = 1;
437       den = 128;
438       break;
439     case "192":
440       num = 1;
441       den = 256
442       break;
443     default:
444       num = "?";
```

```
445       den = "?";
446       break;
447   }
448   loc.getElementsByTagName("duration")[0].setAttribute("num", num);
449   loc.getElementsByTagName("duration")[0].setAttribute("den", den);
450   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t\t"));
451   return 0;
452 }
453
454 function createAugmentationDots(xmlDoc, tmpString, type, loc){
455   var loc;
456   var number = 0;
457   loc = loc.getElementsByTagName(type)[loc.getElementsByTagName(
          type).length-1];
458   var newElem = xmlDoc.createElement("augmentation_dots");
459   loc.appendChild(xmlDoc.createTextNode("\t"));
460   loc.appendChild(newElem);
461   for(cc = 0; cc<tmpString.length; cc++){
462     if(tmpString[cc] == ".")
463       number += 1;
464   }
465   loc.getElementsByTagName("augmentation_dots")[0].setAttribute("
          number", number);
466   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t\t"));
467   return 0;
468 }
469
470 function createNotehead(xmlDoc, tmpString, pos, eventRef, type, loc
       ){
471   if(!xmlDoc.getElementsByTagName("los")[0])
472     createLos(xmlDoc);
473   if(!xmlDoc.getElementsByTagName("part")[0])
474     createPart(xmlDoc, tmpString);
475   if(!xmlDoc.getElementsByTagName("measure")[0])
476     createMeasure(xmlDoc, tmpString);
477   if(!xmlDoc.getElementsByTagName("voice")[0])
478     createVoice(xmlDoc, tmpString, pos);
479   if(!xmlDoc.getElementsByTagName(type)[0])
480     createChord(xmlDoc, tmpString, pos, eventRef);
481   var loc;
```

```
482   loc = loc.getElementsByTagName(type)[loc.getElementsByTagName(
          type).length-1];
483   var newElem = xmlDoc.createElement("notehead");
484   loc.appendChild(xmlDoc.createTextNode("\t"));
485   var newTxt = xmlDoc.createTextNode("\n\t\t\t\t\t\t\t\t");
486   newElem.appendChild(newTxt);
487   loc.appendChild(newElem);
488   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t\t"));
489   createPitch(xmlDoc, tmpString, pos, eventRef, type, loc);
490   if(tmpString.includes("[") || tmpString.includes("("))
491     createTie(xmlDoc, tmpString, loc);
492 }
493
494 function createPitch(xmlDoc, tmpString, pos, eventRef, type, loc){
495   if(!xmlDoc.getElementsByTagName("los")[0])
496     createLos(xmlDoc);
497   if(!xmlDoc.getElementsByTagName("part")[0])
498     createPart(xmlDoc, tmpString);
499   if(!xmlDoc.getElementsByTagName("measure")[0])
500     createMeasure(xmlDoc, tmpString);
501   if(!xmlDoc.getElementsByTagName("voice")[0])
502     createVoice(xmlDoc, tmpString, pos);
503   if(!xmlDoc.getElementsByTagName(type)[0])
504     createChord(xmlDoc, tmpString, pos, eventRef);
505   if(!xmlDoc.getElementsByTagName("duration")[0])
506     createChord(xmlDoc, tmpString, pos, eventRef, type);
507   var loc;
508   loc = loc.getElementsByTagName("notehead")[loc.
          getElementsByTagName("notehead").length-1];
509   var newElem = xmlDoc.createElement("pitch");
510   loc.appendChild(newElem);
511   var step;
512   var octave;
513   var accidental;
514   switch(tmpString.substring(tmpString.search("[A-Za-z]"),tmpString
          .search("[A-Za-z]")+1)) {
515     case "a":
516     case "A":
517       step = "A";
518       break;
```

```
519     case "b":
520     case "B":
521       step = "B";
522       break;
523     case "c":
524     case "C":
525       step = "C";
526       break;
527     case "d":
528     case "D":
529       step = "D";
530       break;
531     case "e":
532     case "E":
533       step = "E";
534       break;
535     case "f":
536     case "F":
537       step = "F";
538       break;
539     case "g":
540     case "G":
541       step = "G";
542       break;
543     default:
544       step = "?";
545       break;
546   }
547   var tmpStringUpper = tmpString.toUpperCase();
548   if(tmpStringUpper.includes("A")||tmpStringUpper.includes("B")||
        tmpStringUpper.includes("C")||tmpStringUpper.includes("D")||
        tmpStringUpper.includes("E")||tmpStringUpper.includes("F")||
        tmpStringUpper.includes("G")) {
549     if(tmpString.includes("A")||tmpString.includes("B")||tmpString.
          includes("C")||tmpString.includes("D")||tmpString.includes("
          E")||tmpString.includes("F")||tmpString.includes("G"))
550       octave = "3";
551     else
552       octave = "4";
553   }
```

```
554   if(tmpStringUpper.includes("AA")||tmpStringUpper.includes("BB")||
          tmpStringUpper.includes("CC")||tmpStringUpper.includes("DD")||
          tmpStringUpper.includes("EE")||tmpStringUpper.includes("FF")||
          tmpStringUpper.includes("GG")) {
555     if(tmpString.includes("AA")||tmpString.includes("BB")||tmpString
            .includes("CC")||tmpString.includes("DD")||tmpString.
            includes("EE")||tmpString.includes("FF")||tmpString.includes
            ("GG"))
556       octave = "2";
557     else
558       octave = "5";
559   }
560   if(tmpStringUpper.includes("AAA")||tmpStringUpper.includes("BBB")
          ||tmpStringUpper.includes("CCC")||tmpStringUpper.includes("DDD
          ")||tmpStringUpper.includes("EEE")||tmpStringUpper.includes("
          FFF")||tmpStringUpper.includes("GGG")) {
561     if(tmpString.includes("AAA")||tmpString.includes("BBB")||
            tmpString.includes("CCC")||tmpString.includes("DDD")||
            tmpString.includes("EEE")||tmpString.includes("FFF")||
            tmpString.includes("GGG"))
562       octave = "1";
563     else
564       octave = "6";
565   }
566   if(tmpStringUpper.includes("AAAA")||tmpStringUpper.includes("BBBB
          ")||tmpStringUpper.includes("CCCC")||tmpStringUpper.includes("
          DDDD")||tmpStringUpper.includes("EEEE")||tmpStringUpper.
          includes("FFFF")||tmpStringUpper.includes("GGGG"))
567     octave = "7";
568   if(tmpString.includes("#"))
569     accidental = "sharp";
570   if(tmpString.includes("-"))
571     accidental = "flat";
572   if(tmpString.includes("n"))
573     accidental = "natural";
574   loc.getElementsByTagName("pitch")[0].setAttribute("step", step);
575   loc.getElementsByTagName("pitch")[0].setAttribute("octave",
          octave);
576   if(accidental)
```

```
577     loc.getElementsByTagName("pitch")[0].setAttribute("
            actual_accidental", accidental);
578   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t\t\t"));
579   return 0;
580 }
581
582 function createTie(xmlDoc, tmpString, loc) {
583   var newElem = xmlDoc.createElement("tie");
584   loc.appendChild(xmlDoc.createTextNode("\t"));
585   loc.appendChild(newElem);
586   loc.appendChild(xmlDoc.createTextNode("\n\t\t\t\t\t\t"));
587   return 0;
588 }
589
590 function createHorizontalSymbols(xmlDoc, tmpString) {
591   if(!xmlDoc.getElementsByTagName("los")[0])
592     createLos(xmlDoc);
593   var loc;
594   if(xmlDoc.getElementsByTagName("horizontal_symbols")[0])
595     loc = xmlDoc.getElementsByTagName("horizontal_symbols")[xmlDoc.
            getElementsByTagName("horizontal_symbols").length-1];
596   else if(xmlDoc.getElementsByTagName("part")[0])
597     loc = xmlDoc.getElementsByTagName("part")[xmlDoc.
            getElementsByTagName("part").length-1];
598   else if(xmlDoc.getElementsByTagName("staff_list")[0])
599     loc = xmlDoc.getElementsByTagName("staff_list")[xmlDoc.
            getElementsByTagName("staff_list").length-1];
600   else if(xmlDoc.getElementsByTagName("metronomic_indication")[0])
601     loc = xmlDoc.getElementsByTagName("metronomic_indication")[
            xmlDoc.getElementsByTagName("metronomic_indication").length
            -1];
602   else if(xmlDoc.getElementsByTagName("agogics")[0])
603     loc = xmlDoc.getElementsByTagName("agogics")[xmlDoc.
            getElementsByTagName("agogics").length-1];
604   else
605     loc = xmlDoc.getElementsByTagName("los")[xmlDoc.
            getElementsByTagName("los").length-1];
606   var newElem = xmlDoc.createElement("horizontal_symbols");
607   var newTxt = xmlDoc.createTextNode("\n\t\t\t");
608   newElem.appendChild(newTxt);
```

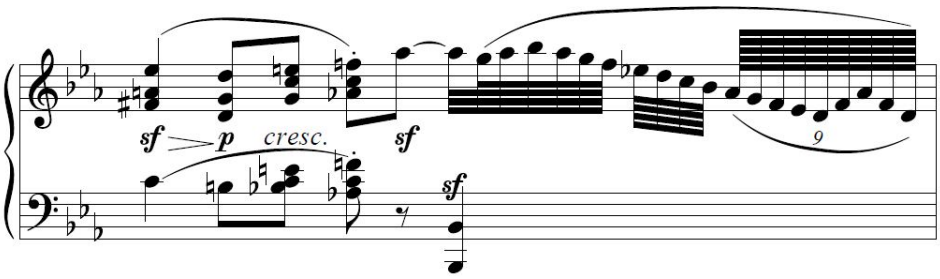
```
609   loc.parentNode.insertBefore(newElem, loc.nextSibling);
610   loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\t"),
          loc.nextSibling);
611   return 0
612 }
613
614 function createSlur(xmlDoc, tmpString, eventRef) {
615   for(n=0; n<tmpString.length; n++){
616     if(tmpString[n] == "(" || tmpString[n] == "[" || tmpString[n] ==
          ")" || tmpString[n] == "]") {
617       if(!xmlDoc.getElementsByTagName("los")[0])
618         createLos(xmlDoc);
619       if(!xmlDoc.getElementsByTagName("horizontal_symbols")[0])
620         createHorizontalSymbols(xmlDoc, tmpString);
621       var loc = "";
622       if(xmlDoc.getElementsByTagName("slur")[0])
623         loc = xmlDoc.getElementsByTagName("slur")[xmlDoc.
              getElementsByTagName("slur").length-1];
624       if(loc != "") {
625         if(tmpString[n] == "(" || tmpString[n] == "[") {
626           var newElem = xmlDoc.createElement("slur");
627           loc.parentNode.insertBefore(newElem, loc.nextSibling);
628           xmlDoc.getElementsByTagName("slur")[xmlDoc.
                getElementsByTagName("slur").length-1].setAttribute("
                start_event_ref",eventRef);
629           xmlDoc.getElementsByTagName("slur")[xmlDoc.
                getElementsByTagName("slur").length-1].setAttribute("
                end_event_ref","");
630           loc.parentNode.insertBefore(xmlDoc.createTextNode("\n\t\t\
                t\t"), loc.nextSibling);
631         }
632         else {
633           for(z=0; z<xmlDoc.getElementsByTagName("slur").length; z
                ++) {
634             if(xmlDoc.getElementsByTagName("slur")[z].getAttribute("
                  end_event_ref") == "") {
635               xmlDoc.getElementsByTagName("slur")[z].setAttribute("
                    end_event_ref",eventRef);
636           }
637         }
```

```
638         }
639       }
640       else {
641         loc = xmlDoc.getElementsByTagName("horizontal_symbols")[
              xmlDoc.getElementsByTagName("horizontal_symbols").length
              -1];
642         var newElem = xmlDoc.createElement("slur");
643         loc.appendChild(xmlDoc.createTextNode("\t"));
644         loc.appendChild(newElem);
645         xmlDoc.getElementsByTagName("slur")[0].setAttribute("
              start_event_ref",eventRef);
646         xmlDoc.getElementsByTagName("slur")[0].setAttribute("
              end_event_ref","");
647         loc.appendChild(xmlDoc.createTextNode("\n\t\t\t"));
648       }
649     }
650   }
651   return 0
652 }
```

Listing A.3: losFunctions.js

Appendice B

Esempio di Conversione




```
!!!COM: Beethoven, Ludwig van
!!!CDT: 1770///-1827///
!!!OTL: Piano Sonata no. 8 in C minor
!!!OTP: path&eacute;tique
!!!OPS: 13
!!!OMV: 1
**kern **kern **dynam
*staff2 *staff1 *staff1
*Ipiano *Ipiano *Ipiano
*>[I,A,A1,A,A2,B] *>[I,A,A1,A,A2,B] *>[I,A,A1,A,A2,B]
*>norep[I,A,A2,B] *>norep[I,A,A2,B] *>norep[I,A,A2,B]
*>I *>I *>I
*clefF4 *clefG2 *clefG2
```

```
*k[b#e#a#] *k[b-e-a-] *k[b-e-a-]
*c: *c: *c:
*met(c) *met(c) *met(c)
*M4/4 *M4/4 *M4/4
*MM40 *MM40 *MM40
=4 =4 =4
(4.c\ 4f#\ 4an\ (4ee-\ .
8Bn\ 8d/ 8g/ 8dd/L .
8B-\ 8c\ 8en\ 8g/ 8cc/ 8een/J .
8A-'\ 8c'\ 8fn'\) 8a-'\ 8cc'\ 8ff'\L) .
8r [8aa-\J .
4BBB-/ 4BB-/ (32aa-\LLL] .
. 96gg\L .
. 96aa-\ .
. 96bb-\ .
. 96aa-\ .
. 96gg\ .
. 96ff\JJJJ .
. 64ee-\LLLL .
. 64dd\ .
. 64cc\ .
. 64b-\JJJJ .
. 96a-/LLLL .
. 96g/ .
. 96fn/ .
. 96e-/ .
. 96d/ .
. 96f/ .
. 96a-/ .
. 96f/ .
. 96d/JJJJ .
== == ==
*- *- *-
!!!ENC: Craig Stuart Sapp
!!!END: 2008/02/25/
!!!hum2abc: -s 0.65 --spacing 1.2
```

Listing B.1: file oriinale .krn

```
<ieee1599 version="1.0">
 <general>
  <description>
   <main_title>Piano Sonata no. 8 in C minor</main_title>
   <author type="composer">Beethoven, Ludwig van</author>
   <other_title>pathetique</other_title>
   <number>1</number>
   <work_number>13</work_number>
  </description>
  <notes>
   composer dates: 1770///-1827///
   encoder of electronic document: Craig Stuart Sapp
  </notes>
 </general>
 <logic>
  <spine>
   <event id="piano_1_clef" timing="0" hpos="0"/>
   <event id="piano_2_clef" timing="0" hpos="0"/>
   <event id="piano_1_keySign" timing="0" hpos="0"/>
   <event id="piano_2_keySign" timing="0" hpos="0"/>
   <event id="piano_1_timeSign" timing="0" hpos="0"/>
   <event id="piano_2_timeSign" timing="0" hpos="0"/>
   <event id="piano_1_1" timing="1024" hpos="1024"/>
   <event id="piano_2_1" timing="0" hpos="0"/>
   <event id="piano_1_2" timing="512" hpos="512"/>
   <event id="piano_2_2" timing="0" hpos="0"/>
   <event id="piano_1_3" timing="512" hpos="512"/>
   <event id="piano_2_3" timing="0" hpos="0"/>
   <event id="piano_1_4" timing="512" hpos="512"/>
   <event id="piano_2_4" timing="0" hpos="0"/>
   <event id="piano_1_5" timing="512" hpos="512"/>
   <event id="piano_2_5" timing="0" hpos="0"/>
   <event id="piano_1_6" timing="1024" hpos="1024"/>
   <event id="piano_2_6" timing="0" hpos="0"/>
   <event id="piano_2_7" timing="32" hpos="32"/>
   <event id="piano_2_8" timing="32" hpos="32"/>
   <event id="piano_2_9" timing="32" hpos="32"/>
   <event id="piano_2_10" timing="32" hpos="32"/>
   <event id="piano_2_11" timing="32" hpos="32"/>
```

```
40    <event id="piano_2_12" timing="32" hpos="32"/>
41    <event id="piano_2_13" timing="64" hpos="64"/>
42    <event id="piano_2_14" timing="64" hpos="64"/>
43    <event id="piano_2_15" timing="64" hpos="64"/>
44    <event id="piano_2_16" timing="64" hpos="64"/>
45    <event id="piano_2_17" timing="32" hpos="32"/>
46    <event id="piano_2_18" timing="32" hpos="32"/>
47    <event id="piano_2_19" timing="32" hpos="32"/>
48    <event id="piano_2_20" timing="32" hpos="32"/>
49    <event id="piano_2_21" timing="32" hpos="32"/>
50    <event id="piano_2_22" timing="32" hpos="32"/>
51    <event id="piano_2_23" timing="32" hpos="32"/>
52    <event id="piano_2_24" timing="32" hpos="32"/>
53    <event id="piano_2_25" timing="32" hpos="32"/>
54   </spine>
55   <los>
56    <staff_list>
57     <staff id="piano1_staff">
58      <clef event_ref="piano_1_clef" shape="F" staff_step="4"/>
59      <key_signature event_ref="piano_1_keySign">
60       <sharp_num number="3"/>
61      </key_signature>
62      <time_signature event_ref="piano_1_timeSign">
63       <time_indication num="4" den="4"/>
64      </time_signature>
65     </staff>
66     <staff id="piano2_staff">
67      <clef event_ref="piano_2_clef" shape="G" staff_step="2"/>
68      <key_signature event_ref="piano_2_keySign">
69       <flat_num number="3"/>
70      </key_signature>
71      <time_signature event_ref="piano_2_timeSign">
72       <time_indication num="4" den="4"/>
73      </time_signature>
74     </staff>
75    </staff_list>
76    <metronomic_indication num="1" den="4" value="40"
77     event_ref="piano_1_ketSign"/>
78
79    <metronomic_indication num="1" den="4" value="40"
```

```
80     event_ref="piano_2_ketSign"/>
81    <part id="piano">
82     <voice_list>
83      <voice_item id="piano_1"/>
84      <voice_item id="piano_2"/>
85     </voice_list>
86     <measure number="4">
87      <voice voice_item_ref="piano_1">
88       <chord event_ref="piano_1_1">
89        <duration num="1" den="4"/>
90        <augmentation_dots number="1"/>
91        <notehead>
92         <pitch step="C" octave="4"/>
93        </notehead>
94        <tie/>
95       </chord>
96       <chord event_ref="piano_1_2">
97        <duration num="1" den="8"/>
98        <notehead>
99         <pitch step="B" octave="3" actual_accidental="natural"/>
100       </notehead>
101      </chord>
102      <chord event_ref="piano_1_3">
103       <duration num="1" den="8"/>
104       <notehead>
105        <pitch step="B" octave="3" actual_accidental="flat"/>
106       </notehead>
107       <notehead>
108        <pitch step="C" octave="4"/>
109       </notehead>
110       <notehead>
111        <pitch step="E" octave="4" actual_accidental="natural"/>
112       </notehead>
113      </chord>
114      <chord event_ref="piano_1_4">
115       <duration num="1" den="8"/>
116       <notehead>
117        <pitch step="A" octave="3" actual_accidental="flat"/>
118       </notehead>
119       <notehead>
```

```
120         <pitch step="C" octave="4"/>
121        </notehead>
122        <notehead>
123         <pitch step="F" octave="4" actual_accidental="natural"/>
124        </notehead>
125       </chord>
126       <rest event_ref="piano_1_5">
127        <duration num="1" den="8"/>
128       </rest>
129       <chord event_ref="piano_1_6">
130        <duration num="1" den="4"/>
131        <notehead>
132         <pitch step="B" octave="1" actual_accidental="flat"/>
133        </notehead>
134        <notehead>
135         <pitch step="B" octave="2" actual_accidental="flat"/>
136        </notehead>
137       </chord>
138      </voice>
139      <voice voice_item_ref="piano_2">
140       <chord event_ref="piano_2_1">
141        <duration num="1" den="4"/>
142        <notehead>
143         <pitch step="F" octave="4" actual_accidental="sharp"/>
144        </notehead>
145        <notehead>
146         <pitch step="A" octave="4" actual_accidental="natural"/>
147        </notehead>
148        <notehead>
149         <pitch step="E" octave="5" actual_accidental="flat"/>
150        </notehead>
151        <tie/>
152       </chord>
153       <chord event_ref="piano_2_2">
154        <duration num="1" den="8"/>
155        <notehead>
156         <pitch step="D" octave="4"/>
157        </notehead>
158        <notehead>
159         <pitch step="G" octave="4"/>
```

```
160       </notehead>
161       <notehead>
162        <pitch step="D" octave="5"/>
163       </notehead>
164      </chord>
165      <chord event_ref="piano_2_3">
166       <duration num="1" den="8"/>
167       <notehead>
168        <pitch step="G" octave="4"/>
169       </notehead>
170       <notehead>
171        <pitch step="C" octave="5"/>
172       </notehead>
173       <notehead>
174        <pitch step="E" octave="5" actual_accidental="natural"/>
175       </notehead>
176      </chord>
177      <chord event_ref="piano_2_4">
178       <duration num="1" den="8"/>
179       <notehead>
180        <pitch step="A" octave="4" actual_accidental="flat"/>
181       </notehead>
182       <notehead>
183        <pitch step="C" octave="5"/>
184       </notehead>
185       <notehead>
186        <pitch step="F" octave="5"/>
187       </notehead>
188      </chord>
189      <chord event_ref="piano_2_5">
190       <duration num="1" den="8"/>
191       <notehead>
192        <pitch step="A" octave="5" actual_accidental="flat"/>
193       </notehead>
194       <tie/>
195      </chord>
196      <chord event_ref="piano_2_6">
197       <duration num="1" den="32"/>
198       <notehead>
199        <pitch step="A" octave="5" actual_accidental="flat"/>
```

```
200         </notehead>
201         <tie/>
202        </chord>
203        <chord event_ref="piano_2_7">
204         <duration num="1" den="128"/>
205         <notehead>
206          <pitch step="G" octave="5"/>
207         </notehead>
208        </chord>
209        <chord event_ref="piano_2_8">
210         <duration num="1" den="128"/>
211         <notehead>
212          <pitch step="A" octave="5" actual_accidental="flat"/>
213         </notehead>
214        </chord>
215        <chord event_ref="piano_2_9">
216         <duration num="1" den="128"/>
217         <notehead>
218          <pitch step="B" octave="5" actual_accidental="flat"/>
219         </notehead>
220        </chord>
221        <chord event_ref="piano_2_10">
222         <duration num="1" den="128"/>
223         <notehead>
224          <pitch step="A" octave="5" actual_accidental="flat"/>
225         </notehead>
226        </chord>
227        <chord event_ref="piano_2_11">
228         <duration num="1" den="128"/>
229         <notehead>
230          <pitch step="G" octave="5"/>
231         </notehead>
232        </chord>
233        <chord event_ref="piano_2_12">
234         <duration num="1" den="128"/>
235         <notehead>
236          <pitch step="F" octave="5"/>
237         </notehead>
238        </chord>
239        <chord event_ref="piano_2_13">
```

```
240         <duration num="1" den="64"/>
241         <notehead>
242          <pitch step="E" octave="5" actual_accidental="flat"/>
243         </notehead>
244        </chord>
245        <chord event_ref="piano_2_14">
246         <duration num="1" den="64"/>
247         <notehead>
248          <pitch step="D" octave="5"/>
249         </notehead>
250        </chord>
251        <chord event_ref="piano_2_15">
252         <duration num="1" den="64"/>
253         <notehead>
254          <pitch step="C" octave="5"/>
255         </notehead>
256        </chord>
257        <chord event_ref="piano_2_16">
258         <duration num="1" den="64"/>
259         <notehead>
260          <pitch step="B" octave="4" actual_accidental="flat"/>
261         </notehead>
262        </chord>
263        <chord event_ref="piano_2_17">
264         <duration num="1" den="128"/>
265         <notehead>
266          <pitch step="A" octave="4" actual_accidental="flat"/>
267         </notehead>
268        </chord>
269        <chord event_ref="piano_2_18">
270         <duration num="1" den="128"/>
271         <notehead>
272          <pitch step="G" octave="4"/>
273         </notehead>
274        </chord>
275        <chord event_ref="piano_2_19">
276         <duration num="1" den="128"/>
277         <notehead>
278          <pitch step="F" octave="4" actual_accidental="natural"/>
279         </notehead>
```

```
280       </chord>
281       <chord event_ref="piano_2_20">
282        <duration num="1" den="128"/>
283        <notehead>
284         <pitch step="E" octave="4" actual_accidental="flat"/>
285        </notehead>
286       </chord>
287       <chord event_ref="piano_2_21">
288        <duration num="1" den="128"/>
289        <notehead>
290         <pitch step="D" octave="4"/>
291        </notehead>
292       </chord>
293       <chord event_ref="piano_2_22">
294        <duration num="1" den="128"/>
295        <notehead>
296         <pitch step="F" octave="4"/>
297        </notehead>
298       </chord>
299       <chord event_ref="piano_2_23">
300        <duration num="1" den="128"/>
301        <notehead>
302         <pitch step="A" octave="4" actual_accidental="flat"/>
303        </notehead>
304       </chord>
305       <chord event_ref="piano_2_24">
306        <duration num="1" den="128"/>
307        <notehead>
308         <pitch step="F" octave="4"/>
309        </notehead>
310       </chord>
311       <chord event_ref="piano_2_25">
312        <duration num="1" den="128"/>
313        <notehead>
314         <pitch step="D" octave="4"/>
315        </notehead>
316       </chord>
317      </voice>
318     </measure>
319    </part>
```

```
320     <horizontal_symbols>
321      <slur start_event_ref="piano_1_1" end_event_ref="piano_1_4"/>
322      <slur start_event_ref="piano_2_1" end_event_ref="piano_1_4"/>
323      <slur start_event_ref="piano_2_5" end_event_ref="piano_2_6"/>
324      <slur start_event_ref="piano_2_6" end_event_ref="piano_2_6"/>
325     </horizontal_symbols>
326    </los>
327   </logic>
328  </ieee1599>
```

Listing B.2: risultato della conversione

Bibliografia

[1] Craig Stuart Sapp. Online database of scores in the humdrum file format. In *ISMIR*, pages 664–665, 2005.

[2] Adriano Baratè, Goffredo Haus, and Luca Andrea Ludovico. State of the art and perspectives in multi-layer formats for music representation. In *Proceedings of the 2019 International Workshop on Multilayer Music Representation and Processing (MMRP 2019)*, pages 27–34. IEEE CPS, 2019.

[3] David Huron. Music information processing using the humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[4] David Huron. Humdrum and kern: Selective feature encoding. In *Beyond MIDI*, pages 375–401. MIT Press, 1997.

[5] David Huron. Humdrum user's guide. *Online manuscript, Ohio State University*, 1999.

[6] Denis L Baggi and Goffredo Haus. Ieee 1599: Music encoding and interaction. *Ieee Computer*, 42(3):84–87, 2009.

[7] Center for Computer Assisted Research in the Humanities at Stanford University. Kernscore.

Progetto sviluppato presso il Laboratorio di Informatica Musicale
`https://www.lim.di.unimi.it`