Università degli Studi di Milano
Facoltà di Scienze e Tecnologie

Corso di Laurea in
Informatica Musicale

Gestione di licenze ODRL all'interno della piattaforma web IEEE1599

Studente:
Alessia Ruggeri
Matricola 858643

Relatore:
Prof. Paolo Perlasca

Correlatori:
Prof. Adriano Baratè
Prof. Luca Andrea Ludovico

Anno Accademico 2016/2017

Alessia Ruggeri: *Gestione di licenze ODRL all'interno della piattaforma web IEEE1599*. © ottobre 2017.

E-MAIL: alessia.ruggeri.30@gmail.com

INDICE

ELENCO DELLE FIGURE

ALESSIA RUGGERI

GESTIONE DI LICENZE ODRL ALL'INTERNO DELLA PIATTAFORMA WEB IEEE1599

ABSTRACT

Attraverso l'utilizzo del linguaggio ODRL e del player multimediale legato al framework IEEE1599, lo scopo dell'elaborato è quello di integrare l'esperienza di fruizione e di interazione multimediale di un brano musicale con il controllo automatizzato delle licenze correlate ai diversi contenuti digitali connessi al brano, che sono genericamente audio, video e partiture.

KEYWORDS:

- copyright
- licenza
- diritto d'autore
- IEEE1599
- ODRL
- Contenuti musicali

INTRODUZIONE

Il diritto d'autore garantisce la salvaguardia del patrimonio artistico e culturale, tutelando le opere d'ingegno e i loro autori che, con spirito di creatività, danno vita a tali opere.

La necessità di tutelare un'opera è strettamente legata alla sua proprietà intellettuale: il creatore dell'opera possiede il diritto alla paternità dell'opera, a prescindere dagli ulteriori diritti di tipo economico. Nel caso in cui l'opera venga riprodotta illecitamente, i soggetti che ne rivendicano i diritti alla paternità hanno bisogno di ottenere delle tutele che permettano loro di essere riconosciuti come i responsabili della creazione di quell'opera.

La tutela del prodotto intellettuale è un'esigenza estremamente sentita in epoca contemporanea, dove le nuove tecnologie digitali, se da un lato hanno reso estremamente facile la diffusione e la divulgazione delle opere creative, dall'altro hanno reso più difficile il controllo, la distribuzione e l'utilizzo che ne viene fatto.

L'applicazione delle tecnologie informatiche all'analisi e all'elaborazione delle opere multimediali ha permesso la realizzazione di moderni meccanismi digitali di protezione, quali il linguaggio ODRL, che si occupano di far rispettare i permessi espressi dalle licenze d'uso, in modo che le opere possano essere fruite nel pieno rispetto del diritto d'autore.

Per garantire appieno la tutela della proprietà intellettuale su un contenuto digitale, infatti, lo stesso dev'essere fruito tramite un sistema in grado di far rispettare i vincoli che sono stati concessi per l'utilizzo del contenuto digitale, mediante l'accettazione di una licenza che permetta la fruizione dell'opera secondo le disposizioni stabilite dal suo creatore.

Negli ultimi anni, è stato creato lo standard IEEE1599, che consente di interagire tra contenuti musicali e multimediali, come se questi materiali fossero parte di un unico ambiente integrato. La modalità di fruizione unificata offerta da questo standard si diversifica da quelle già esistenti, andando ad ampliare le possibilità legate alla fruizione di un'esperienza multimediale di un brano musicale.

Grazie all'utilizzo di questa tecnologia, infatti, è possibile sincronizzare automaticamente tra loro le diverse tipologie di contenuto multimediale che compongono un brano: audio, video e partitura.

È in questo contesto tecnologico e culturale che si colloca il presente lavoro di tesi.

OBIETTIVI DELLA TESI Attraverso l'utilizzo del linguaggio ODRL e del player multimediale legato al framework IEEE1599, lo scopo dell'elaborato è quello di integrare l'esperienza di fruizione e di interazione multimediale di un brano musicale con il controllo automatizzato delle licenze correlate ai diversi contenuti digitali connessi al brano, che sono genericamente audio, video e partiture.

In questo modo l'utente finale potrà approfondire ogni aspetto multimediale dell'esperienza musicale, in tutte le sue sfaccettature e in maniera completamente rispettosa dei diritti d'autore legati alle singole risorse multimediali che connotano il brano.

STRUTTURA DELL'ELABORATO L'esposizione del lavoro è articolata nel modo seguente:

1. **LO STANDARD IEEE1599** dove verranno descritte le caratteristiche e la struttura del formato IEEE1599: verranno esposte le motivazioni che hanno portato alla creazione di un nuovo standard musicale; si studierà la struttura di un documento IEEE1599 e la sua divisione in layer; inoltre, verranno descritte le caratteristiche e gli elementi fondamentali del formato per il suo corretto funzionamento nell'applicazione musicale.

2. **IL LINGUAGGIO ODRL** nel quale si proporranno le principali caratteristiche del linguaggio ODRL (versione 1.1); in particolare sarà descritta l'architettura del linguaggio e verranno esposte le sue funzionalità per l'espressione e la garanzia dei diritti associati ai contenuti multimediali.

3. **L'INTERAZIONE TRA IL LINGUAGGIO ODRL E LO STANDARD IEEE1599** in cui verranno esposte le modalità di interazione tra il linguaggio ODRL per l'espressione di licenze e lo standard IEEE1599 per la creazione di un'esperienza musicale completa. Nello specifico, verrà descritto il profilo ODRL creato appositamente per la generazione di licenze relative a risorse musicali e adatte alla protezione dei diritti sugli asset coinvolti nell'esperienza musicale del framework IEEE1599. Successivamente, si presenteranno due possibili scenari per la creazione del documento IEEE1599 basato sulle licenze ODRL degli utenti e si riporteranno le motivazioni che hanno condotto alla scelta di quella utilizzata.

4. **PROGETTAZIONE DELLA BASE DI DATI** dove si guiderà il lettore attraverso le scelte progettuali che hanno portato alla strutturazione e alla generazione della base di dati per la piattaforma web IEEE1599. Nel dettaglio, verranno descritte le fasi di progettazione concettuale, logica e fisica che sono state necessarie per prendere decisioni sulle modalità di salvataggio delle informazioni e per implementare il database attraverso il linguaggio SQL.

5. **IMPLEMENTAZIONE** nel quale si descriveranno le varie fasi che hanno portato all'implementazione della piattaforma web IEEE1599, motivando le scelte progettuali che sono state effettuate. Verranno presentati i meccanismi per la gestione dei vincoli ODRL, oltre che i sistemi di autenticazione e di inserimento di nuovi brani all'interno della base di dati. Infine, verrà descritta brevemente la realizzazione dell'interfaccia utente e la struttura del progetto nella sua organizzazione in directory.

1 LO STANDARD IEEE1599

INDICE

In questo capitolo verranno descritte le caratteristiche e la struttura del formato IEEE1599: verranno esposte le motivazioni che hanno portato alla creazione di un nuovo standard musicale; si studierà la struttura di un documento IEEE1599 e la sua divisione in layer; inoltre, verranno descritte le caratteristiche e gli elementi fondamentali del formato per il suo corretto funzionamento nell'applicazione musicale.

Nell'ambito musicale, esistono molte tipologie di fruizione della musica: un singolo brano può essere descritto attraverso la sua partitura, scritta a mano o in forma digitale, oppure tramite una performance dal vivo o una registrazione effettuata in studio, che può essere solo audio o anche video, o ancora attraverso un file MIDI. Dunque l'informazione musicale risulta composta da contenuti estremamente eterogenei tra di loro, poichè coinvolge molte descrizioni diverse di uno stesso brano musicale.

Per questo motivo, nel corso degli anni sono stati sviluppati diversi formati e standard per rappresentare specificatamente una certa tipologia di fruizione della musica. Per esempio, nell'ambito dei file audio sono nati formati come FLAC, WAV, MP3 e AAC; per codificare video si usano spesso i formati MP4, AVI e FLV; nel caso di immagini e partiture si ricorre ai formati JPEG, GIF, TIFF e PDF; mentre per rappresentare performance musicali attraverso un'interfaccia tra strumenti digitali, il protocollo più diffuso è il MIDI. I formati appena citati sono comunemente diffusi e adatti al loro scopo, ma tutti presentano un limite intrinseco nella loro natura: essi sono in grado di descrivere l'informazione musicale solo da uno specifico punto di vista e non possono fornire una descrizione unificata di tutti

gli aspetti di un brano musicale. In poche parole, ciascuno standard si limita a codificare soltanto la tipologia di contenuto musicale per cui è stato appositamente creato, senza preoccuparsi di poter integrare l'esperienza musicale con altre tipologie di contenuti.

Per musicologi, musicisti professionisti, studenti di musica o persone semplicemente interessate all'ambito, sarebbe utile avere la possibilità di fruire contemporaneamente di tutte le informazioni relative ad un brano attraverso un unico ambiente, completo di tutti i contenuti riguardanti quel brano. L'obiettivo dello *standard IEEE1599* è proprio quello di soddisfare questa esigenza, mettendo a disposizione un formato in grado di rappresentare in un singolo documento tutte le diverse modalità di fruizione della musica. Il formato IEEE1599 si propone come *standard musicale* a tutti gli effetti, poichè non si limita ad una porzione di contenuti dell'ambito musicale, ma consiste in una tecnologia per la rappresentazione e la codifica della musica in ogni suo aspetto.

Il formato IEEE1599 presenta alcune caratteristiche fondamentali che lo differenziano dagli altri formati:

- *La ricchezza e la varietà di descrizioni multimediali per uno stesso brano musicale:* i contenuti simbolici, strutturali, grafici, audio e video appartenenti ad uno stesso brano musicale possono essere codificati, o comunque relazionati, attraverso un unico documento.

- *Per ciascun tipo di descrizione multimediale, la possibilità di relazionare vari oggetti digitali:* è possibile relazionare ad un unico file varie performance musicali dello stesso brano o varie scansioni di partiture provenienti da edizioni diverse.

- *Il supporto ad una sincronizzazione temporale completa tra i contenuti:* grazie alla creazione di un player dedicato, i contenuti audio e video sono sincronizzati con lo spartito in modo tale che i media avanzino contemporaneamente. È anche possibile cambiare in tempo reale il contenuto audio o video o la partitura, mentre il brano è ancora in esecuzione, e i contenuti rimarranno comunque sincronizzati.

- *L'interazione user-friendly con i contenuti musicali:* questo formato offre le caratteristiche per implementare applicazioni che consentano all'utente di interagire in modo semplice e intuitivo con il brano musicale. In una piattaforma web IEEE1599, l'utente può esplorare il brano cliccando su un simbolo della partitura e l'audio automaticamente salterà a quel punto del brano; viceversa, è possibile cliccare sulla

barra del tempo dell'audio e sullo spartito verrà automaticamente sincronizzato l'andamento della notazione musicale.

1.1 DESCRIZIONE DELLO STANDARD IEEE1599

Lo standard IEEE1599 è frutto di una ricerca portata avanti al *Laboratorio di Informatica Musicale* (LIM) dell'*Università degli Studi di Milano*. IEEE1599 è il nome ufficiale di *MX*, un acronimo che sta per *Musical application using XML*. Il progetto riguardante lo sviluppo di questo framework ha come obiettivo quello di rappresentare la musica raggruppando in un unico ambiente tutte le informazioni e i contenuti di vario genere appartenenti ad uno stesso brano, in modo da offrirne una fruizione completa.

1.1.1 Utilizzo del linguaggio XML

IEEE1599 è un formato che si basa sul linguaggio XML. L'utilizzo di XML per descrivere informazioni nell'ambito musicale apporta molti vantaggi derivanti dall'affinità tra la natura stessa del linguaggio e le strutture tipiche delle informazioni musicali.

In primo luogo, XML è un linguaggio che permette di descrivere oggetti di ogni tipo, associandogli degli attributi. La rappresentazione delle entità è modulare e gerarchica e segue uno schema ben organizzato, ma che rimane comunque flessibile e completamente personalizzabile a seconda delle esigenze. Tutte queste caratteristiche risultano essenziali per esprimere il linguaggio musicale, che a sua volta è formale e gerarchico, ma necessita di elasticità per essere descritto.

Per di più, la modularità di XML consente di mantenere le entità indipendenti e separate tra di loro, ma supporta anche l'interdipendenza tra gli oggetti, creando delle associazioni tra di essi. Nell'ambito del linguaggio musicale, è importante mantenere separate le identità degli "eventi musicali", ma risulta altrettanto rilevante avere la possibilità di creare legami tra di loro e XML risulta adatto alla loro rappresentazione poiché soddisfa queste proprietà.

I linguaggi che si basano su XML, inoltre, sono estensibili; infatti supportano ampliamenti del linguaggio e l'aggiunta di entità esterne. Questo aspetto è fondamentale per lo sviluppo del linguaggio musicale e per sue future descrizioni.

Infine, il linguaggio XML è aperto al contributo da parte degli utenti, è facile da leggere, decodificare ed editare, risulta molto comprensibile ed è supportato dalla maggior parte delle applicazioni.

Dunque, grazie alla sua struttura, alla sua facilità di lettura e alla sua estensibilità, il linguaggio XML si configura come uno strumento perfetto per la creazione di uno standard musicale come l'IEEE1599.

1.1.2 Struttura multi-stratificata

Un documento IEEE1599 racchiude al suo interno tutte le informazioni per descrivere un singolo brano musicale. Tuttavia, i contenuti musicali da includere nel documento sono di natura eterogenea e si differenziano per tipologia, dimensione e formato. Dunque, è necessario rappresentare tali informazioni attraverso una struttura efficiente e ben organizzata.

Per questo motivo, si è deciso di strutturare il framework IEEE1599 attraverso sei diversi *layer*, ciascuno dedicato alla rappresentazione di un certo tipo di informazione musicale:

- *General:* il general layer fornisce una descrizione generale del contenuto del documento e contiene una serie di informazioni formali riguardanti il brano musicale nella sua interezza. Alcuni esempi di dati contenuti in questo layer sono: i metadati (titolo, autori, anno, ecc.), il numero di catalogo (se il brano appartiene ad una collezione), il genere musicale, alcune note riguardanti il brano.

- *Logic:* il logic layer rappresenta il cuore del formato IEEE1599, poiché contiene una descrizione del brano musicale dal punto di vista simbolico e astratto. Questo strato è a sua volta costituito da tre parti: lo *spine*, che rappresenta lo scheletro logico del brano ed è il riferimento per la localizzazione e la sincronizzazione tempo-spazio degli eventi musicali; il *LOS* (*Logically Organized Symbols*), che descrive la simbologia della partitura dando un significato musicale ai simboli; infine il *layout*, che mantiene informazioni riguardanti l'implementazione grafica dei contenuti simbolici.

- *Structural:* lo structural layer contiene una descrizione esplicita degli "oggetti musicali" e delle relazioni che coesistono tra di loro; in altre parole, questo strato descrive la struttura del brano musicale come un susseguirsi di oggetti che si trasformano e formano dei legami tra di loro.

- *Notational:* il notational layer racchiude tutte le rappresentazioni grafiche della partitura del brano. In questo strato possiamo trovare scansioni o immagini di spartiti scritti a mano o creati digitalmente, ma anche elaborazioni digitali da parte di software che si occupano della gestione degli aspetti notazionali di un brano. Per ciascuna di esse, viene memorizzata la mappatura che lega i simboli agli eventi musicali precedentemente descritti.

- *Performance:* il performance layer rappresenta il brano musicale attraverso una collezione di descrizioni ed esecuzioni musicali basate sulla computazione, attraverso uno specifico formato digitale per l'espressione della musica, come il MIDI o il SASL/SAOL.

- *Audio:* il layer audio accoglie il materiale di tipo audio, ovvero registrazioni e file audio e video, in un certo numero di formati diversi. Ciascuno di questi contenuti possiede la propria mappatura che lega i secondi del file audio o video agli eventi musicali precedentemente descritti.

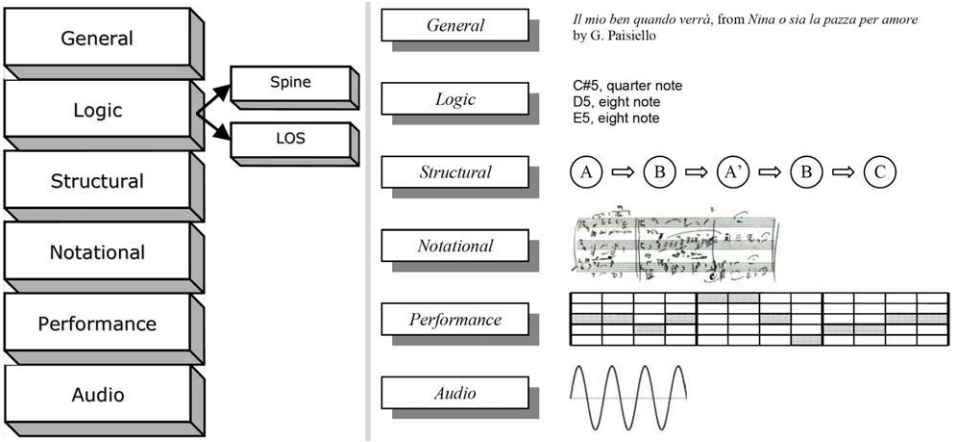

Figura 1. Struttura del formato IEEE1599 basata sui layer (Immagini estratte da "Key Concepts of the IEEE 1599 Standard" e da "Semantic Multimedia", consultare la bibliografia)

La struttura XML del documento IEEE1599 risultante è la seguente:

```
<?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>
        <performance>...</performance>
        <audio>...</audio>
</ieee1599>
```

Al fine di generare un documento IEEE1599 valido, che rispetti le dichiarazioni del DTD (Document Type Definition), non è necessario che siano presenti tutti i layer: solamente il general layer e il logic layer devono essere inseriti e popolati obbligatoriamente affinché il documento possa risultare come tale. Essi, infatti, forniscono informazioni essenziali riguardanti i metadati del brano e la sua strutturazione in eventi musicali, informazioni che non possono mancare per il raggiungimento dei fini che il formato si propone. Gli altri layer possono non essere presenti, oppure essere inseriti e popolati con un numero di contenuti a piacimento. Ovviamente, più il documento IEEE1599 risulta ricco di materiali eterogenei, più completa sarà la descrizione musicale risultante.

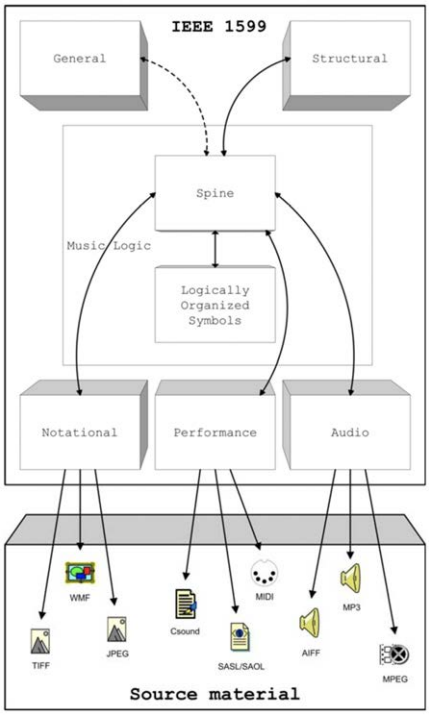

Figura 2. Interconnessioni tra i layer del formato IEEE1599 (Immagine estratta da "IEEE Recommended Practice for Defining a Commonly Acceptable Musical Application Using XML", consultare la bibliografia)

LO SPINE Lo spine è l'unico, tra gli elementi del logic layer, che deve essere inserito obbligatoriamente, poiché costituisce la principale struttura dati del documento IEEE1599. Il suo ruolo è quello di elencare in ordine e definire tutti gli eventi musicali, identificandoli univocamente tramite un'etichetta assegnata a ciascuno di essi. Il concetto di "evento musicale" viene appositamente lasciato vago, in modo che il formato possa rimanere flessibile ed essere impiegato in situazioni diverse secondo le proprie esigenze.

Lo spine mette a disposizione una lista di eventi musicali, che rappresentano i frammenti atomici del brano e la cui granularità può essere liberamente stabilita dall'autore della codifica. Per questo motivo, lo spine rappresenta il "collante" di tutta l'esperienza musicale resa disponibile dal documento IEEE1599; infatti, tutte le descrizioni eterogenee per lo stesso brano musicale possono essere mappate facendo riferimento agli identificatori degli eventi musicali elencati in questo strato. In questo modo, facendo riferimento ad un'unica struttura astratta del brano musicale, tutti i materiali multimediali rimangono sincronizzati tra di loro.

Si riporta un breve esempio di spine nel documento IEEE1599:

```
<ieee1599>
    <logic>
        <spine>
            <event id="e1" timing="0" hpos="0"/>
            <event id="e2" timing="2" hpos="2"/>
            <event id="e3" timing="2" hpos="2"/>
            <event id="e4" timing="1" hpos="1"/>
            <event id="e5" timing="1" hpos="1"/>
            ...
        </spine>
        <los>...</los>
    </logic>
</ieee1599>
```

1.1.3 Sincronizzazione inter-layer e intra-layer

Come affermato precedentemente, un documento IEEE1599 necessita soltanto del general layer e dello spine del logical layer per poter essere considerato valido, ma solitamente esso contiene anche la maggior parte degli

altri layer. Infatti, un documento contenente solo lo spine avrebbe poco significato, dal momento che gli eventi musicali sarebbero definiti da quest'ultimo, ma non verrebbero in nessun modo "utilizzati" dagli altri layer. È nel momento in cui vengono inseriti i riferimenti a contenuti multimediali quali audio, video, performance e partiture che lo spine acquisisce il suo ruolo fondamentale di collante tra materiali eterogenei.

Ciascuna istanza delle diverse tipologie di contenuti viene descritta facendo riferimento all'elenco di eventi atomici descritti nello spine: nel caso di un file audio o video, per esempio, ciascun evento musicale ha inizio in un certo istante, espresso in ore, minuti e secondi trascorsi dall'inizio del file; nel caso di una partitura, invece, ciascun gruppo di simboli del linguaggio musicale viene associato ad un evento specifico tra quelli espressi nello spine.

La ricca rete di riferimenti tra i layer e le loro istanze porta all'individuazione di due tipologie di relazioni descrivibili all'interno del documento IEEE1599:

- *Sincronizzazione intra-layer:* sincronizzazione tra istanze appartenenti ad uno stesso layer (es. sincronizzazione tra due file audio);
- *Sincronizzazione inter-layer:* sincronizzazione tra istanze appartenenti a layer diversi (es. sincronizzazione tra un file audio e una partitura).

1.2 PIATTAFORMA WEB IEEE1599

Per poter fruire dei documenti IEEE1599 in formato XML, è stato sviluppato un apposito player che consenta all'utente di interagire in modo semplice e intuitivo con i contenuti multimediali appartenenti ad un brano musicale. Nella piattaforma IEEE1599 generata, per un determinato brano vengono presentate all'utente le tracce audio e video disponibili e vengono visualizzate le pagine delle partiture associate al brano.

Nel comparto audio del player, l'utente può selezionare la traccia audio che vuole ascoltare cliccando sul relativo nome e questa verrà eseguita; grazie alla barra temporale, è possibile controllare l'esecuzione della traccia, attivandola o mettendola in pausa e spostandosi da un secondo all'altro dell'audio. Nello stesso comparto sono contenuti anche i video, che si comportano esattamente allo stesso modo degli audio, con l'unica differenza che il filmato viene visualizzato in un riquadro apposito.

All'interno del comparto notazionale del player, l'utente può visualizzare la partitura desiderata cliccando sul relativo nome e può sfogliare le pagine dello spartito. Quando una traccia audio è in esecuzione, i simboli dello spartito corrispondenti agli eventi musicali "suonati" dall'audio si illuminano, fornendo una fruizione sincronizzata tra audio e partitura e permettendo all'utente di capire quali suoni corrispondono a quali simboli sullo spartito.

La caratteristica più importante del player IEEE1599 consiste nell'interazione con l'esecuzione sincrona tra contenuti multimediali: cliccando su un simbolo della partitura, la barra del tempo automaticamente salterà a quel punto del brano e la traccia audio corrente continuerà ad essere eseguita dal punto selezionato; allo stesso modo, cliccando sulla barra del tempo dell'audio, verrà automaticamente sincronizzato l'andamento della notazione musicale sulla partitura corrente. È anche possibile cambiare traccia audio/video o partitura in real-time, durante l'esecuzione del brano, e il player continuerà ad eseguirlo, senza pause o ritardi, seguendo il flusso di eventi musicali.

2 IL LINGUAGGIO ODRL

INDICE

Nel seguente capitolo si proporranno le principali caratteristiche del linguaggio ODRL (versione 1.1); in particolare, sarà descritta l'architettura del linguaggio e verranno esposte le sue funzionalità per l'espressione e la garanzia dei diritti associati ai contenuti multimediali.

L'Open Digital Right Language (ODRL, versione 1.1) è un linguaggio proposto alla comunità dei *Digital Right Management* (DRM) per la standardizzazione dell'espressione delle informazioni relative ai diritti applicabili a certi contenuti. L'ODRL intende quindi fornire meccanismi flessibili e interoperabili per supportare un utilizzo trasparente e innovativo delle risorse digitali nell'ambito della divulgazione, della distribuzione e della fruizione di pubblicazioni elettroniche, immagini digitali, risorse audio e video, supporti per l'apprendimento, software e qualsiasi altra creazione di natura digitale. L'ODRL non è coperto da alcuna licenza ed è disponibile come software *open-source*.

Il linguaggio ODRL è un linguaggio di specifica dei diritti, quindi il suo obiettivo è quello di interpretare i vincoli di una licenza esistente procurando degli equivalenti in forma digitale e supportando un'ampia varietà di nuovi servizi che possono essere utilizzati da qualsiasi contenuto digitale presente nel Web. Nell'ambito di creazioni fisiche, ODRL può comunque essere sfruttato per permettere una gestione automatizzata dei diritti, basata quindi sull'elaborazione digitale.

ODRL offre un linguaggio e un vocabolario standard per l'espressione dei termini e delle condizioni su un bene, detto *asset*. A questo scopo, ODRL possiede una semantica di base che consente di definire i *rights holder*, ovvero i titolari dei diritti, e gli utilizzi permessi per quella specifica manifestazione del bene. I diritti possono essere specificati per una singola manifestazione dell'asset oppure possono essere applicati ad un gruppo di manifestazioni dello stesso asset.

ODRL è un linguaggio basato su XML e definisce una semantica di base per l'espressione dei diritti; tuttavia, la semantica di base di ODRL può essere arricchita con l'aggiunta di ulteriori stratificazioni di semantica da parte di terze parti. Ogni aggiunta di elementi al linguaggio deve essere accompagnata dal *Data Dictionary* (DD) relativo, che serve a integrare il Data Dictionary di base di ODRL per la definizione dei nuovi elementi del linguaggio.

2.1 QUADRO GENERALE SULLA GESTIONE DEI DIRITTI DIGITALI

L'espressione Digital Rights Management (DRM) si riferisce all'insieme di sistemi tecnologici che consentono ai detentori di diritti d'autore e dei diritti connessi di esercitare tali diritti nel contesto digitale.

La diffusione di Internet su larga scala sta cambiando la modalità di distribuzione dei contenuti digitali da un flusso unico (dall'editore all'utente finale) ad un ciclo molto più interattivo, dove le creazioni vengono modificate, riutilizzate ed estese all'infinito. In tutti i casi i diritti hanno bisogno di essere gestiti e rispettati tramite sistemi affidabili.

Al giorno d'oggi, i DRM vengono utilizzati per incorporare delle misure di sicurezza all'interno dei sistemi digitali per la fruizione dei contenuti multimediali. In questo modo, le opere d'ingegno tutelate dal diritto d'autore hanno la possibilità di essere protette e di essere identificabili e tracciabili, per esempio attraverso l'inserimento di metadati nei file, con informazioni accessorie per l'identificazione della risorsa e per la sua fruizione, oppure mediante l'utilizzo della crittografia, per rendere illeggibile il contenuto protetto nel caso in cui non si disponga della chiave di cifratura.

L'intento dei sistemi DRM è quello di garantire una diffusione e una fruizione dei file multimediali da parte degli utenti il più possibile controllata, consentendo l'utilizzo della risorsa nei limiti dei diritti espressi nella

licenza d'uso ad essa associata. In particolare, gli obiettivi principali delle tecnologie DRM sono i seguenti:

- la certificazione di legittimità dell'uso e la piena titolarità dei diritti d'autore, che consiste nella possibilità di identificare univocamente l'esemplare digitale, così da riuscire ad individuare eventuali copie illegali del bene;
- il controllo dell'accesso, ovvero la verifica che l'accesso al contenuto da parte di un individuo sia regolarmente consentito nella licenza associata al file;
- il controllo delle copie illegali, attraverso meccanismi che permettano di risalire all'iniziale possessore del bene originale e quindi di individuare il responsabile di eventuali violazioni del diritto d'autore;

Tra le varie tecnologie utilizzabili in un sistema DRM si trovano anche i linguaggi REL (*Rights Expression Language*), linguaggi che consentono di specificare le condizioni d'uso applicabili sulla risorsa in esame. I linguaggi REL vengono quindi utilizzati per l'espressione delle licenze e dei diritti d'autore sotto forma di metadati associabili ad un certo bene digitale. Molti linguaggi REL utilizzano la tecnologia XML, poichè si presta bene alla definizione degli elementi delle licenze e alla costruzione di un'architettura adatta per l'espressione dei diritti d'autore; tuttavia potenzialmente i linguaggi REL possono essere espressi in qualsiasi formato a seconda delle esigenze e necessità.

Lo scopo di un linguaggio REL è quello di specificare chi può accedere ad un bene e sotto quali condizioni, descrivendo la licenza d'uso in termini di permessi e limitazioni sulla fruizione del contenuto digitale. Il linguaggio REL ODRL si occupa di fornire una semantica adatta e versatile per le espressioni DRM in sistemi aperti e affidabili, senza occuparsi dei meccanismi utilizzati per rendere il sistema sicuro.

2.2 ODRL FOUNDATION MODEL

Il linguaggio ODRL si basa su un modello estendibile per l'espressione dei diritti, il quale coinvolge un certo numero di entità fondamentali (core entities) e tutte le relazioni che intercorrono tra di esse. Questo modello viene chiamato Expression Language (EX) e si occupa di definire la natura

degli elementi principali del linguaggio, a cui poi viene dato un significato semantico nel Data Dictionary (DD).

Il modello consiste principalmente di tre unità fondamentali:

- Asset
- Rights
- Party

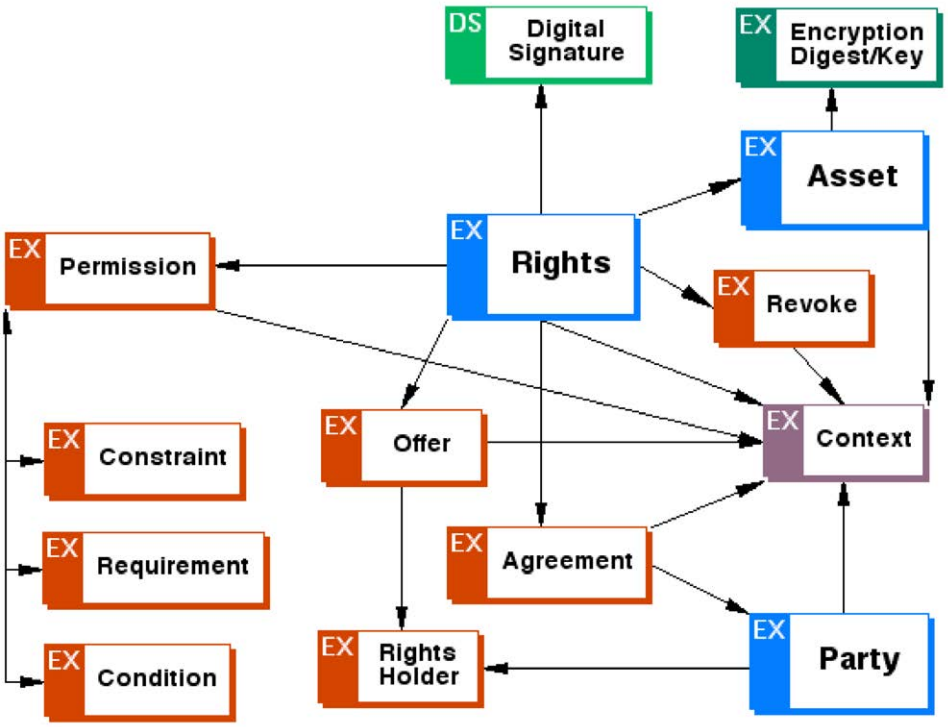

Figura 3. ODRL Foundation Model (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

ASSET Un asset è un qualsiasi contenuto fisico o digitale, oggetto dei diritti espressi nella licenza. Ciascun asset deve essere univocamente identificato, potrebbe essere composto da sotto-parti ed essere disponibile in diversi formati; in ogni caso, ciascuna manifestazione dell'asset deve essere identificata in modo univoco. Gli asset possono anche essere espressioni non tangibili di un lavoro o manifestazioni che si concretizzano solo in particolari situazioni. Gli asset possono essere criptati per permettere di distribuire i contenuti in modo sicuro.

RIGHTS I rights costituiscono l'involucro globale all'interno del quale trovano spazio tutti gli elementi del linguaggio ODRL volti a descrivere i diritti applicabili sugli asset. I rights contengono al loro interno una gerarchia di entità: ciascuno di essi contiene un certo numero di *permission* e all'interno di ogni permission possiamo trovare *constraint*, *requirement* e *condition*. Un permission costituisce a tutti gli effetti un'attività o un utilizzo che è consentito sull'asset in questione (es. visualizzare un'immagine, ascoltare un audio, stampare un documento, ecc.). I constraint all'interno di un permission identificano i limiti entro i quali è possibile esercitare il permission (es. ascoltare un audio solo fino al minuto 1:20). I requirement all'interno di un permission descrivono gli obblighi che devono essere soddisfatti per poter esercitare il permission (es. pagamento in anticipo per usufruire del video). I condition hanno una funzione simile ma contraria rispetto ai constraint: anch'essi esprimono delle condizioni, ma, se queste condizioni si verificano, il permission viene invalidato e non è più possibile esercitarlo (es. se la carta di credito scade, non è più possibile visualizzare il video).

PARTY I party includono gli utenti finali e i rights holder; possono essere persone singole, gruppi di persone, organizzazioni o ruoli definiti. Solitamente gli utenti finali sono coloro che vogliono usufruire dell'asset; mentre i rights holder di solito si identificano con le parti che hanno ricoperto un certo ruolo nella creazione, produzione e distribuzione dell'asset in questione e che quindi possono rivendicare una certa forma di proprietà intellettuale nei confronti dell'opera e dei permission applicati ad essa. In alcuni casi, i rights holder percepiscono delle percentuali di guadagno direttamente dal commercio del bene.

Grazie alle tre entità di base appena descritte, il Foundation Model di ODRL è in grado di esprimere le *offer* e gli *agreement*. Una offer consiste in una proposta, emessa dai rights holder, di permission che possono essere applicati su un determinato asset o gruppo di asset sotto certe condizioni, espresse tramite constraint, requirement e condition. Un agreement viene generato quando le figure all'interno del party decidono di accettare una specifica offer ed esprime il "contratto" risultante dall'accordo basato sulla offer presa in considerazione. In altre parole, l'agreement contiene tutti i permessi, con le relative restrizioni, che uno specifico utente finale o

gruppo di utenti può esercitare sull'asset in questione. Sia le offer che gli agreement possono subire delle revoche, previste nel modello ODRL.

La rappresentazione delle offer e degli agreement è un aspetto essenziale di ODRL, in quanto questi due elementi chiarificano quali sono i permessi che i detentori dei diritti d'autore concedono sulle opere e quali sono le condizioni che gli specifici utenti devono rispettare per poter usufruire delle opere.

Per uno stesso asset, possono essere create varie offer per andare incontro a diversi modelli di business; le offer possono essere legate tra di loro in modo da formate una gerarchia di opzioni per gli utenti finali. Un agreement consiste nell'effettiva trasformazione di una offer in una licenza vera e propria, che si riferisce a permessi applicabili su uno specifico asset da parte di uno specifico utente. Nonostante ciò, non esiste nessuna condizione per cui un agreement debba per forza derivare da una offer: essendo conseguenza di un'interazione tra parti umane, un agreement può essere generato per esprimere i termini e le condizioni accettate, senza essere necessariamente legato ad alcuna offer.

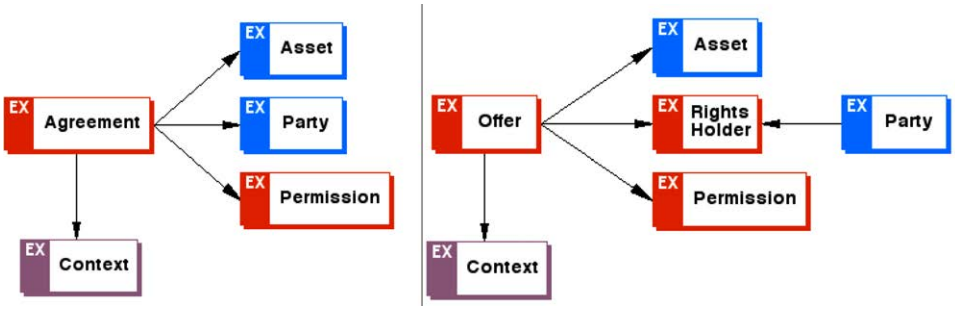

Figura 4. Modelli delle offer e degli agreement (Immagini estratte da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

La maggior parte delle entità del modello supportano l'eventuale presenza della sotto-entità *context*. Un context, che è sempre relativo all'entità nel quale è contenuto, può fornire ulteriori informazioni riguardo a quell'entità o alle relazioni esistenti tra l'entità in questione e le altre entità (es. il context di un agreement può riportare la data in cui sono stati presi gli accordi tra le parti).

Anche se non obbligatorio, l'utilizzo del context per assegnare un identificatore univoco alle entità è fortemente raccomandato. In particolare, nel caso della definizione di un identificatore specifico sia per una offer che per un agreement derivante dalla offer, è possibile fare riferimento all'identificatore per collegare l'agreement alla offer da cui ha avuto

origine.

Di seguito viene riportato un esempio in formato XML delle espressioni appartenenti al Foundation Model di ODRL.

```
<rights>
    <context>.
        <uid> ... </uid>
    </context>

    <offer>
        <asset> ... </asset>
        <permission>
            <permissiontype>
                <requirement> ... </requirement>
                <constraint> ... </constraint>
            </permissiontype>
            <condition> ... </condition>
        </permission>
        <party>
            <context> ... </context>
            <rightsholder> ... </rightsholder>
        </party>
    </offer>

    <agreement>
        <context> ... </context>
        <party> ... </party>
        <permission> ... </permission>
        <asset> ... </asset>
    </agreement>

</rights>
```

ODRL dedica una sezione apposita per la definizione delle "parole chiave" in linguaggio XML secondo le specifiche ODRL, la quale prende il nome di Data Dictionary (DD). Questa sezione racchiude l'ampia semantica messa a disposizione dal linguaggio ODRL per esprimere tutte le entità legate all'espressione dei diritti sugli asset e riguardanti la sfera della tutela del diritto d'autore e della proprietà intellettuale.

All'occorrenza, il Data Dictionary può essere esteso attraverso la creazione di nuovi profili ODRL che meglio si adattino alle esigenze e necessità specifiche per il loro utilizzo.

Dunque, gli schemi XML messi a disposizione da ODRL sono principalmente due: L'Expression Language, che contiene gli elementi di sintassi fondamentali per l'espressione tecnica dei permessi, dei vincoli, delle parti coinvolte e della natura delle risorse, e il Data Dictionary, che invece racchiude la semantica necessaria a riconoscere i costrutti più adatti per l'espressione delle specifiche entità in maniera univoca e inequivocabile. Grazie al DD, gli elementi del linguaggio ODRL possono essere riconosciuti automaticamente dai sistemi digitali dedicati alla gestione e all'elaborazione delle licenze.

Seguono le descrizioni delle principali entità del Foundational Model, ovvero permission, constraint, requirement, condition, rights holder e context. Per ciascuna di esse, viene riportato un esempio della sintassi ODRL che la descrive e un'immagine che ne rappresenta intuitivamente le caratteristiche, le sotto-entità e i legami con gli altri elementi del modello.

2.2.1 Permission Model

```
<permission>
    <display/>
    <print>
        <constraint> ... </constraint>
    </print>
    <annotate/>
</permission>
```

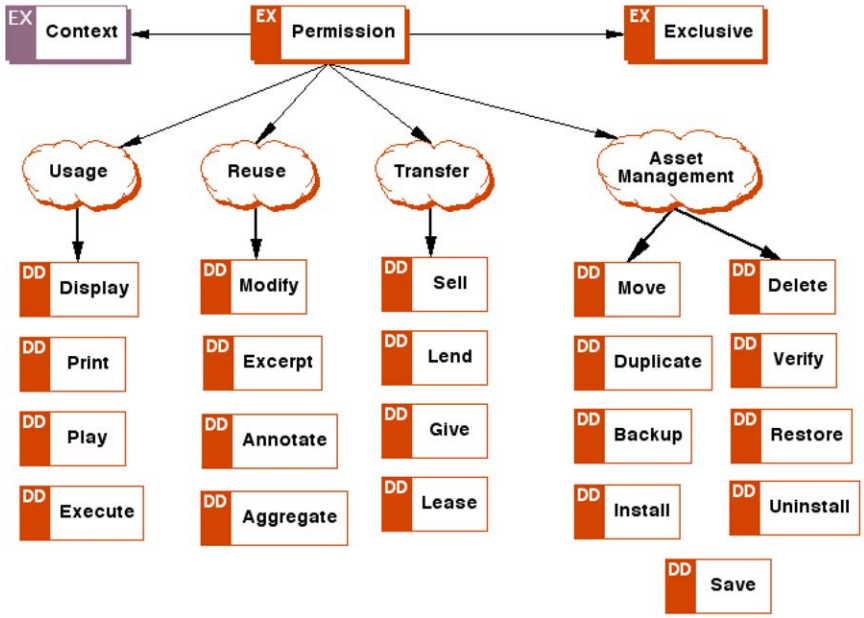

Figura 5. Modello dei permission (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

2.2.2 Constraint Model

```
<display>
    <constraint>
        <cpu/>
    </constraint>
</display>
<print>
    <constraint>
```

```
        <count>5</count>
    </constraint>
</print>
```

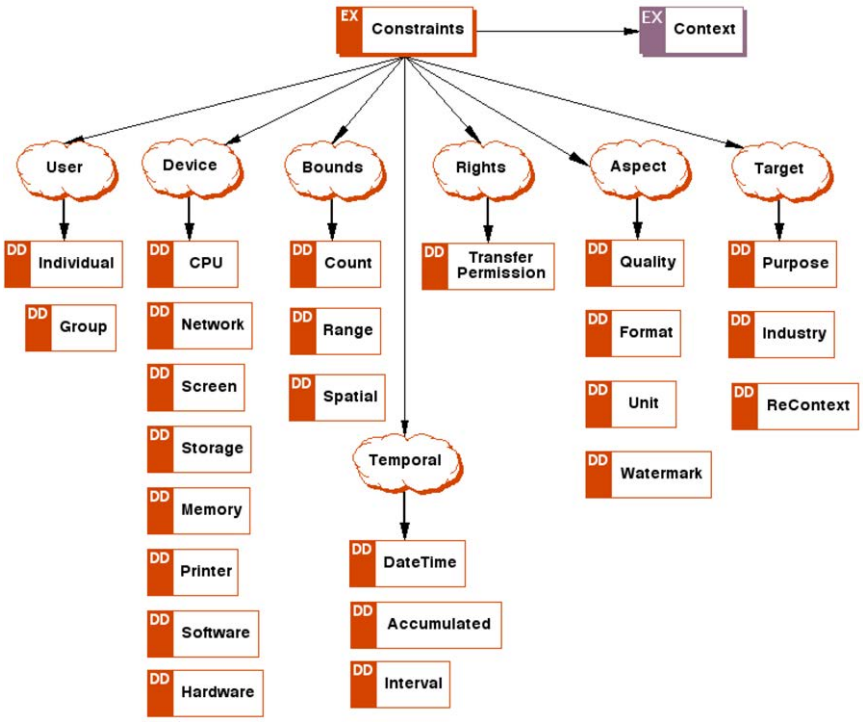

Figura 6. Modello dei constraint (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

2.2.3 Requirement Model

```
<play>
    <requirement>
        <peruse>
            <payment>
                <amount currency="AUD">20.00</amount>
                <taxpercent code="GST">10.0</taxpercent>
            </payment>
        </peruse>
    </requirement>
</play>
```

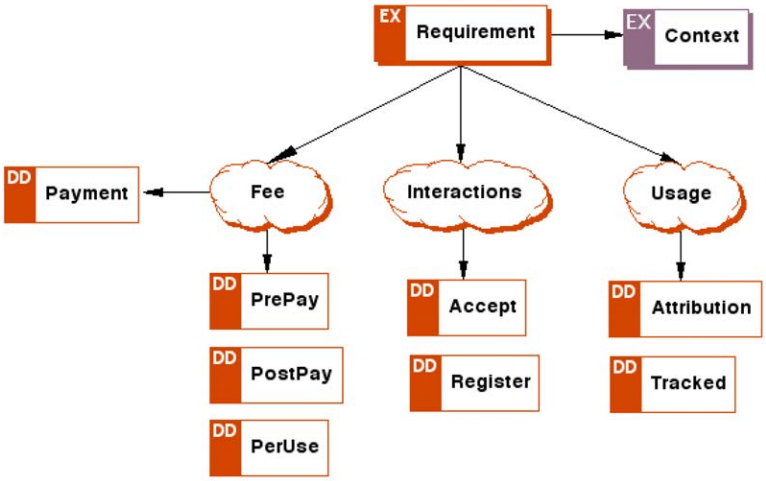

Figura 7. Modello dei requirement (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

2.2.4 Condition Model

```
<play>
    <condition>
        <constraint>
            <software> ... </software>
        <constraint/>
    </condition>
</play>
```

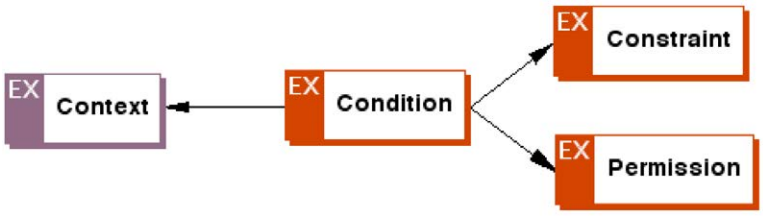

Figura 8. Modello dei condition (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

2.2.5 Rights Holder Model

```
<party>
    <context> ... </context>
    <rightsholder>
        <percentage>90</percentage>
    </rightsholder>
</party>
<party>
    <context> ... </context>
    <rightsholder>
        <percentage>10</percentage>
    </rightsholder>
</party>
```

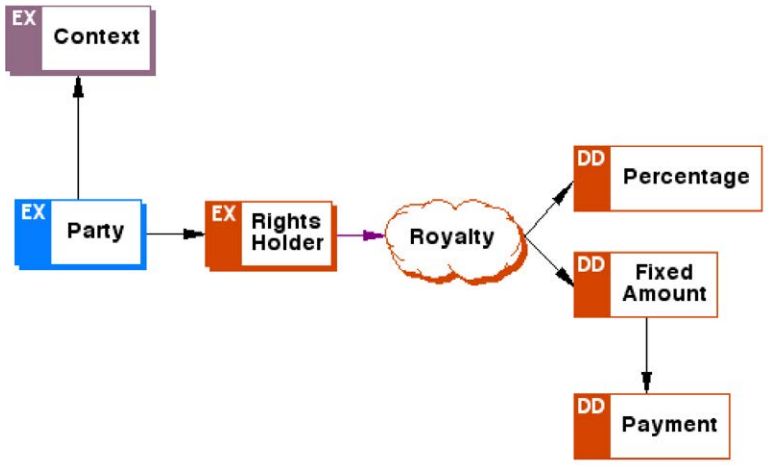

Figura 9. Modello dei rights holder (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

2.2.6 Context Model

```
<party>
    <context>
        <uid>x500:c=EX;o=FederalLibrary;cn=MariaKBrown</uid>
        <name>Maria Brown</name>
        <role>onix:AO1</role>
        <reference>http://www.kbrown.com/vcard.xml</reference>
```

```
    </context>
</party>
```

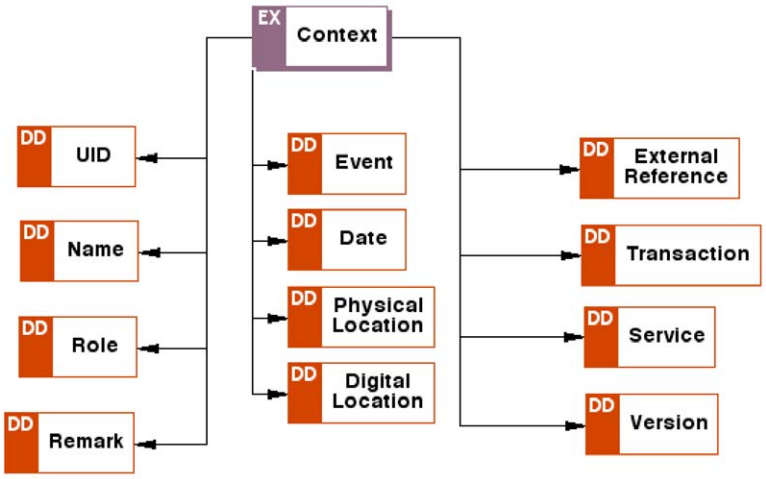

Figura 10. Modello dei context (Immagine estratta da "Open Digital Rights Language (ODRL) Version 1.1", consultare la sitografia)

3

L'INTERAZIONE TRA IL LINGUAGGIO ODRL E LO STANDARD IEEE1599

INDICE

Nel capitolo che segue verranno esposte le modalità di interazione tra il linguaggio ODRL per l'espressione di licenze e lo standard IEEE1599 per la creazione di un'esperienza musicale completa. Nello specifico, verrà descritto il profilo ODRL creato appositamente per la generazione di licenze relative a risorse musicali e adatte alla protezione dei diritti sugli asset coinvolti nell'esperienza musicale del framework IEEE1599. Successivamente, si presenteranno due possibili scenari per la creazione del documento IEEE1599 basato sulle licenze ODRL degli utenti e si riporteranno le motivazioni che hanno condotto alla scelta di quella utilizzata.

Un documento IEEE1599, come già anticipato nel primo capitolo, contiene al suo interno i riferimenti a risorse musicali di varia natura e mette a disposizione un'esperienza musicale capace di descrivere un brano sotto molti punti di vista diversi. Il formato IEEE1599 risulta quindi molto efficace per una ricca descrizione di un lavoro musicale, collegando tra di loro contenuti di diverso genere in grado di fornire all'utente una visione a tutto tondo del brano.

Dunque, lo standard IEEE1599 ha spesso a che fare con una grande quantità di risorse digitali di diverso genere, ciascuna delle quali può essere protetta o meno dal diritto d'autore e da licenze che ne consentono una fruizione limitata. Lo standard non si occupa della gestione dei vincoli esistenti sulla fruizione dei contenuti digitali a cui è legato, ma un'integrazione di questo tipo risulterebbe molto utile, poichè ciascuna risorsa possiede una specifica proprietà intellettuale, che si differenzia a seconda della natura intrinseca del contenuto e dei permessi stabiliti da colui che ne possiede la paternità.

Per poter gestire i diritti d'autore sulle risorse a cui un documento IEEE1599 fa riferimento e per poterne garantire la protezione attraverso licenze d'uso, è necessario fare affidamento sui sistemi di Digital Right Management e, in particolare, su un Right Expression Language. Infatti, un linguaggio REL è in grado di esprimere il contenuto delle licenze attraverso costrutti che ne permettano la gestione e l'elaborazione digitale, mediante sistemi automatici di controllo dei diritti e dei permessi applicati su ciascun asset. Attraverso l'utilizzo di questi linguaggi, è possibile garantire la protezione della grande varietà di contenuti e formati collezionati dallo standard IEEE1599, ognuno caratterizzato da permessi specifici.

Per la specifica dei diritti d'autore relativi ai contenuti digitali coinvolti nell'esperienza IEEE1599, si è scelto di utilizzare il linguaggio REL ODRL. Lo standard musicale necessita infatti di strumenti che si adattino alla sua estrema flessibilità, che gli permette di rappresentare in un singolo documento tutte le diverse modalità di fruizione della musica. Il sistema di gestione dei diritti di ODRL, basato anch'esso sul linguaggio XML, permette una notevole personalizzazione e flessibilità nell'espressione dei permessi relativi agli asset, rivelandosi quindi il linguaggio per la gestione dei diritti che rispecchia al meglio le esigenze del formato IEEE1599. Inoltre, ODRL è uno standard open-source ed è affiliato al W3C Group, che rappresenta una delle massime autorità nell'ambito dello sviluppo e della diffusione dei principali standard per il World Wide Web; queste caratteristiche lo rendono appropriato all'ampliamento di uno standard che nasce come ricerca universitaria.

3.1 IL PROFILO ODRL DI IEEE1599

Il linguaggio ODRL, come anticipato nel capitolo precedente, mette a disposizione dei meccanismi di estensione della semantica del linguaggio, attraverso i quali è possibile aggiungere delle funzionalità dedicate allo specifico scopo per il quale si è deciso di impiegare ODRL come sistema per l'espressione dei diritti sulle risorse. L'estensione delle funzionalità di ODRL avviene attraverso la definizione di nuovi *profili ODRL*, in cui vengono specificate le nuove espressioni semantiche che si vogliono incorporare nel linguaggio.

Uno dei progetti di tesi sviluppati al *Laboratorio di Informatica Musicale* di Milano, realizzato dall'ex studentessa Agnese Farinelli, si è concentrato proprio sullo sviluppo di un profilo ODRL 1.1 in grado di estendere

le funzionalità di ODRL: l'obiettivo era adattare l'espressione dei diritti alle esigenze specifiche del formato musicale, che non risultano soddisfatte dalla semantica di base del linguaggio REL di ODRL. L'estensione del linguaggio per la gestione dei diritti è stata realizzata in maniera dissociata dagli originali Expression Language e Data Dictionary di ODRL, così da consentirne una migliore portabilità. In questo modo, il profilo ODRL sviluppato per lo standard IEEE1599 consiste di file separati rispetto ai documenti originali di ODRL disponibili sul sito ufficiale e basterà essere in possesso di questi file per soddisfare le esigenze del nuovo profilo IEEE1599.

Il profilo ODRL per lo standard IEEE1599 è stato rilasciato come *XML Schema*; infatti, tutti i frammenti di codice scritti con linguaggio XML necessitano di essere validati per assicurarsi che i dati e le strutture contenuti nel codice soddisfino precisi criteri: un XML Schema viene utilizzato per validare il documento XML in questione, garantendo che tutti i valori degli elementi e degli attributi specificati nel codice seguano determinate regole. Proprio per la loro versatilità e per il fatto che la loro tecnologia viene già sfruttata da ODRL per definire i suoi EX e DD, gli XML Schema sono stati impiegati per la creazione del profilo IEEE1599.

Per la costruzione del nuovo profilo IEEE1599, si utilizza la proprietà dei *substitution group*. Il substitution group è un costrutto che permette di specificare un certo gruppo di elementi che può sostituire un altro elemento all'interno del documento XML. In altre parole, un substitution group consente di specificare una collezione di elementi ai quali è possibile riferirsi usando al loro posto un elemento generico. Nel caso in cui sia presente nello Schema un gruppo di elementi tra loro in relazione ed essi siano a tutti gli effetti interscambiabili all'interno di un'istanza, allora è possibile fare riferimento al substitution group per quegli elementi all'interno dell'intero Schema; infatti, all'interno di un substitution group, tutti gli elementi in esso contenuti possono essere utilizzati ciascuno al posto di un altro senza che il documento XML perda di significato e la possibilità di essere correttamente validato.

Attraverso l'utilizzo dei substitution group, è possibile aggiungere nuovi elementi al linguaggio senza intaccare i preesistenti XML Schema di ODRL 1.1, all'interno dei quali esiste già un'ottima definizione degli elementi di base necessari per la dichiarazione dei diritti. I substitution group del profilo IEEE1599 sono stati creati inserendo tipologie di elementi ispirati allo standard IEEE1599, ma che fanno riferimento ad un elemento di testa già definito nel linguaggio standard di ODRL.

Attraverso l'analisi dei costrutti utilizzati nello standard IEEE1599 per la rappresentazione degli elementi, nel profilo ODRL di IEEE1599 sono stati estesi i seguenti elementi corrispondenti:

- *eventBase:* rappresenta l'evento musicale, ovvero la più piccola risorsa presa in considerazione dal formato.
- *fragmentBase:* rappresenta una collezione di eventi musicali, sia che si tratti di una serie di eventi elencati dal primo all'ultimo, sia che si tratti semplicemente di una successione di eventi della quale però si rendono noti soltanto il primo e l'ultimo della lista: tale rappresentazione risulta possibile poiché, all'interno dei layer audio e notational, gli eventi musicali sono sempre enumerati secondo un'ordine prestabilito che non può essere modificato, così da garantirne la posizione e la successione in qualsiasi circostanza.
- *scoreBase e trackBase:* ereditano dal tipo fragmentBase tutte le sue caratteristiche, estendendole per comprendere anche quelle esclusive di tracce e partiture.
- *partBase:* esplicita le parti dedicate ai differenti strumenti musicali di cui solitamente si compongono le partiture; partBase estende fragmentBase poichè potrebbe racchiudere in sé le informazioni inerenti a quali eventi musicali compongono quella specifica parte di spartito.
- *pageBase:* descrive la natura degli elementi di pagina nelle quali è suddiviso lo spartito e fa parte dell'elemento scoreBase; pageBase non estende il tipo fragmentBase poiché lo spartito contiene già tutti i riferimenti necessari alla connotazione di una sequenza di eventi musicali: l'informazione riguardante la pagina rappresenta solamente un arricchimento della descrizione relativa allo spartito.
- *audioBase:* rappresenta il comparto audio come l'insieme di tutte le tracce audio, nella loro interezza, contenute nel formato: arricchisce la definizione di fragmentBase aggiungendo al tipo una sequenza di elementi "track", pensati per indicare tutte le tracce presenti nel comparto audio.
- *scoresBase:* rappresenta la collezione di spartiti come l'insieme di tutte le partiture, nella loro interezza, contenute nel formato: aggiunge alla definizione del tipo fragmentBase una sequenza di elementi score, ciascuno dei quali riservato alla descrizione di una delle tante partiture disponibili per il brano preso in considerazione.

- *containerType:* è il mezzo adeguato per esprimere raggruppamenti di entità accomunate da un medesimo scopo od utilizzo; si è scelto di estendere questo tipo complesso nell'ambito del profilo IEEE1599, così da rendere possibile l'aggregazione di più asset all'interno di un elemento container.

- *assetType:* descrive l'elemento asset e si è scelto di estenderlo per consentire a questo tipo di ospitare come sotto-elementi entità di tipo containerType.

Una volta definita la rappresentazione dei contenuti digitali all'interno del profilo IEEE1599, è stata creata una nuova terminologia per l'espressione dei permission, dei constraint e delle altre entità del linguaggio ODRL, in modo da renderle più adatte alle necessità del formato per una precisa descrizione del diritto d'autore. A questo scopo, sono stati definiti i seguenti elementi:

- *synchronization:* permission che indica il permesso rilasciato dal rights holder di fruire del brano musicale, attraverso gli asset riportati nella licenza, in maniera sincrona; l'elemento synchronization è ti dipo permissionType ed è contemporaneamente membro del substitution group dell'elemento permissionElement.

- *nuberOfPages:* constraint pensato per limitare il numero delle pagine di uno spartito; estende il tipo constraintType.

- *numberOfSeconds:* constraint che punta a limitare la fruizione dell'asset all'interno di un arco temporale stabilito e contato in secondi; estende il tipo constraintType.

- *numberOfMeasures:* constraint che limita qualsiasi permission applicato al bene ad una serie di battute prestabilite; estende il tipo constraintType.

- *qualityOfResource:* constraint che si occupa di limitare il consumo della risorsa a seconda del livello di qualità espresso dal vincolo

- *EventFamily:* constraint che indica la tipologia di eventi ammessi dal vincolo.

3.2 ANALISI E CONFRONTO DELLE ARCHITETTURE PER LA GESTIONE DELLE LICENZE

Per poter gestire le licenze ODRL all'interno del player IEEE1599, è necessario sviluppare un meccanismo in grado di applicare i vincoli espressi nella licenza al contenuto del documento IEEE1599, in modo da dare la possibilità all'utente di eseguire e visualizzare soltanto le risorse multimediali per le quali possiede i permessi.

Nel caso del formato IEEE1599, i permessi da gestire non si limitano soltanto all'esecuzione di file audio o video e alla visualizzazione di partiture, ma comprendono anche il permesso di sincronizzazione tra contenuti multimediali, che siano della stessa natura o meno. La gestione del permesso di sincronizzazione deve tenere conto non solo dei vincoli sulla sincronizzazione in sé, ma anche degli altri permessi coinvolti nella sincronizzazione stessa e dei differenti limiti di fruizione applicati ad ogni risorsa.

Dunque occorre escogitare un sistema che sia in grado di coniugare tra di loro i vincoli sui diversi permessi coinvolti, per fornire un'esperienza musicale che rispetti i diritti d'autore relativi ai contenuti multimediali contenuti nel documento IEEE1599.

Per la progettazione di un sistema di questo tipo, sono state considerate *due possibili architetture*, entrambe in grado di gestire l'utilizzo di risorse per le quali si possiede la licenza e contemporaneamente coordinare la sincronizzazione tra di loro attraverso il formato IEEE1599. L'idea di base ad entrambe le proposte prevede di controllare il permesso di sincronizzazione agendo direttamente sui materiali forniti all'utente: per rispettare i vincoli sulla sincronizzazione tra contenuti, all'utente vengono offerti esclusivamente i dati per i quali è autorizzato all'accesso.

Entrambe le proposte prevedono un'elaborazione lato server del documento IEEE1599 completo, per poi generare il documento specifico per l'utente secondo due diverse maniere:

- nel primo caso, il documento IEEE1599 completo viene analizzato ed elaborato, e le sotto-parti per le quali l'utente non ha i permessi di accesso vengono rimosse;
- nel secondo caso, viene generato un nuovo documento IEEE1599 a partire dal documento completo, prelevando dal secondo e inseren-

do nel primo solamente le sotto-parti di cui l'utente è autorizzato a usufruire.

Le architetture proposte contengono le seguenti componenti:

- *IEEE1599 Server:* si occupa dell'autenticazione e dei processi di valutazione dei servizi richiesti; per poter eseguire il suo lavoro, ha bisogno di ricevere le risorse necessarie dagli altri server, in modo da poterle elaborare e da poter generare le risorse richieste da inviare al lato client.
- *Licence Server:* gestisce le licenze ODRL e risponde alle domande di accesso ad esse effettuate dal server IEEE1599 per soddisfare le richieste del client.
- *Content Repository Server:* si preoccupa della gestione delle risorse digitali e risponde alle domande di accesso ad esse effettuate dal server IEEE1599 per soddisfare le richieste del client.
- *IEEE1599 Repository Server:* si dedica alla gestione dei documenti IEEE1599 e risponde alle domande di accesso ad esse effettuate dal server IEEE1599 per generare il documento IEEE1599 finale ad-hoc al fine di soddisfare le richieste del client.

Il client si interfaccia con l'IEEE1599 Server, che si occupa di selezionare i tipi di materiale che dovrà inviare al client. Il server IEEE1599 recupera la licenza dell'utente dal Licence Server e il documento IEEE1599 adatto dall'IEEE1599 Repository Server. L'IEEE1599 Repository Server restituisce, nel caso della prima architettura proposta, il documento IEEE1599 completo oppure, nel caso della seconda architettura proposta, uno scheletro del documento IEEE1599, che dovrà essere riempito in seguito. Successivamente, in entrambi i casi, l'IEEE1599 Server recupera le risorse digitali dal Content Repository Server attenendosi alle informazioni specificate del documento IEEE1599 e nella licenza ODRL dell'utente recuperate precedentemente.

A questo punto, nel secondo caso l'IEEE1599 Server richiede all'IEEE1599 Repository Server le sotto-parti necessarie per completare lo scheletro del documento IEEE1599. Quindi, l'IEEE1599 Server genera un nuovo documento IEEE1599 contenente sia i dati necessari per la sincronizzazione dei contenuti che le informazioni che regolano l'accesso ai contenuti secondo la licenza.

A seconda del caso considerato, dunque, il documento IEEE1599 finale viene generato attraverso uno dei due seguenti metodi:

- rimuovendo dal documento IEEE1599 completo le sotto-parti relative ai contenuti ai quali l'utente non può accedere;
- inserendo nello scheletro del documento IEEE1599 le sotto-parti che gestiscono la sincronizzazione esclusivamente per i contenuti ai quali l'utente può accedere.

Infine, viene generato un pacchetto di dati che viene inviato al client.

Si mostrano di seguito i due possibili scenari rappresentati sotto forma di schemi di flusso: la *Figura 11* riproduce la prima architettura proposta, mentre la *Figura 12* riproduce la seconda architettura proposta.

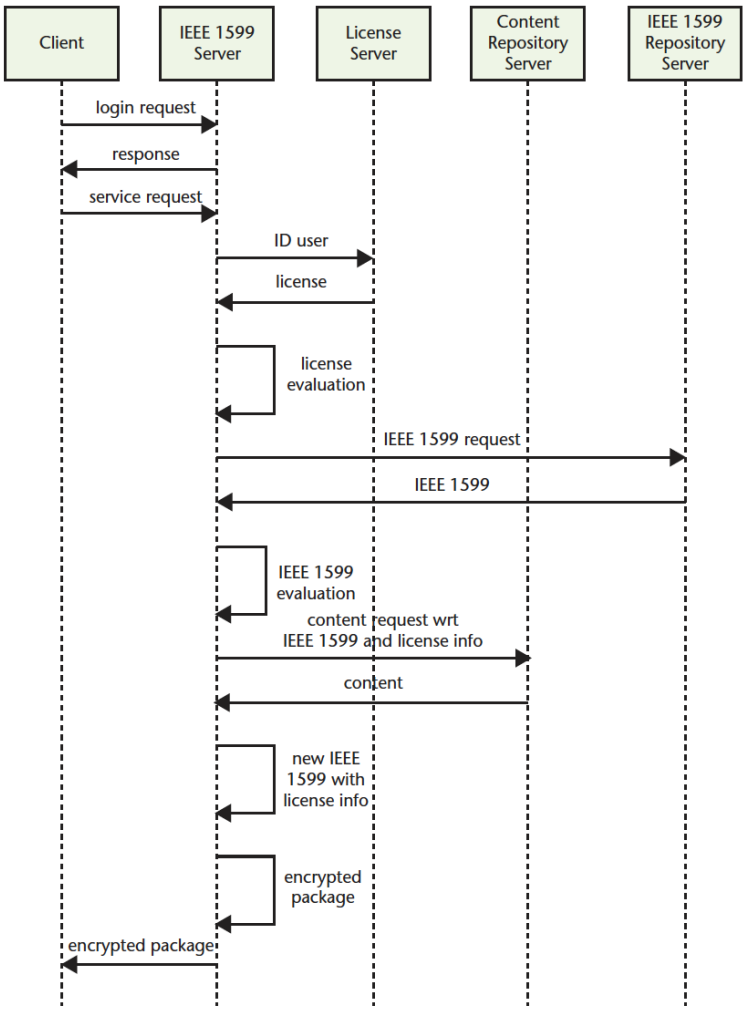

Figura 11. Prima architettura proposta (Immagine estratta da "Managing Intellectual Property in a Music Fruition Environment", consultare la bibliografia)

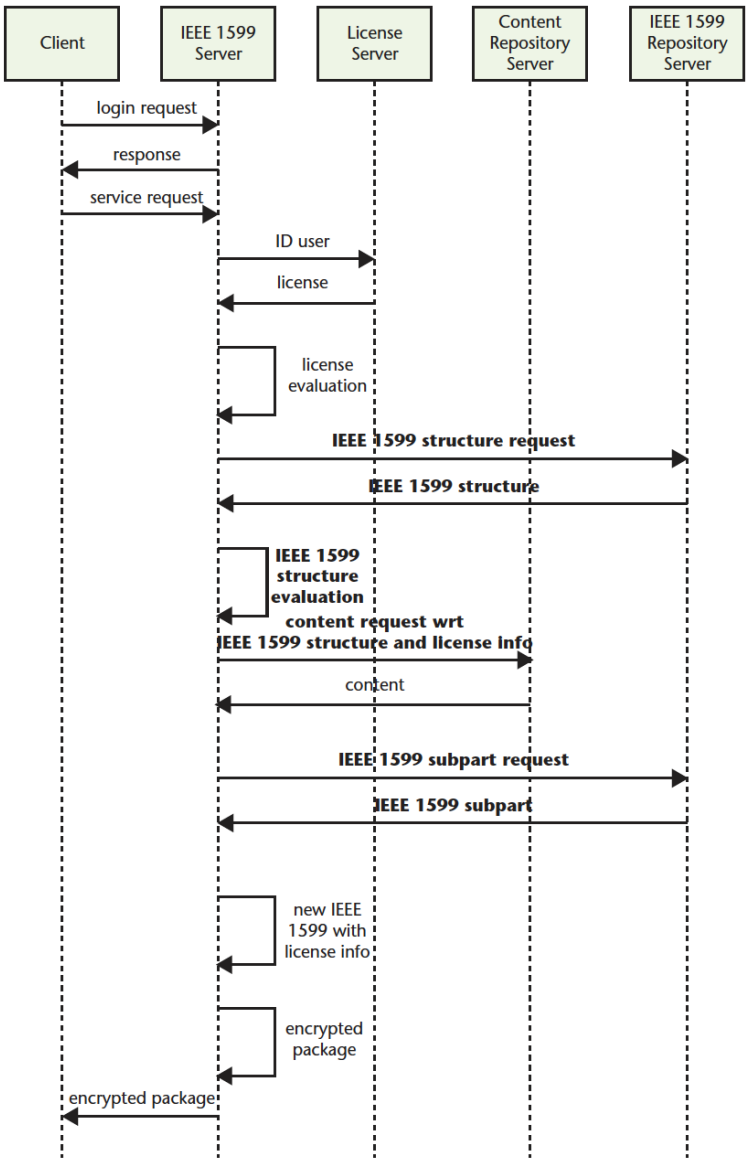

Figura 12. Seconda architettura proposta (Immagine estratta da "Managing Intellectual Property in a Music Fruition Environment", consultare la bibliografia)

4 PROGETTAZIONE DELLA BASE DI DATI

INDICE

In questo capitolo si guiderà il lettore attraverso le scelte progettuali che hanno portato alla strutturazione e alla generazione della base di dati per la piattaforma web IEEE1599. Nel dettaglio, verranno descritte le fasi di progettazione concettuale, logica e fisica che sono state necessarie per prendere decisioni sulle modalità di salvataggio delle informazioni e per implementare il database attraverso il linguaggio SQL.

La piattaforma web IEEE1599 necessita di lavorare con diverse tipologie di dati e di informazioni per poter offrire le sue funzionalità: deve poter gestire file multimediali quali audio, video e immagini per gli spartiti; richiede l'importazione di file XML per la fruizione dell'esperienza IEEE1599 e per la lettura delle licenze; ha bisogno, inoltre, di poter immagazzinare i dati relativi agli utenti e altre informazioni accessorie per la gestione dei loro profili.

Per questo motivo, è stata realizzata una base di dati in grado di memorizzare in modo pratico ed efficiente tutte le informazioni essenziali per il corretto funzionamento del sito. In particolare, l'architettura del database è stata ideata con il fine di organizzare i dati in modo da facilitare la gestione automatizzata dei diritti degli utenti e delle licenze in vigore sui brani musicali.

A questo proposito, come modello per la rappresentazione dei dati si è scelto il *modello relazionale*, che è generalmente il più diffuso tra i DBMS (DataBase Management System). Infatti, il modello relazionale fornisce le

strutture opportune per la rappresentazione di diverse entità, caratterizzate da alcune informazioni, e per l'espressione delle associazioni esistenti tra di esse.

Nel seguente capitolo verranno descritte le diverse fasi che hanno portato alla strutturazione e alla creazione del database e le scelte progettuali che hanno determinato le modalità di salvataggio delle informazioni.

4.1 PROGETTAZIONE CONCETTUALE

Come modello per la progettazione concettuale della base di dati è stato utilizzato il *modello Entità-Relazione*, il quale possiede una tipologia di rappresentazione semplice e ricca semanticamente, basata sull'utilizzo di metodi sia grafici che linguistici. La rappresentazione grafica di uno schema concettuale ER, detta *diagramma ER*, consente di esprimere in modo chiaro ed intuitivo i costrutti tipici di questo modello, tra cui i principali sono *entità*, *associazione* e *attributo*. Attraverso i tre costrutti di base dello schema e alcune loro estensioni, è possibile rappresentare tutti i concetti che devono essere inseriti nella base di dati.

Come primo passo per la progettazione concettuale, è stato necessario individuare i concetti essenziali per la creazione di uno scheletro sul quale costruire la base di dati. In altre parole, l'analisi dei requisiti e delle informazioni necessarie alla piattaforma web per offrire i suoi servizi ha portato all'identificazione di alcuni elementi fondamentali e delle relazioni che devono esistere tra di loro.

Successivamente, è stato deciso quali costrutti del modello ER utilizzare per modellare i vari concetti ricavati dall'analisi dei requisiti. Questa fase assegna un costrutto ad ogni elemento da modellare, basandosi sulle esigenze e necessità espresse nei requisiti.

Infine, si è giunti alla generazione dello schema ER, il quale è stato costruito attraverso una *strategia top-down*: partendo da uno schema semplice ed essenziale, con pochi costrutti che descrivono solo i concetti fondamentali, questo è stato arricchito aggiungendo sempre maggiori dettagli, fino a rappresentare tutte le informazioni richieste dalle specifiche.

4.1.1 Analisi dei requisiti

Il player web IEEE1599 necessita di memorizzare diverse tipologie di dati, tra cui informazioni riguardanti brani musicali, file audio, video e grafici

(spartiti), licenze, vincoli, dati degli utenti e informazioni di controllo. Gli elementi fondamentali e i dati legati ad essi, individuati attraverso l'analisi dei requisiti, sono i seguenti:

BRANO MUSICALE: la piattaforma web contiene un'ampia libreria di *brani musicali*, ciascuno caratterizzato da alcune informazioni, quali *titolo*, *autore* e *anno di creazione*; ad ogni brano sono legati anche un'*immagine* e un *documento IEEE1599*. Ciascun brano musicale possiede una serie di contenuti multimediali, i quali possono essere *audio*, *video* o *partiture*; per ciascuno di essi, a loro volta, è necessario memorizzare alcuni dati, come il *formato*, la *qualità*, la *durata*, l'*anno di creazione*, il *nome dell'autore o interprete* e, nel caso delle immagini delle partiture, il *numero di pagina*.

OFFER: ciascun brano musicale necessita di un documento ODRL che esprima la licenza ad esso applicata, nel quale, dunque, vengano esplicitate le *offer* relative al brano.

UTENTE: per poter consentire agli utenti di registrarsi e di accedere al sito con le proprie credenziali, la piattaforma web deve memorizzare nella base di dati le informazioni relative a ciascun *utente*, quali *mail*, *username*, *password*, *nome*, *cognome*, *genere*, *data di nascita* e *città*. Prendendo in considerazione la possibilità di creare diversi tipi di abbonamenti standard da sottoscrivere, tra i quali l'utente può scegliere, è opportuno salvare i dati relativi alla *categoria* a cui appartiene ciascun utente, tra cui il *tipo di abbonamento* e la *data di inizio e di fine* della sua validità.

AGREEMENT: l'*agreement* è un elemento essenziale per la costruzione del database, poichè racchiude in un documento ODRL i permessi, i limiti, i requisiti e le condizioni relazionate alla fruizione di uno specifico brano da parte di uno specifico utente, a seconda della licenza valida sul brano e del tipo di abbonamento sottoscritto o degli acquisti effettuati dall'utente.

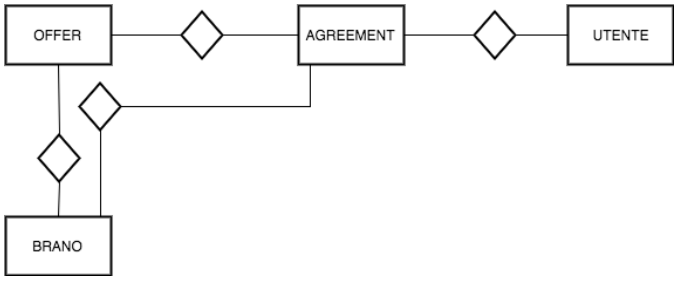

Figura 13. Legami tra gli elementi di base del database

4.1.2 Scelta dei costrutti

La scelta dei costrutti da utilizzare per modellare i diversi concetti individuati dipende da un'attenta analisi dei requisiti, per poter comprendere il ruolo che ogni elemento avrà nel sito.

Se un concetto presente nei requisiti descrive un insieme omogeneo di oggetti, appartenenti allo stesso dominio e aventi un insieme di proprietà comuni, allora il costrutto più idoneo per la sua rappresentazione nello schema ER è quello di *entità*.

Se un concetto esprime un concetto elementare, senza alcuna proprietà associata, allora il costrutto più idoneo per la sua rappresentazione è quello di *attributo*.

Se un concetto denota un legame logico tra concetti che sono stati modellati come entità, allora il costrutto più idoneo per la sua rappresentazione è quello di *associazione*.

Nel caso della base di dati per il player web IEEE1599, gli elementi sono stati modellati come segue:

BRANO Il concetto di brano musicale è stato modellato con l'entità BRANO, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
id_brano	una chiave surrogata per identificare le istanze dell'entità	String
titolo	il titolo del brano musicale	String
autore	l'autore del brano musicale	String
anno	l'anno a cui risale il brano musicale	Integer
ref_img	il path dove è salvata l'immagine legata al brano musicale	String
ref_ieee1599	il path dove è salvato il documento IEEE1599 relativo al brano musicale	String

Ciascun brano musicale, ovvero ciascuna istanza dell'entità BRANO, può essere legato ad una serie di contenuti multimediali, rappresentati da file audio, video e immagini (partiture); ogni tipologia di file multimediale è stata descritta con un'entità separata.

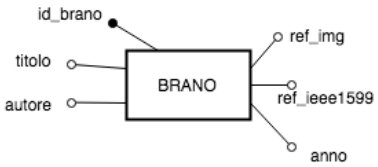

Figura 14. Entità brano

AUDIO Il concetto di traccia audio è stato modellato con l'entità `AUDIO`, caratterizzata da alcuni attributi:

Attributo	**Semantica**	**Tipo**
id_audio	una chiave surrogata per identificare le istanze dell'entità	String
nome_file	il nome con cui è salvato il file audio	String
interprete	l'interprete della performance	String
anno	l'anno a cui risale la traccia audio	Integer
durata	la durata della traccia audio	Time
qualità	la qualità della traccia audio	Char
formato	il formato del file audio	String

Un'istanza appartenente a questa entità equivale ad un file audio memorizzato nella directory "AUDIO" che si trova all'interno della directory relativa al brano musicale in questione.

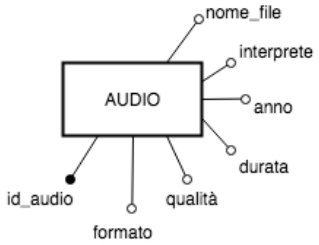

Figura 15. Entità audio

VIDEO Il concetto di video è stato modellato con l'entità `VIDEO`, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
id_video	una chiave surrogata per identificare le istanze dell'entità	String
nome_file	il nome con cui è salvato il file video	String
interprete	l'interprete della performance	String
anno	l'anno a cui risale il video	Integer
durata	la durata del video	Time
qualità	la qualità del video	Char
formato	il formato del file video	String

Un'istanza appartenente a questa entità equivale ad un file video memorizzato nella directory "AUDIO" che si trova all'interno della directory relativa al brano musicale in questione.

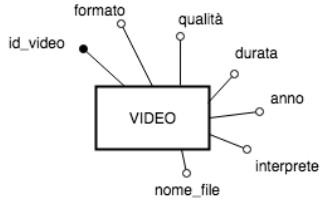

Figura 16. Entità video

PARTITURA Il concetto di partitura è stato modellato con l'entità `PARTITURA`, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
id_partitura	una chiave surrogata per identificare un'intera partitura	String
pag	il numero di pagina della partitura	Integer
nome_file	il nome con cui è salvata l'immagine	String
autore	l'autore che ha scritto la partitura	String
anno	l'anno a cui risale la partitura	Integer
qualità	la qualità dell'immagine	Char
formato	il formato dell'immagine	String

Un'istanza appartenente a questa entità equivale ad un'immagine rappresentante una singola pagina di una partitura, memorizzata nella directory "SCORES" che si trova all'interno della directory relativa al brano musicale in questione. Ogni istanza è identificata dalla coppia `id_partitura + pag`.

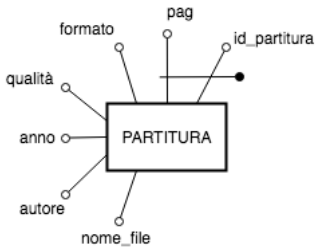

Figura 17. Entità partitura

UTENTE Il concetto di utente è stato modellato con l'entità `UTENTE`, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
mail	la mail dell'utente, che ne identifica l'istanza	String
username	lo username scelto dall'utente per l'autenticazione	String
password	la password scelta dall'utente per l'autenticazione	String
nome	il nome dell'utente	String
cognome	il cognome dell'utente	String
genere	il genere dell'utente	Char
data_n	la data di nascita dell'utente	Date
città	la città in cui l'utente abita	String
accessi	il numero di accessi contemporanei effettuati dall'utente	Integer

Un'istanza appartenente a questa entità equivale ad un utente registrato al sito. Un utente può aver sottoscritto un tipo di abbonamento tra quelli standard: in questo caso appartiene ad una categoria e devono essere memorizzate alcune informazioni a riguardo. Si è scelto di modellare il concetto di categoria, legato singolarmente a ciascun utente, attraverso un'entità a parte.

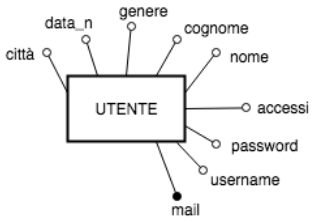

Figura 18. Entità utente

CATEGORIA Il concetto di categoria è stato modellato con l'entità `CATEGORIA`, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
tipo	il tipo di categoria al quale l'utente appartiene, che dipende dall'abbonamento da lui sottoscritto	String
data_inizio	la data a partire dalla quale l'abbonamento è valido	Date
data_fine	la data fino alla quale l'abbonamento è valido	Date
ultima_modifica	timestamp dell'ultima modifica applicata all'abbonamento	Timestamp
ref_certificato	il path dove è salvato il documento relativo alla sottoscrizione dell'abbonamento	String

Un'istanza appartenente a questa entità equivale ad una tupla contenente informazioni che riguardano un singolo utente appartenente ad una certa categoria. L'entità `CATEGORIA` ha come chiave primaria la chiave esterna `mail` di `UTENTE`; infatti in `CATEGORIA` può esistere al massimo una tupla per ogni utente in `UTENTE`.

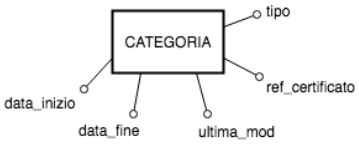

Figura 19. Entità categoria

OFFER Il concetto di licenza di un brano è stato modellato con l'entità OFFER, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
ultima_modifica	timestamp dell'ultima modifica applicata al documento che esprime la licenza	Timestamp
ref_offer	il path dove è salvato il documento relativo alla licenza	String

Un'istanza appartenente a questa entità equivale ad un documento ODRL che esprime la licenza applicata ad un singolo brano. L'entità OFFER ha come chiave primaria la chiave esterna id_brano di BRANO; infatti in OFFER può esistere al massimo una tupla per ogni brano in BRANO.

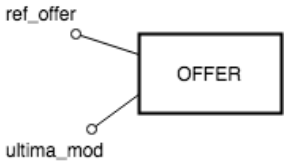

Figura 20. Entità offer

AGREEMENT Il concetto di agreement di un brano è stato modellato con l'entità `AGREEMENT`, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
ultima_modifica	timestamp dell'ultima modifica applicata al documento che esprime l'agreement	Timestamp
ref_agreement	il path dove è salvato il documento relativo all'agreement	String

Un'istanza appartenente a questa entità equivale ad un documento ODRL che esprime l'agreement relativo ad una singola coppia brano-utente. L'entità `AGREEMENT` ha come chiave primaria la coppia di chiavi esterne `id_brano` di `BRANO` e `mail` di `UTENTE`; infatti in `AGREEMENT` può esistere al massimo una tupla per ogni coppia di brano in `BRANO` e `utente` in `UTENTE`.

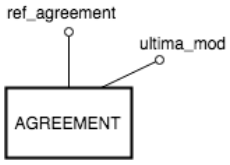

Figura 21. Entità agreement

Le condizioni espresse nell'agreement necessitano di memorizzare alcune informazioni accessorie relative ai vincoli applicati sulla fruizione di uno specifico brano da parte di uno specifico utente. Per questo motivo, sono state create alcune entità legate ad `AGREEMENT`, che salvano le informazioni necessarie per ciascun vincolo contenuto nell'agreement.

VINCOLO Per memorizzare le informazioni accessorie relative ai vincoli contenuti negli agreement, è stata modellata l'entità `VINCOLO`, caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
id_vincolo	una chiave surrogata per identificare le istanze dell'entità	String
tipo	il tipo di permesso su cui è applicato il vincolo	String
asset	l'asset su cui è applicato il vincolo (il nome del file)	String
permission	il permission al quale appartiene il vincolo	String
valido	un booleano che indica se il vincolo è valido o meno	Boolean

Un'istanza appartenente a questa entità equivale ad un elemento di uno specifico permission contenuto in uno specifico agreement. Le tuple dell'entità `VINCOLO` avrebbero potuto essere identificate dall'insieme di attributi `tipo`, `permission` e dalla chiave primaria di `AGREEMENT`, ma si è scelto di utilizzare come chiave primaria di `VINCOLO` una chiave surrogata per comodità.

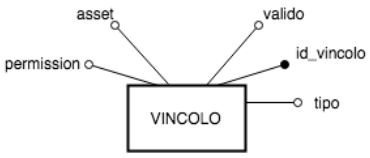

Figura 22. Entità vincolo

Per ciascun elemento di questa entità, vengono memorizzati i dati di utilizzo relativi all'utente a cui è legato l'agreement contenente il vincolo, i quali sono stati modellati tramite entità separate.

TEMPO L'entità TEMPO modella i constraint di tipo temporale ed è caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
data_inizio	la data a partire dalla quale l'elemento del permission ha validità	Date
data_fine	la data fino alla quale l'elemento del permission ha validità	Date

L'entità TEMPO ha come chiave primaria la chiave esterna id_vincolo di VINCOLO; infatti in TEMPO può esistere al massimo una tupla per ogni elemento in VINCOLO.

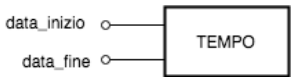

Figura 23. Entità tempo

UTILIZZO L'entità UTILIZZO modella i constraint di tipo cumulativo ed è caratterizzata da un solo attributo:

Attributo	Semantica	Tipo
num	il numero di volte che l'utente ha già fruito dell'asset	Integer

L'entità UTILIZZO ha come chiave primaria la chiave esterna id_vincolo di VINCOLO; infatti in UTILIZZO può esistere al massimo una tupla per ogni elemento in VINCOLO.

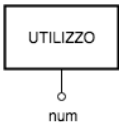

Figura 24. Entità utilizzo

PAGAMENTO L'entità `PAGAMENTO` modella i requirement riguardanti il pagamento ed è caratterizzata da alcuni attributi:

Attributo	Semantica	Tipo
data_inizio	la data a partire dalla quale il pagamento ha validità	Date
data_fine	la data fino alla quale il pagamento ha validità	Date
ref_ricevuta	il path dove è salvata la ricevuta del pagamento	String

L'entità `PAGAMENTO` ha come chiave primaria la chiave esterna `id_vincolo` di `VINCOLO`; infatti in `PAGAMENTO` può esistere al massimo una tupla per ogni elemento in `VINCOLO`.

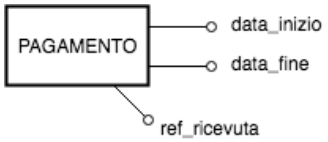

Figura 25. Entità pagamento

4.1.3 Generazione del diagramma ER

Il processo di generazione del diagramma ER, come già anticipato, è stato eseguito mediante un approccio top-down, dunque partendo da uno schema con pochi costrutti e con le informazioni essenziali, fino ad arrivare ad uno schema completo di tutti gli elementi da rappresentare.

Per la costruzione dello schema ER, si è partiti dalla rappresentazione delle entità `BRANO`, `AUDIO`, `VIDEO` e `PARTITURA`. Per ciascun brano musicale, è possibile avere da zero ad un numero indefinito di file audio, file video e file grafici di partiture; per questo motivo, è stata creata un'associazione dall'entità `BRANO` a ciascuna delle altre tre entità. In tutti e tre i casi, l'entità `BRANO` può prendere parte o meno all'associazione e può partecipare N volte; mentre l'altra entità deve partecipare obbligatoriamente e al massimo una volta all'associazione, poichè ciascun file può appartenere ad un solo brano.

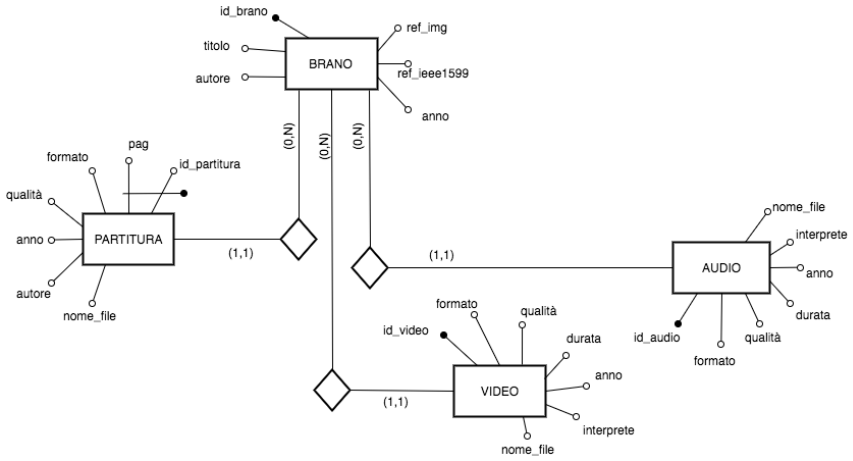

Figura 26. Costruzione dello schema ER: fase 1

Successivamente, sono state descritte le entità `UTENTE` e `CATEGORIA`. Ogni utente può aver sovrascritto un abbonamento, in base al quale appartiene ad una certa categoria, oppure può non aver sovrascritto alcun abbonamento e quindi non appartenere ad alcuna categoria. Di conseguenza, è necessaria un'associazione tra l'entità `UTENTE` e l'entità `CATEGORIA`: ciascun istanza di `UTENTE` può decidere se partecipare o meno, ma può partecipare al massimo una volta, poichè un utente non può sovrascrivere più abbonamenti contemporaneamente; ciascun istanza di `CATEGORIA`, invece, deve partecipare obbligatoriamente e al massimo una volta, poichè contiene informazioni che si riferiscono al singolo utente.

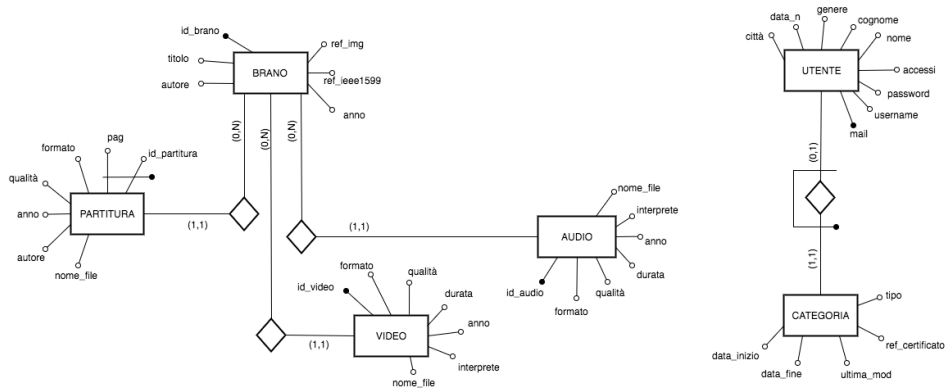

Figura 27. Costruzione dello schema ER: fase 2

In seguito, sono state aggiunte le entità `OFFER` e `AGREEMENT`. Ad ogni brano è solitamente associata una sola licenza, che contiene l'offer relativa a tutti i contenuti musicali legati al brano. A sua volta, dalla offer deriva l'agreement, relativo sia al singolo brano che allo specifico utente che usufruisce di quel brano. L'agreement può derivare dalla offer, oppure essere creato a prescindere dall'esistenza della offer, per situazioni particolari in cui i permessi comprati da uno specifico utente relativamente ad un brano non rientrino tra quelli "standard" espressi nella offer. Dunque, sono state create quattro associazioni:

- la prima, tra l'entità `BRANO` e l'entità `OFFER`, permette al brano di non avere offer associate o di averne massimo una, mentre l'offer deve essere obbligatoriamente legata almeno e soltanto ad un brano;
- la seconda, tra l'entità `OFFER` e l'entità `AGREEMENT`, associa ad ogni offer da zero a N agreement derivati e permette all'agreement di non partecipare all'associazione o di essere legato al massimo ad una offer;

- la terza, tra l'entità `BRANO` e l'entità `AGREEMENT`, tratta il caso in cui l'agreement sia indipendente dalla offer, legando al brano da zero a N agreement indipendenti e associando a ciascuno di essi obbligatoriamente un solo brano;
- la quarta, tra l'entità `AGREEMENT` e l'entità `UTENTE`, relaziona almeno e un solo utente ad ogni agreement e associa a ciascun utente da zero a N agreement.

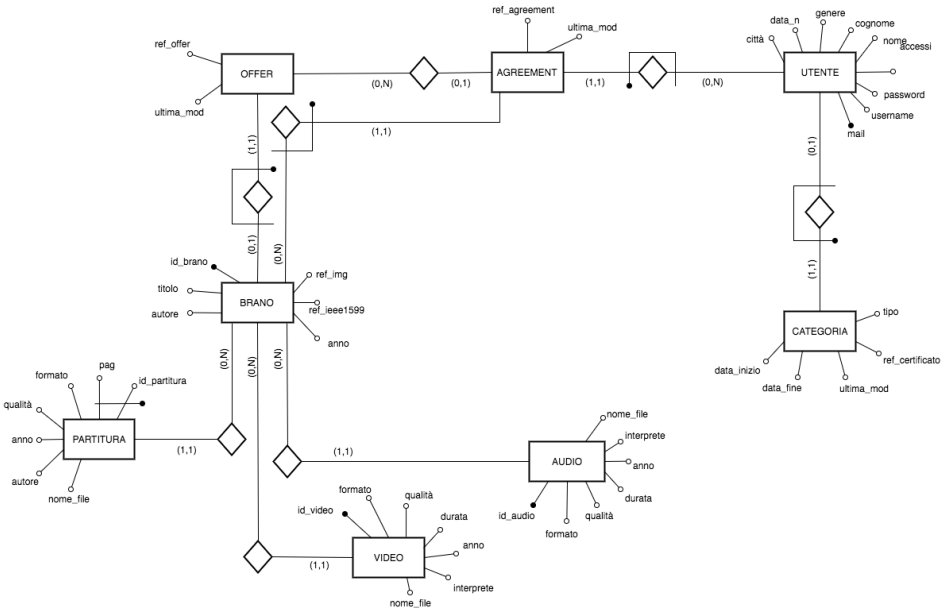

Figura 28. Costruzione dello schema ER: fase 3

Infine, sono state rappresentate le entità `VINCOLO`, `TEMPO`, `UTILIZZO` e `PAGAMENTO`. L'associazione inserita tra l'entità `AGREEMENT` e l'entità `VINCOLO` lega a ciascun agreement da zero a N vincoli, mentre un vincolo deve partecipare obbligatoriamente e solo una volta all'associazione. Inoltre, all'entità `VINCOLO` sono legate tre associazioni identiche con le entità `TEMPO`, `UTILIZZO` e `PAGAMENTO`. In tutti e tre i casi, l'entità `VINCOLO` può prendere parte o meno all'associazione e può partecipare al massimo una volta; mentre l'altra entità deve partecipare obbligatoriamente e al massimo una volta all'associazione, poichè ciascun controllo può appartenere ad un solo vincolo.

In questo modo, è stato creato lo schema ER finale, provvisto di tutti i costrutti necessari e di tutti gli elementi da rappresentare.

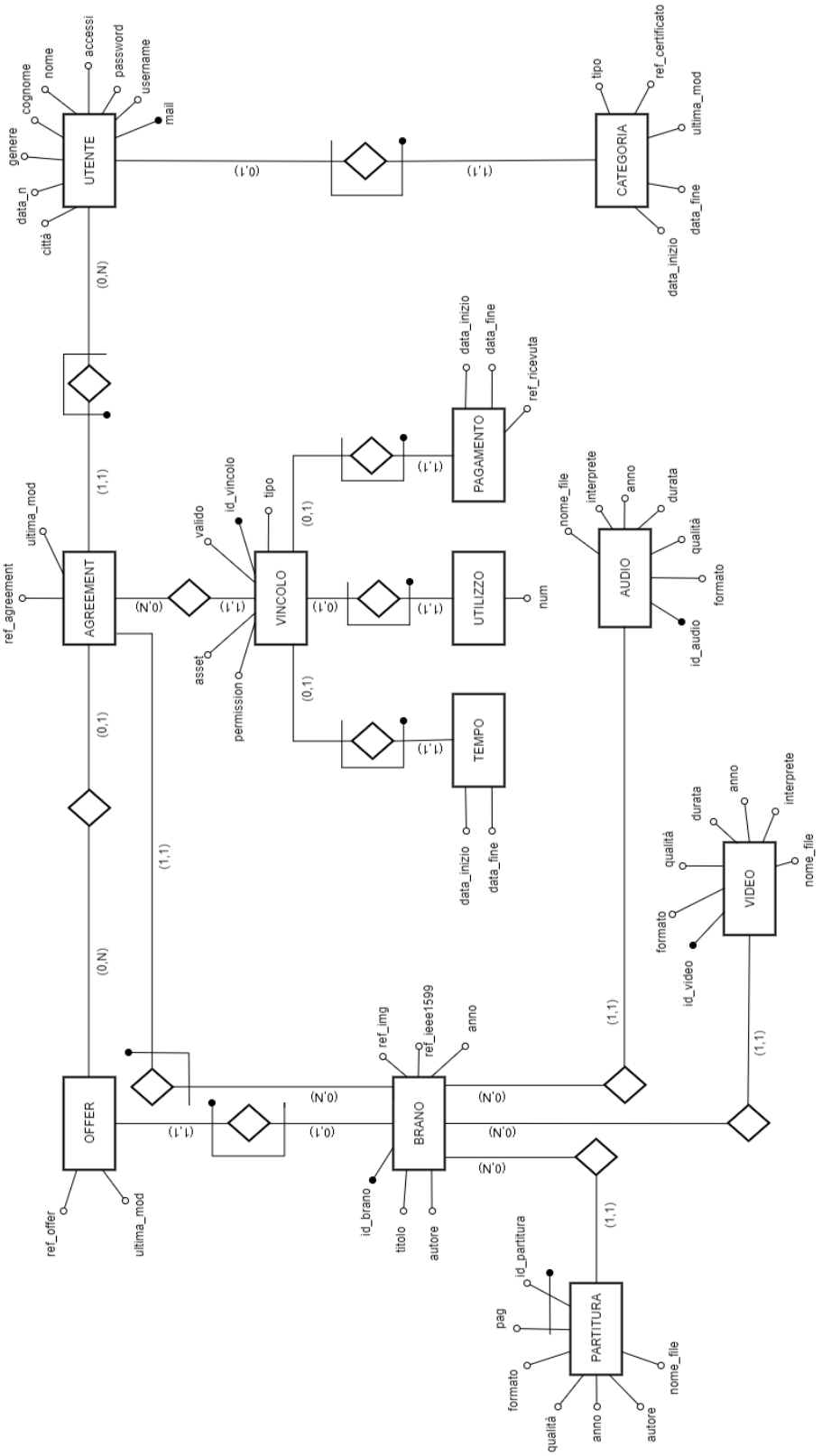

Figura 29. Schema ER finale

4.2 PROGETTAZIONE LOGICA

La progettazione concettuale ha prodotto uno schema ER che descrive ad alto livello il contenuto della base di dati; questo schema è utile per la generazione dello *schema relazionale*, che serve come punto di partenza per l'implementazione della base di dati. Dunque, mentre lo schema concettuale rappresenta in modo completo e formale i dati di interesse, lo schema relazionale descrive il contenuto della base di dati attraverso una struttura direttamente implementabile in uno specifico DBMS.

La prima parte della progettazione logica consiste nella *ristrutturazione* dello schema ER: l'obiettivo di questa fase è quello di generare uno schema ER semplificato, ma equivalente a quello di partenza, che contenga solo i costrutti traducibili nello schema relazionale. Dunque, la ristrutturazione rimodella i costrutti dello schema ER in modo da semplificarli e facilitarne la traduzione logica. Nel caso specifico della base di dati progettata per la piattaforma web IEEE1599, questa fase non ha apportato alcun cambiamento allo schema ER, poichè i costrutti utilizzati della progettazione concettuale descrivono i dati già in modo semplice e attraverso una struttura traducibile nello schema relazionale.

La seconda parte della progettazione logica riguarda la vera e propria *traduzione* dello schema ER in uno schema relazionale equivalente. La traduzione avviene attraverso una serie di regole che semplificano e rendono uniforme la trasformazione dei costrutti dello schema ER nel modello relazionale. Alla fine di questa fase, si ottiene uno schema logico, che rappresenta i dati secondo la struttura che verrà utilizzata per memorizzarli nella base di dati, e una documentazione di supporto, in cui vengono espressi i vincoli di integrità e le condizioni che non possono essere tradotti nello schema relazionale.

4.2.1 Fase di traduzione dello schema ER

Per la fase di traduzione dello schema ER, sono state seguite le regole di traduzione che consentono di mappare i costrutti del modello ER in costrutti equivalenti del modello relazionale.

Ciascuna entità dello schema ER è stata descritta nello schema relazionale attraverso una relazione, la quale contiene un attributo per ciascun attributo appartenente all'entità. Se nell'entità erano presenti chiavi esterne, anch'esse sono state tradotte in corrispondenti chiavi esterne nella relazione; esse rappresentano anche le associazioni con cui l'entità veniva

identificata esternamente o in modo misto. In questo caso specifico, tutte le associazioni sono state tradotte attraverso la presenza di chiavi esterne all'interno delle relazioni derivanti dalla traduzione delle entità: non è stato necessario creare alcuna relazione dedicata alla mappatura esplicita di un'associazione dello schema ER. Le condizioni e i vincoli di integrità associati ai dati sono stati espressi nella documentazione di supporto allo schema logico.

Di seguito viene riportato lo schema relazionale risultante dalla progettazione logica, accompagnato dal relativo documento di supporto per la descrizione dei vincoli di integrità.

SCHEMA RELAZIONALE

BRANO (id_brano, titolo, autore, anno$_o$, ref_ieee1599$_o$, ref_img$_o$)

AUDIO (id_audio, brano(BRANO), interprete, anno$_o$, durata, formato, qualità, nome_file)

VIDEO (id_video, brano(BRANO), interprete, anno$_o$, durata, formato, qualità, nome_file)

PARTITURA (id_partitura, pag, brano(BRANO), autore, anno$_o$, formato, qualità, nome_file)

UTENTE (mail, username, password, accessi, nome, cognome, genere$_o$, data_n$_o$, città$_o$)

CATEGORIA (utente(UTENTE), tipo, ultima_mod, data_inizio, data_fine$_o$, ref_certificato)

OFFER (brano(BRANO), ultima_mod, ref_offer)

AGREEMENT (brano(BRANO), utente(UTENTE), offer(OFFER)$_o$, ultima_mod, ref_agreement)

VINCOLO (id_vincolo, tipo, asset, permission, brano(AGREEMENT), utente(AGREEMENT), valido)

TEMPO (vincolo(VINCOLO), data_inizio, data_fine$_o$)

UTILIZZO (vincolo(VINCOLO), num)

PAGAMENTO (vincolo(VINCOLO), data_inizio, data_fine$_o$, ref_ricevuta)

VINCOLI DI INTEGRITÀ

- l'attributo "anno" nelle relazioni brano, audio, video può assumere solo valori di anni antecedenti o uguali a quello attuale;
- l'attributo "qualità" nelle relazioni audio, video e partitura può assumere solo i valori "l" (low), "m" (medium) e "h" (high);
- l'attributo "nome_file" nelle relazioni audio, video e partitura deve contenere il nome del file senza estensione;
- due utenti non possono avere lo stesso username;
- l'attributo "genere" nella relazione utente può assumere solo i valori "m" (male) e "f" (female);
- l'attributo "data_n" nella relazione utente può assumere solo valori di date antecedenti a quella attuale;
- l'attributo "tipo" nella relazione categoria può assumere solo i valori "stud" (studenti), "doc" (docenti) e "prof" (professionisti);
- l'attributo "data_inizio" nelle relazioni categoria, tempo e pagamento può assumere solo valori di date antecedenti a quelle espresse nell'attributo "data_fine";
- se la chiave esterna "offer" è diversa da NULL, allora le chiavi esterne "offer" e "brano" nella relazione agreement devono assumere lo stesso valore (ovvero l'id dello stesso brano);
- ciascuna istanza dell'entità vincolo deve partecipare ad almeno una tra le tre associazioni con le entità tempo, utilizzo e pagamento.

4.3 PROGETTAZIONE FISICA

Nella fase di progettazione fisica, il modello logico della base di dati estrapolato direttamente dallo schema relazionale è stato tradotto in linguaggio SQL per la sua implementazione nel DBMS. Attraverso il linguaggio SQL, per ciascuna relazione dello schema logico è stata generata una tabella nel database, contenente una colonna per ogni attributo; inoltre, sono stati implementati i vincoli di integrità descritti nel documento di supporto, aggiungendo delle condizioni nella creazione delle tabelle. Tutte queste

operazioni sono state memorizzate all'interno di un file SQL che, una volta importato nel DBMS, genera automaticamente tutta la struttura della base di dati.

Per la creazione, l'implementazione e la gestione della base di dati, è stato utilizzato il tool phpMyAdmin compreso nel pacchetto open source XAMPP. phpMyAdmin si occupa dell'amministrazione di database sul Web e supporta la maggior parte delle funzioni e delle operazioni di mySQL e MariaDB.

Tuttavia, alcune funzionalità del linguaggio SQL non sono supportate, tra cui l'implementazione dei vincoli `CHECK`, utilizzati per verificare l'effettiva applicazione di alcuni vincoli di integrità. Per questo motivo, nel file di creazione della base di dati, i vincoli `CHECK` sono stati inseriti come semplici commenti, per consentirne un eventuale utilizzo futuro.

Di seguito si riportano le operazioni SQL utilizzate per la generazione della base di dati progettata per il player web IEEE1599.

```
-- Creazione del DB
CREATE DATABASE db;

-- Creazione tabella "brano"
CREATE TABLE brano (
        id_brano varchar(10) PRIMARY KEY,
        titolo varchar(70) NOT NULL,
        autore varchar(60) NOT NULL,
        anno int(4),
        ref_ieee1599 varchar(200),
        ref_img varchar(200)
);

-- Creazione tabella "audio"
CREATE TABLE audio (
        id_audio varchar(10) PRIMARY KEY,
        brano varchar(10) NOT NULL,
        interprete varchar(60) NOT NULL,
        anno int(4),
        durata time NOT NULL,
        formato varchar(8) NOT NULL,
        qualita char(1) NOT NULL,
        nome_file varchar(100) NOT NULL,
```

```
        -- CHECK (qualita IN("l","m","h")),
        FOREIGN KEY (brano) REFERENCES brano(id_brano)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "video"
CREATE TABLE video (
        id_video varchar(10) PRIMARY KEY,
        brano varchar(10) NOT NULL,
        interprete varchar(60) NOT NULL,
        anno int(4),
        durata time NOT NULL,
        formato varchar(8) NOT NULL,
        qualita char(1) NOT NULL,
        nome_file varchar(100) NOT NULL,
        -- CHECK (qualita IN("l","m","h")),
        FOREIGN KEY (brano) REFERENCES brano(id_brano)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "partitura"
CREATE TABLE partitura (
        id_partitura varchar(10),
        pag int(5),
        brano varchar(10) NOT NULL,
        autore varchar(60) NOT NULL,
        anno int(4),
        formato varchar(8) NOT NULL,
        qualita char(1) NOT NULL,
        nome_file varchar(100) NOT NULL,
        -- CHECK (qualita IN("l","m","h")),
        PRIMARY KEY (id_partitura, pag),
        FOREIGN KEY (brano) REFERENCES brano(id_brano)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "utente"
CREATE TABLE utente (
        mail varchar(70) PRIMARY KEY,
```

```
        username varchar(30) NOT NULL,
        password varchar(30) NOT NULL,
        accessi int(2) NOT NULL DEFAULT 0,
        nome varchar(40) NOT NULL,
        cognome varchar(40) NOT NULL,
        genere char(1),
        data_n date,
        citta varchar(50),
        -- CHECK (genere IN("m","f")),
        UNIQUE (username)
);

-- Creazione tabella "categoria"
CREATE TABLE categoria (
        utente varchar(70) PRIMARY KEY,
        tipo varchar(4) NOT NULL,
        ultima_mod timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
        data_inizio date NOT NULL,
        data_fine date,
        ref_certificato varchar(200) NOT NULL,
        -- CHECK (tipo IN("stud","doc","prof","amm")),
        FOREIGN KEY (utente) REFERENCES utente(mail)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "offer"
CREATE TABLE offer (
        brano varchar(10) PRIMARY KEY,
        ultima_mod timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
        ref_offer varchar(200) NOT NULL,
        FOREIGN KEY (brano) REFERENCES brano(id_brano)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "agreement"
CREATE TABLE agreement (
        brano varchar(10),
        utente varchar(70),
        offer varchar(10),
```

```
        ultima_mod timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
        ref_agreement varchar(200) NOT NULL,
        PRIMARY KEY (brano, utente),
        FOREIGN KEY (offer) REFERENCES offer(brano)
        ON DELETE SET NULL ON UPDATE CASCADE,
        FOREIGN KEY (brano) REFERENCES brano(id_brano)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (utente) REFERENCES utente(mail)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "vincolo"
CREATE TABLE vincolo (
        id_vincolo varchar(10) PRIMARY KEY,
        tipo varchar(20) NOT NULL,
        asset varchar (100) NOT NULL,
        permission varchar (100) NOT NULL,
        brano varchar(10) NOT NULL,
        utente varchar(70) NOT NULL,
        valido boolean NOT NULL DEFAULT TRUE,
        FOREIGN KEY (brano) REFERENCES agreement(brano)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (utente) REFERENCES agreement(utente)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "tempo"
CREATE TABLE tempo (
        vincolo varchar(10) PRIMARY KEY,
        data_inizio date NOT NULL,
        data_fine date,
        FOREIGN KEY (vincolo) REFERENCES vincolo(id_vincolo)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "utilizzo"
CREATE TABLE utilizzo (
        vincolo varchar(10) PRIMARY KEY,
        num int(5) NOT NULL,
```

```
        FOREIGN KEY (vincolo) REFERENCES vincolo(id_vincolo)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Creazione tabella "pagamento"
CREATE TABLE pagamento (
        vincolo varchar(10) PRIMARY KEY,
        data_inizio date NOT NULL,
        data_fine date,
        ref_ricevuta varchar(200) NOT NULL,
        FOREIGN KEY (vincolo) REFERENCES vincolo(id_vincolo)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

5 IMPLEMENTAZIONE

INDICE

In questo capitolo si descriveranno le varie fasi che hanno portato all'implementazione della piattaforma web IEEE1599, motivando le scelte progettuali che sono state effettuate. Verranno presentati i meccanismi per la gestione dei vincoli ODRL, oltre che i sistemi di autenticazione e di inserimento di nuovi brani all'interno della base di dati. Infine, verrà descritta brevemente la realizzazione dell'interfaccia utente e la struttura del progetto nella sua organizzazione in directory.

Il player web 1599, descritto alla fine del primo capitolo, è in grado di fornire all'utente l'esperienza musicale resa possibile grazie al formato IEEE1599. Il player è stato specificatamente pensato per essere adattabile ai documenti IEEE1599, i quali, per l'appunto, richiedono modalità di gestione ad-hoc; ma esso non si occupa in alcun modo dei meccanismi di controllo delle licenze e dei diritti d'autore esistenti sui contenuti multi-

mediali. Inoltre, la nuova versione del player web IEEE1599 che si vuole sviluppare al momento è indipendente da qualsiasi contesto: consiste semplicemente in una pagina web priva di un sito vero e proprio di contorno, dove è possibile visualizzare soltanto un brano. Attualmente, il player è privo di un sistema di autenticazione, di un'interfaccia per selezionare il brano da visualizzare, o di un sistema back-end per aggiungere nuovi brani e nuovi contenuti, poichè è sprovvisto anche di una base di dati vera e propria.

Il progetto di tesi realizzato dall'ex studentessa Agnese Farinelli, già citato nel terzo capitolo, comprende una parte implementativa nella quale sono stati realizzati dei moduli, lato server e in linguaggio PHP, che si dedicano alla gestione dei vincoli ODRL sul permesso di sincronizzazione, agendo direttamente sul documento IEEE1599 in questione. Nonostante ciò, non vi è alcun controllo sui vincoli ODRL applicati agli altri due permission fondamentali per il formato IEEE1599, ovvero il *play* e il *display*. Difatti, il permission play autorizza un utente all'esecuzione di un brano o di un video, mentre il permission display autorizza un utente alla visualizzazione di uno spartito. Il permesso di sincronizzazione implica necessariamente l'esistenza di uno di questi due permessi per ogni risorsa multimediale; infatti, in mancanza di un permesso di esecuzione o di visualizzazione, il contenuto in questione risulterebbe completamente inaccessibile per l'utente e, di conseguenza, anche la sincronizzazione su quel contenuto sarebbe proibita.

Lo scopo del progetto sul quale si concentra questo elaborato è proprio quello di estendere le funzionalità del player: si vuole ottenere una migliore esperienza utente, integrare il controllo dei vincoli sui permessi di esecuzione e visualizzazione dei contenuti digitali, e gestire tutte le informazioni che ruotano attorno all'esperienza musicale in modo integrato, attraverso la costruzione di una base di dati che permetta una gestione di tutte queste informazioni. In particolare, gli obiettivi fondamentali della fase di implementazione sono i seguenti:

- salvataggio dei dati nella base di dati generata appositamente per il player e strutturazione delle directory contenenti i file multimediali ed i documenti XML;
- interazione tra la base di dati e il player;
- creazione di un sistema di autenticazione;
- integrazione del player web IEEE1599 con i controlli sul permesso di sincronizzazione gestiti dagli appositi moduli lato server;

- implementazione dei controlli sui vincoli ODRL riguardanti i permission play e display;
- creazione di una pagina dedicata alla scelta tra una lista di brani;
- generazione di una pagina back-end per l'inserimento di nuovi brani nel database;
- miglioramento dell'esperienza utente tramite un'interfaccia semplice ed elegante.

Dunque, il progetto si concentra sulla ristrutturazione del player web esistente attraverso la creazione e l'integrazione di nuove funzionalità inerenti alle licenze ODRL: si vuole implementare un controllo automatico non solo dei permessi di sincronizzazione, ma anche di esecuzione e visualizzazione, in modo che l'utente possa godere dell'esperienza musicale senza rischiare di violare i diritti d'autore. Le funzionalità preesistenti del player vengono mantenute, ma il player verrà inserito all'interno di una piattaforma web più completa, che comprenda un sistema di autenticazione per gli utenti e un elenco di brani tra cui scegliere. Naturalmente, saranno implementate solo le funzionalità di base e verrà generato più che altro uno scheletro iniziale del sito, che dovrà poi essere sviluppato e completato in lavori futuri.

5.1 ELEMENTI INIZIALI

Come già anticipato, il lavoro svolto si basa sulla ristrutturazione e sull'integrazione di materiali già precedentemente sviluppati presso il *Laboratorio di Informatica Musicale* di Milano. Nello specifico, i materiali preesistenti da cui ha inizio il progetto sono i seguenti:

- profilo ODRL di IEEE1599 e moduli PHP per la gestione dei permessi di sincronizzazione, frutti del progetto di tesi di Agnese Farinelli;
- player web IEEE1599 per la fruizione dell'esperienza musicale fornita dal formato, sviluppato da alcuni professori e ricercatori dell'Università degli Studi di Milano.

5.1.1 Moduli PHP per la gestione dei permessi di sincronizzazione

Gli elementi del profilo ODRL generato appositamente per il formato IEEE1599 sono già stati presentati nel terzo capitolo; ma si ritiene necessario descrivere in breve anche l'implementazione dei meccanismi lato server per la gestione dei diritti di sincronizzazione sui contenuti multimediali.

La logica applicata per i controlli sui permessi di sincronizzazione rispecchia la seconda architettura per la gestione delle licenze descritta nel terzo capitolo: partendo dal documento IEEE1599 completo, viene creato un nuovo documento IEEE1599, il quale viene costruito inserendo sotto-parti prelevate dal primo, a seconda delle informazioni alle quali l'utente può accedere. Infatti, per ciascun brano, vengono recuperati il documento IEEE1599 originale del brano e la relativa licenza per lo specifico utente. Viene generato uno scheletro del documento IEEE1599, il quale viene integrato e completato inserendo le sotto-parti estratte dal documento originale: tramite il controllo sulla licenza ODRL, nel nuovo documento saranno inserite soltanto le sotto-parti per le quali l'utente ha i permessi di accesso.

I due moduli PHP per la gestione dei permessi di sincronizzazione, che prendono i nomi di `licence_parser.php` e `new_ieee1599_parser.php`, lavorano a stretto contatto tra di loro: il primo si occupa di analizzare la licenza ODRL ed estrapolarne le informazioni sui vincoli; il secondo si dedica alla creazione del nuovo documento IEEE1599 a partire dal documento originale e applicando i vincoli ricavati dal primo modulo. A questo fine, sono state generate delle classi PHP che modellino gli elementi principali del linguaggio ODRL, ovvero asset, permission, constraint, requirement, condition, party e agreement; per ciascuna di esse, si sono definiti alcuni attributi che facilitino la descrizione e la manipolazione degli elementi ODRL.

L'intera procedura svolta dai moduli si basa sull'utilizzo della libreria *SimpleXML* di PHP per la gestione dei documenti XML: grazie a questa libreria, ciascun file XML viene trasformato in un'immensa struttura dati nella quale è possibile accedere direttamente ad ogni suo elemento, senza dover ogni volta rileggere l'intero file. Inoltre, tale libreria consente di effettuare ricerche mirate all'interno della gerarchia XML grazie alle funzionalità di *XPath*.

Il primo modulo, ovvero il `licence_parser.php`, contiene una serie di funzioni, ciascuna dedicata all'analisi di un singolo elemento della licenza ODRL (es. `asset_reader`, `constraint_reader`, ecc.): la funzione `visit_licence` chiama al suo interno le funzioni del modulo `licence_parser.`

`php` e genera una struttura dati che contiene tutte le informazioni relative alla licenza attraverso una configurazione dei dati facilmente accessibile.

L'output della funzione `visit_licence` viene passato come argomento alle funzioni contenute nel secondo modulo, ovvero `new_ieee1599_parser.php`: questo file PHP contiene una funzione per ciascun layer del formato IEEE1599, la quale si occupa di generare il layer in questione partendo dal documento IEEE1599 originale ed estrapolando solo le parti del layer che rispettano i vincoli espressi dalla licenza ODRL. Le funzioni per la creazione dei layer vengono chiamate all'interno della funzione principale del modulo, che prende il nome di `new_ieee1599standard_file_generator`, la quale si occupa di costruire il documento IEEE1599 finale.

5.1.2 Player web IEEE1599

Il player web generato appositamente per la fruizione del formato IEEE1599 si occupa di leggere il documento IEEE1599 di un brano e di concretizzare l'esperienza musicale descritta dal formato musicale in modo da metterla a disposizione degli utenti finali.

Il player è costituito da una parte *lato server* e una *lato client*: la prima ospita tutte le risorse associate ad ogni brano nel formato IEEE1599 e fornisce al lato client il documento IEEE1599; la seconda naviga le strutture dei layer IEEE1599 e popola la pagina web con gli elementi presenti in essi. Dunque, dato un brano specifico, il server estrae il documento IEEE1599 relativo dalla repository di risorse; successivamente il documento viene inviato al lato client, il quale lo analizza, estraendo i path dei contenuti multimediali e tutte le altre informazioni relative all'esperienza musicale: le risorse multimediali vengono prelevate dalla repository, inserite nella pagina web e collegate tra di loro tramite le mappature di eventi musicali contenute nel documento IEEE1599. In questo modo, viene generata un'esperienza musicale del brano che permette all'utente di fruire liberamente dei multimedia associati a quel brano.

Il lato client del player è costituito da una serie di moduli *Javascript* per l'analisi del documento IEEE1599 che insieme formano la libreria MX, la quale risulta essenziale per concretizzare le informazioni ottenute dal formato IEEE1599. I moduli fondamentali all'interno della libreria MX sono:

- `MXPlayer.js`: modulo che attiva tutto il player, servendosi dei moduli di lettura del documento IEEE1599 e di controllo dei media di audio, video e partiture;

- `IEEE1599.js`: modulo che, ricevuto un documento IEEE1599, ne estrapola le informazioni per ottenere i riferimenti ai contenuti multimediali e i rispettivi legami con ciascun evento musicale;
- `MXMedia.js`: modulo che gestisce l'esecuzione dei file audio e video;
- `MXSeekbar.js`: modulo che controlla le azioni dell'utente sulla barra del tempo;
- `MXGraphics.js`: modulo che gestisce la visualizzazione delle pagine degli spartiti.

L'esperienza musicale finale viene fornita a livello client dal modulo `index.js`, il quale, al caricamento della pagina, richiama il modulo MXPlayer, il quale gli restituisce tutte le informazioni riguardanti i contenuti multimediali e il controllo del player: a questo punto, `index.js` non fa altro che inserire queste informazioni all'interno della pagina HTML.

5.2 DESCRIZIONE DEL LAVORO SVOLTO

Come affermato in precedenza, l'obiettivo del progetto prevede una ristrutturazione e integrazione del player web IEEE1599 attraverso l'inserimento dei moduli per il controllo sui permessi di sincronizzazione e l'aggiunta di nuove funzionalità, riguardanti sia i controlli sugli altri permessi, sia lo sviluppo di alcune operazioni e il miglioramento dell'esperienza utente.

Il lavoro è stato svolto partendo dalla ristrutturazione dei materiali preesistenti e dalla loro fusione, costruendo anche una struttura generale del sito dove ospitare le singole pagine; successivamente si è passati all'implementazione di funzioni PHP per l'analisi e l'estrapolazione delle informazioni dalle licenze ODRL. È stato ideato un meccanismo semplice e versatile per la trasmissione dei vincoli estrapolati lato server verso il lato client: grazie alla creazione di funzioni *JQuery*, queste informazioni vengono elaborate per applicare i vincoli sui relativi contenuti multimediali.

In seguito, è stato generato il sistema di autenticazione, che utilizza delle query sulla base di dati per identificare gli utenti, e la pagina di visualizzazione della lista di brani, per consentire all'utente di sceglierne uno tra quelli per cui possiede la licenza. Si è generata anche un'interfaccia backend per facilitare l'inserimento di un nuovo brano, con relativi contenuti multimediali, nel database: il salvataggio avviene attraverso funzioni che

si interfacciano con la base di dati per i controlli sui valori inseriti e per il salvataggio delle nuove istanze nelle tabelle corrispondenti.

Infine, sono stati apportati alcuni semplici miglioramenti all'interfaccia utente tramite l'utilizzo degli strumenti forniti da *Twitter Bootstrap* e sono state generate delle licenze di prova per testare il corretto funzionamento del sito e delle nuove funzionalità inserite.

Grazie all'applicazione contemporanea dei moduli PHP per i controlli sul permesso di sincronizzazione e delle funzioni JQuery per i controlli sui permessi di esecuzione e di visualizzazione, l'utente finale potrà usufruire dell'esperienza musicale fornita dal documento IEEE1599 senza che vengano violati i vincoli imposti dalla licenza ODRL relativa. Di seguito si riporta un diagramma di flusso che descrive il funzionamento del player basato sull'utilizzo di un documento IEEE1599 originale e della relativa licenza ODRL:

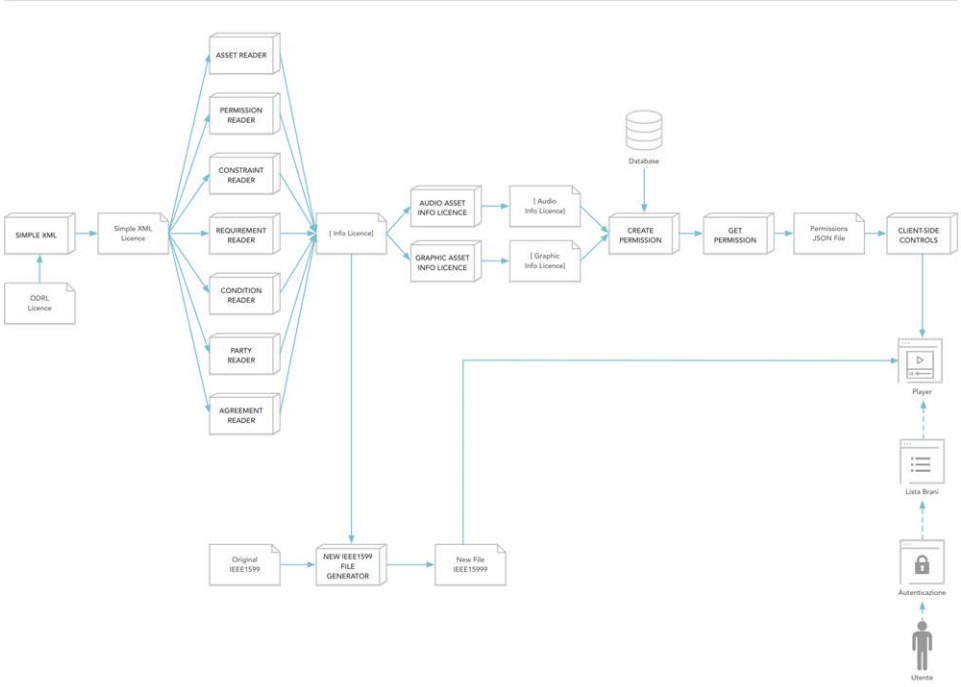

Figura 30. Diagramma di flusso del sistema di gestione dei vincoli ODRL

Nelle sottosezioni che seguiranno, verranno descritte più in dettaglio le varie fasi implementative che hanno condotto al raggiungimento degli obiettivi e alla conclusione del lavoro svolto.

5.2.1 Ristrutturazione dei moduli PHP e del player web IEEE1599

I materiali preesistenti dai quali ha avuto inizio il progetto, ovvero il player web IEEE1599 e i moduli PHP per la gestione dei permessi di sincronizzazione, erano oggetti indipendenti e completamente disgiunti l'uno dall'altro. Il player è nato come una pagina web a sé stante, pensata esclusivamente per la fruizione dell'esperienza musicale fornita dal formato IEEE1599 e priva di qualsiasi controllo sulle licenze legate ai contenuti musicali coinvolti; allo stesso modo, i moduli PHP sono stati generati con l'unico scopo di ottenere un documento IEEE1599 filtrato tramite il controllo sui vincoli ODRL riguardanti i permessi di sincronizzazione delle licenze, senza poi occuparsi dell'effettivo utilizzo del documento IEEE1599 in questione.

Inoltre, proprio a causa del fatto che, originariamente, ciascuno di questi materiali è stato pensato come un oggetto indipendente e autonomo, entrambi sono stati creati con una logica poco incline alla versatilità e alla portabilità: i metodi e le funzioni utilizzate al loro interno utilizzano spesso valori statici e costanti al posto di variabili; inoltre il player non possiede una struttura di base di un sito che lo contenga.

Per poter integrare entrambi i materiali all'interno di un unico sito, si è agito proprio su questi aspetti: in entrambi i casi, si è reso il codice più modulare e adattabile a situazioni diverse, trasmettendo alcune informazioni in modo dinamico invece che statico e andando ad applicare dei controlli per garantirne il corretto funzionamento. Nel caso del player, è stata creata una struttura esterna tramite un file `index.php`, il quale richiama il player come una pagina specifica al proprio interno; in questo modo si è resa possibile la creazione e la gestione di altre pagine nel sito.

Inoltre, è stato integrato all'interno del player il controllo sui permessi di sincronizzazione effettuato dai moduli PHP: il player, invece di utilizzare il documento IEEE1599 originale del brano, chiama la funzione `new_ieee 1599standard_file_generator`, che costruisce il documento IEEE1599 finale privo delle sotto-parti non accessibili all'utente; sarà il nuovo documento creato dalla funzione PHP ad essere utilizzato dal player per generare l'esperienza musicale richiesta.

5.2.2 Funzioni PHP per l'estrapolazione delle informazioni dalle licenze ODRL

Una volta ottenuto uno scheletro del sito contenente il player web IEEE1599 e dopo aver integrato i moduli per i controlli sui permessi di sincronizzazione, ci si è occupati della gestione degli altri due permessi applicabili ai contenuti multimediali, ovvero *play* e *display*. Per poter implementare la gestione di queste due tipologie di permission, risulta necessario disporre di una struttura dati che contenga le informazioni essenziali ricavate dalla licenza ODRL: le informazioni devono essere memorizzate attraverso una configurazione che ne permetta un accesso semplice e immediato.

A questo fine, è stato generato il file `licencefunction.php`, che contiene al suo interno tutte le funzioni relative alla gestione delle informazioni delle licenze ODRL. Per la fase iniziale di estrapolazione delle informazioni dalla licenza ODRL, è stata utilizzata la stessa funzione `visit_licence` creata per i moduli di gestione del permesso di sincronizzazione: il file ODRL contenente la licenza è stato convertito in una struttura dati PHP grazie all'utilizzo della libreria *SimpleXML*; successivamente, la struttura dati contenente le informazioni è stata elaborata dalle funzioni specializzate nella memorizzazione dei dati relativi ad un singolo elemento della licenza (es. `asset_reader`, `constraint_reader`, ecc.). I valori ritornati da queste funzioni sono stati memorizzati all'interno di un array nominato `info_licence`, il quale contiene tutte le informazioni relative a tutti i permessi contenuti nella licenza ODRL.

L'array `info_licence` contiene complessivamente sette celle:

- la cella `[0]` contiene l'elenco di tutti gli asset coinvolti nella licenza ODRL, con i relativi dati riguardanti l'id, il tipo, il path, i file name, ecc.;
- la cella `[1]` contiene l'elenco dei permission contenuti nella licenza, con informazioni sulle tipologie di permessi, sugli asset a cui vengono applicati e sui constraint, requirement e condition associati;
- la cella `[2]` contiene l'elenco dei constraint definiti nella licenza ODRL e ne memorizza i valori impostati;
- la cella `[3]` contiene l'elenco dei condition definiti nella licenza ODRL e ne memorizza i valori impostati;

- la cella `[4]` contiene l'elenco dei party coinvolti nella licenza ODRL, con informazioni sul loro ruolo, sugli asset di cui possiedono i diritti, sulle percentuali a cui hanno diritto, ecc.;
- la cella `[5]` contiene l'elenco degli agreement contenuti nella licenza (per come sono state strutturate le licenze per i documenti IEEE1599, ogni licenza contiene sempre e solo un agreement);
- la cella `[6]` contiene l'elenco dei requirement definiti nella licenza ODRL e ne memorizza i valori impostati;

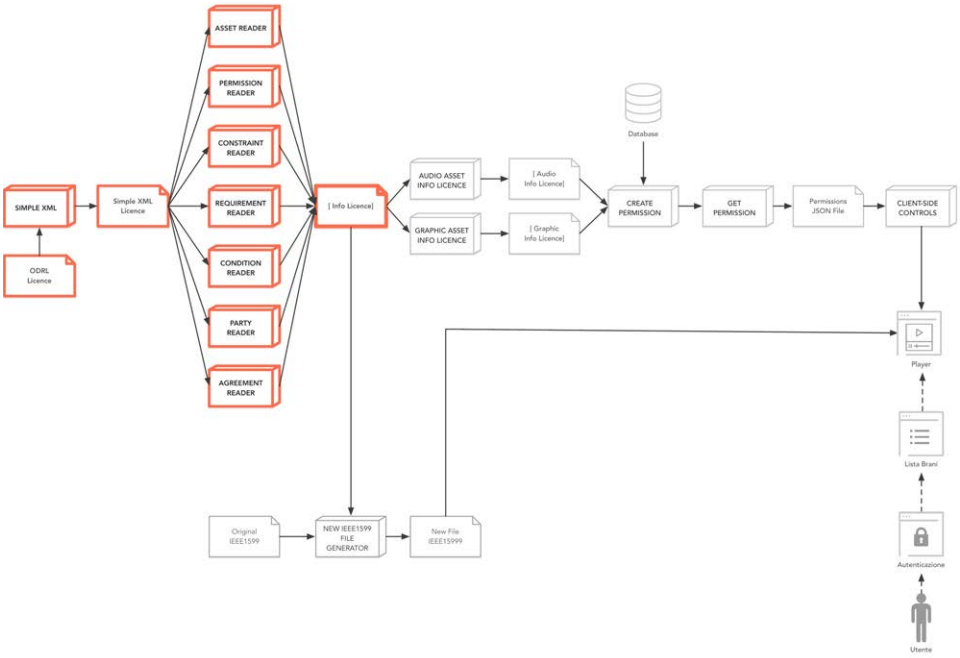

Figura 31. Diagramma di flusso 1

L'array `info_licence` ottenuto permette di raggruppare in un'unica struttura dati tutte le informazioni contenute all'interno della licenza ODRL in questione; ma, proprio per la sua completezza, risulta scomoda se si necessita di accedere a poche informazioni essenziali. Inoltre, per poter applicare i vincoli di esecuzione e visualizzazione ai contenuti musicali, è necessario strutturare le informazioni della licenza in funzione di ciascuna delle risorse: conoscendo il codice identificativo o il path di una data risorsa, deve essere possibile accedere ad una sezione della struttura dati contenente tutte e sole le informazioni riguardanti quella risorsa. Nell'array `info_licence` questo non è fattibile, poichè le informazioni sono struttu-

rate in funzione degli elementi ODRL della licenza e non in funzione dei singoli asset.

Ne deriva l'esigenza di generare una nuova struttura dati, a partire dall'array `info_licence`, che memorizzi, per ogni risorsa, tutti e soli i permission, i costraint, i requirement e i condition associati a quella risorsa, tralasciando per ciascuno di essi le informazioni superflue. Questo compito è affidato alle funzioni `audio_assets_infolicence`, per i contenuti di tipo audio o video, e `graphic_assets_infolicence`, per le partiture: entrambe le funzioni esplorano l'array `info_licence` e, per ciascun asset, memorizzano in un array i valori dei vincoli associati all'asset.

L'output della funzione `audio_assets_infolicence` è un array con una cella per ciascuna risorsa audio o video, contenente le relative informazioni sui vincoli; allo stesso modo, l'output della funzione `graphic_assets_infolicence` è un array con una cella per ciascuna partitura, contenente le relative informazioni sui vincoli. Nel caso in cui, nella licenza ODRL, un insieme di asset fosse identificato come un'unica risorsa (es. tutti gli score), significa che i vincoli applicati a quella risorsa valgono su tutti gli asset appartenenti al gruppo: in questo caso, le strutture dati generate considerano quel gruppo di asset come un'unica risorsa e memorizzano i vincoli applicati a tutto l'insieme una sola volta.

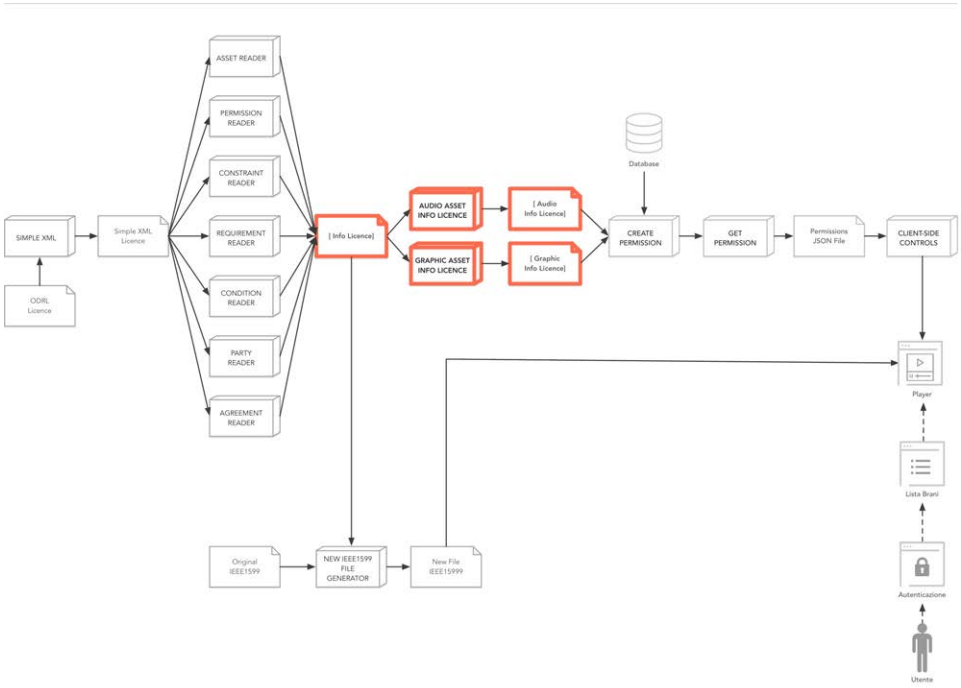

Figura 32. Diagramma di flusso 2

L'array `audio_infolicence`, generato dalla funzione `audio_assets_infolicence`, ha una struttura di questo tipo:

```
Array (
    [0] => Array (
        [asset] => track
        [path] => "..."
        [file_name] => ...
        [permission_audio] => Array (
            [play] => Array (
                [constraint] => Array (
                    [numberOfSeconds] => Array (
                        [startSecond] => "..."
                        [number] => "..."
                    )
                    [spatial] => "..."
                    [count] => "..."
                    ...
                )
                [requirement] => Array (...)
                [condition] => Array (...)
            )
            [synchronization] => Array (
                [constraint] => Array (
                    [qualityOfResource] => "..."
                    ...
                )
                [requirement] => Array (...)
                [condition] => Array (...)
            )
        )
        ...
    )
    [1] => Array (
        [asset] => track
        [path] => "..."
        [file_name] => ...
        ...
    )
    ...
```

```
)
```

L'array `graphic_infolicence`, generato dalla funzione `graphic_assets_infolicence`, ha una struttura di questo tipo:

```
Array (
    [0] => Array (
        [asset] => scores
        [path] => .../SCORES/
        [permission_scores] => Array (
            [display] => Array (
                [constraint] => Array (
                    [numberOfPages] => Array (
                        [startPage] => "..."
                        [number] => "..."
                    )
                    [qualityOfResource] => "..."
                    [count] => "..."
                    [format] => "..."
                    ...
                )
                [requirement] => Array (...)
                [condition] => Array (...)
            )
            [synchronization] => Array (...)
        )
    )
)
```

Grazie alle due nuove strutture dati generate, che memorizzano le informazioni della licenza ODRL in funzione di ciascuna risorsa, è possibile accedere direttamente alla sezione della struttura dati contenente tutte e sole le informazioni riguardanti la risorsa richiesta.

5.2.3 Meccanismo di trasmissione dei vincoli tramite JSON

Il player web IEEE1599 immagazzina i dati relativi al formato all'interno di una repository lato server; ma tutti i meccanismi di controllo del player sono implementati lato client in linguaggio Javascript. Di conseguenza,

anche i controlli sui vincoli espressi nelle licenze ODRL dovranno essere implementati lato client e, per questo motivo, è necessario realizzare un meccanismo che sia in grado di trasmettere queste informazioni dal lato server al lato client.

Per realizzare un sistema del genere, esisterebbero diversi metodi, ma probabilmente il più semplice e versatile tra tutti consiste nell'utilizzare JSON, un formato adatto all'interscambio di dati fra applicazioni client-server. JSON permette di memorizzare le informazioni secondo una gerarchia di dati molto semplice, simile a quella di XML e facilmente interpretabile, che consente un accesso rapido e diretto alle informazioni.

A questo punto, si hanno a disposizione tutti gli strumenti necessari per ottenere le informazioni delle licenze ODRL lato client: il meccanismo consiste nella traduzione degli array `audio_infolicence` e `graphic_infolicence`, precedentemente creati e strutturati in funzione degli asset, in un unica stringa in formato JSON.

Per tradurre gli array PHP contenenti le informazioni della licenza ODRL in una stringa in formato JSON, è stata generata lato server la funzione PHP `createPermission`: questa funzione si occupa di codificare in sintassi JSON tutte le informazioni contenute nei due array e alcune informazioni prelevate dalla base di dati, che riguardano dati di utilizzo dell'utente relativi alla fruizione dello specifico brano. La funzione `createPermission` esplora i due array, ricavando i valori dei vincoli applicati ad ogni contenuto multimediale, ed interroga il database per ottenere le informazioni di cui ha bisogno relative alla fruizione del brano da parte dell'utente. Le informazioni vengono memorizzate secondo una struttura costante, che viene replicata per rappresentare i vincoli applicati a ciascuna risorsa. La struttura di base della stringa JSON risultante, per ciascun asset, è la seguente:

```
{"elements": {
    "(md5code)": {
        "asset": "...",
        "path": "...",
        "synchronization": {
            "numberOfMeasures": {
                "startMeasure": "...",
                "number": "..."
            },
            "numberOfSeconds": {
                "startSecond": "...",
```

```
            "number": "..."
        },
        "numberOfPages": {
            "startPage": "...",
            "number": "..."
        },
        "qualityOfResource": "..."
    },
    "media": {
        "play": {
            "numberOfSeconds": {
                "startSecond": "...",
                "number": "..."
            }
        },
        "display": {
            "numberOfPages": {
                "startPage": "...",
                "number": "..."
            }
        },
        "common": {
            "qualityOfResource": "...",
            "spatial": "...",
            "count": "...",
            "used": "...",
            "datetime": {
                "start": "...",
                "end": "...",
            },
            "format": "...",
            "prepay": {
                "amount": "...",
                "currency": "..."
            },
            "postpay": {
                "amount": "...",
                "currency": "..."
            },
```

```
                "peruse": {
                    "amount": "...",
                    "currency": "..."
                }
            }
        }
    },
    ...
}
```

Tutte le informazioni sono contenute in un racchiuse generale *elements*: al suo interno, per ogni asset, viene generata la struttura appena mostrata. La gerarchia di vincoli appartenente a ciascun asset è identificata da un codice (sopra indicato come `md5code`), ottenuto codificando il path relativo dell'asset tramite la funzione hash crittografica MD5; idealmente, in un futuro sviluppo del progetto, questo valore dovrà essere sostituito dall'effettivo identificatore dell'asset nella base di dati. La gerarchia, costante per ogni risorsa, contiene tutti i campi necessari per salvare ogni tipo di informazione relativa alla licenza ODRL e alla fruizione del brano da parte dell'utente. Nel caso di un asset di tipo audio o video, i campi relativi al permesso display vengono automaticamente impostati vuoti; viceversa, nel caso di un asset di tipo partitura, i campi relativi al permesso play vengono automaticamente impostati vuoti.

La stringa in formato JSON generata dalla funzione `createPermission` viene inserita nel file `getPermission`, che si occupa di passare i giusti parametri alla funzione e di trasformare la codifica del file in un vero e proprio file JSON, che sia accessibile dal lato client. Infatti, `getPermission` è stato appositamente generato come file *stand-alone*, ovvero indipendente dalle funzionalità dell'applicativo web: necessita soltanto dell'array `info_licence` e dell'oggetto PHP per la connessione al database, che sono stati appositamente memorizzati come variabili di sessione. Nel file `getPermission` vengono chiamate le funzioni `visit_licence`, `audio_assets_infolicence` e `graphic_assets_infolicence` per la creazione dei due array `audio_infolicence` e `graphic_infolicence`; successivamente, le informazioni appena ottenute vengono passate come parametri alla funzione `createPermission`, la quale restituisce una "stampa" della stringa JSON generata. A questo punto, il file `getPermission` contiene un'istruzione che modifica il suo header, in modo tale da essere letto in formato JSON e non più come file PHP; in questo modo, quando il file verrà ri-

chiesto dal lato client, il client riceverà un header di risposta di tipo JSON invece che PHP e quindi potrà accedere al file come un comune file JSON ogni volta che ne avrà bisogno.

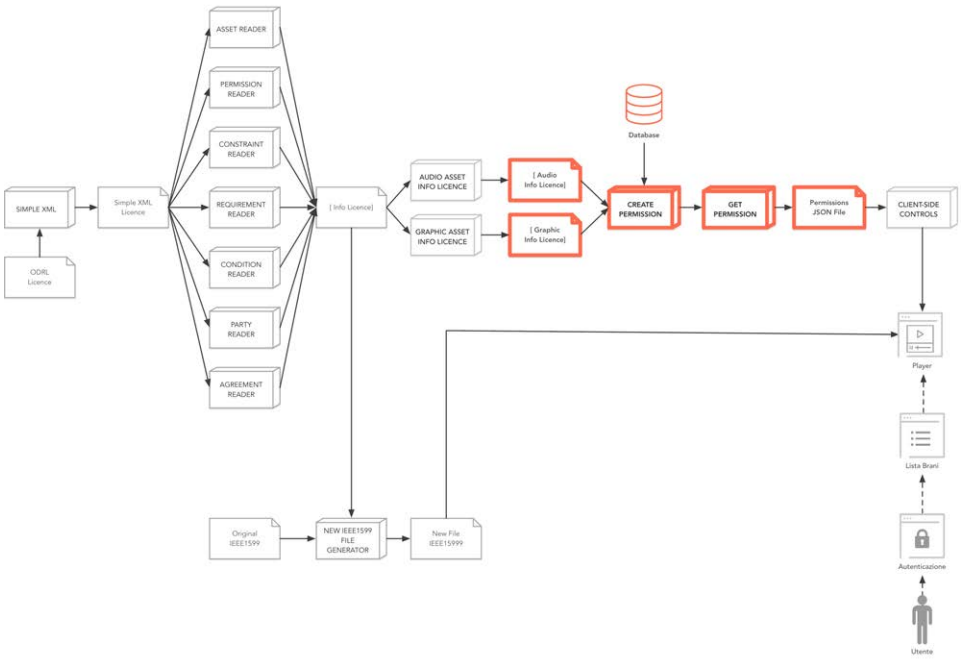

Figura 33. Diagramma di flusso 3

5.2.4 Funzioni JQuery per l'applicazione dei vincoli sulla fruizione dei contenuti multimediali

Come già anticipato, la gestione del player avviene completamente lato client, dunque anche i controlli sui vincoli espressi nelle licenze ODRL sono stati implementati lato client. Per non apportare troppi cambiamenti alla libreria MX utilizzata nel player, ci si è limitati ad aggiungere qualche istruzione esclusivamente nel file `MXPlayer.js`, mentre la reale implementazione dei controlli sui vincoli è stata generata all'interno di un altro file, nominato `MXPermission.js`: questo file può essere visto come un modulo che estende la libreria MX, integrando la verifica dei vincoli ODRL sulla fruizione dei contenuti multimediali.

Il file `MXPermission.js` contiene un oggetto `MXPermission`, che a sua volta racchiude alcune funzioni JQuery per il controllo delle informazioni estrapolate dalla licenza ODRL su un determinato asset. L'utilizzo della

libreria JQuery di Javascript è motivato dal fatto che JQuery consente di semplificare enormemente la gestione e la manipolazione degli eventi e degli elementi DOM nelle pagine HTML; inoltre, implementa le funzionalità Ajax per lo scambio di dati tra client e server.

L'oggetto `MXPermission` prende come parametro il path dell'asset sul quale si vogliono applicare i controlli della licenza. Al suo interno, la funzione `getPermissions` utilizza le funzionalità Ajax per accedere al file `getPermission.php`, il quale, come spiegato precedentemente, viene ricevuto dal client come file in formato JSON. Dal file JSON, vengono ricavate e restituite in opportune variabili tutte le informazioni della licenza necessarie per effettuare le verifiche sui vincoli relativi all'asset in questione.

All'interno dell'oggetto `MXPermission` sono contenuti anche i metodi specifici per ciascun controllo: ogni vincolo contenuto nel JSON possiede una funzione dedicata che, tramite la funzione `getPermissions`, ne controlla il valore e stabilisce se l'utente sta rispettando o meno il vincolo. Ai fini del progetto, ci si è limitati all'implementazione dei vincoli *spatial*, *timedate*, *numberOfSeconds* e *count*: il controllo sul campo `spatial` verifica che l'utente si trovi in uno dei paesi dai quali è consentito accedere all'asset; il controllo sul campo `timedate` verifica che la data in cui l'utente accede al contenuto rientri all'interno del periodo di tempo stabilito dalla licenza; il controllo sul campo `numberOfSeconds` assicura che l'esecuzione dell'audio o video in questione venga limitata all'intervallo di secondi consentito; infine, il controllo sul campo `count` assicura che l'utente non superi il numero di volte per cui gli è concesso accedere all'asset. Ciascuna funzione di controllo effettua la verifica sul vincolo e, nel caso in cui il vicolo non venga rispettato, blocca la fruizione del contenuto in questione e fa comparire un messaggio di avvertimento per l'utente, in cui viene indicata la natura del problema.

Per poter funzionare correttamente, l'oggetto `MXPermission` deve essere inizializzato all'interno del file `MXPlayer.js`, poichè è proprio all'interno di questo script che vengono controllati gli eventi del player. Tramite le funzioni contenute in `MXPermission`, i controlli specifici per ogni vincolo vengono effettuati, a seconda del tipo di vincolo, ad ogni *update* del player (ogni 10 millisecondi) oppure ad ogni evento di play e pause sulla barra del tempo di esecuzione. In questo modo, tutti gli eventi e le informazioni del player saranno sempre sottoposti ai controlli relativi alla licenza, sia che l'utente cambi risorsa da eseguire o visualizzare, sia che si sposti sulla barra del tempo, sia che visualizzi lo stesso contenuto per un certo numero

di volte, e così via.

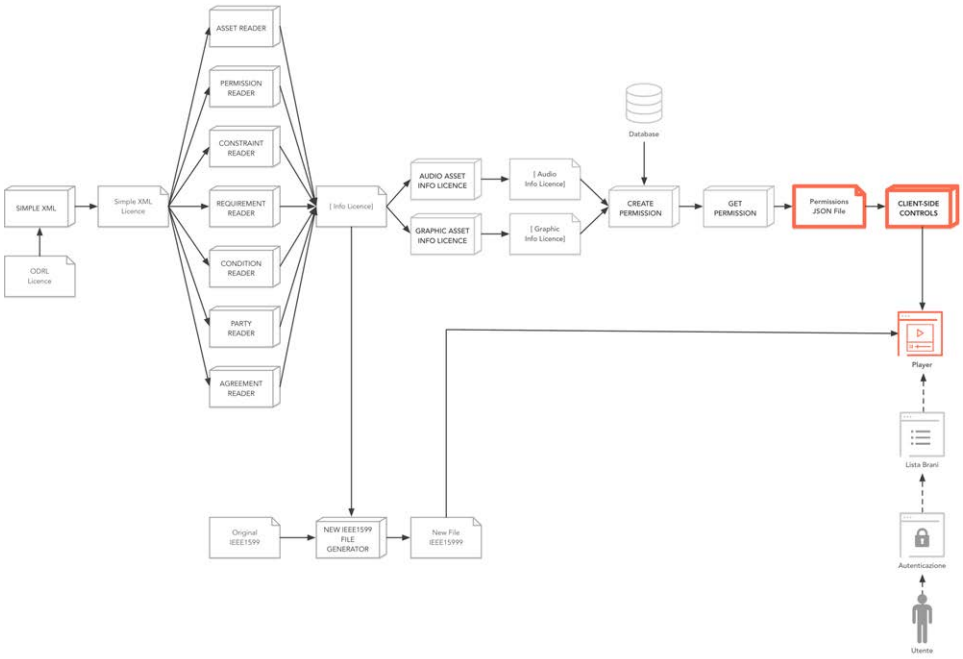

Figura 34. Diagramma di flusso 4

5.2.5 Sistema di autenticazione e pagina di visualizzazione dei brani

Ogni licenza ODRL si riferisce alla fruizione di uno specifico brano da parte di uno specifico utente. Il player web IEEE1599 originario non eseguiva alcun controllo sui permessi associati ai contenuti di un brano; dunque, per un corretto funzionamento, necessitava solamente del documento IEEE1599 relativo al brano scelto. Nel nuovo player web IEEE1599, è necessario un sistema per identificare gli utenti che usufruiscono dell'esperienza musicale legata a ciascun brano: per i brani contenenti risorse su cui è valido il diritto d'autore, risulta essenziale memorizzare, per ciascun utente, una licenza relativa al brano che indichi con quali restrizioni lo specifico utente può usufruire della licenza.

Per questo motivo, è stato creato un sistema di autenticazione, che permetta agli utenti iscritti al sito di accedere con le proprie credenziali e di essere identificati mediante una ricerca nella base di dati. L'accesso di un utente al sito mediante le credenziali consente al sistema di conoscere e di visualizzare soltanto quei brani e quei materiali ai quali l'utente è

autorizzato ad accedere.

Il sistema si autenticazione è stato implementato tramite un file front-end, che contiene una form HTML dove l'utente deve inserire username e password, e un file back-end, che interroga la base di dati per cercare l'utente tra quelli iscritti al sito. Se le credenziali dell'utente in questione non vengono trovate all'interno della base di dati, viene visualizzato un errore.

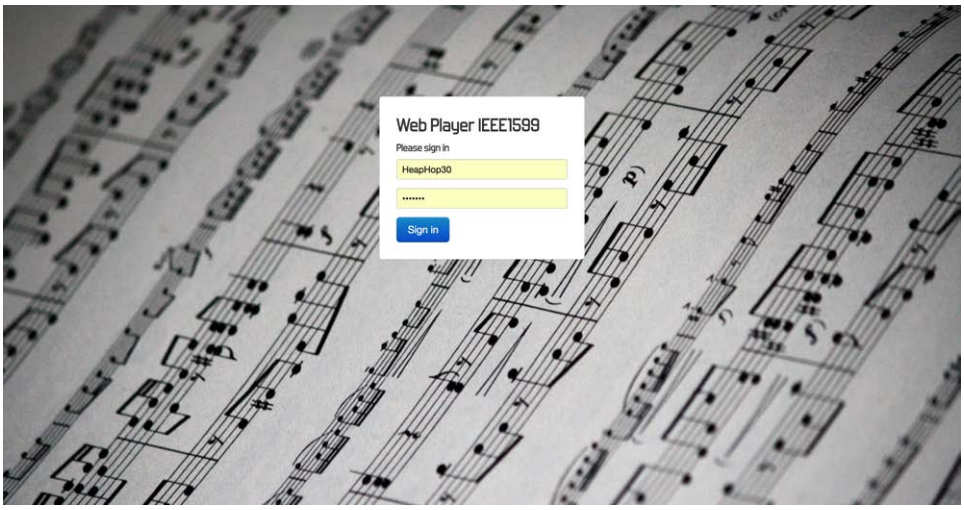

Figura 35. Pagina di login

Nel caso in cui l'autenticazione vada a buon fine, viene aperta una nuova sessione e viene visualizzata una pagina web contenente la lista dei brani ai quali l'utente può avere accesso, poichè possiede la relativa licenza. La lista di brani accessibili dall'utente è ottenuta mediante un'interrogazione sulla base di dati, che controlla quali sono i brani associati a tutte le licenze che l'utente possiede. Nella pagina HTML, per ciascun brano, è possibile cliccare il pulsante "View" per visualizzare il player IEEE1599 e fruire dell'esperienza musicale relativa al brano selezionato.

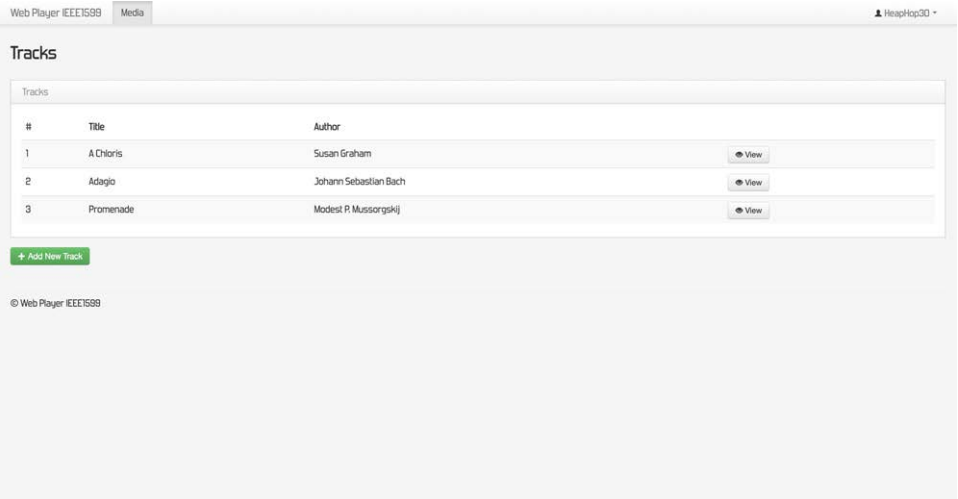

Figura 36. Pagina con l'elenco dei brani

5.2.6 Interfaccia back-end per l'inserimento di un nuovo brano

Ai fini del progetto, molte funzionalità che prevedono il prelevamento, l'inserimento, la modifica o l'eliminazione di informazioni dalla base di dati non sono state implementate. Tuttavia, la creazione di un nuovo brano musicale, insieme ai contenuti multimediali associati, risulta essere una funzione particolarmente utile e importante, oltre che complessa da gestire; dunque, si è ritenuto necessario implementare sin da subito questa funzionalità.

Di conseguenza, è stata ideata un'interfaccia back-end dove l'admin del sito web può inserire tutti i valori degli attributi associati ad un brano da aggiungere al database. Insieme alla specifica degli attributi relativi al brano, è possibile definire le informazioni relative ai singoli file audio, video o partiture che si vogliono associare la brano stesso.

A livello tecnico, la form è stata suddivisa in quattro parti (*fieldset*) corrispondenti ad ognuna delle quattro parti che definiscono un brano: attributi del brano (obbligatori e quindi visibili all'ingresso in pagina), audio, video e partiture. Gli ultimi tre campi sono opzionali e attivabili da un apposito pulsante, per evitare all'utente un sovraccarico cognitivo nella compilazione della form. Il sistema, tuttavia, si assicura che almeno uno tra audio, video o partitura sia stato compilato e aggiunto al brano. Prima dell'invio dei dati alla base di dati, vengono eseguiti prima dei controlli di validazione lato client (tramite funzioni di validazione inline JQuery) e successivamente lato server.

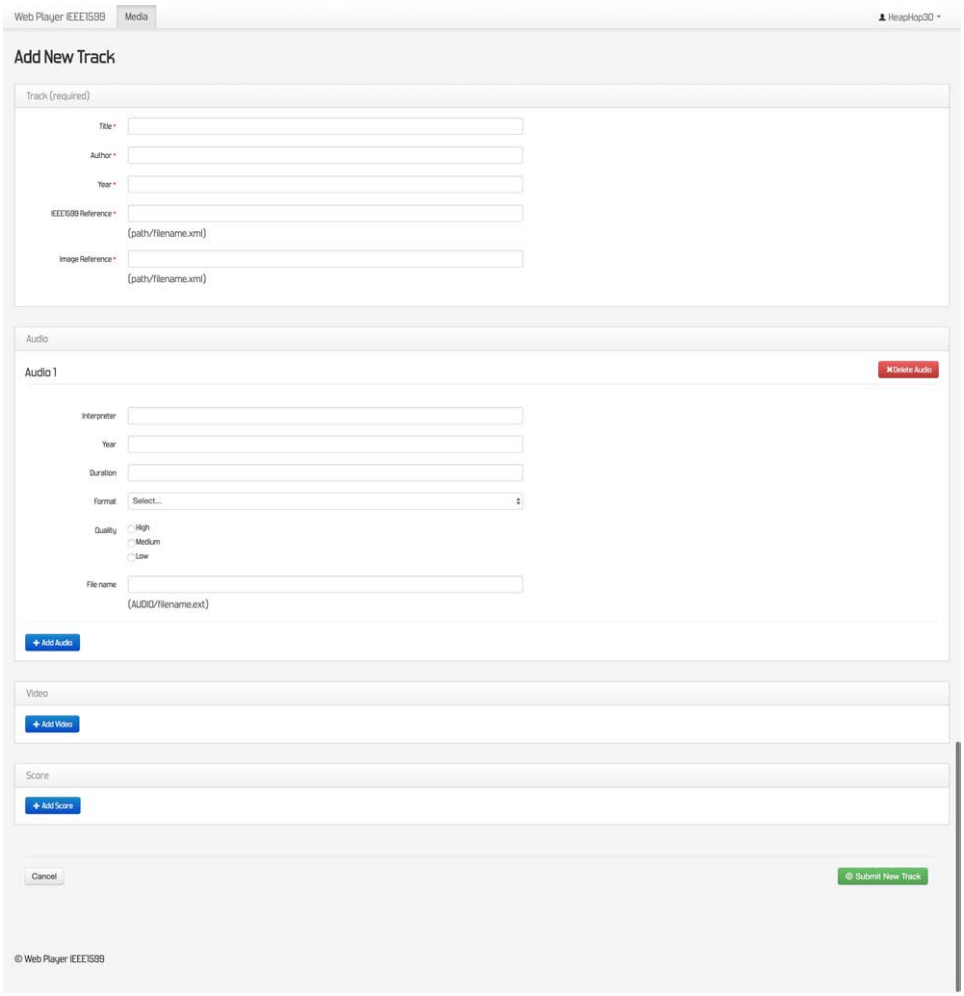

Figura 37. Pagina per l'inserimento di un nuovo brano

5.2.7 Interfaccia front-end

Per migliorare l'esperienza di fruizione del sistema, è stato scelto di utilizzare per l'interfaccia utente un *template* basato sul framework *Twitter Bootstrap* (versione 3.1). Il template scelto è in grado di fornire una serie di pattern evoluti di interfaccia utente utili a migliorare l'usabilità e la piacevolezza dell'utilizzo del sistema, consentendo allo stesso modo estrema semplicità e flessibilità di implementazione.

La scelta di utilizzare Bootstrap, infatti, garantisce l'utilizzo di funzionalità e design pattern moderni direttamente *out+of-the-box*. Twitter Boostrap si è imposto da anni come *standard-de-facto* nella realizzazione di interfacce utente, proprio grazie alla sua sua semplicità di adozione e alla vasta documentazione disponibile, oltre che per la presenza di una forte community

di supporto.

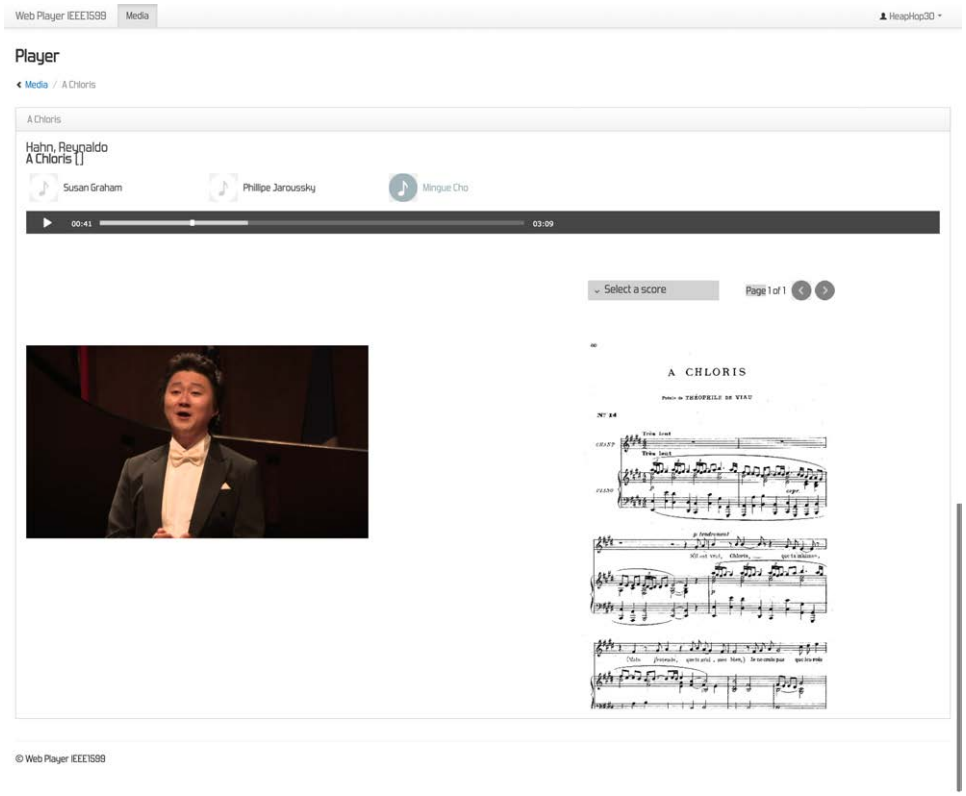

Figura 38. Player web IEEE1599

5.3 STRUTTURA DEL PROGETTO

Il file del sistema implementato sono stati organizzati mediante la seguente gerarchia di directory:

- *common:* contiene le funzioni PHP inerenti all'elaborazione delle licenze ODRL, all'interazione con la base di dati, al sistema di autenticazione e di aggiunta di un nuovo brano e alla creazione del file JSON per la validazione dei vincoli ODRL;
- *doc:* contiene i documenti relativi alla categoria di appartenenza di ciascun utente;
- *js:* contiene i file JQuery per la creazione di un nuovo brano e per l'applicazione dei vincoli espressi nelle licenze ODRL sulle relative risorse;

- *licences:* contiene gli XML Schema del linguaggio ODRL e del nuovo profilo IEEE1599;
- *media:* contiene una directory per ogni brano, con al suo interno la offer ODRL e il documento IEEE1599 relativi al brano, oltre alle seguenti directory: "AUDIO", che racchiude i file audio e video, "SCORES", che racchiude le pagine delle partiture, e "AGREEMENTS", che racchiude gli agreement ODRL associati a ciascun utente;
- *parsers:* contiene i moduli PHP per la gestione dei permessi di sincronizzazione e la generazione del documento IEEE1599 finale;
- *player:* contiene tutti i file PHP e Javascript per il corretto funzionamento del player web IEEE1599;
- *template:* contiene i file relativi al template di Bootstrap utilizzato per l'interfaccia utente;
- *view:* contiene i file corrispondenti ad ognuna delle parti in cui è stata opportunamente segmentata l'interfaccia utente.

CONCLUSIONI E SVILUPPI FUTURI

Il progetto trattato come oggetto di questo elaborato si inserisce in un contesto tecnologico in continuo sviluppo, dove risulta sempre più difficile tutelare le opere d'ingegno digitali e i loro autori. L'evoluzione tecnologica, infatti, continua a facilitare la diffusione e la divulgazione dei contenuti digitali, rendendone però più difficile il controllo. Nel momento in cui non sia possibile regolare la fruizione di un'opera, i soggetti coinvolti nella sua creazione, e quindi possessori dei diritti d'autore, non possono godere del diritto alla paternità dell'opera in quanto responsabili della sua creazione. Per questo motivo, è necessaria la realizzazione di meccanismi digitali di protezione dei diritti d'autore, che tutelino i creatori di un'opera contro l'utilizzo illecito della stessa.

Lo scopo del progetto sul quale si concentra questo elaborato consiste proprio nella generazione di un sistema di controllo dei diritti esistenti sui contenuti digitali integrati nel formato IEEE1599. Prima che venissero applicate le opportune modifiche al funzionamento del player web IEEE1599, creato per poter fruire dell'esperienza musicale offerta dallo standard IEEE1599, i contenuti multimediali coinvolti nell'esperienza erano sempre accessibili, senza che venisse effettuata alcuna validazione sui diritti d'autore associati.

Grazie all'utilizzo del linguaggio ODRL per l'espressione dei diritti d'autore e alla creazione di meccanismi lato client e lato server per il controllo dei vincoli espressi nelle licenze, è stato generato un sistema in grado di fornire le basi per la tutela dei prodotti intellettuali forniti dal player. La piattaforma web generata include già alcuni controlli sui permessi di cui gode l'utente per ogni brano musicale e fornisce tutti gli strumenti necessari per l'estensione di queste funzionalità.

Possibili sviluppi futuri del progetto consistono nell'evoluzione del lavoro già svolto e descritto da questo elaborato: la piattaforma web generata dovrebbe essere estesa e migliorata, rendendola completamente dipendente dal database per quanto riguarda l'estrazione delle informazioni e delle risorse digitali utilizzate; l'espansione del lavoro dovrebbe essere indirizzata soprattutto verso l'implementazione di nuovi controlli sui vincoli ODRL, in modo da ottenere un sistema più flessibile e sicuro per i detentori dei diritti che concedono al sito l'utilizzo delle loro opere.

Un altro aspetto migliorabile potrebbe essere quello riguardante l'e-

sperienza utente: risulterebbe utile estendere la piattaforma web attraverso l'aggiunta di nuove pagine navigabili, che forniscano informazioni e funzionalità aggiuntive agli utenti finali, e attraverso la creazione di un'interfaccia in grado di semplificare ulteriormente l'esplorazione del sito.

Una terza prospettiva di sviluppo potrebbe prevedere l'ultimazione dell'interfaccia back-end, per facilitare la gestione dei contenuti della piattaforma web da parte di chi la amministra: risulterebbe opportuno implementare funzioni che consentano un controllo completo delle informazioni memorizzate nella base di dati. Un'evoluzione determinante per le funzionalità back-end includerebbe la creazione automatica dell'agreement ODRL per ciascun brano fruito da uno specifico utente, a partire dalla categoria alla quale appartiene l'utente e dalla offer ODRL relativa al brano in questione.

BIBLIOGRAFIA

Barbara Catania, Elena Ferrari, Giovanna Guerrini
SISTEMI DI GESTIONE DATI. CONCETTI E ARCHITETTURE
Milano, CittàStudi, 2006

Agnese Farinelli
ANALISI, SPECIFICA ED ATTUAZIONE DI TERMINI E CONDIZIONI PER LA GESTIONE DELLA PROPRIETÀ INTELLETTUALE PER LO STANDARD IEEE1599
Milano, Università degli Studi di Milano, 2015

Adriano Baratè, Goffredo Haus, Luca A. Ludovico, Paolo Perlasca
MANAGING INTELLECTUAL PROPERTY IN A MUSIC FRUITION ENVIRONMENT
Milano, Università degli Studi di Milano, 2016

Luca A. Ludovico
KEY CONCEPTS OF THE IEEE 1599 STANDARD
Milano, Università degli Studi di Milano

IEEE Computer Society
IEEE RECOMMENDED PRACTICE FOR DEFINING A COMMONLY ACCEPTABLE MUSICAL APPLICATION USING XML
New York, The Institute of Electrical and Electronics Engineers, 2008

Bianca Falcidieno, Michela Spagnuolo,Yannis Avrithis, Ioannis Kompatsiaris, Paul Buitelaar (Eds.)
SEMANTIC MULTIMEDIA
Genova, Second International Conference on Semantic and Digital Media Technologies, 2007

SITOGRAFIA

Le risorse elencate nel repertorio qui presentato si intendono consultate ed accessibili a ottobre 2017.

EMIPIU

emipiu.dico.unimi.it

OPEN DIGITAL RIGHTS LANGUAGE (ODRL) VERSION 1.1

www.w3.org/TR/odrl/

ODRL COMMUNITY GROUP

www.w3.org/community/odrl/

BOOTSTRAP

getbootstrap.com

STACK OVERFLOW

stackoverflow.com

GRUPPO UTILIZZATORI ITALIANI DI TEX

www.guitex.org/home/

RINGRAZIAMENTI

Un ringraziamento sincero è rivolto al relatore di questo elaborato, il Prof. Paolo Perlasca, per la grande disponibilità, pazienza, attenzione, interesse e gentilezza dimostrati durante la redazione dello stesso, e ai correlatori Adriano Baratè e Luca Andrea Ludovico, per il sostegno e la disposizione dimostrati in caso di dubbi o problemi nello sviluppo del lavoro svolto.

Un ringraziamento doveroso è rivolto ai miei genitori, per l'indispensabile sostegno economico e per il continuo incoraggiamento. Grazie ad entrambi per aver sempre creduto in me e nelle mie capacità, anche nei momenti più difficili, e grazie per avermi alleggerita nella pratica da tutte le questioni esterne da risolvere, consentendomi di concentrarmi a pieno sul completamento di questo elaborato.

Ringrazio con affetto mia sorella Roberta e mio fratello Andrea, che durante lo sviluppo del progetto sono sempre stati presenti, incoraggiandomi in ogni occasione e rendendosi sempre disponibili nei momenti in cui avevo bisogno di loro. Grazie anche per il vostro modo di scherzare e di farmi ridere anche in situazioni di difficoltà, per alleggerire la mia tensione e ricordarmi sempre che ci siete per me.

Un ringraziamento va a tutti i miei parenti, che mi sono stati di sostegno e che, anche da lontano, mi sono stati vicini, facendomi sentire tutto il loro affetto. In particolare ringrazio mia zia Alessandra e mia cugina Valentina, le quali, senza esitazione, si sono mostrate disposte a sacrificare il loro tempo e la loro comodità per venire incontro alle mie esigenze. Ringrazio anche mia cugina Serena, che mi ha fornito un aiuto indispensabile per conciliare il lavoro sul progetto con la preparazione della lettera motivazionale per l'accesso al corso di laurea magistrale.

Desidero ringraziare tutti i miei amici, da Bologna e da Milano, che mi hanno supportato con un consiglio, un parere, o una semplice chiacchierata. In particolar modo, ringrazio Francesca Zerbini, che con il suo spirito vivace e premuroso mi ha sempre trasmesso grande affetto e allegria e nello stesso tempo ha saputo incoraggiarmi e sostenermi. Ringrazio Valentina Olivi e Francesca Tirotta per le bellissime chiacchierate, per aver ascoltato tutti i miei racconti e per avermi incentivata nella stesura di questo elaborato. Ringrazio Giulia Testoni e Laura Fabbri per le serate in baracchina a ridere e scherzare, in grado di tirarmi sempre su di morale anche in un periodo impegnativo come quello degli ultimi mesi. Ringrazio Linda

Ceroni, che essendo stata la mia coinquilina, ha vissuto insieme a me i momenti di difficoltà degli ultimi tre anni ed è sempre stata un supporto indispensabile. Ringrazio Alessandro Reggiani, che ha insistito perchè io intraprendessi un percorso legato alla musica e senza il quale probabilmente non avrei scelto di laurearmi in Informatica Musicale. Ringrazio Benedetta Menaggia e Veronica Curioni, che sono state persone preziose ed essenziali durante la mia permanenza a Milano per la laurea triennale e lo rimarranno anche da lontano. Ringrazio Ivan Rossi per l'incoraggiamento e per l'interesse mostrato verso gli argomenti che ho trattato nell'elaborato e Marco Tiraboschi per essersi reso sempre disponibile a risolvere qualsiasi mio dubbio. Ringrazio tutti gli altri amici che, in un modo o nell'altro, mi sono stati di supporto da vicino e da lontano.

Infine, un sentito ringraziamento va a Roberto Falcone, che mi è rimasto accanto durante tutto il percorso di sviluppo del progetto e stesura di questo elaborato. Grazie per il tuo sostegno, la tua forza, il tuo incoraggiamento, la tua capacità di trasmettermi sicurezza, la tua premura nei miei confronti e il tuo pieno coinvolgimento, senza i quali tutto questo non sarebbe stato possibile. Grazie per aver affrontato insieme a me questa sfida e per avermi aiutato in mille modi, dagli aspetti pratici agli aspetti emotivi. Grazie per il tuo affetto e la tua dolcezza e per essermi stato accanto sempre, considerandomi la tua prima priorità e sacrificando il tuo tempo e i tuoi impegni pur di sostenermi. Ti sono infinitamente grata per tutto e rimarrai sempre la mia principale fonte di *nefer*.

COLOPHON

Questo lavoro è stato realizzato con LaTeX 2ε utilizzando uno stile d'impaginazione ispirato alla celebre opera *Gli elementi dello stile tipografico* di Robert Bringhurst e ricreato da Roberto Falcone.
Tutti gli accorgimenti tipografici e relativi all'impaginazione sono stati appositamente adottati per offrire al lettore la migliore esperienza di lettura.

Versione finale redatta a Lugano il 9 ottobre 2017 alle ore 10:03.