UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Corso di Laurea Triennale in
Scienze e Tecnologie della Comunicazione Musicale

PROTOTIPO WEB PER LA MAPPATURA AUDIO DI DOCUMENTI IEEE 1599

Relatore: Prof. Luca Andrea Ludovico
Correlatore: Dott. Stefano Baldan

Autore: Daniele Crepaldi
matr. 678844

Anno Accademico 2011/2012

Indice

Capitolo 1

Introduzione

1.1 Descrizione multistrato dell'informazione musicale

Come può essere analizzata e descritta l'informazione contenuta in un brano musicale? Si possono distinguere diversi livelli di astrazione. È possibile ricavare alcune informazioni di carattere *generale* (ad esempio: titolo, autore, durata), si può descrivere la successione *logica* degli eventi musicali (note, pause, tonalità), e si può analizzare gli aspetti *strutturali* (strofe, ritornelli, motivi). Gli eventi musicali possono inoltre avere diverse rappresentazioni grafiche *notazionali* (si pensi anche solo alle partiture manoscritte od a quelle stampate). Il brano in questione potrebbe essere suonato da musicisti, dove la loro *performance*, ovvero le azioni meccaniche eseguite sugli strumenti, potrebbero essere "catturate", ed infine il risultato di questa esecuzione potrebbe essere "registrato" sotto forma di *audio* o video. Questi differenti aspetti sono fortemente correlati tra di loro, e proprio le relazioni che intercorrono tra di essi, permettono un'analisi ed una rappresentazione completa dell'informazione presente nell'oggetto musicale.

Tuttavia, per quanto negli ultimi decenni siano state sviluppate tecnologie e metodi di codifica per ciascuno degli aspetti sopra citati, non esiste ancora uno standard riconosciuto e condiviso a livello internazionale che ne permetta l'unificazione in una descrizione integrata e definita. Per sopperire a questa mancanza, IEEE [W01] ad inizio anni 2000, aprì la proposta di progetto n°1599 intitolata *Recommended Practice for Definition of a Commonly Acceptable Musical Application Using the XML Language*. XML [W02], a cui IEEE fa esplicitamente riferimento nella proposta, è un linguaggio di marcatura estensibile, ovvero basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi creati. Una peculiarità di questo linguaggio è la struttura gerarchica ed annidata, che ben si conforma alla natura multistrato dell'informazione musicale.

1.2 Il progetto MX

Alla luce di questa "proposta" da parte di IEEE, il Laboratorio di Informatica Musicale dell'Università degli Studi di Milano (LIM) [W03] ha avviato il progetto MX. Lo scopo di questo progetto è la realizzazione di una codifica basata su XML che possa:

> *[...] descrivere, collegare e sincronizzare l'informazione musicale in un ambiente multistrato. L'obiettivo è il raggiungimento dell'integrazione tra i livelli di rappresentazione simbolico, strutturale, notazionale, performance e audio. Inoltre, lo standard proposto deve integrare la rappresentazione musicale con gli standard comuni già definiti e accettati.* [B01]

Descrizione, collegamento e sincronizzazione sono attuati per mezzo dei sei aspetti visti in precedenza, ossia: *general*, *structural*, *logic*, *notational*, *performance*

e *audio*. Questi differenti strati (denominati *layer*) rappresentano i diversi livelli di astrazione dell'informazione presente all'interno dell'oggetto musicale. Per quanto riguarda l'integrazione con gli standard già esistenti si deve osservare che non tutti i livelli racchiudono informazione musicale vera e

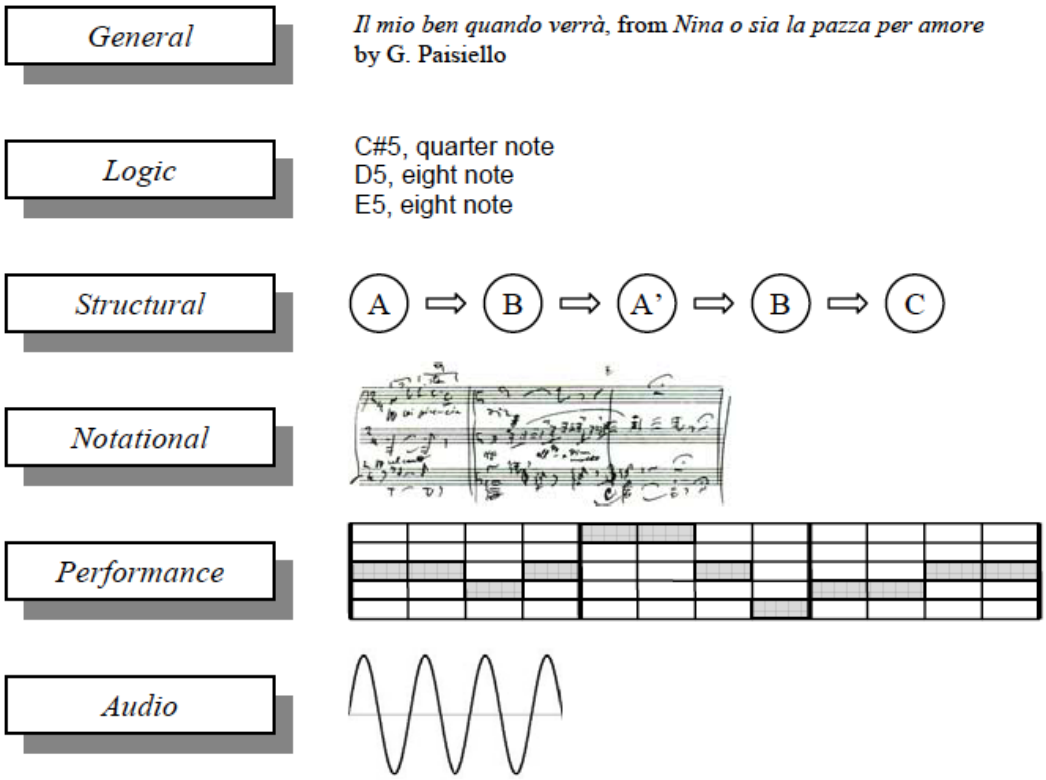

Figura 1.1: I diversi livelli dell'informazione musicale

propria: *general*, *structural* e *logic* codificano i contenuti di loro pertinenza, mentre *notational*, *performance* e *audio* includono riferimenti tra l'informazione presente all'interno della codifica ed i file multimediali esterni. La specifica non crea quindi nuovi formati per codificare l'informazione (nuovi formati audio, nuovi formati grafici notazionali o nuovi formati per la descrizione delle performance), ma lascia gli oggetti multimediali nel proprio formato originale, all'esterno del documento XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ieee1599 SYSTEM "http://www.mx.dico.unimi.it/ieee1599.dtd">
<ieee1599>
      <general>...</general>
      <logic>...</logic>
      <structural>...</structural>
      <notational>...</notational>
      <perfomance>...</performance>
      <audio>...</audio>
</ieee1599>
```

Figura 1.2: Struttura di un documento XML IEEE 1599

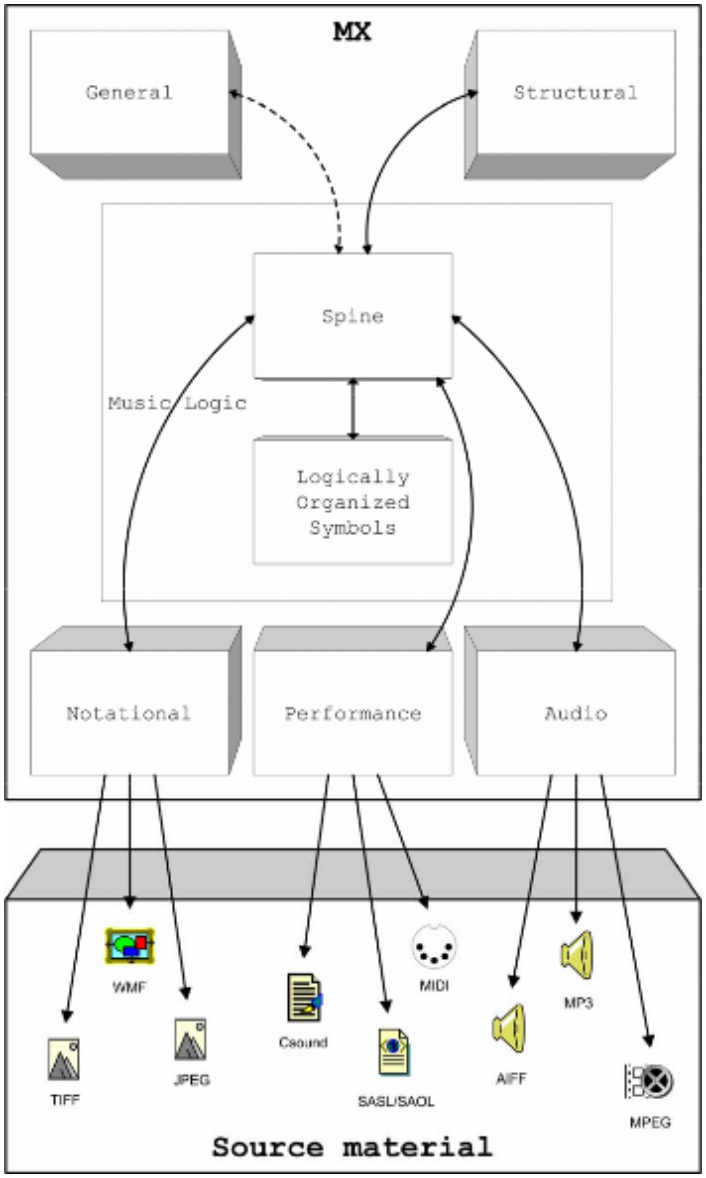

Figura 1.3: Struttura gerarchica di IEEE 1599

Verranno ora descritte con maggiore dettaglio le parti della codifica più rilevanti allo scopo del progetto in esame: *layer audio* e *spine*.

1.2.1 Lo spine

Il componente forse più importante e rivoluzionario della specifica si trova all'interno del layer *logic* ed è costituito da un elemento denominato *spine*. Quest'ultimo è la vera e propria "spina dorsale" della codifica in quanto permette la sincronizzazione globale di tutti i livelli rappresentando un comune punto di contatto tra di essi. Lo *spine* è costituito da una sequenza di eventi musicali ordinati ed etichettati, a cui gli elementi degli altri layer possono far riferimento. Ogni evento, quindi ogni tag `event` presente all'interno dello spine, contiene tre attributi. Attributo `id`: identificativo univoco, viene usato per far corrispondere ad un unico evento musicale tutte le sue istanze (suono, simbolo in partitura, ecc.). Attributo `timing`: rappresenta la collocazione temporale relativa all'evento precedente; il suo valore è espresso sotto forma di VTU (unità di tempo virtuali). Le VTU codificano il tempo in maniera relativa, sta all'utente scegliere di volta in volta che valore assoluto assegnare a queste unità per accelerare o rallentare l'esecuzione dei contenuti multimediali all'interno del documento. Attributo `hpos`: definisce i rapporti di distanza spaziale relativa tra i vari eventi musicali (ad esempio la distanza tra una nota e l'altra sullo spartito). Lo *spine* contiene dunque soltanto la mappatura spazio-temporale degli eventi, ma non fornisce di per se alcuna informazione aggiuntiva riguardo ad essi.

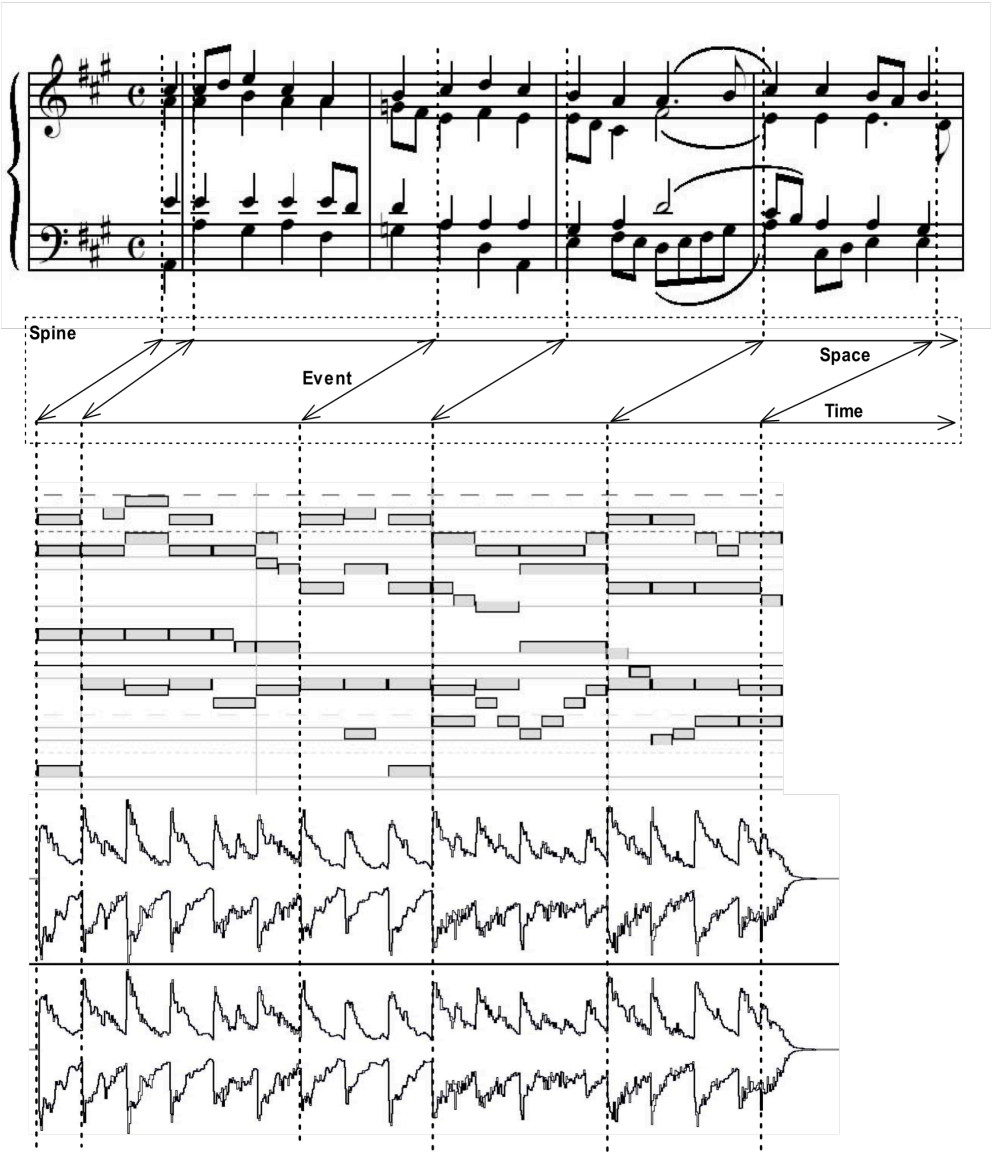

Figura 1.4: Lo spine come elemento di sincronizzazione

1.2.2 Il layer audio

Per quanto il nome possa supporne l'utilizzo unicamente legato all'audio, questo layer, facoltativo nelle specifiche della codifica ma essenziale per lo scopo di questo elaborato, non si occupa solamente di sincronizzare contenuti audio relativi ad un determinato brano ma tecnicamente anche anche quelli video possono essere gestiti in modo analogo. Agli eventi inclusi nello *spine* vengono associati (in gergo "mappati") i vari istanti in cui essi si presentano

all'interno dei diversi file audio/video. Le relazioni tra i vari flussi multimediali create in questo modo possono poi essere utilizzate per gestire la riproduzione rispetto al tempo "logico" del brano (numero di battuta, note suonate, ecc.) anziché rispetto al tempo "fisico" (minuti e secondi). Ogni traccia audio/video è rappresentata dall'elemento `track`, i cui attributi `file_name`, `file_format` e `file_encoding` riportano informazioni sull'URI della risorsa e sulla sua codifica. All'interno di ciascuna `track` compaiono gli elementi `track_event`, i cui attributi `event_ref` e `start_time` contengono rispettivamente l'identificativo univoco dell'evento associato nello *spine* e l'istante di occorrenza di questo evento all'interno della traccia audio/video, generalmente espresso in secondi.

1.3 Mappatura di un documento IEEE 1599 via web

Nel 2008, il formato prima noto come MX, dopo circa sei anni di intenso sviluppo, è stato ufficialmente promosso a standard internazionale e da allora è conosciuto come IEEE 1599. Trattandosi di un formato aperto è possibile consultare il DTD della specifica [W04] ed usarlo per scopi di validazione. Per informazioni più dettagliate si può inoltre fare riferimento alla documentazione ufficiale [B02] [B03] oppure [W05].

Negli ultimi anni IEEE 1599 ha trovato utilizzo in alcune soluzioni software impiegate prevalentemente in ambiente desktop, in particolare modo in applicazioni di carattere dimostrativo e nella suite IEEE 1599 Framework sviluppata e quotidianamente usata presso il LIM. Grazie a quest'ultimo applicativo è infatti possibile (tra le varie funzionalità che offre) eseguire in modo semi-automatico la mappatura audio/video di un documento conforme allo standard. L'utilizzo online del formato è in continua fase di studio e

sperimentazione ed è stato solo recentemente implementato con successo grazie alla realizzazione di una applicazione di streaming di contenuti multimediali basati giustappunto su tecnologia IEEE 1599 [B04]. Proprio in quest'ottica di studio e sperimentazione, il lavoro descritto in questo elaborato si pone come obiettivo la progettazione e realizzazione di un prototipo web per la mappatura audio di documenti IEEE 1599.

1.4 HTML5

Il linguaggio più importante del web arriva alla sua quinta versione portando con se numerose novità, sia dal punto di vista della descrizione semantica dei contenuti presenti in una pagina web sia per le nuove possibilità di sviluppo in campo multimediale [B05]. Fino pochi anni fa era infatti prerogativa delle soluzioni proprietarie quella di permettere agli utenti la fruizione di contenuti multimediali online tramite l'utilizzo di appositi plugin (Adobe Flash il più diffuso), con tutte le problematiche ad essi legate. Oggigiorno grazie ad HTML5, che pur essendo ancora ufficialmente un "draft", a livello pratico è ben supportato dalla maggiorparte dei principali browser ed approvato da enti come il W3C [W06] e il WHATWG [W07], lo sviluppo di un'applicazione web multimediale può contare su tecnologie innovative e fornire un'esperienza interattiva che fino poco tempo fa poteva sembrare impensabile.

1.4.1 Le API audio

Tra gli elementi multimediali introdotti con HTML5 troviamo l'elemento `<audio>`. Grazie al suo utilizzo è possibile aggiungere contenuto audio ad una pagina web semplicemente includendo all'interno del suo tag l'attributo `src`,

specificando in esso l'URL della risorsa da riprodurre. Quando si parla di `<audio>` è opportuno evidenziare che molte delle sue caratteristiche sono condivise anche dall'elemento `<video>`. Le funzionalità di questi due elementi possono essere notevolmente ampliate per mezzo di apposite API per JavaScript, che in particolare forniscono la possibilità di controlli avanzati sia del flusso multimediale e sia sul monitoraggio del tempo di riproduzione. Proprio quest'ultima funzionalità si è rilevata molto utile al fine dello scopo del progetto, ossia eseguire la mappatura audio di un documento IEEE 1599 via web.

Capitolo 2

Definizione del progetto

Dopo una fase introduttiva dove sono state presentate le tecnologie ed illustrati gli obbiettivi alla base dell'applicazione, si passerà ora a prendere in rassegna gli aspetti progettuali legati ad essa.

2.1 Analisi degli obiettivi

Lo scopo principale, come detto in precedenza, è la progettazione e la realizzazione di un prototipo web per la mappatura audio di documenti IEEE 1599. L'applicazione web dovrà includere granparte delle caratteristiche contenute nell'applicativo Audio Mapper 3.1 presente nella suite IEEE 1599 Framework ed avere un comportamento del tutto analogo. Dovranno essere disponibili in particolare le seguenti funzionalità:

- Possibilità di importare all'interno dell'applicazione un documento XML IEEE 1599 con cui si vuole eseguire la mappatura.

- Possibilità di importare nell'applicazione un file audio con il quale si vuole operare.

- Assegnazione del valore di VTU da parte dell'utente nel caso questo non fosse già stato appositamente dichiarato in precedenza nel documento.

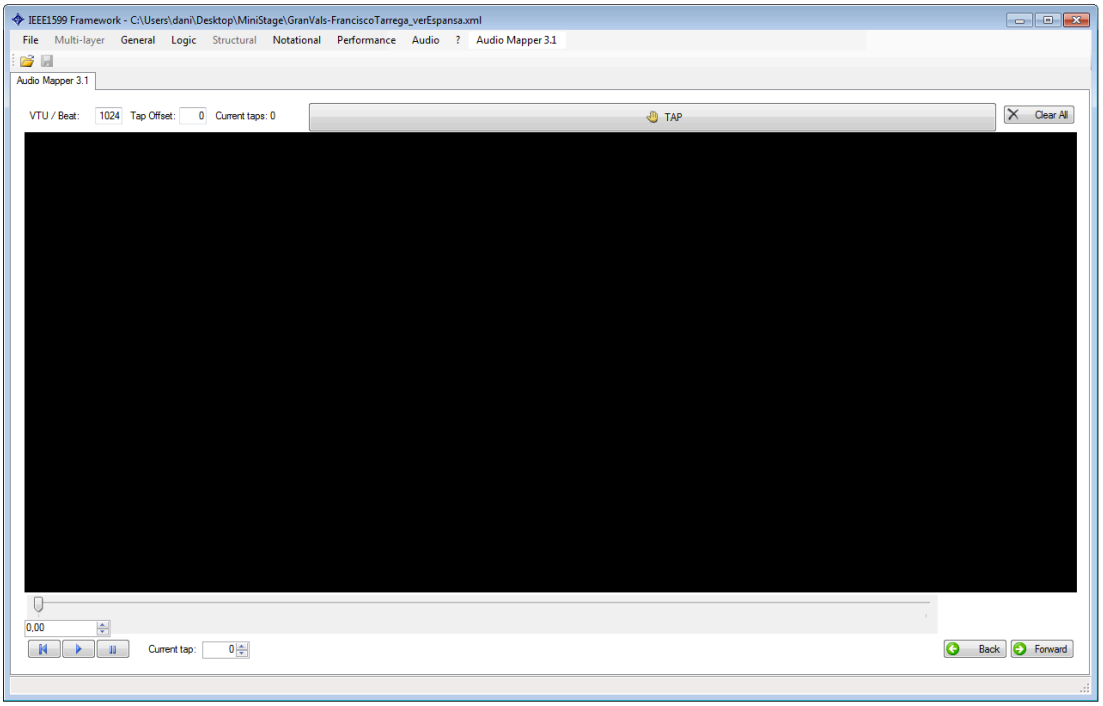

Figura 2.1: Applicativo Audio Mapper 3.1 della suite IEEE 1599 Framework di cui l'applicazione web dovrà cercare di riprodurne le funzionalità principali.

- Controllo della riproduzione con un player che supporti le funzioni base: play, pausa, indicazione tempo di playback. Un comando di stop potrà essere incluso nella funzione più generica reset ed implementato con rispettivo pulsante.

- Utilizzo della barra spaziatrice della tastiera del computer come interfaccia fisica per eseguire l'azione di mappatura. Presenza di un feedback visivo indicante il numero di pressioni (tap).

- Possibilità di scaricare il documento mappato (ed i rispettivi file audio utilizzati) una volta terminato il processo.

Il prototipo dovrà inoltre possedere le seguenti caratteristiche:

1. Funzionamento "out-of-the-box", ovvero senza la necessità dell'installazione di plugin esterni da parte dell'utente.

2. Universalità nella mappatura di documenti IEEE 1599, estesa per lo meno a tutti quelli prodotti fino ad oggi presso il LIM.

3. Compatibilità verso tutte le principali versioni desktop dei browser HTML5 in circolazione: Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Internet Explorer, Opera.

2.2 Scelta della piattaforma di sviluppo

La scelta della piattaforma di sviluppo è stata praticamente da subito indirizzata verso l'utilizzo di HTML5 e JavaScript. Oltre a rispettare i requisiti posti in precedenza, uno dei motivi principali che ha fatto propendere per questa soluzione è dato dalla possibilità di sfruttare l'eccellente supporto delle tecniche di sviluppo AJAX nella manipolazione di documenti XML e dunque, di riflesso, anche nella manipolazione di documenti IEEE 1599.

Uno dei maggiori problemi da superare è riconducibile al fatto che non tutti i browser web supportano le specifiche HTML5 allo stesso modo, e sopratutto, non esiste ad oggi un formato audio riconosciuto come standard web e supportato in egual misura da tutti i produttori.

Al fine di ovviare all'eterogeneità di comportamento dei diversi browser nell'interpretare il codice JavaScript e nell'implementare le specifiche HTML5, si sono dovute affrontare sessioni periodiche di debug. Uno strumento che si è rivelato molto utile allo scopo è stato il noto plugin per sviluppatori denominato *Firebug*, disponibile come estensione per Mozilla Firefox (ed in maniera meno completa anche per Google Chrome).

2.3 Progettazione dell'infrastruttura

Dopo aver scelto la piattaforma di sviluppo si può passare a considerare la progettazione dell'infrastruttura, di per sè comunque piuttosto semplice per questa applicazione. Vediamo più nello specifico come sono stati assegnati i compiti all'interno del progetto.

2.3.1 Lato server

Il server durante l'utilizzo dell'applicazione si dovrà essenzialmente occupare di svolgere i seguenti compiti:

- Salvataggio del documento IEEE 1599 caricato dall'utente tramite l'apposita interfaccia in modo che sia successivamente reperibile per essere manipolato.

- Salvataggio del file audio caricato dall'utente e conversione nei formati supportati dai principali browser in modo che sia in seguito riproducibile e processabile.

- Impacchettamento del documento IEEE 1599 mappato ed i file audio utilizzati in un'unica cartella compressa rendendola disponibile al download da parte dell'utente.

Il server in questione potrà essere un normale web server. I compiti di salvataggio ed impacchettamento saranno svolti da script in linguaggio PHP, per quanto riguarda la conversione dei file audio verrà utilizzato (sempre tramite script in PHP) l'applicativo *ffmpeg*.

2.3.2 Lato client

Dal punto di vista del client, l'utilizzo dell'applicazione dovrà avvenire nella maniera più fluida possibile. Per garantire che ciò accada, le risorse audio caricate dall'utente e convertite dal server dovranno essere rese disponibili appena consentito. Questa accessibilità potrà avvenire compatibilmente con l'ampiezza di banda disponibile in fase di upload e dalla dimensione del file audio, che in caso di formati non compressi, può raggiungere facilmente la grandezza di decine di megabyte, comportando più tempo non soltanto per la fase di caricamento ma anche per quella di conversione.

2.4 Progettazione del software

A livello di software bisognerà progettare sostanzialmente tre componenti:

1. Un componente lato server che si occupi del salvataggio e della conversione dei file caricati dall'utente.

2. Un componente, lato client, che si occupi di analizzare il file XML acquisito (ovvero effettuare il parsing del documento), manipolarlo all'interno di un DOM, e re-inviarlo al server per poter essere salvato e successivamente scaricato dall'utente una volta che l'operazione di mappatura sia terminata.

3. Un'interfaccia grafica che renda pratico ed intuitivo all'utente il processo di mappatura di un file IEEE 1599.

Capitolo 3

Descrizione dell'applicazione

Affrontati nelle precedenti pagine gli aspetti legati alle tecnologie utilizzate e definiti gli scopi del progetto, ovvero permettere di eseguire la mappatura audio di un documento IEEE 1599 via web, in questo capitolo si cercherà di illustrare il funzionamento degli algoritmi utilizzati e la risoluzione dei problemi legati alla realizzazione dell'applicazione.

3.1 Conversione dei file audio

Una conversione dei file audio forniti in fase di upload dall'utente è consigliabile sia per ottimizzare il consumo di banda durante la riproduzione degli stessi, sia per consentirne un idoneo processamento durante il procedimento di mappatura. A differenza della codifica IEEE 1599, che supporta un grande varietà di codec e di container per l'audio, HTML5 (anche nelle sue più recenti implementazioni) non offre una piena compatibilità a tutti i formati dovendo soprattutto dipendere dalle scelte "aziendali" dei produttori

di browser che, nonostante la necessità di adottare uno standard condiviso sia sempre più sentita, stanno proseguendo imperterriti su percorsi diversi. La tabella che segue può illustrare abbastanza efficacemente la situazione.

BROWSER	ogg/vorbis	wav/pcm	mp3	aac
Google Chrome	YES (9.0+)	YES	YES	YES
Mozilla Firefox	YES (3.5+)	YES (3.5+)	NO	NO
Internet Explorer	NO	NO	YES (9.0+)	YES (9.0+)
Opera	YES (10.5+)	YES (11+)	NO	NO
Apple Safari	NO	YES (3.1+)	YES (3.1+)	YES (3.1+)

Figura 3.1: Formati audio supportati dai diversi browser web

I dati sopra presentati evidenziano come attualmente non esista una scelta predefinita nel caso ci si trovi a lavorare con file audio (e multimediali in generale) in ambiente web. Per questo motivo i formati da utilizzare per garantire la piena compatibilità con i browser in circolazione risultano essere il diffusissimo formato MP3 ed il suo corrispettivo "open" OGG (codec Vorbis). Con il primo offriremo compatibilità agli utenti che utilizzano Microsoft Internet Explorer ed Apple Safari, con il secondo a quelli che navigano con Mozilla Firefox ed Opera. Google Chrome merita una menzione a parte in quanto è l'unico browser che attualmente offre pieno supporto a tutti i formati e per questo risulta estremamente versatile.

3.2 La mappatura audio

Lo *spine* abbiamo visto essere per definizione il "collante" del formato IEEE 1599, ovvero la struttura logica che implementa l'integrazione dei layer *notational*, *performance* ed *audio* con il layer *logic*, in cui esso stesso è contenuto. Il

suo obiettivo è quindi costruire una struttura astratta cui fanno riferimento tutti gli strati che descrivono le proprietà del materiale originario.

Mappare significa creare un collegamento tra un evento fisico che si verifica in una certa istanza dell'opera ed il relativo evento logico presente nello *spine*. Nel caso specifico della mappatura audio creare, ad esempio, un collegamento tra una nota suonata nel file audio di un'esecuzione ed il corrispettivo evento logico collocato all'interno dello *spine*. In questo modo si potrà anche generare un ulteriore riferimento nel caso in cui siano presenti altre istanze dell'opera (spartiti, libretti, file MIDI, e perchè no, versioni audio differenti dello stesso brano).

L'operazione di mappatura è una procedimento in cui l'apporto della componente umana risulta tutt'oggi imprescindibile. Nonostante ci siano studi ed esperimenti in corso per essere in grado di modellare e di riconoscere efficacemente l'informazione musicale in forma sonora e grafica [B06] [B07] [B08], i risultati ottenuti sono ancora lontani da poter essere utilizzati con profitto in ambito produttivo e professionale (dovendo in primo luogo considerare che talvolta si deve lavorare con documenti manoscritti o con riproduzioni fonografiche datate e degradate) . L'applicazione oggetto di questo elaborato, analogamente alla versione desktop inclusa nella suite IEEE 1599 Framework, consentirà all'utente di mappare un documento in maniera semi-automatica, ossia non ci sarà la necessità di andare a scrivere letteralmente in linguaggio XML la codifica dell'evento, ma bensì sarà sufficiente segnalarne l'occorrenza premendo (o meglio "tappando") sulla barra spaziatrice della tastiera del computer, sarà poi l'applicativo a ricavarne l'equivalente codificato secondo i parametri della specifica. Queste informazioni verranno inserite all'interno del layer *audio* il quale si occupa della sincronizzazione relativa ai contenuti multimediali. Ogni file audio (o video) collegato, è rappresentato

dall'elemento `track`, il quale a sua volta contiene una serie di elementi `track_event` che rappresentano i riferimenti agli eventi codificati nello *spine*. Ogni `track_event` ha due attributi principali: l'identificativo univoco dell'evento corrispondente, denominato `event_ref`, e l'istante in cui l'evento accade all'interno del flusso multimediale, chiamato `start_time` (espresso solitamente in secondi). Le API audio di HTML5 offrono praticamente tutte le funzionalità necessarie per implementare un sistema di mappatura. In particolare il metodo `currentTime()` si è rivelato molto utile se non addirittura fondamentale allo scopo. Ogni volta che è stato innescato dalla pressione della barra spaziatrice, `currentTime()` ha infatti consentito di rilevare e memorizzare l'istante di tempo in cui un evento musicale si è presentato nel brano.

Bisogna ora valutare il criterio alla base della mappatura del documento. Potremmo considerare di mappare il brano scandendo i quarti di battuta sulla tastiera. Rimane il problema di capire che valore temporale attribuire ad un quarto. Come già accennato nel paragrafo 1.2.1 uno degli attributi inclusi nell'elemento *spine* è `timing`, il quale rappresenta una temporizzazione virtuale ed il cui valore viene espresso in VTU (Virtual Time Units). Esistono delle norme per scegliere l'unità di misura, che non viene fissata a priori: i VTU non hanno un valore di tempo fisico (come secondi o minuti), ma l'utente sceglie che significato temporale attribuirgli di volta in volta. Nella scelta del VTU di riferimento è consigliabile utilizzare valori grandi, divisibili per molti divisori e potenze di due per i valori più lunghi. Ad esempio, in un corale scritto sostanzialmente a quarti si può assegnare il valore 1024 al quarto. Le potenze di due consentono di rappresentare molti valori con numeri interi (1024 = semiminima, 512 = croma, 256 = semicroma, ecc.); un valore base divisibile per 3, per 5, per 7 consente di rappresentare con numeri interi le note che costituiscono terzine, quintine, settimine. In ogni caso, si tratta di una scelta del

tutto arbitraria. La valutazione si deve basare anche su considerazioni di granularità degli eventi da descrivere. Se abbiamo a che fare con un corale di Bach in cui si usano solo metà, quarti e ottavi, si può effettuare l'assegnamento: croma = 1, semiminima = 2, minima = 3. La granularità risultante è molto grezza: nessun valore intero potrebbe rappresentare ad esempio un ottavo con il punto, in quanto la sua durata sarebbe di 1,5. Il valore di VTU/quarto è riportato in un attributo denominato `vtu_amount` contenuto all'interno del tag `time_indication` a sua volta annidato nell'elemento *logic*. La dichiarazione dell'attributo `vtu_amount` nella specifica tuttavia è opzionale. Non potendoci quindi basare con assoluta certezza nella presenza di questo attributo si è scelto di offrire la possibilità all'utente di assegnare il valore di VTU per quarto (un'opzione analoga di inserimento è disponibile anche all'interno dell'applicativo Audio Mapper 3.1 della suite IEEE 1599 Framework).

Anche se ad ogni "tap" sulla barra spaziatrice della tastiera corrispondesse un quarto, bisognerebbe ad ogni modo considerare che non tutti gli eventi presenti nello *spine* si verificano con la medesima frequenza l'uno dall'altro, basti pensare solamente ad un'opera polifonica o meglio ad un accordo, dove sul pentagramma abbiamo il concretizzarsi simultaneo di più note disposte verticalmente. Nella codifica IEEE 1599 questo aspetto è evidenziato analizzando l'attributo `timing`. Nell'ipotesi che VTU/quarto sia uguale a 1024 ed il valore dell'attributo sopra citato sia `0`, significherà che l'evento in questione avviene nel medesimo istante di quello precedente e di conseguenza anche `start_time` contenuto nel tag `track_event` sarà identico; nel caso in cui il valore di `timing` sia invece 1024, un suo multiplo o un suo divisore (ad esempio: 512, 256, ecc.), significherà che gli eventi sono in successione e quindi anche `start_time` avrà un valore progressivo corrispondente al valore che in quel momento è stato acquisito grazie al metodo `currentTime()`.

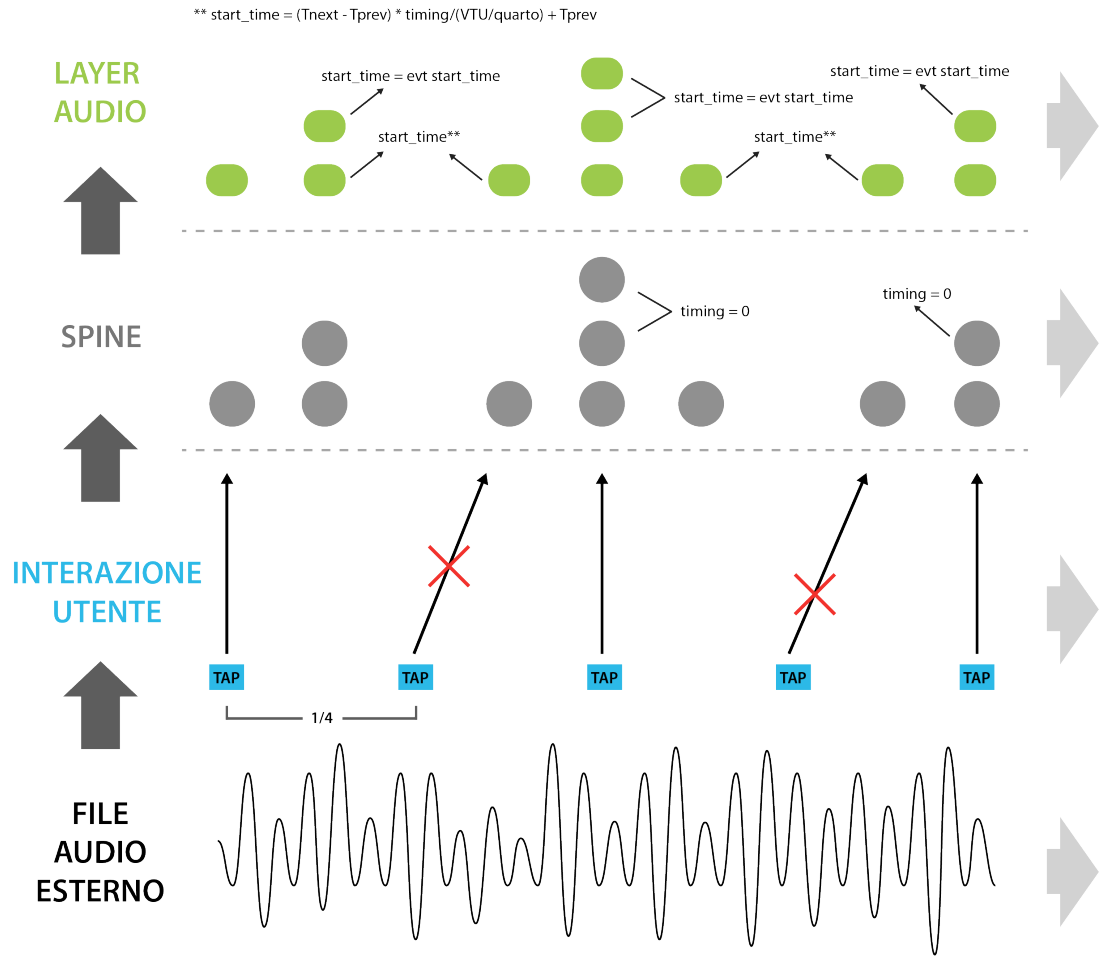

Figura 3.2: Schema semplificato del processo di mappatura audio

Lo schema sopra riportato cerca di esplicitare il processo alla base della mappatura audio effettuata dall'applicazione web. Riassumendo:

- Viene mandato in esecuzione il brano precedentemente caricato e l'utente inizia a "tenere il tempo" battendo i quarti sulla barra spaziatrice della tastiera (tap).

- Non sempre ad un tap corrisponde un evento presente nello *spine*, sarà comunque importante incrementare ad ogni "battuta" del valore di VTU/quarto (ad esempio 1024) per procedere nella progressione e mantenere una certa sincronizzazione. I tap che non coincidono ad un

evento logico non verranno comunque presi in considerazione ai fini della creazione di un corrispettivo riferimento nel *layer audio*.

- Agli eventi dello *spine* con valore dell'attributo `timing` pari a `0`, quindi simultanei all'evento che li precedeva, verrà creato un corrispettivo riferimento nel *layer audio* (tag `track_event`) con valore dell'attributo `start_time` uguale anch'esso all'elemento precedente, essendosi verificati nel medesimo istante.

- Agli eventi dello *spine* con un valore di `timing` diversi da `0`, dal valore di VTU/quarto o da un suo multiplo, verrà creato un corrispettivo riferimento nel *layer audio* (tag `track_event`) con il valore di `start_time` uguale a: (Tempo Successivo - Tempo Precedente) * [`timing`/(VTU/quarto)] + Tempo Precedente.

3.3 Ricerca delle informazioni nel documento

Come si è visto nel paragrafo 1.2 la codifica IEEE 1599 è basata sul linguaggio XML. Grazie a questa caratteristica, e grazie all'utilizzo delle tecniche AJAX, è possibile gestire i documenti in maniera funzionale, analizzandoli e generando un DOM che crea la struttura dati principale dell'applicativo. Questa struttura consentirà di ricercare informazioni e manipolare il documento secondo le proprie esigenze. Più nel dettaglio:

- Ricerca dell'elemento `<audio>`. Nel caso non esista già nel documento dovrà essere generato essendo alla base del processo di mappatura.

- Ricerca all'interno del documento di una precedente mappatura audio/video. Questo dato è ottenibile, oltre che dalla presenza dell'elemento `track`, anche grazie alla presenza o meno di un elemento `<track_indexing>`. La mappatura che si andrà ad eseguire non dovrà infatti sostituire una preesistente ma, nel caso dovesse esserci, dovrà "affiancarla" offrendosi come possibile alternativa o come informazione complementare (ad esempio due diversi strumenti musicali che eseguono lo stesso brano, registrati però su due file audio differenti).

- Ricerca dell'attributo `vtu_amount` che contiene informazioni riguardo al valore dei VTU assegnati per quarto, questo elemento non è comunque ritenuto obbligatorio nella specifica.

- Ricerca degli elementi `event` presenti nello *spine* ed in particolare degli attributi `id` e `timing`, essenziali alla fine del rilevamento di eventi simultanei e della sincronizzazione.

3.4 Interfaccia utente

Al fine di rendere l'applicazione fruibile per mezzo di browser web è stata progetta e implementata un'intefaccia grafica utilizzando come linguaggi di codifica HTML5, CSS3 e JavaScript sfruttando la famosa libreria jQuery. L'interfaccia riproduce il modello concettuale di funzionamento presente nell'applicativo Audio Mapper 3.1, permette così agli utenti che già utilizzano il programma della suite IEEE 1599 Framework di usare con profitto da subito anche la versione web. L'applicazione si sviluppa su cinque schermate:

1. Schermata introduttiva: come mappare un documento IEEE 1599.

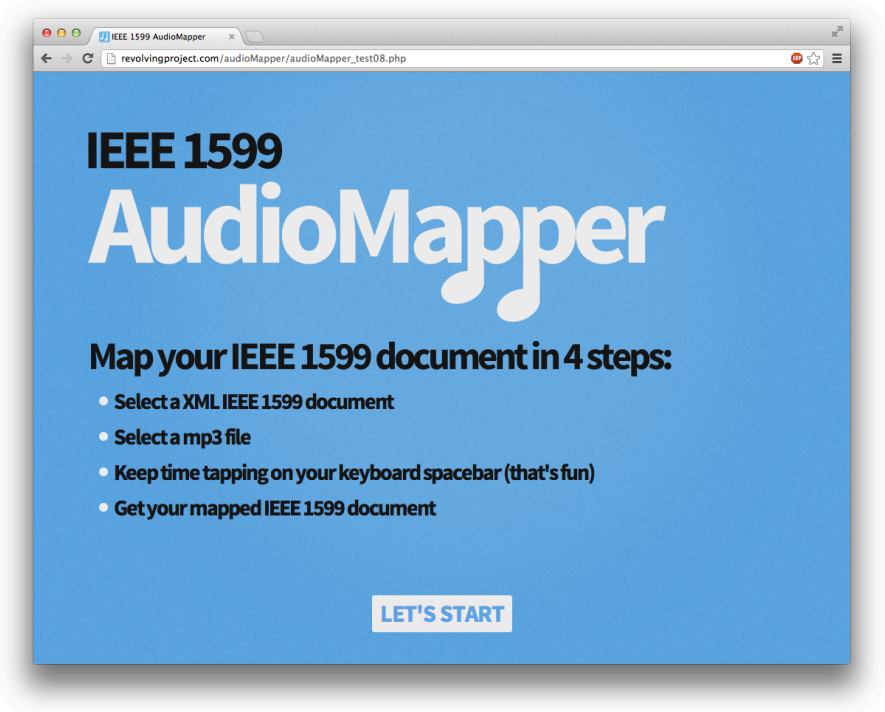

Figura 3.3: Schermata introduttiva dell'applicazione web.

2. Schermata che consente di selezionare (e caricare) dal proprio computer il documento IEEE 1599 su cui vuole essere eseguita la mappatura.

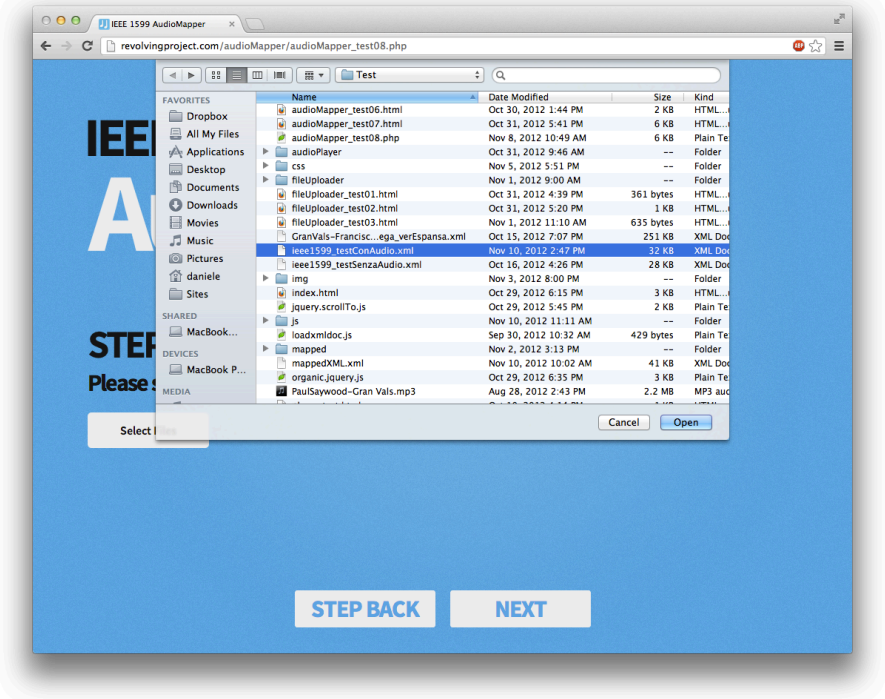

Figura 3.4: Schermata n°2 - Selezione ed upload del documento IEEE 1599

3. Schermata che permette di eseguire l'upload del file audio con il quale vuole essere effettuata la mappatura.

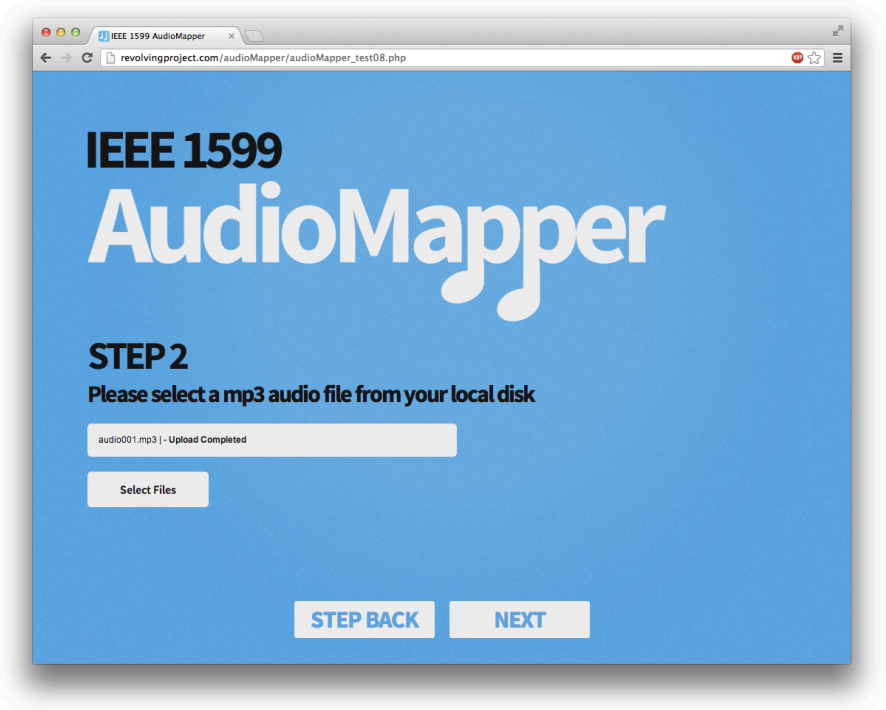

Figura 3.5: Schermata n°3 dell'applicazione - Upload del file audio

4. Schermata in cui viene effettivamente svolto il processo di mappatura ed in cui si concretizza l'interazione con l'utente. La si può considerare quindi come il fulcro dell'intera applicazione. In questa schermata grazie ad un player dalle funzionalità basilari è possibile mandare in esecuzione il file audio precedentemente caricato e per mezzo della barra spaziatrice tenere il tempo marcando, ad esempio, i quarti di battuta. Il feedback visivo all'utente verrà dato dal contatore presente sotto il player che aggiornerà di volta in volta il numero di tap. Nel caso non si fosse soddisfatti della propria prestazione è disponibile un pulsante "reset" che riporta la fase di mappatura alla situazione originaria, azzerando i tap battuti fino a quel momento e riportando il tempo di riproduzione del brano all'inizio.

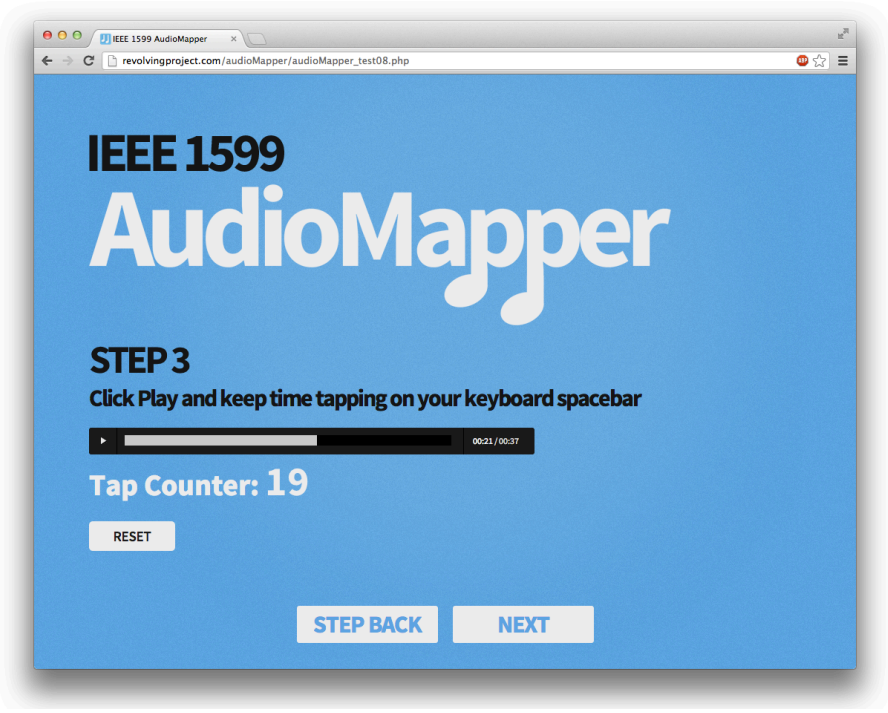

Figura 3.6: Schermata n°4 - Viene messo in riproduzione il brano ed eseguita la mappatura

5. Schermata finale. Una volta terminato il processo di mappatura è possibile scaricare il documento IEEE 1599 mappato ed i file audio.

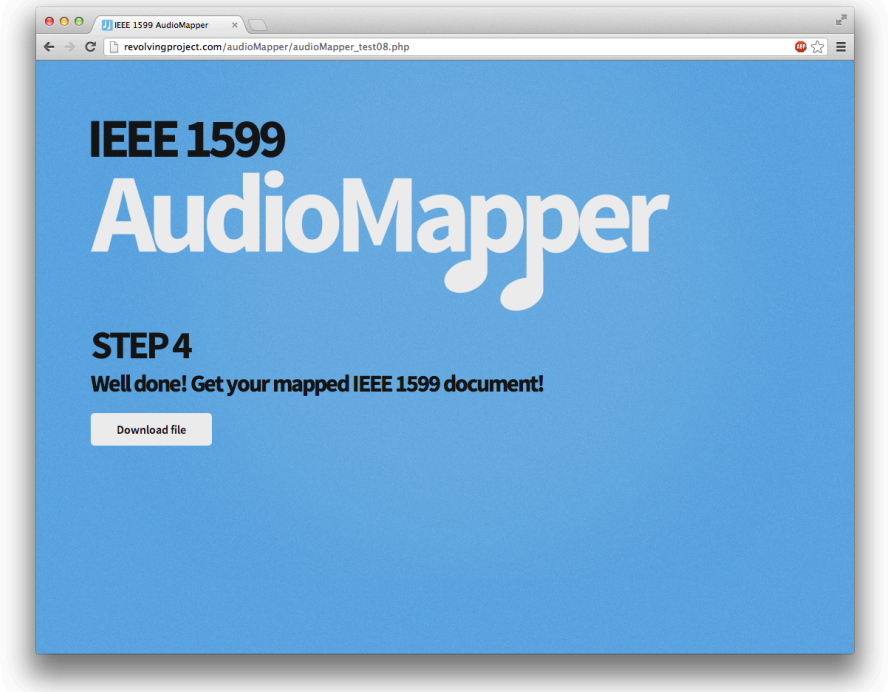

Figura 3.7: Schermata n°5 - Download documento IEEE 1599 mappato e file audio

Capitolo 4

Sviluppo del software

Dopo aver analizzato tecnologie, obbiettivi e processi che stanno alla base della applicazione web, si andrà ora ad esaminare e commentare le parti di codice più significative, al fine di chiarire il funzionamento e la realizzazione pratica dell'applicativo. Come già detto le funzionalità principali saranno sviluppate con HTML5 e JavaScript, utilizzando tecniche AJAX.

4.1 Inizializzazione: `initVtu()`, `getTracksNumber()` e `initAudio()`

Queste funzioni predispongono le condizioni appropriate necessarie al fine della mappatura audio vera e propria. Come le altre parti di codice descritte nelle sezioni a seguire, sono contenute all'interno del file audioMapper.js.

La prima funzione, `initVtu()`, verifica se all'interno del documento sia già stato assegnato il valore dei VTU/quarto.

```
function initVtu() {
  var fileXMLUrl = document.getElementById('XMLFileName').innerHTML;
  var XMLUrl = 'uploads_xml/' + fileXMLUrl;

  xhttp = new XMLHttpRequest();
  xhttp.open('GET', XMLUrl, false);
  xhttp.send();
  xmlDoc = xhttp.responseXML;

  var timeIndications =
  ➥ xmlDoc.getElementsByTagName('time_indication');
  var vtuAmount = timeIndications[0].getAttribute('vtu_amount');

  if (vtuAmount == undefined) {
    var unitVtuVal =
    ➥ prompt('Please insert a valid VTU value', '1024');
    unitVtu = parseInt(unitVtuVal);
  }
  else {
    var unitVtuVal = vtuAmount;
    unitVtu = parseInt(unitVtuVal);
  }
  return unitVtu;
}

nextXML.addEventListener('click', initVtu, false);
```

Ad inizio funzione viene effettuata una chiamata AJAX e quindi caricato il documento IEEE 1599 in un DOM. Si ricava l'elemento `time_indication` e si verifica se al suo interno sia presente o meno l'attributo `vtu_amount` il quale, come detto in precedenza, dovrebbe contenere il valore dei VTU per quarto. Nel caso l'attributo non fosse stato dichiarato viene chiesto all'utente, tramite una funzione di `prompt`, di assegnare un valore a sua discrezione per poter effettuare la mappatura.

`getTracksNumber()` svolge il compito di controllare se nel documento siano già presenti o meno mappature precedenti. La funzione viene attivata dall'evento "click" in seguito alla pressione del pulsante "next" presente nella schermata di upload the file XML, quindi mentre si passa ad uno "step" successivo (dall'upload del file xml all'upload del file audio).

```
var trackIndexingCounter = 0;

function getTracksNumber() {

  var fileXMLUrl = document.getElementById('XMLFileName').innerHTML;
  var XMLUrl = 'uploads_xml/' + fileXMLUrl;

  xhttp = new XMLHttpRequest();
  xhttp.open('GET', XMLUrl, false);
  xhttp.send();
  xmlDoc = xhttp.responseXML;

  if (xmlDoc.getElementsByTagName('audio').length != 0) {
    var tracksIndexing =
    ➥ xmlDoc.getElementsByTagName('track_indexing');
    trackIndexingCounter = tracksIndexing.length;
  }
  return trackIndexingCounter;
}
```

Viene innanzitutto dichiarata la variabile globale `trackIndexingCounter` ed assegnatole il valore di `0`. Questa variabile dovrà svolgere il ruolo di contatore, ovvero memorizzare il numero di elementi `track_indexing` contenuti nel documento (che denotano la presenza di mappature audio/video preesistenti). Verrà successivamente recuperato lo URL del file XML per mezzo delle variabili locali `fileXMLUrl` e `XMLUrl`. Lo URL servirà alla successiva chiamata AJAX per ottenere il documento IEEE 1599 e renderlo disponibile in un DOM. Dopodiché si verificherà la presenza o meno del tag `audio`. Nel caso fosse presente si

provvederà ad ottenere una lista degli elementi `track_indexing` inclusi ed il numero di questi andrà ad aggiornare il valore della variabile `trackIndexingCounter`. Si è scelto ti controllare il numero di elementi `track_indexing` e non solamente la presenza o meno di elementi `track` perchè `track_indexing` risulta essere l'elemento genitore di `track_event` ed averlo già a disposizone tornerà utile in futuro quando si andrà a manipolare ulteriormente il DOM.

La funzione `initAudio()` ha il compito principale di rendere disponibile alla riproduzione il file audio appena caricato dall'utente. Viene innescata dall'evento "click" collegato al pulsante "next" presente nella terza schermata. Ecco il codice:

```
function initAudio() {
  var fileAudioUrl =
  ➥ document.getElementById('AudioFileName').innerHTML;
  var audioUrl = 'uploads_audio/' + fileAudioUrl;
  var audioSource = document.getElementById('audioSource');
  audioSource.src = audioUrl;
  var fileXMLUrl = document.getElementById('XMLFileName').innerHTML;
  var XMLUrl = 'uploads_xml/' + fileXMLUrl;
  var fileAudioUrl =
  ➥ document.getElementById('AudioFileName').innerHTML;
  var XMLAudioFileUrl = 'audio/' + fileAudioUrl;
  xhttp = new XMLHttpRequest();
  xhttp.open('GET', XMLUrl, false);
  xhttp.send();
  xmlDoc = xhttp.responseXML;

  if (xmlDoc.getElementsByTagName('audio').length == 0) {
    var audioEl = xmlDoc.createElement('audio');
    var ieee1599El = xmlDoc.getElementsByTagName('ieee1599')[0];
    ieee1599El.appendChild(audioEl);
  }
```

```
  var trackEl = xmlDoc.createElement('track');
  trackEl.setAttribute('file_name', XMLAudioFileUrl);
  trackEl.setAttribute('file_format', 'audio_mp3');
  trackEl.setAttribute('encoding_format', 'audio_mp3');
  audioEl = xmlDoc.getElementsByTagName('audio')[0];
  audioEl.appendChild(trackEl);
  var trackIndexingEl = xmlDoc.createElement('track_indexing');
  trackIndexingEl.setAttribute('timing_type', 'seconds');
  trackEl.appendChild(trackIndexingEl);

  myAudio.load();

  return xmlDoc;
}

nextAudio.addEventListener('click', initAudio, false);
```

Nella funzione `initAudio()` viene prima di tutto recuperato lo URL del file audio uploadato ed aggiornato di conseguenza l'attributo `src` dell'elemento audio HTML5 in modo da consentirne la corretta localizzazione e la conseguente riproduzione nel player. Viene caricato in un DOM il file IEEE 1599 per mezzo di una chiamata AJAX come già avvenuto nella funzione precedente. Una volta che il documento è disponibile per essere manipolato si inizia a creare la struttura dove in un secondo tempo, per mezzo del processo di mappatura vero e proprio, verranno aggiunti i vari elementi `track_event`. Si effettua l'operazione di controllo per verificare l'esistenza dell'elemento `audio`, nel caso non sia presente verrà creato. Successivamente verranno inseriti gli elementi facenti parte della specifica IEEE 1599 relativi al *layer audio*, in particolare: elemento `track` con i rispettivi attributi `file_name`, `file_format`, `encoding_format` e l'elemento `track_indexing` con al suo interno l'attributo `timing_type`. Come ultima operazione verrà caricata la risorsa audio.

4.2 Conversione file audio: `convertAudio()`

Per quanto svolga un lavoro "dietro le quinte" la funzione `convertAudio()` ha un compito fondamentale: consentire l'idoneo processamento del file audio bypassando le differenze di formati supportati dai diversi browser web. Inoltre come pregio aggiuntivo permette di ottimizzare il consumo di banda in fase di playback. La funzione `convertAudio()` viene innescata dall'evento "click" generato dal tasto "next" presente nella schermata di caricamento del file audio (come per la funzione `initAudio()`).

```
function convertAudio() {
  $.ajax({
    type: "POST",
    url: "convertAudio.php",
  });
}

nextAudio.addEventListener('click', convertAudio, false);
```

4.3 Mappatura audio: `mapAudio()`

La funzione `mapAudio()` può essere considerata come il cuore dell'applicazione web. È dove viene eseguita effettivamente la mappatura, ovvero dove viene generato un collegamento tra un evento che si verifica nel file audio ed il corrispettivo evento logico presente nello *spine*. La funzione `mapAudio()` viene eseguita ad ogni pressione della barra spaziatrice della tastiera. Questo il codice che la innesca:

```
var tap = 0;
```

```
$(document).bind('keyup', function(evt) {
  if (evt.keyCode == 32) {
    tap++;
    $('#counter').text(tap);
    mapAudio ();
  }
});
```

Come si nota da queste linee, ad ogni pressione della barra spaziatrice, evento identificabile con la proprietà `keyCode == 32`, sarà incrementato il contatore definito per mezzo della variabile `tap` (a cui è stato attribuito un valore iniziale di `0`) e verrà eseguita la funzione `mapAudio()`.

Si prenderà ora in esame il codice inerente alla funzione `mapAudio()`.

```
var spineVtu = 0;
var tapVtu = 0;
var eventIndex = 0;
var eventIndexPrev = 0;
var eventIndexUnc = [];
var listUncTimingEvts = [];

function mapAudio() {
  var myAudio = document.getElementById('myAudio');
  var audioTime = myAudio.currentTime.toFixed(2);
  var events = xmlDoc.getElementsByTagName('event');
  var trackIndexingEl =
  ➥ xmlDoc.getElementsByTagName('track_indexing')
  ➥ [trackIndexingCounter];
```

Vengono inizializzate le variabili globali che permetterano di memorizzare i valori che di volta in volta assumeranno gli attributi `timing` degli `event` ed il loro corrispettivo indice per poter scorrere e rimanere sincronizzati agli eventi dello `spine`. All'interno di `mapAudio()` si procederà a recuperare l'elemento HTML5 `audio` oppurtunamente identificato e grazie al metodo `currentTime()`

si andrà a ricavare il valore di tempo relativo all'istante di riproduzione nel momento in cui la barra spaziatrice è stata premuta. Verranno inoltre create le variabili `events`, la quale conterrà una lista di tutti gli eventi dello `spine`, e `trackIndexingEl` che consentirà un corretto inserimento dei `track_event` in caso di mappature preesistenti. Procedendo con il codice:

```
if (spineVtu == tapVtu) {
  var conNextStartTime = myAudio.currentTime.toFixed(2);
  var evtsTiming = events[eventIndex].getAttribute('timing');

  if ((evtsTiming % unitVtu) == 0) {
    var trackEventEl = xmlDoc.createElement('track_event');
    trackEventEl.setAttribute('event_ref',
    ➥ events[eventIndex].getAttribute('id'));
    trackEventEl.setAttribute('start_time', audioTime);
    trackIndexingEl.appendChild(trackEventEl);
    eventIndex++;
  }
```

Si esegue un primo controllo per verificare che il valore dell'`event` preso in considerazione sia effettivamente quello a cui si è arrivati con il numero di tap (ad esempio: considerando un `event` con `timing` 2048 significherebbe che per essere arrivati ad esso si dovrà aver battuto due tap sulla tastiera nel caso in cui si fosse deciso di marcare i quarti con valore di VTU 1024). Nell'eventualità in cui questa condizione non fosse verificata il documento non verrà manipolato e si procederà con l'analisi degli eventi seguenti. In caso positivo si controllerà che il `timing` dell'evento in questione sia quindi un multiplo di `unitVtu` (cioè il valore assegnato ai VTU per quarto) o uguale a `unitVtu`, se la condizione sarà soddisfatta si inserirà all'interno dell'elemento `audio` IEEE 1599 un elemento `track_event` con valore di `start_time` pari a quello recuperato in precedenza grazie a `.currentTime`. Infine si incrementerà l'indice degli eventi per avanzare con la mappatura.

```
var vtuTot = 0;

for (var y = 0; y < listUncTimingEvts.length; y++) {
  vtuTot +=
  ➥ parseFloat(events[eventIndexUnc[y]]
  ➥ .getAttribute('timing'));
}

vtuTot += parseFloat(events[eventIndex-1].getAttribute('timing'));

for (var y = 0; y < listUncTimingEvts.length; y++) {
  var trackEventEl = xmlDoc.createElement('track_event');
  trackEventEl.setAttribute('event_ref', listUncTimingEvts[y]);
  var trackEvents =
  ➥ trackIndexingEl.getElementsByTagName('track_event');
  var referTrackEventEl = trackEvents[trackEvents.length-1];
  var prevTrackEventEl = trackEvents[trackEvents.length-2-y];
  var eventVtus =
  ➥ parseFloat(events[eventIndexUnc[y]].getAttribute('timing'));
  var prevStartTime =
  ➥ parseFloat(trackEvents[trackEvents.length-2
  ➥ y].getAttribute('start_time'));
  var prevRelStartTime =
  ➥ parseFloat(trackEvents[trackEvents.length-2]
  ➥ .getAttribute('start_time'));
  var timeVtu = ((conNextStartTime - prevStartTime)
  ➥ * eventVtus / vtuTot + prevRelStartTime).toFixed(2);
  trackEventEl.setAttribute('start_time', timeVtu);
  trackIndexingEl.insertBefore(trackEventEl, referTrackEventEl);
}

eventIndexUnc.length = 0;
listUncTimingEvts.length = 0;
```

Nella parte di codice soprastante si procede ad inserire nel documento i `track_event` relativi agli `event` che hanno un `timing` uguale ad un frazione di `unitVtu` (nel caso in cui `unitVtu` valesse 1024 per quarto gli elementi considerati saranno identificabili come ottavi, sedicesimi, trentaduesimi, ecc.).

Questi elementi, come si vedrà poco più avanti nella funzione, verranno salvati in due array e da questi recuperati solamente durante l'esecuzione successiva di `mapAudio()` nel caso in cui ne risultasse presenza nelle liste. Il valore di `start_time` di questi elementi sarà ricavato interpolando i valori di `start_time` di quelli precedente e successivo. Gli array saranno in ultima battuta resettati permettendo il loro ripopolamento nel momento in cui si dovessero presentare altri casi analoghi.

```
eventIndexPrev = eventIndex;

while (events[eventIndex].getAttribute('timing') == 0) {
  var nextCurrentId = events[eventIndex].getAttribute('id');
  var nextTrackEventEl = xmlDoc.createElement('track_event');
  nextTrackEventEl.setAttribute('event_ref', nextCurrentId);
  nextTrackEventEl.setAttribute('start_time', audioTime);
  trackIndexingEl.appendChild(nextTrackEventEl);
  eventIndex++;
}
```

Queste linee di codice controllano se l'evento successivo nello `spine` ha `timing` uguale a zero ed in tal caso inseriscono l'elemento `track_event` all'interno di `audio` con un valore di `start_time` identico. Come accaduto anche in precedenza ad ogni ciclo viene incrementato `eventIndex` avanzando così con la mappatura.

```
var vtuTotal = 0;

while ((events[eventIndex].getAttribute('timing') != unitVtu &&
➥ (events[eventIndex].getAttribute('timing') % unitVtu) != 0) ||
➥ events[eventIndex].getAttribute('timing') == 0) {
  var nextCurrentId = events[eventIndex].getAttribute('id');
  eventIndexUnc.push(eventIndex);
  listUncTimingEvts.push(nextCurrentId);
  eventIndex++;
```

```
      vtuTotal +=
      ➥ parseInt(events[eventIndex-1].getAttribute('timing'));
      if (vtuTotal >= unitVtu) {
        break;
      }
    }
```

Le istruzioni sopra riportate hanno il compito di consentire una corretta mappatura degli eventi con un valore di `timing` frazionario rispetto a quello di `unitVtu` inserendoli in due array da dove successivamente (come già visto nei passaggi precedenti) verranno recuperati e opportunamente gestiti, precisamente `eventIndexUnc` sarà destinato a contenere l'indice degli eventi interessati e `listUncTimingEvts` conterrà gli eventi veri e propri. Nel momento in cui la somma del valore dei `timing` degli elementi inseriti negli array sarà uguale o maggiore al valore di `unitVtu` si uscirà dal ciclo significando, ad esempio, che un quarto è trascorso.

```
    if (listUncTimingEvts.length != 0 &&
    ➥ events[eventIndex].getAttribute('timing') != 0) {
      var fakeEl = xmlDoc.createElement('event');
      fakeEl.setAttribute('id', 'fakeEl');
      fakeEl.setAttribute('timing', '0');
      var spineEl = xmlDoc.getElementsByTagName('spine')[0];
      var referNode = events[eventIndex];
      spineEl.insertBefore(fakeEl, referNode);
    }
  }
```

Questa istruzione `if` permette, con un piccolo espediente, di gestire comunque con successo un'anomalia che si presenta eseguendo il codice di mappatura degli eventi con `timing` frazionario di `unitVtu`. Nel caso in cui l'evento interessato non fosse seguito da un evento dello spine con valore di `timing` uguale a `0`, si procederà ad inserire un elemento fittizio sempre con `timing`

impostato su zero che ne consentirà una corretta gestione. Questi elementi fittizi saranno successivamente rimossi una volta completato tutto il processamento.

```
  tapVtu += unitVtu;

  if (spineVtu < tapVtu && eventIndexPrev <= eventIndex) {
    while (spineVtu < tapVtu && eventIndexPrev <= eventIndex) {
      spineVtu +=
      ➥ parseInt(events[eventIndexPrev].getAttribute('timing'));
      eventIndexPrev++;
    }
  }
```

Con le ultime linee della funzione `mapAudio()` si andrà ad aumentare sempre e comunque il valore di `tapVtu` con quello di `unitVtu` ad ogni tap sulla barra spaziatrice ed inoltre nel caso in cui `spineVtu`, ossia la somma dei valori degli attributi `timing` degli `event` finora considerati, fosse minore di `tapVtu` ed i due indici fossero correttamente allineati, si eseguirà un ciclo per avanzare all'`event` successivo, sommando il valore del relativo `timing` a quello di `spineVtu` ed incrementando l'indice di volta in volta.

Istruzioni che non fanno parte di `mapAudio()` ma che sono strettamente legate ad essa sono quelle che implementano il "reset" della mappatura.

```
$('#resetButton').click(function(){
  var trackIndexingEl = xmlDoc.getElementsByTagName('track_indexing')
  ➥ [trackIndexingCounter];
  var trackEvents =
  ➥ trackIndexingEl.getElementsByTagName('track_event');
  $(trackEvents).remove();
  spineVtu = 0;
  tapVtu = 0;
  eventIndex = 0;
  var audioPlayer = document.getElementById('myAudio');
  audioPlayer.currentTime = 0;
```

```
  tap = 0;
  $('#counter').text(tap);
});
```

Quando il pulsante "reset" viene premuto vengono cancellati tutti i `track_event` generati nel corso della corrente mappatura, gli indici vengono azzerati ed il tempo di riproduzione del brano viene resettato.

4.4 Salvataggio documento mappato: `saveMappedXML()`

Come suggerisce il nome stesso la funzione `saveMappedXML()` permette di effettuare il salvataggio del documento XML una volta che è stato manipolato. La funzione verrà attivata nel momento in cui la procedura di mappatura può ritenersi conclusa, ossia quando l'utente preme sul pulsante "next" nella schermata in cui si trova il player per accedere all'ultima sezione, quella destinata al download del file mappato.

```
function saveMappedXML() {
  var events = xmlDoc.getElementsByTagName('event');
  var trackIndexingEl = xmlDoc.getElementsByTagName('track_indexing')
  ➥ [trackIndexingCounter];
  var trackEvents =
  ➥ trackIndexingEl.getElementsByTagName('track_event');
  var fakeString = 'fakeEl';
  for (var q = 0; q < events.length; q++) {
    var detectFakeId = events[q].getAttribute('id');
    if (detectFakeId == fakeString) {
      $(events[q]).remove();
      $(trackEvents[q]).remove();
    }
  }
  var xmlStringRaw = new XMLSerializer().serializeToString( xmlDoc );
  var xmlString = vkbeautify.xml(xmlStringRaw);
```

```
  $.ajax({
    type: "POST",
    url: "saveMappedXML.php",
    data: { 'xmlString': xmlString }
  });
}

nextMapper.addEventListener('click', saveMappedXML, false);
```

Come si può notare il codice è molto semplice. Si inizia con eliminare dal DOM gli elementi fittizi generati nella funzione `mapAudio()` per gestire correttamente gli eventi con `timing` frazionario del valore di `unitVtu`. Viene successivamente dichiarato un nuovo oggetto XMLSerializer [W08] che grazie al suo utilissimo metodo `serializeToString()` permette di convertire il DOM manipolato (quindi incluso della mappatura appena eseguita) in una stringa testuale. Questa stringa verrà in seguito correttamente indentata per mezzo del plug-in vkbeautify [W09] e successivamente inviata al server mediante AJAX per essere salvata come file XML.

Capitolo 5

Conclusioni e sviluppi futuri

In quest'ultima sezione dell'elaborato verranno trattati gli aspetti legati al testing dell'applicazione nell'ambiente web per il quale è stata progettata e soprattutto gli sviluppi e le migliorie future che potrebbero essere ulteriormente implementate.

5.1 Testing

Oltre alle quotidiane sessioni di debugging, nella fase finale del progetto sono stati condotti alcuni test per verificare il corretto funzionamento dell'applicazione ed accertare il raggiungimento degli obiettivi inizialmente fissati. Si è provato a ripetere il processo di mappatura su alcuni documenti IEEE 1599 su cui era già stata precedentemente eseguita per mezzo dell'applicativo Audio Mapper 3.1, con lo scopo di verificare eventuali discrepanze e diversità nei risultati. Il procedimento di mappatura (che è stato effetuato in prevalenza su server locale grazie ad un'installazione di MAMP) ha

coinvolto i cinque web browser per cui era stato preventivato il supporto in fase di progettazione: Mozilla Firefox, Opera, Apple Safari, Microsoft Internet Explorer e Google Chrome. In ogni browser sono state testate tutte le funzioni rese disponibili dall'applicazione:

- Upload del documento XML IEEE 1599
- Upload del file audio
- Utilizzo del player e della tastiera del computer per eseguire la mappatura
- Download del documento IEEE 1599 mappato e relativi file audio

Durante le fasi di testing tutti i browser hanno dimostrato un ottimo comportamento e un completo supporto alle tecnologie utilizzate per lo sviluppo dell'applicazione, ad eccezione di Microsoft Internet Explorer che sia nella sua versione 9 che 10 ha mostrato incompatibilità con l'oggetto `XMLSerializer` e/o con il plugin `vkbeautify.js` producendo un documento XML non conforme alle specifiche del formato IEEE 1599. A causa di questi problemi Internet Explorer è l'unico browser di cui viene sconsigliato l'utilizzo, non riuscendo ad offrire un'esperienza accettabile ai fini del processo di mappatura audio.

In sintesi i risultati ottenuti si possono ritenere soddisfacenti, salvo per l'ultima eccezione appena citata. L'uso dell'applicazione risulta affidabile e pratico ma soprattutto conforme agli obiettivi posti in partenza.

5.2 Sviluppi futuri

Tra le tante funzionalità che si potrebbero implementare, ed anche tra le più pratiche e facilmente attuabili dal punto di vista programmativo, una è senza dubbio l'integrazione della mappatura video (contenuti audio-visivi) all'interno dell'applicazione. Lo standard IEEE 1599 come detto in precedenza è già nativamente predisposto per tale procedimento che viene gestito in maniera del tutto analoga a quello audio. Inoltre, molti dei controlli offerti dalle API audio JavaScript, sono allo stesso modo utilizzabili anche attraverso l'elemento `video` ed implementati con delle rispettive API.

La seconda feature oltremodo utile che si potrebbe sviluppare (e che è già presente nell'applicativo Audio Mapper 3.1) è quella di poter visualizzare e manipolare gli eventi o meglio, i riferimenti ad essi, direttamente all'interno dell'applicazione, potendo aggiustare di volta in volta il tempo relativo a ciascun elemento ed offrendo così un controllo sul risultato molto più preciso.

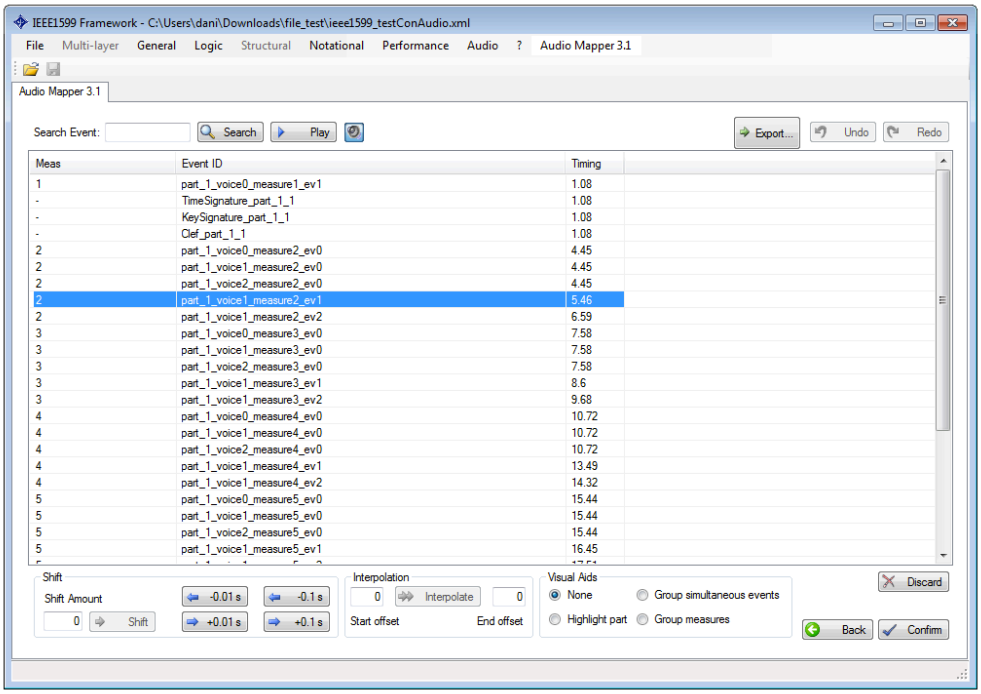

Figura 5.1: Schermata di rifinitura della mappatura audio nell'applicativo Audio Mapper 3.1

Infine si potrebbe considerare l'utilizzo dell'applicazione anche su dispositivi mobili di ultima generazione: tablet e smartphone. JavaScript infatti offre oggigiorno un buon supporto per catturare e gestire eventi attraverso le interfacce touchscreen (ad esempio `touchstart`, `touchend` o `touchmove` solo per citarne alcuni).

5.3 Conclusioni

In conclusione, visti i risultati restituiti in fase di testing, gli obiettivi principali del progetto possono considerarsi raggiunti. Il prototipo realizzato è in grado di eseguire la mappatura audio di documenti IEEE 1599 via web, riproducendo le funzionalità principali offerte dall'applicativo desktop Audio Mapper 3.1, attualmente unica soluzione software che consente di mappare in maniera semi-automatica un file conforme alla specifica IEEE 1599. Grazie al lavoro svolto durante la realizzazione di questo progetto è stato inoltre possibile approfondire e testare direttamente alcune delle possibilità offerte dalla nuova specifica HTML5 e dalla codifica IEEE 1599, tecnologie che consentiranno in futuro un'interazione in ambiente web sempre più attiva da parte degli utenti nella creazione e nella manipolazione di contenuti multimediali.

Glossario

AJAX: Tecnica di sviluppo per la realizzazione di applicazioni web interattive, basato su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente.

API: Application Programming Interface. Insieme di procedure disponibili al programmatore, raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma.

DOM: Document Object Model. Forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti, neutrale sia per la lingua che per la piattaforma.

HTML: HyperText Markup Language, ovvero il linguaggio solitamente usato per i documenti ipertestuali disponibili nel World Wide Web.

MAMP: Macintosh Apache MySQL PHP. È un set di programmi liberi comunemente utilizzati insieme per far girare un sito web dinamico sul sistema operativo Mac OS X della Apple.

URI: Uniform Resource Identifier, stringa che identifica univocamente una risorsa generica che può essere un indirizzo Web, un documento, un'immagine, un file, un servizio, un indirizzo di posta elettronica, ecc.

URL: Uniform Resource Locator, particolare tipo di URI rappresentante l'indirizzo mnemonico di una risorsa in Internet.

XML: eXtensible Markup Language. Metalinguaggio di markup, ovvero un linguaggio marcatore che definisce un meccanismo sintattico che consente di estendere o controllare il significato di altri linguaggi marcatori.

Ringraziamenti

Ai miei genitori, per il tempo e lo spazio concessomi nell'inseguire questa mia passione.

A Silvia, per esserci sempre stata.

Al mio relatore prof. Luca Andrea Ludovico, che mi ha dato la possibilità di svolgere questo elaborato e mi ha messo nelle migliori condizioni per farlo.

Al mio correlatore dott. Stefano Baldan, per i preziosi consigli.

Al dott. Adriano Baratè, il cui supporto nella parte finale del progetto è stato inestimabile.

Al prof. Goffredo Haus ed a tutti i professori del corso di Scienze e Tecnologie della Comunicazione Musicale, è stato un bel viaggio e voi siete state ottime guide.

GRAZIE.

Daniele

Bibliografia

[B01] IEEE P1599 Working Group, *Recommended Practice for Definition of a Commonly Acceptable Musical Application using the XML Language*. IEEE Computer Society Press, Washington DC, 2008.

[B02] Luca A. Ludovico, *Manuale di MX*. Laboratorio di Informatica Musicale - Università degli Studi di Milano, Milano, 2005.

[B03] Luca A. Ludovico, *Key Concepts of the IEEE 1599 Standard*. Proceedings della conferenza IEEE CS *The Use of Symbols to Represent Music and Multimedia Objects*, pp. 15-26, IEEE CS Lugano, 2008.

[B04] Stefano Baldan, *GlobeMX: Studio e Realizzazione di una Piattaforma per il Web Streaming Multimediale Basato su Tecnologia IEEE 1599*. Tesi di Laurea Magistrale in Informatica per la Comunicazione, Università degli Studi di Milano, 2011.

[B05] Mark Pilgrim, *HTML5: Up and Running*. O'Reilly Media, Cambridge, 2010.

[B06] Adriano Baratè, Luca A. Ludovico, Alberto Pinto, *A Computer Tool to Visualize Score Analysis*. In: Jensen K (ed.). Proceedings della conferenza *Computers in Music Modelling and Retrieval and Network for Cross-Disciplinary Studies of Music and Meaning Conference*, pp. 315-326. Re: New, Copenhagen, 2008.

[B07] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R.S. Marcal, Carlos Guedes, Jaime S. Cardoso, *Optical Music Recognition: State-Of-The-Art and Open Issue*. Springer-Verlag London Limited, 2012.

[B08] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, Malcom Slaney, *Content-Based Music Information Retrieval: Current Directions and Future Challenges*, pp. 668-696. University of London, London, 2008.

Webografia

[W01] Sito Ufficiale IEEE: http://www.ieee.org

[W02] Sito Ufficiale formato XML: http://www.w3.org/XML/

[W03] Sito Ufficiale LIM: http://www.lim.dico.unimi.it

[W04] DTD IEEE 1599:
http://standards.ieee.org/downloads/1599/1599-2008/ieee1599.dtd

[W05] Documentazione specifica IEEE 1599:
http://mx.lim.dico.unimi.it/reference_manual/index.php

[W06] Sito Ufficiale W3C: http://www.w3.org

[W07] Sito Ufficiale WHATWG: http://www.whatwg.org

[W08] Documentazione XMLSerializer:
https://developer.mozilla.org/en-US/docs/XMLSerializer

[W09] Documentazione plugin vkbeautify:
http://code.google.com/p/vkbeautify/