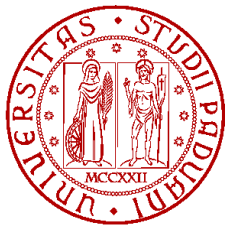
UNIVERSITÀ DI PADOVA  **FACOLTÀ DI INGEGNERIA**

TESI DI LAUREA

Progettazione e sviluppo di una piattaforma mobile di digital learning per promuovere l'alfabetizzazione digitale attraverso attività culturali

Laureando: Luca Bianconi

Relatore: Prof. Sergio Canazza
Correlatore: Ing. Niccolò Pretto

Corso di Laurea Magistrale in Ingegneria Informatica

Padova, 04/04/2017

Anno Accademico 2016/17

Abstract

La tesi descrive il progetto e lo sviluppo di una piattaforma informatica finalizzata all'apprendimento tecnologicamente aumentato tramite strumenti *social*.

Nel primo capitolo vengono presentati il contesto di diffusione della tecnologia informatica, applicata alla cultura e all'insegnamento, e le motivazioni che hanno ispirato il progetto.

Nel secondo e terzo capitolo, dopo aver delineato lo stato dell'arte, vengono individuati i requisiti dello strumento ideato e vengono presentate le risorse utilizzate per realizzarlo. Inoltre viene descritta l'architettura della piattaforma creata e l'implementazione delle sue componenti *client* e *server*.

Nel quarto capitolo vengono esposte le conclusioni e proposti dei possibili sviluppi futuri.

Ringraziamenti

Desidero ringraziare il Prof. Sergio Canazza e l'Ing. Niccolò Pretto per il prezioso aiuto e il supporto nel lavoro a questa tesi.

Ringrazio i miei genitori, per i loro sacrifici, e in modo particolare mia madre che mi è stata sempre affianco.

Ringrazio i miei familiari, amici e conoscenti che mi hanno motivato in questo percorso.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Da diversi anni i sistemi informatici sono stati impiegati nell'ambito della cultura e dell'insegnamento permettendo di evidenziarne potenzialità e limiti. I dispositivi *mobile* con cui oggi siamo tutti abituati a interagire offrono una opportunità notevole per migliorare l'insegnamento e l'apprendimento nelle scuole e possono essere mezzo fruttuoso per la divulgazione e la promozione culturale.

In questo primo capitolo si vuole dare una visione panoramica del contesto in cui il progetto si colloca sottolineando le ragioni per cui può risultare efficace e di successo. Si discuterà quindi della crescita del mercato dei *device* portatili e del ruolo della tecnologia nella cultura e nell'insegnamento. Si mostrerà perché i dispositivi odierni rappresentano un'occasione per questi settori e infine si illustrerà lo stato dell'arte riportando dei progetti simili.

1.1 La diffusione dei dispositivi mobili

Negli ultimi dieci anni, l'enorme diffusione e la popolarità dei dispositivi portatili hanno determinato la loro adozione da parte della quasi totalità dei consumatori. Anche le persone che non hanno mai utilizzato un *personal computer* oggi sfruttano tali dispositivi come strumento per la ricerca e la fruizione di informazioni, la comunicazione, la collaborazione, per lavoro o per l'intrattenimento [9]. L'evoluzione dei dispositivi, in particolar modo *smartphone* e *tablet*, ha reso elementare il loro uso, portando a interagire con essi sia bambini in età infantile [13] sia persone anziane, con maggiori difficoltà [2].

Il derivante interesse economico ha alimentato notevolmente l'innovazione nel campo del software, specialmente nel settore dell'*application intelligence* [27]. Questo campo di ricerca sviluppa tecniche di *machine learning* per creare applicazioni che interpretano i dati, elaborano delle previsioni e propongono molteplici soluzioni. Approccio, riassunto dal termine *smart*, che ha contribuito al successo dei dispositivi portatili insieme alla ricerca di interfacce grafiche sempre più semplici, gradevoli e intuitive.

La crescita dell'uso dei dispositivi mobili non sembra rallentare: le ricerche web generate da *smartphone* e *tablet* nel 2013 rappresentavano, secondo *comScore*, il 25% del totale, il 29% nel 2014 e divenendo, secondo *Google*, la maggioranza nel 2015 [15].

Tali strumenti sono sempre più impiegati anche in ambito lavorativo dove le politiche di tipo BYOD, *Bring Your Own Device*, danno lo spunto per soluzioni specifiche come i *tablet* ibridi: una integrazione fra *notebook* e *tablet*, che sono ora uno dei segmenti in maggiore sviluppo dell'industria dei computer [9].

Il loro impiego per la creazione di contenuti

Prima dell'arrivo dello *smartphone* i telefoni cellulari, i palmari e i lettori musicali erano utilizzati quasi esclusivamente con lo scopo di leggere dei contenuti: fruire di informazioni ricavate da diverse sorgenti. La loro capacità e praticità nella produzione dei contenuti erano limitate a piccoli documenti testuali e a fotografie di scarsa qualità. La realizzazione dei documenti digitali era riservata all'ambiente *desktop*, PC fissi o *notebook*, e a strumenti specifici, per esempio macchine fotografiche, registratori e tavole grafiche.

I dispositivi portatili moderni consentono di essere utilizzati anche per la creazione di contenuti: testi, immagini, registrazioni o dati di diversa tipologia. Il loro equipaggiamento con svariati sensori quali fotocamere, ricevitori GPS, accelerometri, giroscopi, magnetometri e altri li rende sorgenti potenziali di grandi quantità di dati. Risultano quindi strumenti molto versatili, sfruttabili in molteplici situazioni per produrre contenuti spesso più completi di quanto non sia possibile ottenere con un *personal computer* o dispositivi tradizionali. Grazie a uno *smartphone* oggi è possibile, ad esempio, documentare un luogo da molteplici punti di vista (fotografie, georeferenze, ecc...) con semplici operazioni eseguibili individualmente.

Queste possibilità hanno contribuito alla crescita del mercato *mobile* e hanno messo in secondo piano l'ecosistema *desktop*, che vede lentamente scendere la percentuale dei suoi utilizzatori [14].

1.1.1 Il ruolo dell'interfaccia utente

Abbiamo già detto come la facilità di utilizzo degli attuali dispositivi mobili sia chiave del loro successo. A sua volta la semplicità di interazione dipende dall'interfaccia utente: livello di astrazione grafico mediante il quale l'utilizzatore inserisce dati e ricava informazioni.

Così come le prime macchine fotografiche e automobili, anche i primi computer erano usabili solo da persone che dedicavano tempo e sforzi allo studio della tecnologia. Con l'evolversi della tecnologia le cose sono cambiate. La scienza interdisciplinare della *human-computer interaction* prende vita combinando i metodi per reperire informazioni e una base di sperimentazioni psicologiche con l'efficienza e la diffusione dei dispositivi informatici [26]. A queste si aggiungono i contribuiti di psicologi dell'educazione e dell'industria, insegnanti e grafici, esperti di ergonomia, antropologi e sociologi.

La tecnologia delle *User Interface* (UI) è nel mezzo della sua fase evolutiva. *Smartphone* e *tablet* che usano Google Android, Apple iOS e Windows Phone offrono tutti approcci differenti.

La differenza delle interfacce è intenzionale: le compagnie vogliono distinguersi dalle altre per identificarsi e conquistare una fetta del mercato. Sebbene questa competitività sia positiva per il miglioramento globale delle UI rappresenta anche una sfida per gli sviluppatori di applicazioni e siti web per questi dispositivi.

Per creare questo tipo di applicazioni occorrono:

- abilità in diverse tecnologie di sviluppo software;
- conoscenza di un insieme vasto e in costante cambiamento di dispositivi;
- conoscenza di diverse convenzioni e standard;
- sforzi per scrivere il codice in diverse piattaforme (*porting*);
- collaudi multidispositivo.

Lo sviluppo web *mobile* invece rappresenta una soluzione più efficiente nei costi. Esistono infatti strumenti che consentono di perseguire l'idea *"write once, run everywhere"* [31] e liberano gli sviluppatori dall'obbligo di conoscere approfonditamente le diverse piattaforme e reimplementare il codice per ognuna di queste.

In ogni caso sviluppare UI *mobile* ha le sue intrinseche implicazioni:

- fattori di forma ridotti;
- interazione *touch*;
- sensibilità all'orientazione;
- animazione pervasiva;
- simulazione del comportamento fisico degli oggetti nell'interfaccia;
- adattabilità a molteplici dimensioni.

Un aspetto chiave per la realizzazione di interfacce grafiche su dispositivi mobili è il *responsive design*. Esso consiste nell'insieme di pratiche atte a visualizzare correttamente l'interfaccia utente su dispositivi di diversa dimensione in maniera da ottimizzare le possibilità che tale fattore di forma offre [21].

L'obiettivo del *responsive design* è rendere ogni interfaccia come se fosse appositamente realizzata per ogni *device* e *browser* in cui viene mostrata. *Layout* complessi sono indicati per schermi di dimensioni generose mentre diventano inutilizzabili su schermi ridotti. Viceversa *layout* semplici adatti agli schermi piccoli risultano noiosi da navigare ed esageratamente scarni su quelli grandi. Un approccio possibile e largamente utilizzato è quello di riformattare un *layout* multicolonna in uno a singola colonna quando la risoluzione del dispositivo è limitata, figura 1.1.

Oltre al *layout* è opportuno rendere *responsive* anche il contenuto. È utile quindi identificare le informazioni principali e quelle secondarie facendo in modo che le prime appaiano in risalto nel flusso di utilizzo su qualsiasi dispositivo. Alcuni contenuti saranno quindi ridotti, spostati o

(a) Layout iPad multicolonna (b) Layout iPhone a colonna singola

Figura 1.1: Responsive layout *in Apple iOS*

nascosti a seconda dello spazio disponibile per visualizzarli.

Un importante aspetto è il riutilizzo di *design pattern* comuni che facilita la fruizione del software. Un *design pattern* è una soluzione comprovata per uno specifico problema ricorrente. Alcuni di questi sono validati da test di usabilità formali mentre altri vengono mantenuti per effetto di rete. Dato che i *pattern* sono comuni in diverse applicazioni, essi risultano familiari agli utenti che non hanno bisogno di apprenderne di nuovi.

1.2 La tecnologia informatica nell'istruzione

Nel mondo dell'educazione il campo dell'*Information and Communication Technology* (ICT) viene da sempre considerato utile per supportare l'insegnamento. Diversi studi sono stati condotti per migliorare il rapporto che la tecnologia informatica ha con l'istruzione [28]. Il risultato è un insieme di modelli sviluppati per descrivere le conoscenze necessarie a rendere proficua l'adozione della tecnologia nelle scuole. È stato evidenziato come la conoscenza dell'ICT da parte degli insegnanti giochi un ruolo fondamentale in questo processo [28]. Si è quindi sviluppato il concetto di *ICT-related Pedagogical Content Knowledge* (ICT-PCK), rappresentante le differenti conoscenze che concorrono a formare la base di sapere necessaria all'insegnamento. Esse comprendono nozioni pedagogiche, conoscenze dell'area in cui la materia si colloca, conoscenza degli studenti e, infine, dell'ICT. Con conoscenza dell'ICT si definiscono le nozioni necessarie per l'uso del computer, di molteplici strumenti software nonché delle loro caratteristiche e potenzialità.

L'ICT-PCK può quindi essere descritto come: l'insieme delle modalità con cui la conoscenza degli strumenti tecnologici, della pedagogia, del contenuto, degli allievi e del contesto educa-

tivo porta a una consapevolezza di come particolari argomenti possono essere trasmessi a uno specifico pubblico sfruttando l'ICT in modo tale da aggiungere valore all'insegnamento [3].

Si identificano cinque fasi per la realizzazione dell'ICT-PCK:

1. Identificazione degli argomenti spiegabili tramite l'ICT e dove l'ICT può significativamente aggiungere valore all'insegnamento. È il caso di argomenti difficili da comprendere per gli studenti o difficili da trasmettere efficacemente dall'insegnante.

2. Identificazione delle rappresentazioni che permettono di trasformare ciò che viene spiegato in maniera tradizionale in forme più facilmente comprensibili dagli allievi e non supportate dagli strumenti tradizionali.

3. Identificazione delle strategie di insegnamento non realizzabili con metodi tradizionali. Per esempio applicazione di concetti a contesti non disponibili realmente, insegnamenti interattivi o che necessitano di adattarsi ai bisogni di uno specifico studente.

4. Selezione degli strumenti di ICT che hanno le idonee caratteristiche per supportare le trasformazioni dei contenuti e le strategie di insegnamento identificate.

5. Adozione delle relative attività basate sull'ICT in classe.

Similmente con il termine *eLearning*, apprendimento digitale, si identifica il ricorso alle tecnologie informatiche per migliorare la qualità dell'attività didattica e dell'apprendimento. Esso mira non solo all'apprendimento tramite il mezzo tecnologico, ma anche alla promozione della conoscenza del mezzo stesso per l'arricchimento delle competenze in materia di nuove tecnologie [10].
L'*eLearning* può essere di tre tipologie [8]:

- **recettivo**: la conoscenza è acquisita in maniera passiva con poca o nulla interazione con il contenuto da parte dello studente;

- **direttivo**: la conoscenza è acquisita tramite un percorso guidato che passo dopo passo stimola la ricerca di una risposta;

- **scoperta guidata**: l'apprendimento avviene tramite una alta interattività in simulazioni, giochi, sperimentazioni e risoluzione di problemi.

Il tipo di insegnamento è quindi determinato dal mezzo multimediale che viene adottato.

L'interattività è l'elemento fondamentale per la realizzazione di una didattica attiva e dipende dalle caratteristiche di personalizzazione e adattabilità dell'applicazione. È stato infatti provato che un approccio multimediale interattivo nel processo educativo comporta migliori risultati di apprendimento, permette agli studenti di essere maggiormente coinvolti e autonomi nel loro percorso didattico [28].

1.3 La tecnologia al servizio del patrimonio culturale

1.3.1 Conservazione del patrimonio culturale

Sono innumerevoli i ricorsi alla *computer science* nell'ambito della conservazione culturale. L'obiettivo è quello di usare le moderne tecnologie per migliorare il mantenimento delle opere riducendone il naturale deterioramento, impedendone l'alterazione o lo smarrimento. Ad esempio, è possibile proteggere quei manufatti sensibili alla manipolazione creandone copie qualitativamente identiche destinate a essere fruite dagli studiosi o chiunque ne sia interessato. È il caso della conservazione e la valorizzazione dei nastri magnetici [25], ambito di ricerca del Centro di Sonologia Computazionale (CSC) dell'Università di Padova. In tale applicazione la tecnologia digitale consente di allegare al contenuto stesso un insieme di informazioni, metadati, che lo completano. La digitalizzazione permette l'archiviazione e la protezione di grandi moli di documenti storici trasferendone il contenuto su supporti moderni. Sebbene i nuovi supporti possano donare nuova vita ai documenti non si ha la certezza che questi possano durare per un lungo periodo. Inoltre non è spesso possibile riprodurre l'opera senza perdere dell'informazione. Questo significa che la digitalizzazione permette solamente la creazione di copie fruibili, portatili, di facile e diffuso utilizzo che non possono tuttavia rimpiazzare l'opera autentica .Se però è possibile abbandonare l'idea di "conservare l'originale" in favore dell'approccio "preservare il contenuto, non il supporto" allora la digitalizzazione si rivela il migliore strumento [6].

Oltre a eliminare il problema di mantenere l'integrità del manufatto, si slega il contenuto dallo specifico strumento adatto a riprodurlo il quale può essere simulato, figura 1.2, ricreando l'esperienza d'uso originale in maniera filologicamente corretta [5].

Attraverso la rete, si può consentire l'accesso simultaneo a più persone in diversi luoghi, non limitato ai soli utenti umani. Un documento digitalizzato può essere infatti analizzato anche attraverso procedure software con vari obiettivi: lettura, ricerca ed elaborazione.

Altri esempi di ricorso alla tecnologia informatica nel campo della salvaguardia dei beni culturali si hanno con la scansione tridimensionale del patrimonio culturale attraverso la *computer vision* [19], figura 1.3(a), la ricostruzione di siti archeologici grazie a modelli computerizzati [16] o la riproduzione di manufatti tramite stampa tridimensionale [1], figura 1.3(b).

La tecnologia consente infine di creare delle opere culturali nuove: per esempio fotografie, immagini, filmati, registrazioni e tracce musicali.

1.3.2 Promozione del patrimonio culturale

Sotto l'aspetto della promozione culturale l'idea è sempre stata quella di arricchire l'esperienza del visitatore. Qualsiasi sia la tipologia dell'eredità culturale una qualche forma di materialità e fisicità è solitamente più propensa a creare interesse e condivisione. I musei scientifici sfruttano l'interazione fisica come una via efficace per invogliare le persone a esplorare concetti, idee e oggetti [24].

Musei più tradizionali, invece, tendono a organizzare delle sessioni controllate dove un ristretto numero di articoli, che possono essere cautamente manipolati, vengono messi a disposi-

Figura 1.2: *Schermata principale dell'applicazione REMIND per la simulazione dell'esperienza d'ascolto filologicamente corretta di un nastro magnetico*

(a) *Point cloud* del palazzo Dolmabahce, Istanbul

(b) Stampa 3D della testa di un putto acefalo, chiesa Castello di San Martino dall'Argine, Mantova

Figura 1.3: *Tecnologia applicata alla salvaguardia dei beni culturali*

zione dei visitatori. I problemi di conservazione possono impedire di poter toccare gli oggetti in esposizione. Con l'avvento della tecnologia digitale, sempre più opere culturali sono state arricchite da un corredo di dati multimediali e sistemi interattivi che generano una esperienza multisensoriale e consentono talvolta una totale coinvolgimento dei sensi. C'è quindi l'opportunità di integrare la tecnologia per sfruttare l'esperienza fisica dei visitatori anche se questo comporta delle sfide da superare.

La progettazione di sistemi digitali per realizzare questa esperienza fisica deve prendere in considerazione l'equilibrio fra obiettivi dei promotori e delle necessità dei visitatori [24]

- Il mezzo digitale può arricchire l'esperienza, ma può facilmente spostare l'attenzione dal contenuto, limitando la riflessione e il pensiero critico.
- I modelli digitali possono ricreare fedelmente gli articoli e il loro contesto ma possono contribuire alla diminuzione del valore percepito dell'originale (la sua "aura" e autenticità).
- I contenuti digitali spesso impegnano il visitatore in quiz, giochi o dettagliate letture distogliendolo dalla esperienza reale: la visita.
- Il ritmo di una visita rischia di essere determinato dal mezzo digitale impedendo all'individuo di concentrarsi maggiormente su ciò che più gli interessa.
- Le tecnologie interattive spesso interferiscono con la socializzazione fra i visitatori. È il caso, ad esempio, delle audio guide che isolano le persone, ma anche di piattaforme interattive (e.g. giochi) che prevedono un singolo utente per sessione.
- L'impiego della stessa soluzione digitale, (e.g. applicazione), in diversi luoghi può essere considerata generica e non esaustiva. Mentre soluzioni specifiche sono più costose e lunghe da implementare.
- Diverse età dei visitatori hanno bisogno di diverse attenzioni.

1.3.3 La realtà aumentata

Con realtà aumentata (AR) si intende la "volontà di arricchire la visione del mondo sovrapponendo oggetti virtuali alla realtà, in maniera da persuadere l'osservatore che l'oggetto virtuale faccia parte dell'ambiente reale" [17]. Gli operatori museali sono da sempre interessati alla realtà aumentata: sia i manufatti che le aree sono spesso corredate da documenti multimediali come immagini, mappe e filmati. Altri esempi sono le installazioni nelle sale, come il progetto *Harmonic Walk* [22], che consentono al visitatori di comunicare con l'ambiente e reagire agli stimoli che l'ambiente produce.

Un primo riscontro dell'utilizzo della realtà aumentata applicata alla cultura, dopo le ormai vetuste audioguide, si ha nel 2001 con il sistema Archeoguide [17, 29] che permetteva ai visitatori della città di Olimpia, in Grecia, di ricevere informazioni contestualizzate rispetto alle loro posizione e orientazione rilevate tramite GPS. Questo sistema necessitava di uno dispositivo

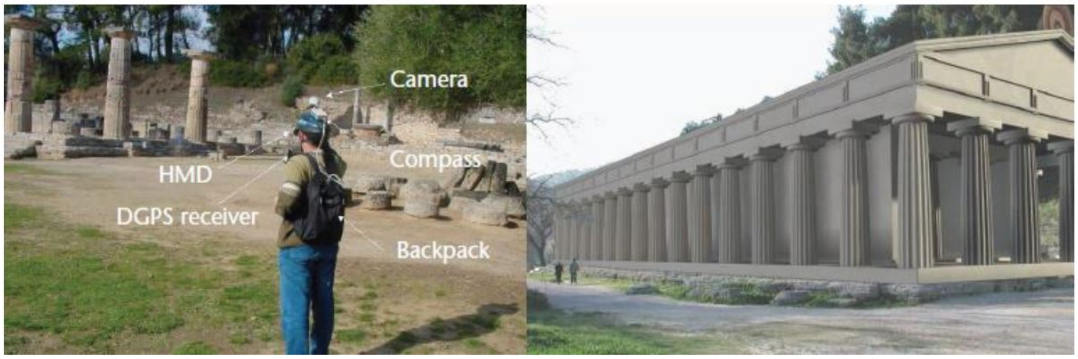


Figura 1.4: *Il sistema Archeoguide e la sua simulazione virtuale*

con display, indossabile sul capo, contenente una videocamera esterna, una bussola digitale e collegato al ricevitore GPS posto all'interno di uno zainetto, figura 1.4.

L'idea di realtà aumentata ben si sposa con i dispositivi portatili odierni. La loro capacità di calcolo e i loro sensori permettono l'ideazione di servizi di realtà aumentata che rimuovono il vincolo di trovarsi in un'area specificamente equipaggiata. Attualmente i moderni *smartphone* e *tablet* sono in grado di sostenere applicazioni di realtà aumentata senza l'ausilio di ulteriori dispositivi. Per una esperienza più immersiva oggi esistono strumenti come visori per *virtual reality* (VR), *smart glasses* e *smartwatch*.

Nei sistemi *mobile* di realtà aumentata sia il divertimento che l'utilità percepita sono importanti per l'utilizzo di questa tecnologia in contesti culturali. Una applicazione di questo genere dovrebbe quindi focalizzarsi anche su questi due aspetti.

1.4 L'opportunità didattica dei dispositivi mobili

Il *mobile computing* è stato introdotto gradualmente negli ultimi due decenni all'interno del contesto educativo [7, 28]. La tecnologia permette oggi alla gran parte delle persone di avere con sé un piccolo calcolatore: un *laptop*, un *tablet* PC, uno *smartphone* o un ebook *reader*. Queste capacità di calcolo e portabilità, combinate alla possibilità di comunicazione *wireless* e a molteplici sensori, forniscono al *one-to-one computing* [7] un grande potenziale nel campo dell'insegnamento.

Gli effetti dell'introduzione di dispositivi portatili nell'educazione sono studiati da alcuni anni [28]. Appare chiaro che migliorano il livello di apprendimento rispetto sia alle lezioni tradizionali sia all'uso di dispositivi *desktop*. All'interno del mondo portatile si nota inoltre un effetto più accentuato nell'utilizzo di dispositivi *handheld*, ovvero *smartphone*, *tablet* ed *e-reader*, rispetto ai meno recenti *notebook* e *laptop*. Le caratteristiche uniche dei dispositivi *mobile* sono in grado di migliorare la funzionalità di specifici metodi di insegnamento e, quindi, promuovere i risultati educativi. Il possesso di un proprio dispositivo crea una individualità nello studio che diventa più autonomo e autoregolato nei tempi. Inoltre si risolvono alcune difficoltà di intro-

durre valutazioni istantanee dell'apprendimento, per esempio valutazione in attività all'aperto di eguale difficoltà.

L'arricchimento dell'insegnamento e dell'apprendimento deriva anche dalla capacità dei dispositivi di rilevare diverse informazioni del contesto in cui operano permettendo agli allievi di estrapolare informazioni dell'ambiente visitato, registrare e reagire ai dati ottenuti attraversando diversi luoghi di interesse, come nel caso di musei o attività sul campo.

Oltre alle caratteristiche intrinseche dei dispositivi, si è evidenziata l'importanza delle modalità con cui questi vengono sfruttati. In particolare ambienti di insegnamento informale hanno maggior successo rispetto a quelli formali così come adozioni di breve-media durata risultano più efficaci rispetto a progetti di lungo termine. Inoltre metodologie *activity-based*, caratterizzate dallo svolgimento per *step* di un progetto o inchiesta, danno migliori risultati rispetto all'utilizzo in congiunzione con le lezioni, allo studio autonomo, all'apprendimento in gruppo e attraverso il gioco. Queste considerazioni sono soggette a specifiche variazioni soprattutto in relazione al target dell'insegnamento e alla constatazione che le ricerche effettuate non sono tali da rendere questi risultati assoluti [28].

Nel contesto di forte sviluppo della consapevolezza del mezzo digitale presentato, nelle sezioni 1.1 e 1.2 non si può fare a meno di cogliere una opportunità dal punto di vista didattico.

Smartphone e *tablet* si collocano in questa categoria di strumenti. La loro diffusione, semplicità e praticità consentono il loro impiego nell'educazione da parte sia degli studenti che degli insegnanti i quali sfrutterebbero così, in maniera proficua, la tecnologia a loro familiare. Ciò, oltre ai vantaggi già esposti, consente di riqualificare l'uso del dispositivo come strumento proattivo per la crescita dell'individuo. L'oggetto personale conosciuto permette di ridurre, o eliminare, i tempi di apprendimento di tecnologie specifiche, sia da parte dello studente che dell'insegnante e di ridurre i costi che le istituzioni scolastiche sostengono per l'adozione di tecnologie destinate a diventare presto obsolete [20].

1.5 Stato dell'arte

Il progetto che verrà descritto nel capitolo 2 si colloca all'interno di un contesto già parzialmente occupato da altre soluzioni. Di seguito ne sono mostrate alcune.

1.5.1 Moodle

Moodle, acronimo di *Modular Open-source Object oriented Learning Enviroment*, è un software per la gestione dell'insegnamento, in inglese *Learning Managment System* (LMS), gratuito e distribuito con licenza *General Public License* (GNU) che consente la modifica e ridistribuzione del codice. È scritto in linguaggio di programmazione PHP e si interfaccia con un DBMS, *Database Managment System*, di tipo relazionale.

Originariamente fu sviluppato da Martin Dougiamas per aiutare gli educatori a creare corsi online orientati all'interazione e alla costruzione collaborativa dei contenuti. La prima versione venne rilasciata nell'Agosto 2002. Ora Moodle ha raggiunto la versione 3.2 ed è sviluppato dalla

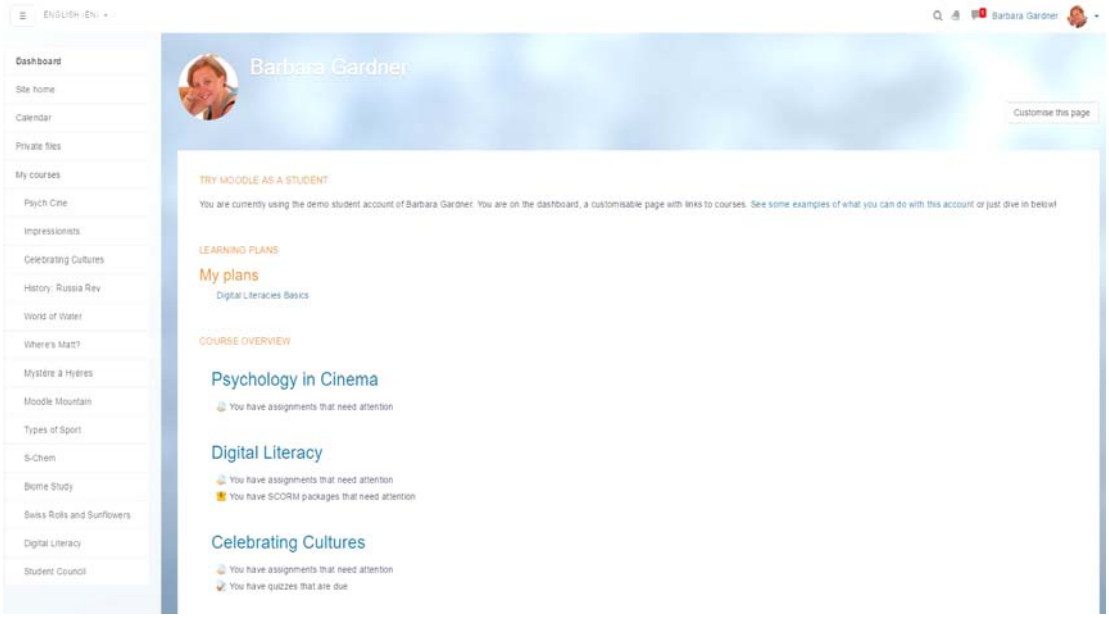


Figura 1.5: Dashboard *del sito dimostrativo Moodle*

compagnia australiana Moodle HQ [12], assistita da una rete di programmatori *open source*. Basata su principi pedagogici, la piattaforma può migliorare l'ambiente di insegnamento ma è usata anche per l'allenamento, lo sviluppo e a fini aziendali.

Risulta uno degli strumenti leader nel settore delle piattaforme di insegnamento virtuale secondo, per diffusione, solo a Blackboard. Viene principalmente usato nell'istruzione secondaria avanzata, scuole superiori e università, figura 1.5.

Fattori del successo di Moodle sono:

- **Approccio pedagogico**
  La filosofia di Moodle include una idea costruttivista e sociale dell'educazione enfatizzando la capacità degli studenti di contribuire all'esperienza di apprendimento. In questo modo Moodle diventa strumento per comunità di apprendimento e non per singole persone.

- **Plugin**
  Moodle sfrutta una struttura modulare. I *plugin* sono un insieme di strumenti flessibile che permette di estendere le funzionalità di base della piattaforma. Esistono centinaia di *plugin* archiviati nella *Moodle plugin directory*.

- **Temi grafici**
  È possibile modificare l'estetica e la funzionalità del LMS attraverso l'uso di diversi temi grafici applicati all'intero sito o al singolo corso.

- **Traduzioni**
  Moodle supporta più di 100 lingue diverse. Le traduzioni sono realizzate da persone da tutto il mondo.

- **Portabilità**
  Moodle funziona su diversi sistemi operativi quali Unix, Linux, FreeBSD, Windows, OSX e in generale su ogni sistema che supporti il linguaggio PHP e un DBMS. Questo permette la sua installazione sui comuni servizi di *web hosting*.

1.5.2 Blackboard Learn

Blackboard Learn è un *virtual learning managment system* sviluppato da Blackboard Inc. Consiste in un software server *web-based* che implementa la gestione dei corsi, una architettura personalizzabile e un design scalabile. Blackboard nacque nel 1997 dall'idea di due consulenti in educazione, Matthew Pittinsky e Michael Chasen, di fornire standard tecnici per applicazioni di insegnamento online. L'anno successivo la fusione con CourseInfo LLC portò al rilascio del primo prodotto per l'*eLearning*. Una serie di acquisizioni di altre aziende del settore conduce Blackboard ad alimentare nel 2004 l'80% dei *LMS* del nord America. Nel 2006 contava 12 milioni di utenti in 60 paesi del mondo [23] ed è attualmente il sistema di *eLearning* con la fetta più ampia di mercato [18].

Come Moodle anche Blackboard viene principalmente utilizzato nell'istruzione avanzata ma, a differenza del primo, viene licenziato commercialmente ed è *closed source*.

1.5.3 SeeSaw: The Learning Journal

SeeSaw è un'applicazione per dispositivi mobile che consente di realizzare un portafoglio digitale del lavoro svolto dagli alunni. Questi possono documentare e condividere online ciò che stanno imparando a scuola attraverso foto, video, documenti testuali, link e disegni. Gli insegnanti possono osservare ciò che gli studenti hanno prodotto e organizzare i contenuti caricati. Possono appuntare contenuti ritenuti interessanti e creare contenuti a loro volta. Gli utenti della piattaforma sono organizzati in classi: i contenuti caricati possono essere commentati da tutti gli iscritti a una determinata classe. Un'altra possibilità offerta è la messaggistica fra gli utenti della piattaforma
Dalla descrizione e dall'interfaccia, figura 1.6, di SeeSaw appare chiaro come l'applicazione si rivolga a un target di studenti delle scuole primarie o secondarie di primo grado. Viene data la possibilità ai genitori degli alunni di registrarsi e visionare l'operato dei loro figli. Questi "spettatori", nella filosofia dell'applicazione, incoraggiano gli alunni a lavorare meglio.

SeeSaw non è un pacchetto software scaricabile e installabile su server auto gestito: la piattaforma è quindi la stessa per ogni istituto che voglia adottarla. La licenza è proprietaria.

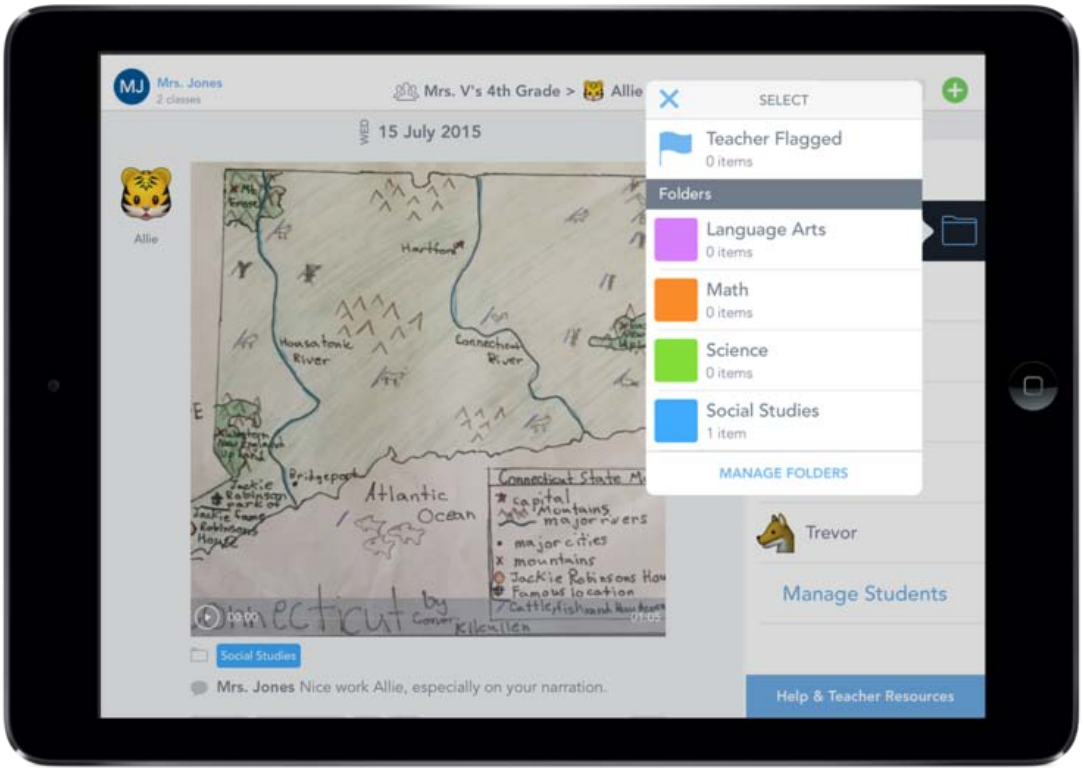


Figura 1.6: *Schermata dell'interfaccia di SeeSaw*

Capitolo 2

Il progetto

In questo capitolo verrà illustrata l'idea del progetto: una *Mobile Learning Platform to Foster Digital Literacy Through Cultural Activities*, che riassumeremo con l'acronimo MOLLY. Verranno quindi illustrati i requisiti che il *software* dovrà soddisfare una volta ultimato. Si riporteranno varie tecnologie disponibili per realizzarlo e infine gli strumenti scelti per lo sviluppo.

2.1 Obiettivo

L'obiettivo del progetto MOLLY è la realizzazione di una piattaforma informatica per l'apprendimento digitale mirata a favorire la partecipazione attiva degli studenti e l'interazione con la realtà studiata.

Vuole essere uno strumento didattico che consente agli alunni di muoversi sul territorio per andare a mappare dei luoghi con fotografie, video, registrazioni audio e documenti testuali.

Gli insegnanti avrebbero la facoltà di aprire un nuovo corso (o classe) al quale aggiungerebbero gli studenti partecipanti e altri insegnanti collaboratori. All'interno del corso potrebbero aggiungere un nuovo tema di mappatura che descriverebbe il luogo e le caratteristiche da studiare. In questo tema ogni studente, oltre agli insegnanti, potrebbe inserire i contenuti da lui creati affinché siano visibili, commentabili e valutabili dagli altri. L'insieme di questi contenuti multimediali formerebbe la base su cui gli studenti, insieme agli insegnanti, realizzerebbero un documento (articolo o *report*) del luogo studiato che chiameremo *dossier*.

Insegnanti e studenti potrebbero interagire fra loro commentando e recensendo i contenuti altrui con lo scopo di migliorare la qualità complessiva del *dossier*. Terminata l'attività un sottoinsieme dei contenuti formerebbe il *dossier*, che potrà essere reso pubblico o meno. Un *dossier* pubblicato sarà visibile a chiunque via *web*.

I genitori degli studenti potranno accedere al sistema e visionare il profilo dei loro figli osservando i contenuti da loro creati e le valutazioni che hanno ricevuto.

Studenti, genitori e professori potranno comunicare tra loro attraverso un sistema di messaggistica.

Il *target* degli studenti a cui è rivolto il progetto è composto da bambini e ragazzi con età compresa fra i 6 e i 18 anni.

2.2 Requisiti funzionali

Primari

- Gestione della piattaforma da parte di amministratori autenticati che potranno effettuare qualsiasi operazione attraverso un apposito pannello di amministrazione.
- Possibilità per gli amministratori di aggiungere e rimuovere utenti alla piattaforma e assegnarne i ruoli.
- Registrazione per qualsiasi nuovo utente come insegnante, studente o genitore.
- Ogni utente dovrà poter eseguire tutte e sole le operazioni previste per il suo ruolo. Deve quindi essere previsto un sistema di autenticazione e autorizzazione.
- Un insegnante può creare un nuovo corso. Può gestire un corso da lui creato modificandolo o eliminandolo. Può aprire un nuovo *dossier*, modificarlo, cancellarlo e archiviarlo o pubblicarlo.
- Ogni studente deve poter richiedere l'iscrizione a un corso. È da definire come tale richiesta avvenga.
- L'insegnante del corso ha facoltà di approvare o meno le richieste di iscrizione degli studenti.
- L'insegnante di un corso può iscrivervi uno studente senza che questo ne abbia fatto richiesta attraverso modalità da definire.
- All'interno di un suo corso un insegnante può rendere disponibile del materiale generico che chiamiamo "materiale del corso". Questo consiste in *file pdf* o immagini che ogni utente del corso può visualizzare.
- L'insegnante può registrare altri insegnanti come collaboratori. Gli utenti di un corso sono formati quindi da un solo insegnante (il creatore), zero o più collaboratori (insegnanti non creatori) e gli studenti iscritti al corso.
- L'insegnante può all'interno del corso creare degli avvisi testuali visibili a tutti gli utenti del corso.
- Un tema può venire assegnato a un collaboratore. Un collaboratore può modificare e archiviare un tema che gli è stato assegnato dove può aggiungere, approvare, sospendere e valutare qualsiasi contenuto. La cancellazione è, invece, ristretta ai soli suoi contenuti.
- Un collaboratore ha possibilità di aggiungere e modificare i materiali del corso e di cancellare i materiali da lui creati.

- All'interno di un *dossier* l'insegnante, un collaboratore e gli studenti possono creare contenuti di varia tipologia come immagini, registrazioni audio, ecc. Rimangono da definire le tipologie di contenuti multimediali disponibili nonché la quantità di dati (numero, dimensione, durata...) che questi contenuti possono avere.
- Possibilità da parte degli utenti di un corso di commentare testualmente i contenuti multimediali caricati da altri. Ogni commento può essere eliminato dal suo creatore, dall'insegnante del corso o da un collaboratore all'interno di un tema a lui assegnato.
- Possibilità da parte degli utenti di un corso di recensire i contenuti multimediali caricati da altri tramite un sistema semplice di valutazione.
- I contenuti multimediali non possono più essere recensiti o commentati dopo l'archiviazione del *dossier* cui sono associati.
- Un genitore può associare come suo figlio uno studente iscritto alla piattaforma. Sono da definire le modalità di tale associazione.
- Il genitore può visualizzare il profilo dei suoi figli accedendo ai contenuti da loro creati e alle valutazioni ricevute.
- Fruibilità per chiunque via *web* dei *dossier* pubblicati.
- Insegnanti e studenti usano tutte le funzionalità a loro disposizione anche attraverso un dispositivo *mobile* (*smartphone* o *tablet* Android). Per gli studenti si può decidere che la fruizione della piattaforma avvenga esclusivamente tramite dispositivo *mobile*.

Secondari

  - Modulo di messaggistica per la comunicazione fra gli utenti della piattaforma. Le conversazioni avvengono sempre con modalità uno a uno. È possibile comunque inviare lo stesso messaggio a più destinatari. Gli studenti iscritti allo stesso corso possono comunicare fra loro.
  - Sistema di messaggistica istantanea di gruppo, *chat*, fra i componenti di una classe.
  - Un insegnante titolare del corso può inviare un messaggio con destinatari tutti gli studenti iscritti, tutti i genitori di questi e tutti i collaboratori. Un collaboratore può inviare a tutti gli studenti. Studenti e genitori non possono inviare un messaggio a diversi destinatari.
  - Funzionalità di valutazione degli studenti da parte del titolare del corso riguardo un contenuto, un tema oppure globalmente. Questa valutazione è indipendente dalle recensioni ricevute dallo studente ed è inoltre riservata: solo lo studente interessato, i suoi genitori e l'insegnante possono vederla.
  - Funzionalità di semplice *feedback* sui commenti ai contenuti (es: "mi piace").

- Sistema di notifica degli eventi accaduti configurabile.
- Commentabilità e recensibilità dei *dossier* pubblici da parte degli utenti autenticati. Sarabbe commentabile il *dossier* nella sua interezza e non i singoli contenuti.
- Registrazione alla piattaforma come utente *guest*, con la possibilità di commentare e recensire i *dossier* pubblici.
- Cancellazione posposta di contenuti e commenti: una volta cancellati i contenuti entrano in uno stato transitorio di durata definita durante il quale sono visibili per verifiche solo all'insegnante e ai collaboratori.
- Possibilità di operare anche con dispositivo *mobile* Apple iOS e Windows.
- Fruibilità agevole del servizio da parte di studenti di tutte le età a cui è rivolto. In particolare quindi è richiesto che ogni funzionalità sia raggiungibile in maniera intuitiva e semplice.

2.3 Requisiti tecnici

2.3.1 Hardware

La natura del progetto rende indispensabile il ricorso a una infrastruttura *hardware* remota (accessibile a tutti gli utenti via *web*) che gestisca tutte le richieste, memorizzi e renda disponibili tutti i dati. Un'architettura *peer-to-peer*, o che preveda il mantenimento solo temporaneo, di una piccola quantità di dati violerebbe dal principio diversi requisiti funzionali. Non sarebbe possibile, ad esempio, rendere pubblici e accessibili via *web* per lungo tempo i *dossier* creati.

Server

La componente *server-side* deve poter gestire tutte le richieste degli utenti e fornire le risposte in tempi sufficientemente brevi da garantire usabilità e fluidità da parte dei *client*. Deve avere accesso a internet ed essere raggiungibile tramite internet da qualunque dispositivo con accesso *web*. Deve poter processare e immagazzinare tutti i dati inviati dagli utenti. Deve inoltre garantire un sistema di autenticazione e autorizzazione sicuro. Si vuole mantenere contenuta, in particolare, la dimensione, e quindi i costi, dell'*hardware* destinato allo *storage* dei dati.

Client

I *client* devono essere sufficientemente performanti da poter eseguire con buona fluidità tutte le funzionalità offerte da MOLLY. Per quanto riguarda i *client desktop* qualsiasi *pc*

moderno deve essere in grado di completare i *task* necessari in tempi brevi. I *client mobile* devono avere la possibilità di acquisire i vari tipi di contenuti multimediali previsti, di effettuare su questi il *processing* necessario e avere una connettività tale da permettere il trasferimento di dati richiesti dall'applicazione. Questo trasferimento deve avvenire in tempi brevi per le operazioni *real-time* in maniera da non causare ritardi che compromettano l'esperienza d'uso dell'utente. In fase iniziale si può assumere indicativamente come idoneo un qualsiasi *smartphone* Android in grado di eseguire fluidamente almeno la versione di Android OS 5.0 Lollipop e dotato di moduli *WiFi/3G*, *GPS*, accelerometro, fotocamera e microfono.

2.3.2 Software

Server

L'applicativo *server* deve fornire un sistema di autenticazione e autorizzazione sicuro. Deve gestire tutte le richieste e i dati ricevuti dagli utenti. Deve prevedere dei metodi di controllo di tali richieste. In particolare deve adottare dei meccanismi di *rate-limiting* sulle funzionalità esposte, per impedire l'abuso dei servizi offerti e prevenire sovraccarichi di lavoro della piattaforma. Inoltre vanno considerate delle procedure di limitazione della dimensione massima dei dati ricevuti, per evitare sprechi di spazio di memoria e l'eccessivo consumo della larghezza di banda degli utenti. Deve permettere agli utenti l'accesso a tutte le funzionalità a loro destinate, processare, immagazzinare e recuperare in tempi brevi i contenuti multimediali creati. La *web application* e il *database* devono garantire la capacità di servire con successo in ogni momento al carico di lavoro. È quindi necessario che presentino caratteristiche di affidabilità e scalabilità elevate.

Client

L'applicazione *client* deve interfacciarsi con il servizio web, rendere disponibili all'utente tutte le funzionalità in maniera intuitiva e rapida. Deve essere in grado di effettuare tutta l'elaborazione dei dati richiesta e garantire un alto livello di stabilità. Il consumo di energia e memoria deve essere contenuto, in particolare per i dispositivi portatili. La sicurezza delle informazioni dell'utente deve essere garantita.
L'interfaccia utente deve presentare i contenuti in maniera dinamica, pratica e intuitiva. L'applicazione *mobile* si vuole sia ben integrata con il sistema operativo, con sistema di notifiche, costrutti comuni dell'interfaccia, *gestures*, ecc, mentre l'interfaccia *web* deve poter essere correttamente visualizzata sui più diffusi *browser web*. L'interfaccia utente deve quindi rispettare il più possibile i costrutti tipici di un'interfaccia moderna realizzata a regola d'arte.

2.4 Tecnologie disponibili

2.4.1 Web Application

In questa sezione verranno illustrati vari strumenti utilizzabili per lo sviluppo del progetto. L'elenco non vuole essere esaustivo ma riportare solamente le tecnologie più conosciute e a cui più frequentemente si fa ricorso per l'implementazione di piattaforme *client-server* simili, per necessità tecniche, a MOLLY. Si è ritenuto interessante riportare, per ciascuna tecnologia descritta, un esempio di successo della sua adozione [1]. Tali tecnologie sono spesso utilizzate in maniera complementare, per sfruttare meglio le peculiarità di ciascuna.

Webserver e PHP

PHP, acronimo ricorsivo di *"PHP: Hypertext Preprocessor"*, è un linguaggio di *scripting* pensato principalmente per lo sviluppo web che consente anche la realizzazione di applicazioni *"general purpose"*. Nell'ecosistema PHP esistono le funzioni basilari e le librerie necessarie per lo sviluppo dell'applicativo *server* necessario. Si possono trovare infatti librerie per la connessione e interfacciamento con svariati *Database Management System* (DBMS), per il *processing* di immagini e dati audio nonché per la realizzazione di una *web* API. PHP nasce come linguaggio procedurale ma ha da tempo adottato una architettura *object oriented*. È in continuo sviluppo, la sua versione più recente è stata rilasciata il 19 Gennaio 2017, e la sua popolarità lo porta a essere il linguaggio di *scripting* più utilizzato a livello aziendale [4]. Esistono diversi *framework* che consentono di velocizzare l'implementazione di applicativi in PHP fornendo delle funzioni comuni, e non, che andrebbero di volta in volta scritte se si utilizzasse il PHP *"out of the box"*.

Essendo un linguaggio di *scripting*, PHP "soffre" delle problematiche di questo tipo di soluzioni. Necessita di un interprete software ogni volta che viene processato, differentemente da ciò che avviene per un linguaggio compilato, che viene eseguito direttamente perché memorizzato in linguaggio macchina. Questo può far emergere un problema di ottimizzazione, in caso di siti con moli di traffico particolarmente elevate. Per ovviare a questo limite è stata creata HHVM, acronimo di *"Hip Hop Virtual Machine"*, una macchina virtuale per la compilazione del codice PHP *just in time*. Con HHVM il codice PHP viene prima trascritto in un codice intermedio chiamato *HipHop bytecode* che viene poi dinamicamente compilato in linguaggio macchina, ottimizzato e nativamente eseguito.

PHP è il linguaggio più utilizzato per lo sviluppo di applicativi *web*, la sua diffusione lo porta ad alimentare più dell'80% di tutti i siti internet [2]. Questo anche perché, storicamente, la maggior parte dei siti di *web hosting* proponevano PHP come principale soluzione nei loro prodotti commerciali.

Oltre a Moodle, sezione 1.5.1, altri esempi di grande successo dell'uso di PHP sono Facebook, Yahoo e Wikipedia.

Sebbene PHP possa essere usato come linguaggio di programmazione generico, e quindi consenta di implementare un *web server* (ne esiste uno realizzato per scopi di test), viene solita-

[1] informazioni ricavate da stackshare.io

[2] fonte w3techs.com, 13 Marzo 2017

mente utilizzato in coppia con un *web server* HTTP appositamente sviluppato. Il *server* HTTP, per esempio Apache o Nginx, funge da *proxy* per le richieste pervenute e reindirizza all'interprete PHP quelle specificate nella configurazione del *web server*. Scenario comune è quello di lasciare al *server web* la gestione delle richieste verso file statici e all'interprete PHP quelle verso risorse dinamiche.

Java e Spring

Il *framework* Spring è un insieme completo di componenti *software* per la creazione di applicazioni aziendali in linguaggio Java. Spring si preoccupa di fornire dei modelli che formano l'ossatura di ogni applicazione lasciando agli sviluppatori la libertà di focalizzarsi sulla *business logic*, indipendentemente dallo specifico ambiente di sviluppo.

È composto da un *layer* di moduli chiamato *Core Container* che contiene quattro moduli per le funzionalità fondamentali del *framework*. Altri componenti estendono le funzionalità di base come il *layer Data Access/Integration* che realizza una interfaccia di astrazione per l'accesso a dati attraverso diversi sistemi di memorizzazione delle informazioni quali DBMS, XML e *database* non relazionali. Il *layer web* fornisce strumenti per la realizzazione di *web application* secondo architetture MVC e REST/HATEOAS.

Una caratteristica importante di Spring la *Inversion of Control Container* (IoC) che permette la creazione e la gestione di oggetti tramite *reflection*. Questa consente di creare la struttura degli oggetti in file separati, XML, e, tramite *dependency injection*, lasciare all'applicazione il lavoro di collegare questi oggetti fra loro, gestire le loro dipendenze e creare i loro metodi comuni per il loro ciclo di vita.

Spring viene principalmente utilizzato per lo sviluppo di soluzioni aziendali interne. Alcune compagnie che risultano usare il *framework* sono Gazelle, Wix e Ticketmaster.

NodeJs

NodeJs è un ambiente d'esecuzione *runtime*, multipiattaforma per l'esecuzione *server side* di applicazioni *web*. La maggior parte dei suoi componenti è scritta in linguaggio JavaScript. Il codice, sviluppato in linguaggio JavaScript, viene interpretato ed eseguito sfruttando il Google's V8 JavaScript engine.

È stato ideato nel 2009 da Ryan Dahl con l'idea di migliorare i comuni *server web* considerati limitati nella capacità di sostenere un gran numero di connessioni concorrenti. NodeJs implementa quindi un'architettura *event driven* che consente la realizzazioni di *web server* estremamente veloci nel gestire in maniera asincrona le richieste pervenute. Questo è reso possibile da una API di *input/output* di basso livello che risponde ai segnali, inviati dall'*hardware* alla ricezione e all'invio dei pacchetti di rete, invocando direttamente il corrispondente *event handler*. Ciò significa che nonostante il software venga eseguito con un singolo *thread* è capace di gestire un numero elevato di richieste concorrenti. Differentemente, i tradizionali *server* generano un nuovo *thread* per ciascuna richiesta pervenuta che rimane attivo fin quando non termina l'invio della risposta, saturando più velocemente le risorse (memoria RAM, *Random Access Memory*) del sistema. In un modello *event driven* invece il *thread* rimane in esecuzione solamente per il

tempo strettamente necessario alle operazioni di I/O e in *idle* per il resto del tempo. Questo corrisponde a una percentuale molto alta dell'*uptime* di un server considerando le diverse grandezze delle velocità alle quali oggi siamo abituati: decine di *Mbit/s* per trasferimenti tramite internet e decine di *GByte/s* per la *bandwidth* delle memorie RAM.

Un altro vantaggio offerto da NodeJs è la possibilità di sviluppare codice *server side* con lo stesso linguaggio del codice *client side*. Come riportato nella sezione 2.4.3, JavaScript è infatti lo strumento più utilizzato e consigliato per gestire dinamicamente i contenuti di una pagina *web*.

La giovinezza del progetto e le sue potenzialità hanno portato NodeJs a essere utilizzato da diverse compagnie, alcune delle quali molto conosciute e di grande successo come NetFlix, PayPal e LinkedIn

Asp.net

Asp.Net è un insieme di *tool* sviluppato da Microsoft per lo sviluppo di applicazioni *web*. Senza dilungarci nelle sue funzionalità, analoghe a quelle di PHP, Spring e NodeJS, evidenziamo la possibilità offerta da Asp.Net di poter scrivere il codice in molti linguaggi diversi come Visual Basic, C Sharp (C#), Perl e Python. Asp.net è un linguaggio precompilato, quindi potenzialmetne più prestante di un linguaggio di *scripting*. Applicazioni alimentate da ASP.Net conosciute sono StackExcange, Microsoft.com e bbc.co.uk.

Altri

Sostanzialmente ogni linguaggio di programmazione mette a disposizione delle librerie per la comunicazione tramite *socket* internet e quindi può essere utilizzato per lo sviluppo di *web application*. Ne sono esempio di successo C++ sfruttato da una buona parte dei servizi di Google e Python, con cui sono stati sviluppati Youtube e Dropbox.

2.4.2 Database

Il *database* e lo strumento software sfruttato per la memorizzazione delle informazioni create e gestite dalla *web application*. I DBMS si dividono in due macrocategorie:

- **Relazionali**: sono chiamati anche RDBMS o SQL (il linguaggio su cui si basano), sono caratterizzati da una struttura a tabelle dove ciascuna tabella rappresenta un'entità o una relazione fra più entità. Ogni tabella è definita da diverse colonne che rappresentano gli attributi dell'entità. Le righe, chiamate *record*, rappresentano i vari oggetti memorizzati. L'accesso ai dati avviene attraverso ricerca nelle righe delle tabelle e secondo le relazioni stabilite tra di esse. Queste sono generalmente rappresentate da un grafico chiamato schema *entity-relation* (ER) che mostra visivamente la struttura della base di dati. Modelli complessi, con molte tabelle e relazioni, possono facilmente portare a un rallentamento delle *performance* di un RDBMS, principale ragione per cui vengono abbandonati in favore di sistemi non relazionali. I *database* relazionali risultano infatti più difficilmente scalabili e partizionabili, ovvero adottati per servizi via via più diffusi e distribuiti. Alcuni dei più conosciuti progetti sono MySQL, PostgreSQL e SQLite.

- **Non relazionali**: o NoSQL, sviluppano un modello per la memorizzazione e il recupero delle informazioni in maniera diversa da quello tabulare. I diversi approcci pensati vogliono realizzare un sistema più semplice e più scalabile, prevedono strutture dati di tipo dizionario, grafo o documento che consentono delle *performance* migliori rispetto ai DBMS di tipo SQL. In questi modelli viene sacrificata la consistenza dei dati (si potrebbero leggere dati non aggiornati) in favore della velocità, della scalabilità del sistema e della garanzia della disponibilità dei dati. Sono appositamente progettati per gestire moli di dati anche molto elevati rimanendo comunque efficienti e facili da usare. Lo svantaggio principale dell'uso di tali DBMS è la gestione a livello di *business logic* di qualsiasi vincolo fra i dati. Sono disponibili varie realizzazioni di *database* NoSQL anche *open source*, come il *database* MongoDB.

2.4.3 Client application

Con *client application* si intende il *software* destinato a essere usato dagli utenti per usufruire della piattaforma. Diversamente da quanto visto per l'applicazione *server*, qui la scelta di strumenti idonei è meno ampia anche se completa.

Per l'applicazione *client web*, accessibile tramite *browser*, si può ricorrere all'utilizzo della tecnologia HTML5 insieme al linguaggio di *scripting* JavaScript. La prima rappresenta il linguaggio di *markup* decisamente più utilizzato per la presentazione di contenuti sul *world wide web*, quinta versione dello storico HTML, *Hyper Text Markup Language*, di cui estende le funzionalità per far fronte alle necessità dello lo sviluppo delle pagine *web* moderne. Queste sfruttano largamente contenuti multimediali di diversa natura e complessità e possono prevedere un'altissima dinamicità e interattività degli elementi. In questo contesto rappresenta di fatto l'unica soluzione adatta. JavaScript è il linguaggio *client side* più diffuso e completo per la manipolazione degli elementi grafici e dei contenuti di una interfaccia *web*. Il suo ecosistema di sviluppatori e di librerie che ne estendono e semplificano l'uso nei diversi *browser* attuali lo fanno strumento più indicato per lo scopo del progetto. Altre tecnologie usate in passato come applicazioni Flash Player e Applet Java non risultano altrettanto indicate anche perché dichiarate deprecate o obsolete nei nuovi standard di HTML5.

Per quanto riguarda l'applicazione *client* che deve essere eseguita sui dispositivi *mobile* esistono due soluzioni percorribili, di seguito descritte.

App in linguaggio nativo e API

La prima, più utilizzata, è quella di sviluppare un'applicazione per dispositivi mobili scritta in linguaggio nativo per lo specifico sistema operativo del dispositivo: Android, IOS, Windows Phone o altri. Questo porta a creare una nuova applicazione da zero che riproduce ciò che l'interfaccia *web* realizza sulla pagina del *browser*. Per la comunicazione con il *web server* si realizza una API che descrive quei metodi e procedure che l'applicazione può utilizzare, o deve seguire, per inviare e ricevere dati dal *server* remoto. Solitamente tali API vengono realizzate ispirandosi all'architettura REST, (*REpresentational State Transfer*)e ai principi HATEOAS, (*Hypermedia As The Engine Of Application State*).

L'architettura REST considera tutti i contenuti accessibili come *web resource* tutte identificate tramite apposito URI (*Uniform Resource Identifier*). Attraverso questi *link* è possibile accedere alle risorse che possono essere rappresentate in formati JSON (*JavaScript Object Notation*), XML (*eXtensible Markup Language*) o HTML. La comunicazione fra *client* e *server* in una REST API avviene in maniera *stateless*: il *server* non mantiene alcuna informazione di sessione sulle precedenti richieste pervenute da un *client*. Queste informazioni servono, solitamente, a tenere traccia delle operazioni svolte da un utente nelle successive visite a pagine web. È comune ad esempio memorizzare le credenziali di un'utente autenticato. In una REST API se è necessario mantenere qualche tipo di dato sullo "stato" di avanzamento di un insieme di operazioni questo deve essere contenuto all'interno della richiesta stessa. In questo caso, ogni richiesta dovrebbe contenere le informazioni per autenticare un utente, non avendo il *server* modo di riconoscerlo autonomamente.

Sebbene una REST API non sia vincolata all'uso di un particolare protocollo, spesso si sfrutta il protocollo HTTP perché mette a disposizione degli standard, come il formato delle richieste e risposte, i metodi (chiamati "verbi" *POST*, *GET*, *DELETE*, *HEAD*, *PUT*) o schemi di autenticazione, già consolidati.

HATEOAS è un *design pattern* delle REST API che impone al *client* di interagire con il *server* attraverso gli URI, indirizzi che quest'ultimo genera dinamicamente e comunica all'interno delle risposte che invia. In altre parole il *client* non è tenuto a conoscere anticipatamente la struttura della API ma la scopre, a partire dal primo *entry point*, durante la navigazione. Per esempio alla richiesta di informazioni riguardo un conto bancario il *server* potrebbe allegare il *link* per depositare denaro a chiunque e i *link* per inviare, trasferire denaro e chiudere il conto solo in caso in cui l'utente sia il proprietario.

App web based

Un'altra soluzione è quella di sfruttare le tecnologie *web* anche sui dispositivi mobili. Queste infatti consentono di realizzare pagine web appositamente studiate per tali *device*. I sistemi operativi *mobile* consentono di aprire e navigare le pagine *web* all'interno di un'app, per esempio con l'utilizzo della classe WebView di Android. La possibilità di visualizzare tali pagine come si fa con un comune *browser* permette di evitare la realizzazione di una intera nuova applicazione. L'aggiunta di funzionalità ulteriori tramite il linguaggio nativo e i *tool* che il sistema mette a disposizione consente la costruzione di un'app *mobile* completa.

Si elimina così anche la necessità di implementare una API di interfacciamento fra *client* e *server*, il lavoro è svolto essenzialmente da un *browser web*.

Un aspetto negativo di questa strategia è dover ricorrere sul dispositivo portatile a un *browser web* e un'interprete JavaScript, generalmente più pesanti e meno performanti di un applicativo appositamente realizzato.

2.5 Tecnologie utilizzate

Dopo aver esposto gli strumenti che possono essere utilizzati per lo sviluppo di questo tipo di progetto vedremo ora quali strumenti sono stati scelti per l'implementazione.

2.5.1 Web server Nginx

Nginx, pronunciato come *"engine x"* è un HTTP *web server* che può essere usato anche come *reverse proxy*, *mail server*, *load balancer* e server TCP/UDP generico. Nginx usa una architettura *event driven* per la gestione delle richieste che permette al *server* di reagire ai segnali *hardware* generati alle operazioni di *input* e *output* comportamento simile a NodeJs, visto nella sezione 2.4.1. Ciò lo rende estremamente efficiente nella gestione di grandi quantità di richieste concorrenti. Originariamente scritto da Igor Sysoev nel 2002 ha alimentato per anni alcuni siti russi caratterizzati da una grande mole di traffico ed è arrivato nel Febbraio 2017 a essere sfruttato dal 28.34% dei siti *web* "attivi". Esempi del suo impiego di successo sono Netflix, Wordpress.com e Fastmail.fm. Tra le sue caratteristiche spiccano una ottimizzata gestione delle richieste verso *file* statici e un supporto per applicazioni FastCGI, *Fast Common Gateway Interface*: un protocollo che vuole diminuire l'*overhead* generato dalle applicazioni interne che si interfacciano al *web server*, fra cui PHP. Nginx permette infatti di comunicare con tali applicazioni fungendo da *proxy* verso queste. Nello specifico si è accoppiato al *web server* l'interprete PHP-FPM, *PHP fast process manager*, un'alternativa FastCGI di PHP adatta anche a siti molto trafficati.

Una sua versione, attualmente la 1.11.10, viene rilasciata con licenza *2-clause BSD open source*.

Nginx è stato utilizzato come FastCGI *proxy* verso PHP-FPM mentre sono state utilizzate le sue specifiche funzionalità per la gestione delle richieste di *file* statici.

Nginx XSendfile

XSendfile è una funzionalità che permette all'applicazione CGI di restituire il controllo delle richieste al *web server*, dopo l'esecuzione di una procedura preliminare.

Come abbiamo visto, è possibile configurare un *web server* affinché passi, CGI *proxy*, a una applicazione interna (ad esempio PHP) il controllo di alcune richieste. A questo punto è compito dell'applicazione accogliere la richiesta e produrre l'intero *output* che il *web server* invia direttamente al mittente. XSendfile consente, attraverso la scrittura nella risposta di un apposito *header* HTTP, chiamato X-Accel-Redirect, di ripassare il controllo al *server web*.

Lo scenario comune è quello dell'invio di un *file* statico, ad esempio una traccia audio, che l'applicazione deve o meno autorizzare. In caso positivo il *software* CGI dovrebbe gestire l'invio del *file*, leggendone l'intero contenuto dal *filesystem* e scrivendolo in *output* in maniera del tutto analoga a ciò che farebbe il *web server* se non si necessitasse dell'autorizzazione. Un *web server*, come nel caso di Nginx, implementa però delle procedure ottimizzate per l'invio dei *file* ed è sviluppato per gestire in maniera completa questo tipo di richieste. Si pensi per esempio alle richieste di tipo *ByteRange*. Queste permettono al *client* di ottenere solamente la parte dei *file* ritenuta utile, indicando l'intervallo di *byte* che si vuole scaricare. È quello che avviene quando, ad

esempio, si effettua il *seek* della traccia audio. Non viene normalmente scaricato l'intero file, che potrebbe essere molto pesante, ma solamente una porzione successiva al punto di *seek*. L'implementazione tramite applicazione CGI, così come è stata esposta, non permetterebbe questo tipo di operazioni. Ogni tentativo di *seek* comporterebbe una nuova richiesta che prevederebbe come risposta l'invio dell'intero *file*. La gestione efficace ed efficiente della richiesta tramite la *web application* costringerebbe a reinventare funzionalità già esistenti.

Un'altra soluzione prevederebbe di proteggere il *link* alla risorsa che necessita di autorizzazione. Tale *link* verrebbe fornito solo ad autenticazione avvenuta ma a questo punto potrebbe essere usato da chiunque ne venisse in possesso.

Grazie alla funzionalità XSendfile la gestione della richiesta torna al *web server*, Nginx, che la effettua in maniera corretta.

Questa soluzione è stata adottata per l'invio di tutti i *file* allegati ai contenuti caricati dagli utenti della piattaforma.

2.5.2 Yii framework

Yii (*"Yes, It Is"*) è un *framework object oriented*, *component based* e *open source* scritto in linguaggio PHP, studiato per agevolare e velocizzare lo sviluppo di *web application*. Yii è un progetto nato nel 2010, ma la versione Yii2, rilasciata nel 2013, consiste in una totale riscrittura e revisione del codice. La versione più recente è la 2.0.11, Febbraio 2017. È supportato da una larga comunità di sviluppatori e utilizzatori, fornisce un insieme di componenti per lo sviluppo *full-stack*, adotta ovunque il *design pattern* MVC e mantiene come obiettivo primario l'alta prestazionalità. Dispone di moduli per l'implementazione agevolata di sistemi di autenticazione e autorizzazione *role based* oltre a *web service* REST. Altri moduli permettono al *framework* di lavorare con diversi *database* sia relazionali sia non relazionali. A supporto dello sviluppatore vengono fornite sia una guida che una API *reference* entrambe complete e di facile comprensione.

2.5.3 Model View Controller workflow

Yii segue il *design* architetturale chiamato *Model View Controller* (MVC) che separa il *software* in tre parti concettualmente differenti:

- **Model**: o modello, è composto dal dominio degli oggetti che popolano la realtà modellata dall'applicazione. Riguarda i dati, le regole che questi devono rispettare e le procedure per trattarli, indipendenti dall'interfaccia utente. È il componente principale del *pattern*.
- **View**: o vista, descrive la rappresentazione, l'output, con cui le informazioni contenute nei *model* vengono mostrate all'utente. Un *model* può essere rappresentato in diversi modi da diverse *view*, per esempio un grafico o un tabulato per rappresentare delle transazioni.
- **Controller**: o controllore, è il componente che lega *model* e *view*. Riceve l'*input*, lo elabora, e se accettato lo traduce in comandi per i *model*, che mutano il loro stato, e restituisce la *view* opportuna per rappresentare i dati. Un *controller* espone diverse azioni, *entry-point*, ovvero metodi dell'applicazione che possono essere invocati dall'esterno.

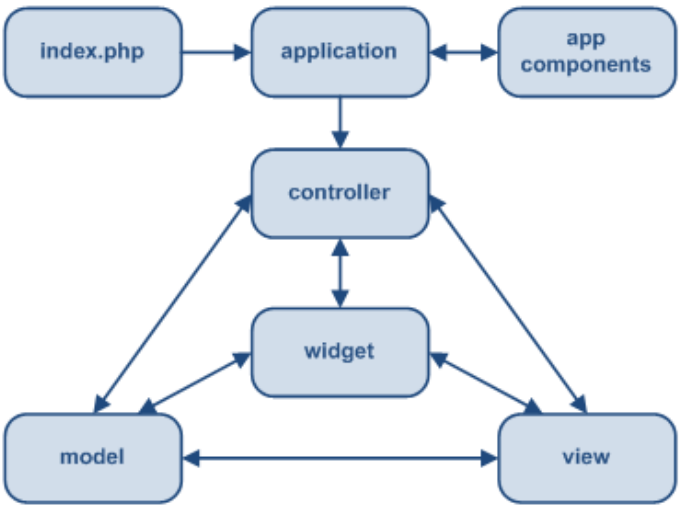


Figura 2.1: *Architettura* Model View Controller *in Yii*

Yii, oltre a implementare il *pattern* MVC, introduce un *front-controller*, chiamato *Application* il quale incapsula il contesto di esecuzione per il processo di una richiesta. *Application* raccoglie alcune informazioni sulla richiesta dell'utente e poi le smista al *controller* appropriato, figura 2.1.

Il *workflow* tipico di un'istanza di Yii è schematizzato dalla figura 2.2. Immaginiamo che una applicazione *client* cerchi di visualizzare le informazioni, ad esempio testo, autore, data di caricamento, contenute nel commento numero 123 del sito "esempio.com". Nel caso proposto la gestione di tale richiesta avviene nei seguenti passi.

1. L'applicazione richiede la risorsa all'indirizzo:

   ```
   http://esempio.com/index.php?r=commento/mostra&id=123
   ```

2. Viene creata un'istanza di *Application*.

3. *Application* riceve in ingresso le informazioni contenute nella richiesta sotto forma di un oggetto della classe *Request*.

4. Vengono determinati il *controller* e l'azione interessate dalla richiesta grazie a un modulo chiamato *urlManager*, nell'esempio il *controller* dei commenti e l'azione "mostra".

5. Un'istanza del *controller* viene creata e vengono applicati dei filtri, come il controllo dell'autenticazione, *rate-limiting* o *benchmark*. associati con questa azione. Se i filtri sono superati con successo l'azione "mostra" viene lanciata.

6. L'azione interroga il *database* tramite il *model* "commento" alla ricerca del record con identificatore "1".

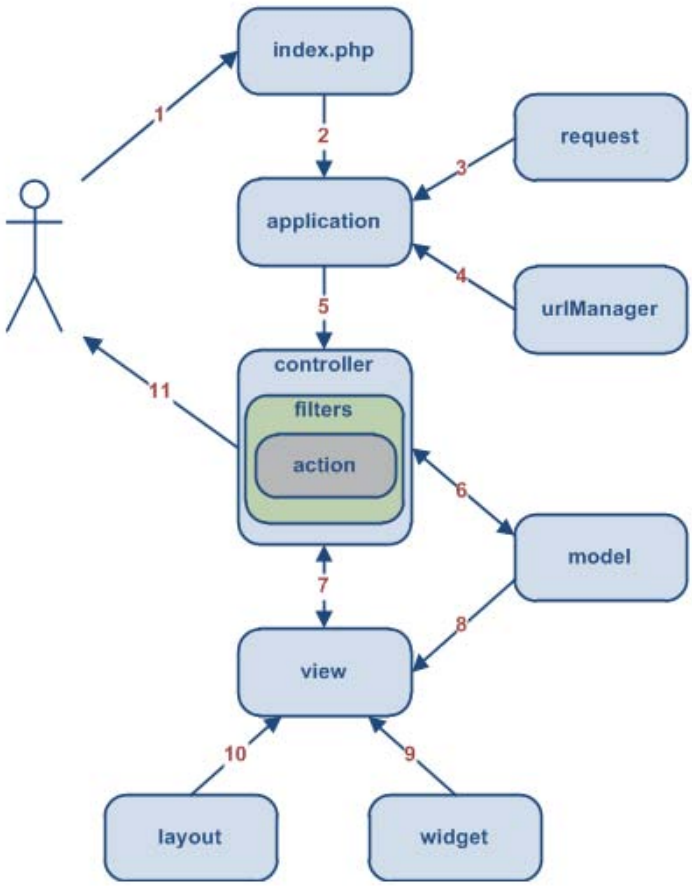


Figura 2.2: Workflow *di Yii*

7. I dati recuperati vengono renderizzati in una *view* che sarà restituita come *output* dell'azione.

8. *Application* restituisce la *view* prodotta all'utente.

Il *framework* Yii è stato utilizzato per la realizzazione della gran parte della *server application*.

2.5.4 Mysql

MySQL è un RDBMS *open source* distribuito con licenza GNU con alcune modifiche proprietarie. Viene sviluppato dalla Oracle Corporation. MySQL è componente fondamentale delle applicazioni basate sul *free stack* LAMP, acronimo di Linux Apache MySQL PHP/Perl/Python che indica rispettivamente il sistema operativo, il *web server*, il DBMS e il linguaggio applicativo impiegati. Come abbiamo visto, è stata impiegata nel progetto una variante di questo *stack* denominata LEMP, o LNMP, che usa il *web server* Nginx anziché Apache.

Invece di MySQL si può utilizzare, in maniera totalmente compatibile, il DBMS MariaDB: un *fork* del progetto di MySQL che vuole rimanere completamente libero sotto licenza GNU GPL.

2.5.5 Android

Android è un sistema operativo pensato principalmente per dispositivi *mobile* attualmente sviluppato da Google Inc. Può essere eseguito su molti *device* come *smartphone*, *tablet*, *tv*, computer di bordo e dispositivi indossabili (*smartwatch*).

Si tratta di un progetto quasi totalmente libero e *open-source* rilasciato con licenza *Apache 2.0*. Alcune componenti *software* hanno invece licenza proprietaria, come i *driver* per i dispositivi inclusi dai produttori *hardware*.

La società Android Inc nasce nel 2003, inizialmente realizzava solamente *software* applicativo per dispositivi *mobile*. Nel 2005 venne acquisita da Google, interessata a un ingresso nel mondo della telefonia mobile. Il team di sviluppo cominciò quindi a progettare un sistema operativo basato sul consolidato *kernel* Linux, pubblicando la prima versione ufficiale del sistema nel 2007. Il primo dispositivo equipaggiato con il "robottino verde" [3] fu rilasciato nel 2008, era l'HTC Dream [30].

Dal 2008 il sistema è in continuo sviluppo. Ogni *release* è caratterizzata da una precisa successione di numeri e di nomi, quelli di dolciumi, in ordine alfabetico. L'ultima versione, rilasciata nel 2017, è la 7.1.2 Nougat.

Android adotta per i *file* sorgente del sistema una politica di tipo *open source*; la licenza *Apache 2.0* consente di modificare e redistribuire liberamente il codice sorgente [11]. Una vasta comunità di sviluppatori realizzano applicazioni per il sistema incrementando così le funzionalità offerte dai dispositivi e migliorando la qualità di quelle già esistenti. Il negozio di *apps*

[3] Il logo del sistema Android è il tradizionale robot (androide) di colore verde

Android più usato, il *Google Play Store*, conta oggi, Marzo 2017, quasi 3 milioni di applicazioni disponibili [4]. La progettazione del sistema operativo è guidata da Google.

Android si basa sul *kernel* Linux 3.x (dalle versioni 4.0 in poi), con *middleware* e API scritte in linguaggio C o C++. La piattaforma *hardware* principale è l'architettura ARM.

Il sistema operativo utilizza una *Dalvik Virtual Machine*, versione più leggera e con ridotte funzionalità della *Java Virtual Machine*, che esegue il codice *bytecode* delle applicazioni. Questo viene prodotto attraverso una compilazione *just in time* del *Dalvik dex-code (Dalvik Executable)*. Il software sviluppato per Android viene quindi generalmente scritto in linguaggio Java, si esegue una prima compilazione e i *file* prodotti, fra cui il *dex-code*, vengono posizionati all'interno di un *file .apk*, simile a un *file zip*, che può essere trasferito sul dispositivo e che consente l'installazione dell'applicazione.

Sebbene Android si basi su kernel Linux, le modifiche apportate da Google a quest'ultimo rendono difficile il *porting* di applicazioni Linux o librerie, scritte in C o C++ su Android. È comunque possibile scrivere parti di software in linguaggio C sfruttando l'NDK, *Native Development Kit* e il *framework* JNI, *Java Native Interface*.

Android SDK

Android SDK (*Software Development Kit*) è l'insieme degli strumenti e API che Android mette a disposizione degli sviluppatori nelle sue versioni. Esso include librerie per la compilazione, il *debug*, il *package management* e l'emulazione di applicazioni. Comprende documentazione, codice di esempio e i *tutorial*. Fino al 2014 supportava l'IDE, *Integrated Development Enviroment*, Eclipse ma a partire dal 2015, abbandonato Eclipse, l'IDE di riferimento è divenuto Android Studio, una versione appositamente modificata della piattaforma IntelliJ.

L'SDK è stato utilizzato per lo sviluppo dell'applicazione client Android.

2.6 Strumenti utilizzati

2.6.1 Google Awareness API

Google Awareness è una API Java sviluppata per il sistema operativo Android che permette al dispositivo di conoscere la situazione attuale dell'utente. Si tratta di un sistema unificato per la gestione di diversi input, chiamati segnali, rilevabili da un dispositivo portatile, in particolare:

- localizzazione tramite GPS;
- luogo geografico, indirizzo;
- condizione meteorologica e temperatura, recuperate attraverso l'informazione posizionale;
- attività svolta (es: "a piedi", "in bici", "in auto",...);

[4] http://www.appbrain.com/stats/number-of-android-apps

- *beacons*, piccoli dispositivi che tramite tecnologia bluetooth inviano informazioni al dispositivo in un ristretto raggio di azione;
- stato degli auricolari;
- orario.

L'API non porta con se la rilevazione di un nuovo tipo di dato bensì uno strumento accentrato per recuperare tali informazioni. Ciò comporta:

- **Facilità implementativa**: si evita di ricorrere a molteplici librerie per recuperare i dati e si semplifica l'integrazione con diversi dispositivi.
- **Miglior qualità dei dati**: I segnali grezzi vengono elaborati per migliorarne la qualità e la significatività. Per esempio vengono utilizzati sofisticati algoritmi per capire l'attività svolta dall'utente con un alto livello di accuratezza a partire dai dati di accelerometro, GPS, giroscopio, ecc.
- **Ottimizzazione**: L'API si preoccupa di gestire efficacemente il consumo di energia e di banda dati.

Implemeta due strategie di accesso ai dati. La prima, chiamata *fence* (recinto), consente di registrarsi in attesa che il dispositivo entri in un particolare contesto. Al quel punto l'applicazione viene notificata e può reagire opportunamente. Ad esempio è possibile consigliare all'utente di spegnere la musica quando viene rilevato che sta guidando con gli auricolari collegati. La seconda chiamata *snapshot* ("istantanea") permette di conoscere il contesto attuale dell'utente in maniera puntuale. Questo significa che l'API restituisce entro un tempo massimo le informazioni che sono state recuperate sui segnali interessanti. È possibile quindi che si ottenga informazione incompleta in questa modalità, perché non si è avuto il tempo necessario a rilevare un contesto valido.

2.6.2 JQuery

JQuery *"Write less, do more"* è una libreria JavaScript che nasce con l'intento di facilitare la scrittura di codice per *task* comuni nelle interfacce *web*. JQuery implementa singoli metodi per l'esecuzione di operazioni che, utilizzando JavaScript *"out of the box"*, richiederebbero decine di righe di codice. JQuery consente inoltre una manipolazione facilitata e intuitiva degli elementi, grafici e non, della struttura ad albero di un documento HTML, chiamata DOM, *Document Object Model*. La gestione di tali elementi in maniera uniforme sui vari *browser web* risulta infatti difficoltosa, e ancora di più lo era in passato quando i vari produttori di questo tipo di applicazioni realizzavano, modificavano, gli interpreti JavaScript secondo le loro necessità. Questo portava alla aggiunta di *feature* o la rimozione di altre che costringevano a riscrivere il codice JavaScript per la stessa funzione in maniera diversa a seconda del *browser* dell'utente. JQuery si fa carico di unificare in unica libreria tali diversità rendendola una libreria multipiattaforma, o multi *browser*.

Altre caratteristiche di JQuery sono la gestione dello stile tramite manipolazione del CSS, *Cascading Style Sheet*, l'implementazione di effetti e animazioni, procedure AJAX, *Asynchronous JavaScript And XML*, e varie *utilities*.

I principi a cui JQuery si ispira sono:

- Separazione fra la struttura del documento e il codice che ne gestisce le funzionalità. JQuery utilizza JavaScript per aggiungere dei gestori di eventi agli elementi del DOM, approccio diverso dall'aggiunta di funzioni JavaScript negli attributi degli elementi stessi. In questo modo lo sviluppatore è incoraggiato a mantenere completamente separato il codice JavaScript dal *markup* HTML.

- Vengono promosse la chiarezza e brevità del codice attraverso la documentazione e le sue linee guida nonché con le funzioni di JQuery stesse.

- Eliminazione delle incompatibilità fra i vari *browser*.

- Estensibilità attraverso la possibilità di aggiungere nuovi eventi, nuovi oggetti e *plugin* alla libreria.

La libreria gode di una vasta comunità di sviluppatori e utilizzatori, alcuni dei quali sono compagnie molto conosciute come Google, Microsoft e Netflix.

JQuery è stato largamente utilizzato nelle pagine *web* dell'interfaccia grafica dell'applicazione per la visualizzazione degli elementi e per l'esecuzione asincrona di richieste HTTP, come il *download* dei contenuti di un dossier.

2.6.3 Twitter Bootstrap

Bootstrap è un *framework* gratuito e *open source* per lo sviluppo di interfacce grafiche di siti e applicazioni *web*. Si occupa solamente e specificamente dello sviluppo di componenti e *design pattern* per lo sviluppo *front-end*, ovvero il gli elementi grafici e funzionali con cui l'utente interagisce per usufruire di un servizio internet.

Originariamente chiamato Twitter Blueprint, fu ideato per il noto *social network* Twitter come strumento per uniformare le interfacce all'interno della piattaforma. Dopo alcuni mesi gli sviluppatori si resero conto che stavano creando qualcosa che aveva un potenziale maggiore di uno strumento interno all'azienda. Rinominato il progetto in Bootstrap ed esteso il team di lavoro il progetto venne rilasciato come *open source* nell'Agosto del 2011. Oggi ha raggiunto la sua terza versione, Bootstrap 3, e una sua nuova versione Bootstrap 4 è in stato di sviluppo *alpha*. È il secondo progetto più "stellato" della famosa piattaforma per condivisione e controllo di versione di progetti chiamata Github [5]. Il primo posto è occupato, marzo 2017, dal progetto freeCodeCamp, una piattaforma per la promozione di conoscenze di programmazione.

Bootstrap consiste in un insieme di pacchetti completamente personalizzabile di componenti HTML, CSS ed estensioni JavaScript che contengono *design pattern* per diversi componenti

[5] sito internet: *github.com*

delle interfacce, come stile tipografici personalizzati, pulsanti, moduli di *input*, elementi per la navigazione e molti altri.

La versione 3 rilasciata nell'Aprile 2013 è contraddistinta da una ricerca di un *design flat* e un approccio che mette in primo piano la resa su dispositivi *mobile*. Per questo, come JQuery, Bootstrap 3 è stato utilizzato per lo sviluppo dell'interfaccia grafica.

2.6.4 FFmpeg

FFmpeg è un potente e completo *framework* per la gestione di dati multimediali. È in grado di effettuare la decodifica e la codifica, la transcodifica, il *muxing* e *demuxing*, lo *streaming*, il filtraggio e la riproduzione dei dati. Sostanzialmente tutto le operazioni ideate nell'ambito del trattamento di documenti audio, video e immagini. Supporta un'enorme varietà di formati diversi, dai più antichi fino ai formati in sperimentazione. È sviluppato in maniera da essere estremamente portabile: può essere compilato su macchine Linux, Mac OS X, Microsoft Windows e Solaris in molte delle loro distribuzioni.

Il progetto FFmpeg vuole fornire la migliore soluzione tecnica per gli svluppatori di applicazioni multimediali e gli utenti finali. Per questo combina il miglior software gratuito disponibile e codice indipendentemente sviluppato per ridurre le dipendenze e migliorare l'uniformità del codice all'interno del progetto.

FFmpeg consiste in 4 applicazioni *standalone* e 7 librerie destinate a essere integrate dagli sviluppatori all'interno del loro *software*.

Le 4 applicazioni sono:

- **FFmpeg**: un performante convertitore audio e video in grado di lavorare sia con *file* che con dati in *stream*. Dall'esterno FFmpeg appare come un *tool* da riga di comando che riceve in *input* un numero arbitrario di flussi di dati, *file*, *pipe*, *network stream* o altro, e, dato un insieme di impostazioni di configurazione produce un arbitrario numero di flussi in *output*.

  L'opzione *-map*, mappatura, di FFmpeg consente di gestire il *muxing* e *demuxing* dei flussi di dati contenuti nei "*file*" di *input* e *output*. Ognuno dei *file* in ingresso può infatti contenere qualsiasi numero di *stream* di dati (audio, video, sottotitoli) previsti dal formato contenitore, ad esempio *mp4*, *mkv*, *mp3* e *3gp*, che possono essere ricopiati, transcodificati, eliminati, fusi nei flussi di *output*. Si può immaginare ad esempio di ottenere da un *file video*, contenente due tracce audio, italiano e inglese, entrambe stereo e una traccia per i rispettivi sottotitoli, un nuovo *file* video con il solo audio in italiano ricodificato a singolo canale.

  Tramite una vasta scelta di opzioni FFmpeg consente di effettuare le principali operazione di manipolazioni dati multimediali.

- **ffserver**: una applicazione server per lo *streaming* di contenuti audio/video. Supporta molteplici trasmissioni in diretta, lo *streaming* da *file* e la scostamento all'indietro del tempo riproduzione delle trasmissioni *live*. L'applicazione riceve quindi in *input file* preregistrati o *flussi* da istanze di FFmpeg e le trasmette attraversi i protocolli RTP, RTSP o HTTP.

- **ffplay**: un semplice e portabile riproduttore di contenuti multimediali usato principalmente a scopo di test.
- **ffprobe**: un *tool* per l'estrapolazione delle informazioni, metadati, riguardo flussi di dati multimediali e la loro scrittura in formati leggibili da una persona fisica. Permette la selezione delle informazioni necessarie, degli *stream* e la stampa in diversi formati, ad esempio JSON e XML.

Le librerie messe a disposizione degli sviluppatori sono:

- **libavutil**: contiene funzioni per facilitare lo sviluppo: come strutture dati, *routine* matematiche e generazione casuale di numeri;
- **libavcodec**: include l'insieme dei codificatori e decodificatori per i *codec* audio/video disponibili;
- **libavformat**: permette il *muxing* e *demuxing* dei vari formati contenitori.
- **libavdevice**: contiene i dispositivi di acquisizione e *rendering* da e verso comuni *framework* multimediali: per esempio Video4Linux e ALSA;
- **libavfilter**: implementa filtri per i dati multimediali;
- **libswscale**: consente la computazione altamente ottimizzata della scalatura di immagini e la conversione del loro spazio di colori;
- **libswresample**: realizza il ricampionamento di dati audio, ovvero la modifica della frequenza di campionamento e la conversione del formato dei campioni.

Data la sua completezza, le sue prestazioni e e la sua solidità il progetto FFmpeg è stato sfruttato per diversi scopi nell'ambito dello sviluppo del progetto. Ha permesso la lettura dei metadati dei contenuti multimediali caricati, la generazione della forma d'onda delle tracce audio, l'estrapolazione dei *frame* dai video per la creazione delle loro miniature e, compilato per Android, il *processing* dei *file* audio, video e immagini ritenuti eccessivamente pesanti ai fini della trasmissione e della riproduzione, per qualità troppo alta o codifica debole (*compression factor* ridotto).

2.6.5 Peaks.js

Peaks.js è un progetto sviluppato dalla BBC Research & Development che comprende quattro *tool* per la generazione, la fornitura e la visualizzazione su interfacce *web* della forma d'onda di tracce audio, *waveform*.

Dei quattro componenti *software*, quello centrale è Peaks.js, omonimo dell'intero progetto, una libreria JavaScript che permette la visualizzazione di un'onda sonora. Legandosi a un *player* audio HTML5, può eseguire la riproduzione e mostrare a diversi livelli di ingrandimento la posizioni corrente nella traccia. Il componente è completamente *client side* e richiede dal *server*

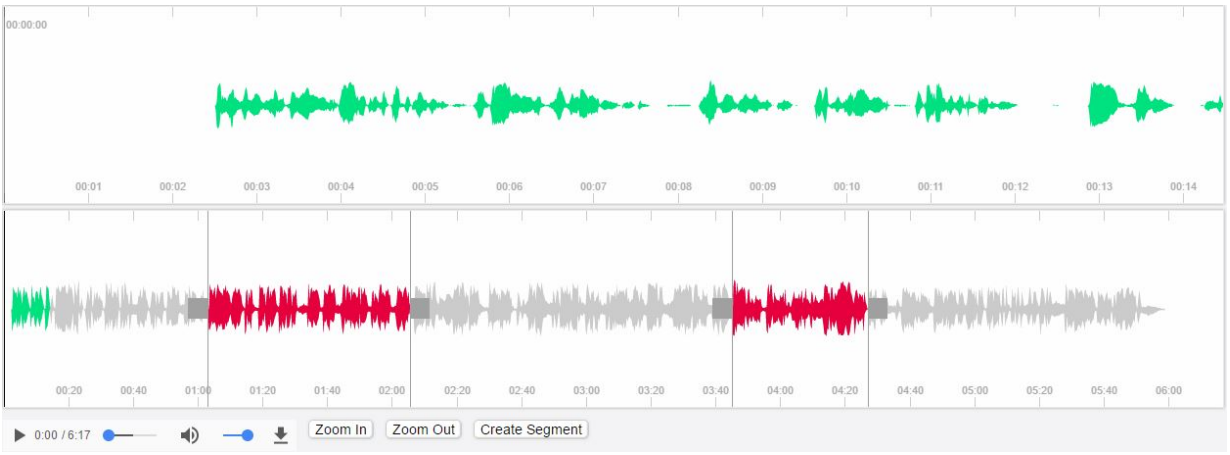


Figura 2.3: *Interfaccia di Peaks.js*

solamente i campioni della *waveform* precalcolata. Per questa si possono utilizzare due formati: binario, più compatto, e JSON, maggiormente compatibile con i diversi browser. Il formato binario necessita del supporto dei *typed arrays* da parte dell'interprete JavaScript.

Peaks.js, figura 2.3, permette inoltre di specificare e visualizzare, all'interno della forma d'onda all'onda, dei segmenti temporali e dei punti, eventi spot evidenziati con una linea a un determinato istante di riproduzione. Entrambi questi tipi di *marker* possono essere modificati, cioè l'utente muoverli tramite il mouse all'istante d'interesse. Questa caratteristica non è ancora stata sfruttata all'interno di MOLLY ma potrebbe esserlo in futuro.

Altra funzionalità della libreria è la capacità usare la Web Audio API per generare la *waveform* lato *client* direttamente dallo *stream* audio. Le tracce audio potrebbero però essere potenzialmente molto lunghe e causerebbero un ritardo ogni qualvolta se ne dovesse calcolare la forma d'onda. Per questo è stato scelto di generare la *waveform* lato *server*, una sola volta, nel momento in cui i contenuti audio vengono caricati.

2.6.6 Mustache

Mustache è un sistema di *template* di tipo *logic less*. Consiste in un piccolo *tool*, disponibile in un gran numero di linguaggi di programmazione, per il *rendering* di informazioni all'interno di *template*. I *template* Mustache sono semplici file testuali che contengono la struttura del documento in cui i dati andranno visualizzati e delle stringhe *tag*, segnaposto che indicano al *template system* il punto esatto in cui collocare le informazioni.

Mustache accetta due parametri: un *frontmatter* in formato YAML, simile a JSON, contenente tutti i dati degli oggetti logici da rappresentare, e un *template*, ovvero una stringa di testo contenente dei *tag*. Una volta effettuato il *parsing* del *template* i *tag* vengono espansi e al loro posto viene scritto il contenuto, identificato attraverso il nome del *tag* e l'attributo dell'oggetto.

La specifica *logic less* sta a indicare che i *template* non contengono al loro interno alcun tipo di struttura di controllo logica, *for, if, switch o statement*. Ci sono solo i *tag* che sono rimpiazzati da valori. Ciò permette a Mustache di essere estremamente leggero e performante. Un altro

sistema di *template* che espande Mustache introducendo strutture logiche è Handlebars.js, che supporta solamente JavaScript.

Il sistema non è vincolato a qualche tipo di documento specifico e può essere usato in qualunque ambito possa risultare utile: HTML, *file* di configurazione, ...

È stato usato per il *rendering* dei *template* per i contenuti di un *dossier*.

Capitolo 3

Implementazione

In questo capitolo verrà descritta l'implementazione del progetto MOLLY realizzata durante il lavoro a questa tesi. Si presenterà quindi la struttura e il funzionamento della *web application*, l'interfaccia *web* e l'app *mobile* Android.

3.1 Panoramica della piattaforma

Nello schema in figura 3.1 viene presentata l'architettura della piattaforma. L'utente interagisce con la componente *server* attraverso l'interfaccia *web* oppure tramite un'app *mobile* Android. Tutte le trasmissioni fra i *client* e il *server* sono effettuate attraverso il protocollo HTTP e sono gestite dal *web server* Nginx.

Nginx completa le richieste che riguardano contenuti statici come *asset* CSS, JavaScript o immagini. Se le richieste riguardano contenuti dinamici viene fornito all'interprete PHP-FPM uno *script*, contenente il codice da eseguire. Nel caso di un'applicazione Yii lo *script* è contenuto nel *file* "index.php", memorizzato nella cartella "web" della *directory* principale del progetto. Tutti gli altri *file* PHP dell'applicazione vengono caricati da questo *entry script* in maniera *lazy*: ovvero si usano solamente le classi e i moduli interessati dalla particolare esecuzione.

Viene quindi lanciata un'istanza di Yii che, acquisiti i parametri della richiesta, la smista al controllore e all'azione appropriati. L'applicazione comunica con un *database*, gestito dal RDMS MySql, dove sono memorizzati i dati dei diversi modelli, e con il *filesystem* per leggere e scrivere tutti i *file* necessari; ad esempio legge le classi PHP e scrive i contenuti caricati dagli utenti. Una volta elaborata la risposta alla richiesta pervenuta, questa viene inviata a Nginx che la trasmette al *client* tramite la rete. Il processo Nginx, l'interprete PHP-FPM, MySql e il *filesystem* di un *server* remoto formano la componente *server side* della piattaforma.

In figura 3.2 viene mostrato lo schema EER, *Enhanced Entity-Relationship*, del *database* modellato. La maggior parte delle tabelle mostrate rispecchia uno dei modelli descritti nella sezione 3.2 mentre le tabelle con suffisso "_map" realizzano relazioni *many-to-many* fra le entità collegate.

La discussione delle entità, dei loro attributi e vincoli viene rimandata alle sezioni 3.2 e 3.3

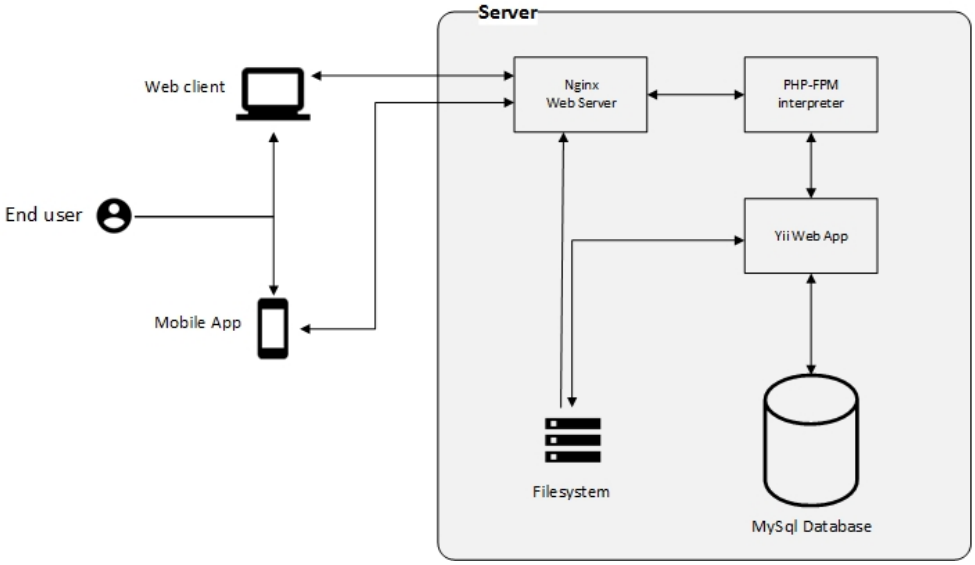


Figura 3.1: *Schema della piattaforma*

3.2 Modelli

Come descritto nella sezione 2.5.3, i modelli rappresentano la componente fondamentale del *pattern* MVC. Di seguito verranno discusse le classi dei modelli realizzati nel progetto.

Quelli che rispecchiano direttamente una delle entità del *database* estendono la classe Yii `ActiveRecord`. Un `ActiveRecord` si interfaccia con una tabella memorizzata nel *database* e implementa un insieme di operazioni comuni a tutti i modelli. Si tratta delle operazioni di inserimento, modifica e cancellazione di un *record* e di operazioni di validazione degli attributi di una particolare istanza del modello.Queste rappresentano una delle funzionalità più importanti di un `ActiveRecord`.

La validazione degli attributi di un modello avviene nel momento precedente al salvataggio nel *database*, ovvero prima di un operazione di inserimento o modifica.

È implementata dal metodo `ActiveRecord::validate()`, eseguito in maniera predefinita con l'invocazione dei metodi `ActiveRecord::save()` e `ActiveRecord::update()`. Le regole di validazione vengono definite attraverso il metodo `ActiveRecord::rules()` e consistono in un insieme di funzioni, o classi, che vengono eseguite sequenzialmente sugli attributi a loro associati.

Se una funzione di validazione fallisce, ovvero l'attributo *target* non rispetta le regole definite, il vettore `ActiveRecord::errors` viene aggiornato, inserendo un nuovo messaggio per l'attributo errato.

In Yii sono disponibili diverse classi di validatori *built in* per controlli comuni, come il formato di un indirizzo *email* o di una data. È possibile costruire validatori personalizzati semplicemente creando una apposita funzione booleana che valuti gli attributi e inserisca gli errori riscontrati.

Al termine del metodo `ActiveRecord::validate()`, se non sono stati rilevati errori, il modello viene effettivamente salvato nel *database*.

In una applicazione Yii i *file* delle classi dei modelli sono solitamente salvati nella *directory* "models" della cartella principale. Di seguito verranno illustrati i modelli realizzati, indicando i loro attributi, i metodi implementati e le regole di validazione specificate. Se non diversamente indicato gli attributi a

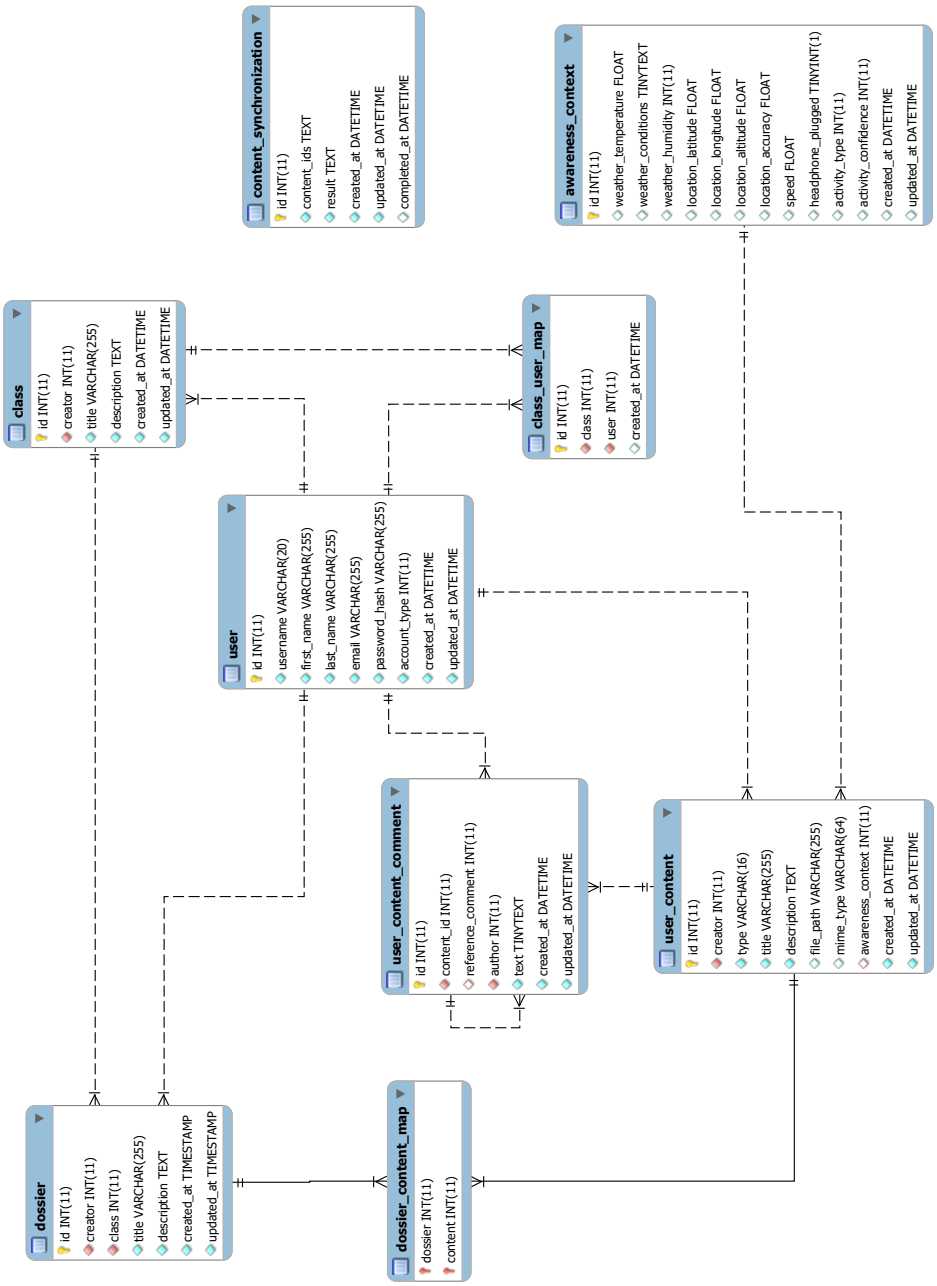


Figura 3.2: *Schema ER del* database

cui non è associata alcuna regola di validazione sono gestiti internamente, non dipendono cioè da valori inviati dal *client* degli utenti, e quindi non necessitano di controlli. Per quanto riguarda i metodi dei vari modelli, sono stati riportati solamente quelli ritenuti interessanti al fine di capire il funzionamento dell'applicazione. Metodi *standard* o comuni a tutti i modelli sono stati omessi: per esempio i metodi `save()`, `insert()`, `delete()` e `rules` (descritto sopra) non vengono mai riportati anche se sono presenti in ogni modello per gli stessi scopi.

3.2.1 User

Descrizione

La classe `User` rappresenta il modello di un utente. Estende `ActiveRecord` e si collega con la tabella `user` del *database*. Implementa inoltre l'interfaccia `IdentityInterface` di Yii pensata per racchiudere i metodi necessari per il tipico modello di un utente che ha facoltà di accedere all'applicazione. `IdentityInterface` permette di associare alla variabile statica `Yii::$app->user->identity`, accessibile globalmente, un'istanza della classe `User`. Questa rappresenta l'utente, se autenticato, all'interno della piattaforma.

Attributi e regole di validazione

- `id [PK]` : identificatore dell'utente.
- `username` : soprannome scelto dall'utente all'interno della piattaforma.
- `first_name` : nome dell'utente.
- `last_name` : cognome dell'utente.
- `email`: indirizzo di posta elettronica dell'utente.
- `password_hash` : *password* criptata dell'utente, generata da `generatePasswordHash()` metodo della classe `Security` di Yii. La criptazione della *password* protegge da attacchi *brute force* e impedisce anche nel caso peggiore (accesso maligno al *database*) di ricavare le *password* degli utenti.
- `account_type` : tipo di profilo dell'utente, non ancora utilizzato, previsto per uso futuro.
- `created_at` : data e orario di creazione del modello.
- `updated_at` : data e orario dell'ultimo aggiornamento del modello.

Nella tabella 3.1 vengono riportate le regole di validazione del modello `User`.

Metodi

- `validatePassword($password)` : verifica se l'attributo `password_hash` di questo modello corrisponde al parametro `$password`.
- `findByUsername($username)` : metodo statico che restituisce, se esiste, il modello corrispondente al nome utente `$username`.
- `findByEmail($email)` : metodo statico che restituisce, se esiste, il modello corrispondente all'indirizzo `$email`.

Attributi	**Nome regola**	**Descrizione**
first_name, last_name	match	*Match* dell'espressione regolare in formato PCRE `/^[a-z ,.\'-]+$/i`
username	match	*Match* dell'espressione regolare in formato PCRE `/^[a-z][a-z0-9]{3,20}$/i`
email	email	L'attributo deve avere il formato di un indirizzo *email* valido
email, username	unique	L'attributo deve essere univoco per questo modello

Tabella 3.1: *Regole di validazione per* `User`

Attributi	**Nome regola**	**Descrizione**
email, password	required	L'attributo deve essere specificato
rememberMe	boolean	L'attributo deve avere un valore booleano
password	validate password	La *password* specificata deve corrispondere alla *password* dell'utente individuato dall'attributo `email`

Tabella 3.2: *Regole di validazione per* `LoginForm`

3.2.2 LoginForm

La classe `LoginForm` realizza il modello corrispondente al modulo di accesso alla piattaforma. Non è collegata a nessuna tabella del *database*.

Attributi

- `email` : indirizzo *email* inserito dall'utente.
- `password` : *password* inserita dall'utente.
- `rememberMe` : indica se l'utente vuole che le sue credenziali siano memorizzate dal `client`, attraverso l'uso dei *cookie* HTTP, per i successivi accessi.

Nella tabella 3.2 vengono riportate le regole di validazione del modello `LoginForm`.

Metodi

- `getUser()` : restituisce, se esiste, l'utente associato all'attributo `email`.
- `login()` : esegue il *login* dell'utente tramite gli attributi impostati. Restituisce un valore booleano: *true* se il *login* ha successo, *false* altrimenti.

Attributi	**Nome regola**	**Descrizione**
title, description	required string	L'attributo deve essere una stringa
creator	exist	L'attributo deve essere un identificatore valido del modello `User`

Tabella 3.3: *Regole di validazione per* `ClassModel`

3.2.3 ClassModel

Descrizione

La classe `ClassModel` implementa il modello di una classe, o corso, creata da un insegnante all'interno della piattaforma. Estende `ActiveRecord` e si collega con la tabella `class` del *database*.

Al momento nella piattaforma è presente una sola classe, per scopo di test, a cui vengono associati tutti gli utenti creati.

Attributi

- `id [PK]` : identificatore del modello.
- `creator [FK]` : identificatore dell'utente creatore della classe, chiave esterna verso il modello `User`.
- `title` : titolo della classe fornito dal suo creatore.
- `description` : descrizione della classe fornita dal suo creatore.
- `created_at` : data e orario di creazione del modello.
- `updated_at` : data e orario dell'ultimo aggiornamento del modello.

Nella tabella 3.3 vengono riportate le regole di validazione del modello `ClassModel`.

3.2.4 Dossier

Descrizione

La classe `Dossier` implementa il modello di un *dossier* creato da un insegnante all'interno di una classe. Estende `ActiveRecord` e si collega con la tabella `dossier` del *database*.

Attributi

- `id [PK]` : identificatore del modello.
- `creator [FK]` : identificatore dell'utente creatore del *dossier*, chiave esterna verso il modello `User`.
- `title` : titolo del *dossier* fornito dal suo creatore.
- `description` : descrizione del *dossier* fornita dal suo creatore.
- `created_at` : data e orario di creazione del modello.

Attributi	Nome regola	Descrizione
title	required string	L'attributo deve essere una stringa di minimo 8 e massimo 40 caratteri alfanumerici
description	string	L'attributo, deve essere una stringa di minimo 10 e massimo 1000 caratteri alfanumerici

Tabella 3.4: *Regole di validazione per* `Dossier`

- `updated_at` : data e orario dell'ultimo aggiornamento del modello.

Nella tabella 3.4 vengono riportate le regole di validazione del modello `Dossier`.

3.2.5 UserContent

Descrizione

La classe `UserContent` rappresenta un contenuto caricato da un utente. Inizialmente era stato pensato per rappresentare un contenuto generico, ma finora è stato solamente utilizzato per i contenuti caricati all'interno di un *dossier*. Estende la classe `ActiveRecord` ed è collegata alla tabella `user_content` del *database*.

Inizialmente si era pensato di costruire una classe per ciascuna tipologia di contenuto possibile: testuale, immagine, video e registrazione audio. Ciascuna di queste avrebbe avuto una corrispondente tabella nel *database* e specifici attributi. Successivamente si è abbandonata questa idea in favore di un'unica classe, soluzione che appare più semplice e più chiara.

Un contenuto può avere un *file* allegato (audio, immagine o video) che identifica la tipologia del contenuto stesso. Se non viene allegato alcun file il contenuto è di tipo testuale.

Attributi

- `id [PK]` : identificatore del contenuto.
- `creator [FK]` : utente che ha caricato contenuto.
- `type` : tipologia del contenuto: può assumere i valori "video", "audio", "image" o "text".
- `title` : titolo del contenuto dato dal suo creatore.
- `description` : eventuale descrizione a corredo di un contenuto se questo è un'immagine, un video o un audio. Se il contenuto è di tipo testuale questo attributo rappresenta il corpo del contenuto.
- `file` : non rappresenta un attributo della tabella ma è un campo che prende il valore del *file* caricato in fase di creazione di un nuovo contenuto. In questo modo è possibile stabilire regole di validazione per il *file*.
- `file_path` : percorso dove viene memorizzato il *file* allegato questo contenuto (attributo vuoto in caso di contenuti testuali). Questo percorso serve anche per recuperare altri file, chiamati risorse, che dipendono dal *file* associato.

Attributi	**Nome regola**	**Descrizione**
title	required string	L'attributo deve essere una stringa di minimo 8 e massimo 40 caratteri alfanumerici
description	string	L'attributo, se specificato, deve essere una stringa di minimo 10 e massimo 1000 caratteri alfanumerici
description	required if text	Se non viene allegato alcun *file*, ovvero il contenuto è testuale, l'attributo deve essere specificato.
file	file	Se è stato allegato un *file* questo deve avere uno dei tipi MIME restituiti dal metodo `contentMimeTypes()`

Tabella 3.5: *Regole di validazione per* `UserContent`

- `mime_type` : tipo MIME, *Multipurpose Internet Mail Extensions*, associato con questo contenuto. Nel caso di contenuti testuali questo viene impostato a "text/plain".
- `awareness_context [FK]` : identificatore del modello `AwarenessContext` eventualmente associato a questo contenuto.
- `created_at` : data e orario di creazione del modello.
- `updated_at` : data e orario dell'ultimo aggiornamento del modello.

Nella tabella 3.5 vengono riportate le regole di validazione del modello `UserContent`.

Metodi

- `resources()` : restituisce, secondo il tipo del contenuto, i nomi delle risorse disponibili. Le risorse sono i *file* collegati a questo contenuto. Queste sono chiamate:
  - *source* : il *file*, audio, video o immagine, allegato al contenuto.
  - *waveform* : il *file*, in formato JSON, con i dati della *waveform* di un contenuto audio.
  - *poster* : l'immagine di copertina, o miniatura, di un contenuto *video*.
- `getResourceUri($resourceName)` : restituisce l'indirizzo all'interno del *filesystem* del *server* dove trovare la risorsa `$resourceName` associata a questo contenuto .
- `fields()` : restituisce i campi del modello quando questo viene convertito in un *array*, ad esempio quando viene inviato un modello in una risposta HTTP in formato JSON. Questo permette di aggiungere, modificare o eliminare delle informazioni del contenuto. Sono stati aggiunti i campi che contengono i *link* per accedere alle risorse del contenuto e il *link* per la sua eliminazione. Inoltre viene allegato al contenuto l'`AwarenessContext` eventualmente associato.
- `afterSave($insert, $changedAttributes)` : sovrascrive il metodo predefinito che viene eseguito in seguito al salvataggio di un `ActiveRecord`. In caso in cui si tratti di un nuovo contenuto viene lanciata l'esecuzione delle procedure di gestione del nuovo *file* caricato, si veda la sezione 3.3.7.

- `contentMimeTypes()` : metodo statico che ritorna una mappa dei tipi MIME accettati per i *file* allegati a un contenuto. Questi MIME sono raggruppati per il `type` corrispondente. Ad esempio la chiave "audio" della mappa contiene i tipi MIME "audio/wav" e "audio/mp3".
- `getFileType($path)` : metodo statico che individua il `type` del contenuto a partire dal percorso `$path`.

3.2.6 AwarenessContext

La classe `AwarenessContext` rispecchia tutte le informazioni sul contesto del dispositivo *mobile* che si possono rilevare tramite la Google Awareness API, sezione 2.6.1, eccetto i *beacon*. Estende la classe `ActiveRecord` ed è collegata alla tabella `awareness_context` del *database*.

Attributi

- `id [PK]`: identificatore del modello.
- `weather_temperature` : temperatura rilevata, viene memorizzata l'informazione in gradi celsius.
- `weather_conditions` : condizioni atmosferiche rilevate, memorizzate come vettore in formato JSON. Ogni elemento è un intero che corrisponde a una delle condizioni definite dall'API (soleggiato, nuvoloso, pioggia, ecc). Non è stato ritenuto necessario normalizzare questo attributo all'interno del *database* perché ogni volta che si accede al modello si recuperano tutte le condizioni, che nel loro insieme descrivono il meteo rilevato.
- `weather_humidity` : umidità atmosferica, rilevata come valore percentuale.
- `location_latitude` : latitudine della posizione, rilevata in gradi sessagesimali.
- `location_longitude` : longitudine della posizione, rilevata in gradi sessagesimali.
- `location_altitude` : altitudine, in metri sopra il livello del mare.
- `location_accuracy` : raggio in metri della circonferenza, centrata nella posizione rilevata, entro la quale cade la posizione reale.
- `speed` : velocità rilevata, in metri al secondo.
- `headphone_plugged` : stato degli auricolari: indica se questi sono collegati o meno.
- `activity_type` : intero che identifica l'attività svolta dall'utente, per esempio "a piedi", "in bici" o "fermo", così come definito dalla API.
- `activity_confidence` : accuratezza, indicata con un valore percentuale, della rilevazione di `activity_type`.
- `created_at` : data e orario di creazione del modello.
- `updated_at` : data e orario dell'ultimo aggiornamento del modello.

Nella figura 3.6 sono riportate le regole di validazione del modello `AwarenessContext`.

Attributi	**Nome regola**	**Descrizione**
weather_temperature, weather_humidity, location_latitude, location_longitude, location_altitude, location_accuracy, speed, headphone_plugged, activity_type, activity_confidence	number	L'attributo, se presente, deve essere un numero, intero o reale in conformità con la grandezza che rappresenta e nei limiti di essa
weather_conditions	integer array	L'attributo, se presente, deve essere un vettore di valori interi
headphone_plugged	boolean	L'attributo, se presente, deve avere valore booleano

Tabella 3.6: *Regole di validazione per* `AwarenessContext`

3.2.7 ContentComment

Descrizione

La classe `ContentComment` realizza il modello per un commento, scritto da un utente, riguardante un `UserContent`. Estende `ActiveRecord` e si collega con la tabella `user_content_comment` del *database*.

Attributi

- `id [PK]` : identificatore del commento.
- `content_id [FK]` : identificatore del contenuto a cui questo commento è associato. Chiave esterna verso il modello `UserContent`.
- `reference_comment [FK]` : `id` di un altro commento allo stesso contenuto di cui questo è una risposta. È chiave esterna verso il modello `ContentComment`. Non ancora utilizzato, pensato per un futuro utilizzo.
- `author [FK]` : identificatore dell'utente autore del commento, chiave esterna verso il modello `User`.
- `text` : stringa corpo del commento.
- `created_at` : data e orario di creazione del modello.
- `updated_at` : data e orario dell'ultimo aggiornamento del modello.

Nella tabella 3.7 sono riportate le regole di validazione per il modello `ContentComment`.

Attributi	Nome regola	Descrizione
text	string	L'attributo deve essere una stringa di minimo 3 caratteri alfanumerici e massimo 255
content_id	exist	L'attributo deve corrispondere ad un `id` esistente per il modello `UserContent`

Tabella 3.7: *Regole di validazione per* `ContentComment`

3.3 Controllori

I controllori, o *controller*, sono le entità che legano i *model* con le *view* nel *design pattern* MVC. Stabiliscono i comandi che possono essere eseguiti dall'applicazione e ne regolamentano l'utilizzo. Nel *framework* Yii i controllori possono derivare da due classi: `yii\web\Controller` se gestiscono le richieste effettuate via *web* e `yii\console\Controller` per le istanze dell'applicazione lanciate da linea di comando. Ogni *controller web* espone degli URL per accedere alle azioni implementati. Questi sono solitamente nella forma "appdomain/nome-controller/nome-azione", ma possono essere completamente personalizzati configurando il componente `UrlManager`.

Le classi dei controllori sono salvate all'interno della cartella "controllers" della *web application*. Di seguito verranno descritti i *controller* finora implementati, i loro metodi, chiamati *action* e le loro regole di accesso.

3.3.1 SiteController

`SiteController` rappresenta il controllore che gestisce le azioni generali della *web application*. Si può pensare come un controllore che ha come modello il sito stesso. Estende `yii\web\Controller`.

Azioni

- `actionLogin()`: se non vengono forniti credenziali per l'autenticazione renderizza la *view* associata al modello `LoginForm` visualizzando i campi per inserire le informazioni. Quando vengono inviati i dati dell'utente l'azione si occupa di lanciare il metodo `LoginForm::login()` e se questo ha successo reindirizza il *client* alla pagina che necessitava di autenticazione. Altrimenti viene nuovamente visualizzato il modulo di *login* mostrando gli errori riscontrati.

- `actionLogout()`: effettua il *logout* dell'utente corrente e reindirizza alla pagina *home*.

- `actionIndex()` **:** renderizza la pagina iniziale del sito, quella a cui l'utente giunge tramite l'URL "appdomain". Questa consiste attualmente in un semplice reindirizzamento verso la pagina di visualizzazione di una classe di esempio.

- `actionAbout()` **:** renderizza una pagina di informazioni riguardo la piattaforma.

3.3.2 UserController

La classe `UserController` implementa il controllore che gestisce i modelli `User`. Estende il controllore base `yii\web\Controller`.

Questo controllore implementa la sola azione `actionNew()` che permette di registrare un nuovo utente nella piattaforma. Se vengono passati i parametri del nuovo modello tramite verbo POST di HTTP viene eseguita la validazione e in caso positivo il modello viene salvato. Prima di scrivere l'utente nel *database* viene generato l'*hash* della *password* inserita in fase di compilazione del *form*.

Inoltre, avvenuto il salvataggio, l'utente viene registrato nell'unica classe presente nel database inserendo un nuovo *record* nella tabella `class_user_map`.

Se non vengono passati parametri, oppure la validazione fallisce, viene visualizzato il *form* per l'inserimento dei dati.

3.3.3 ClassController

La classe `ClassController` implementa il controllore che gestisce i modelli `ClassModel`. Estende il controllore base `yii\web\Controller`.

Questo controllore presenta le azioni per la creazione, l'aggiornamento e la cancellazione di un modello, ma finora viene utilizzata nell'interfaccia solo l'azione `actionView()`.

Azioni

- `actionView($id)`: renderizza una pagina di visualizzazione della classe identificata dal parametro `$id`. Se questa non esiste viene lanciata un'eccezione per la visualizzazione di un errore di tipo "404 - Not Found".
- `actionCreate()`: permette la creazione di una classe se sono stati forniti parametri attraverso il verbo POST di HTTP, altrimenti visualizza la pagina per la creazione di una nuova classe.
- `actionUpdate($id)`: funzionamento analogo a `actionCreate()`, in questo caso il modello identificato da `$id` viene aggiornato.
- `actionDelete($id)`: cancella il modello identificato da `$id`.

3.3.4 DossierController

La classe `DossierController` implementa il controllore che gestisce i *dossier*. Estende il controllore base `yii\web\Controller`.

Azioni

- `actionView($id)`: renderizza una pagina di visualizzazione del *dossier* identificato dal parametro `$id`. Se questo non esiste viene lanciata un'eccezione per la visualizzazione di un errore di tipo "404 - Not Found".
- `actionNew($class_id)`: permette di creare un nuovo *dossier* associato alla classe identificata da `$class_id`. Se non vengono passati dati, tramite verbo POST, viene restituito il *form* per inserire le informazioni.

- `actionNewContent($id)`: è l'azione che permette la creazione di contenuti all'interno di un *dossier* identificato dal parametro `$id`. Se non vengono caricate informazioni, attraverso verbo POST di HTTP, viene renderizzata l'interfaccia che gestisce il *form* dove l'utente potrà inserire i dati del nuovo contenuto. Se vengono forniti i dati l'azione si crea un nuovo modello `UserContent`, viene recuperato il *file* eventualmente allegato e generato il percorso dove salvarlo. A questo punto si effettua la validazione del modello e, se l'esito è positivo, viene memorizzato nel *database* e il *file* salvato nel *filesystem*. Inoltre viene inserito un *record* nella tabella `dossier_content_map` che stabilisce la relazione fra il nuovo contenuto e il *dossier* individuato da `$id`. Se la validazione non va a buon fine vengono restituiti gli errori riscontrati che l'interfaccia si occuperà di visualizzare.

- `actionContents($id, $before_id)`: restituisce in formato JSON gli `UserContent` associati al *dossier* identificato dal parametro `$id`. Vengono restituiti al massimo cinque contenuti, ordinati per data di creazione decrescente, a partire dal contenuto identificato da `$before_id`.

3.3.5 UserContentController

Il controllore `UserContentController` si occupa della gestione del modello `UserContent`. Estende `yii\web\Controller`.

Come riportato nella sezione 3.2.5, un `UserContent` vuole rappresentare un contenuto generico (non necessariamente legato ad un *dossier*). La sua creazione è legata al contesto associato al contenuto. Per questo si è pensato di lasciare ad altri controllori il compito di creare i modelli `UserContent`, come nel caso dell'azione `DossierController::actionNewContent()`.

Azioni

- `actionDelete($id)`: si occupa dell'eliminazione dell'`UserContent` individuato dal parametro `$id`. Se il modello non esiste viene lanciata un'eccezione per la visualizzazione di un errore di tipo "404 - Not Found".

- `actionAwarenessContext($id)`: restituisce in formato JSON le informazioni dell'eventuale modello `AwarenessContext` associato al contenuto identificato dal parametro `$id`. Questa azione non è più utilizzata, si è pensato che ogni richiesta delle informazioni di un contenuto dovrebbe ricevere in risposta anche i dati dell'`AwarenessContext` associato al contenuto. Per questo un'azione che fornisca tali informazioni a parte risulta inutile: queste vengono allegate direttamente al modello, come descritto dal metodo `UserContent::fields()`, sezione 3.2.5.

- `actionResource($id, $resource)`: si occupa di inviare la risorsa con nome `$resource` del contenuto individuato `$id`. Come descritto nella sezione 3.2.5, le risorse sono *file* che dipendono da un `UserContent`. Se il contenuto esiste e la risorsa è disponibile per quel tipo di contenuto, si veda il metodo `UserContent::resources()`, si recupera il percorso del interno del *file*, metodo `UserContent::getResourceUri()`, e si procede all'invio. Per gestire il trasferimento del file si usa il modulo XSendFile del *web server* Nginx, presentato nella sezione 2.5.1. Vengono quindi impostati gli appositi *header* HTTP nella risposta che viene restituita a Nginx che si preoccuperà di trasferire i dati.

3.3.6 ContentCommentController

Il controllore `ContentCommentController` gestisce i commenti scritti dagli utenti riguardo un `UserContent`. Estende `yii\web\Controller`.

Azioni

- `actionIndex($content_id, $before_date)`: l'azione ritorna, un massimo di 20, i commenti associati all'`UserContent` identificato dal parametro `$id`. Vengono restituiti i modelli ordinati per data di creazione decrescente in un *array* formato JSON. Se viene specificato il parametro opzionale `$before_date` allora la ricerca si limita ai commenti antecedenti la data indicata da tale parametro. Nel caso in cui venga restituito il commento meno recente viene aggiunto al termine dell'*array* un elemento nullo, per indicare che non esistono altri elementi.
- `actionNew()`: permette il caricamento di un nuovo commento. I dati del modello vengono passati tramite verbo POST di HTTP e inseriti all'interno di un nuovo `UserContentComment`. Questo viene validato e salvato in caso positivo, oppure vengono restituiti gli errori riscontrati.
- `actionDelete($id)`: consente di cancellare il commento identificato dal parametro `$id`. Se il commento non esiste viene sollevata una eccezione di tipo "404 -Not Found".

3.3.7 PostProcessingController

`PostProcessingController` è la classe che implementa il controllore per la gestione dei *file* allegati a un `UserContent` dopo che questo è stato salvato nel *database*. Estende `yii\console\Controller` e viene quindi lanciato da riga di comando in modo asincrono nel metodo `afterSave()` del modello `UserContent`. Crea le risorse associate ai contenuti e può essere usato per processare lato *server* i *file* al fine di ottimizzare lo spazio occupato.

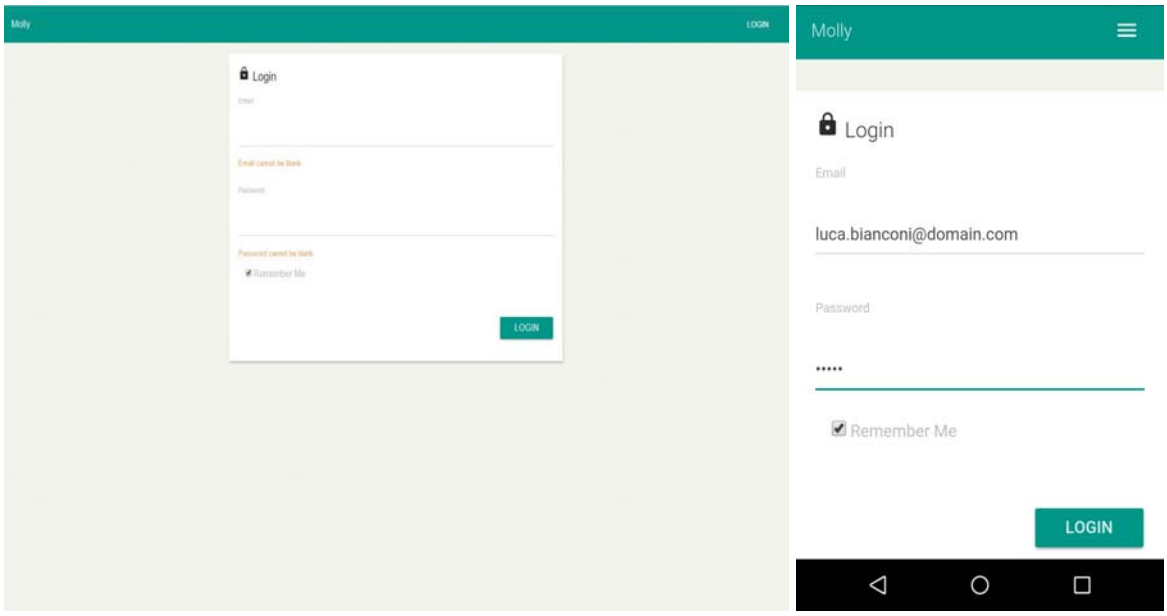
Azioni

- `actionProcess($id)`: lancia le opportune operazioni per la gestione dell'`UserContent` identificato dall'attributo `$id`. Se il contenuto multimediale è di tipo "audio" viene chiamata la funzione `processAudio()`, la funzione che genera la *waveform* della traccia audio. Se il contenuto è di tipo "video" viene eseguita la funzione `processVideo()` che estrapola dal filmato un fotogramma, quello centrale, e lo salva come immagine *poster* del video.

3.4 Interfaccia realizzata

3.4.1 Schermata di login

La schermata di *login*, visibile nella figura 3.9, permette l'inserimento delle credenziali d'accesso di un utente. Viene visualizzato un *form* HTML generato sfruttando la classe `ActiveRecord` di Yii. Il *form* rispecchia gli attributi del modello *LoginForm* descritto in precedenza. Viene validato *client-side* via AJAX rispetto al formato dei vari campi. Se per esempio il campo "email" è vuoto viene visualizzato un apposito messaggio di errore, in arancio nella figura 3.6(a). Quando il *form* viene inviato il *server*



(a) *Desktop* (b) *Mobile*

Figura 3.3: *Schermata di* login

si occuperà di validare le credenziali e, nel caso di errore, risponderà con il *form* corredato dagli errori riscontrati.

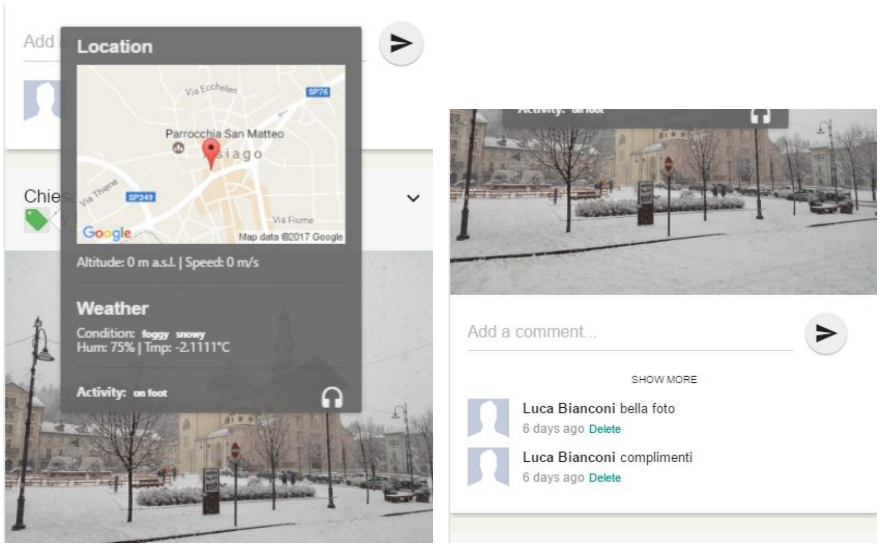
Come si può vedere nella figura 3.6(b), quando la pagina viene visualizzata su schermi di grandezza ridotta, *mobile*, la barra di navigazione è visualizzata in maniera "collassata". Con il bottone *dropdown* posto a destra è possibile accedere alle opzioni disponibili. Questo avviene su ogni schermata della piattaforma.

3.4.2 Vista di un dossier

La schermata, in figura 3.5, mostra le informazioni di un *dossier*. L'interfaccia si presenta con un *layout* a due colonne, nella colonna di sinistra vengono mostrati, in un riquadro, il titolo, la descrizione e il creatore del *dossier*.

La colonna di destra è dedicata alla visualizzazione dei contenuti caricati dagli utenti. Ogni contenuto è disposto all'interno di un riquadro con lo stile di un classico *post* di un *social network*. Per ogni contenuto vengono visualizzati titolo, autore, data di creazione e descrizione, se presente. Nell'angolo in alto a destra del riquadro è presente un pulsante che da accesso a un menù *dropdown*. Questo menù permette la cancellazione di un elemento, se l'utente è l'autore, e la selezione del contenuto per lanciare l'allineamento delle tracce audio, sezione 3.7. Quando un elemento viene selezionato il pulsante in azzurro nella colonna di destra incrementa il numero di contenuti selezionati. Raggiunto il numero minimo di 2 elementi è possibile, cliccando su questo pulsante, lanciare il processo di allineamento, al termine del quale verranno visualizzati i risultati.

Ogni contenuto è visualizzato in maniera diversa secondo la sua tipologia. Un contenuto audio mostra la *waveform* della traccia sonora sfruttando la libreria Peaks.js. La forma d'onda è ottenuta dal *server* gra-



(a) Awareness Context (b) Commenti

Figura 3.4: *Visualizzazione di un contenuto*

zie all'azione `UserController::actionResource()`. Viene inoltre visualizzato un *player* audio per controllare la riproduzione. Un contenuto video visualizza in un riquadro il *player* per la riproduzione, per la produrre i pulsanti di controllo del *player* è stata adottata la libreria JavaScript MediaElement. Anche l'immagine *poster*, visualizzata quando la riproduzione deve ancora essere avviata, di un video è ottenuta come risorsa dal *server*.

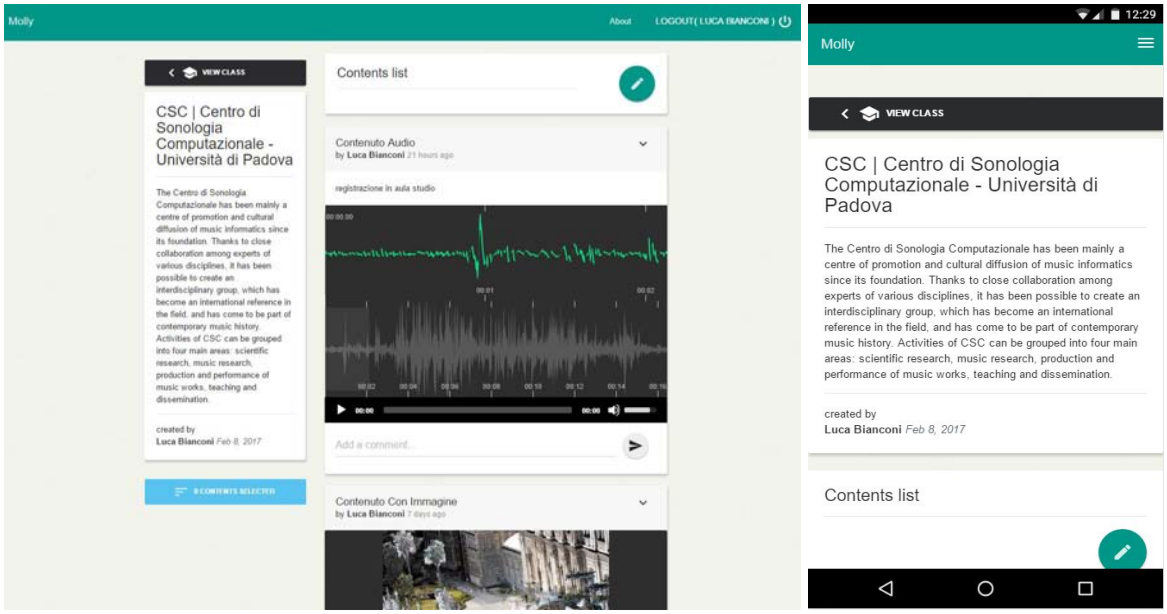
Oltre ai *file* già indicati, per le immagini, gli audio e i video il *file* sorgente associato al contenuto viene recuperato come risorsa. Per i contenuti testuali viene semplicemente mostrata la loro descrizione.

Sotto al titolo di ogni contenuto è possibile leggere la data di creazione, visualizzata in tempo trascorso rispetto all'ora attuale, il creatore del contenuto ed è renderizzato un pulsante per l'apertura di un *popover* che mostra, se disponibile, l'*awareness context* associato al contenuto, figura 3.4(a). Per commentare, ai piedi di ogni riquadro è mostrato un *form* posto sopra alla lista di commenti già salvati, figura 3.4(b). Questa, una volta caricata la pagina, mostra solamente i due commenti più recenti. Tramite un apposito pulsante è possibile espandere la lista, più volte, fino a visualizzare tutti gli elementi.

La lista dei riquadri viene renderizzata grazie alla libreria Mustache.js

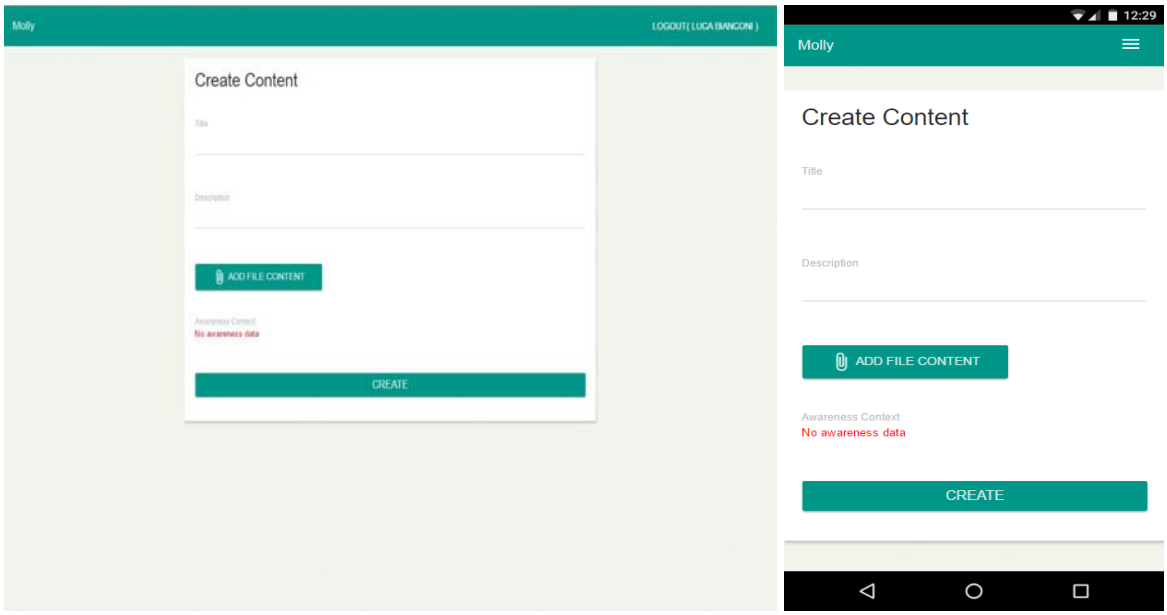
3.4.3 Schermata nuovo contenuto

Nella figura 3.6, è visibile la schermata per la creazione di un nuovo contenuto. Questa viene renderizzata grazie alla classe `ActiveForm` di Yii che opera su un'istanza del modello `UserContent`. Viene effettuata una validazione *client-side*, tramite AJAX, dei campi del *form* e una corrispondente validazione *server side*. Nella validazione *server side* viene controllato anche il *file* caricato. Avviando l'*upload* del contenuto tramite il pulsante "create" viene mostrato lo stato di avanzamento, in percentuale, della richiesta. Questo permette di avere un *feedback* sul processo in corso che altrimenti, con i comuni *browser*, sarebbe assente.



(a) *Desktop* (b) *Mobile*

Figura 3.5: *Schermata di un* dossier



(a) *Desktop* (b) *Mobile*

Figura 3.6: *Schermata nuovo contenuto*

3.5 Autorizzazione

Una volta definiti i controllori e le loro azioni è necessario un sistema di autorizzazione che ne regolamenti l'utilizzo.

Un approccio che si può seguire è denominato *Role Based Access Control*, RBAC. Questo consente di definire quali utenti e in quali circostanze possono accedere a determinate funzionalità della piattaforma. Il *framework* Yii permette di implementare tale sistema grazie al componente `authManager` [1]. Definendo ruoli, permessi e regole, è possibile configurare l'applicazione in maniera da regolamentare l'accesso ai suoi *entry point*.

Il componente `authManager` può sfruttare sia il *database* che il *filesystem* per la memorizzazione e la lettura dei permessi. La documentazione del *framework* indica l'uso del *filesystem* come preferibile, a meno di ruoli molto dinamici.

Di seguito verrà descritto come è stato usato questo componente applicato nel caso del controllore `UserCommentController`. Si ritiene che tale esempio possa essere d'aiuto per chi continuerà lo sviluppo del progetto. Supponiamo di voler autorizzare l'accesso a `ContentCommentController` e alla sua azione `delete`, in maniera tale che possano cancellare il commento solamente l'autore oppure un utente con il ruolo di insegnante. Per determinare il ruolo di un utente si utilizza l'attributo `role`, appositamente creato, del modello `User`. È necessario innanzitutto definire una classe PHP che implementi la regola per riconoscere il ruolo dell'utente corrente, per esempio la classe `UserRoleRule`:

```
class UserRoleRule extends Rule{
  public $name = 'userRole';
  public function execute($user, $item, $params){
    if(!Yii::$app->user->isGuest){
      //l'utente deve essere autenticato
        if($item->name == 'teacher'){
        //recuperiamo l'attributo role dell'utente corrente
        $role = Yii::$app->user->identity->role;
        return $role == 'teacher';
      }else if($item->name == 'user'){
        //tutti gli utenti hanno ruolo user
        return true;
    }
  }
  return false;
  }
}
```

Successivamente viene definita la regola che determina se un utente è l'autore di un commento:

```
class AuthorRule extends Rule{
  public $name = 'isAuthor';
  public function execute($user, $item, $params){
    if($params['comment'])
      return $params['comment']->author == $user
    return false;
  }
}
```

[1] http://www.yiiframework.com/doc-2.0/guide-security-authorization.html

Scritte queste due semplici regole è possibile passare alla configurazione dei ruoli, dei permessi e delle regole che li governano. Questo viene realizzato in una classe chiamata `RbacController` che rappresenta un controllore la cui azione `init()` costruisce la gerarchia voluta. L'implementazione di tale classe è stata riportata nell'appendice A.

Questo controllore può essere eseguito da linea di comando e genera all'interno della cartella "rbac" i file necessari a implementare tale configurazione. Questi vengono usati dal componente `authManager` che gestisce l'autorizzazione.

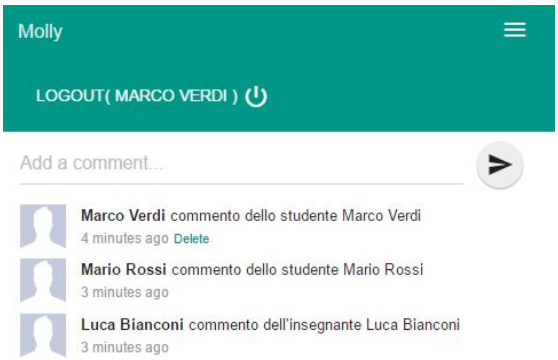
Ora, modificando opportunamente la funzione `ContentCommentController::behaviors()`, si definiscono le azioni dove i controlli vengono applicati:

```
...
'rules' => [
    [
      'allow' => true,
      'actions' => ['delete'],
      'matchCallback' => function($rule, $action){
        $id = Yii::$app->request->get('id');
        if(!empty($id)){
          $comment = ContentComment::findOne($id);
          $user =Yii::$app->user;
                return $comment &&
                        $user->can('deleteComment', [
                                'comment'=>$comment
                                ]);
        }
      return false;
      },
    ],
...
```

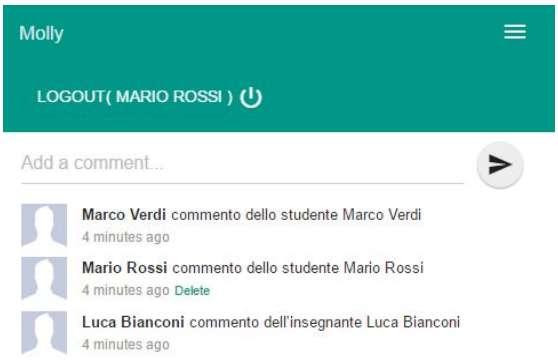
Infine si modifica la funzione `ContentComment::fields()` affinché ritorni il *link* per eliminare il commento solo se l'utente ne ha la possibilità.

```
        ...
        if(Yii::$app->user->can('deleteComment', ['comment'=>$this])){
            $fields['links'] = function($model, $field){
                return [
                'delete' => Url::to([
                    'content-comment/delete',
                    'id'=>$model->id
                    ])
                 ];
            };
        }
        ...
```

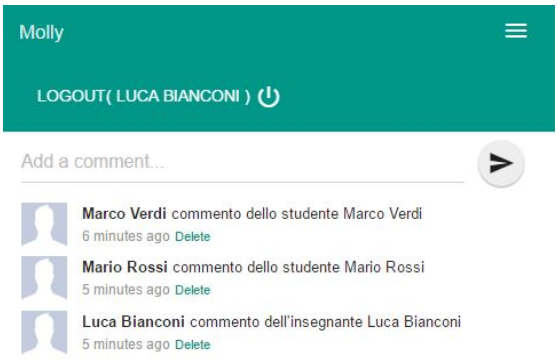
Il risultato dell'implementazione descritta mostrato in figura 3.7. In maniera analoga è possibile gestire l'autorizzazione basata su ruoli di qualsiasi controllore dell'applicazione.



(a) *Studente Marco Verdi*



(b) *Studente Mario Rossi*



(c) *Insegnante*

Figura 3.7: *Controllo sui commenti con diversi utenti autenticati*

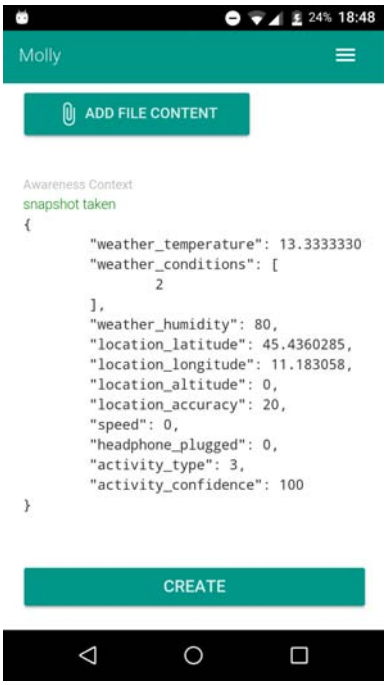


Figura 3.8: *Informazioni dello* Snapshot *della Google Awareness API*

3.6 Android App

È stata realizzata un'app per il sistema operativo Android per l'accesso alla piattaforma. Lo scopo di tale applicazione è aumentare le funzionalità offerte dall'interfaccia *web* quando viene utilizzato un dispositivo *mobile*.

Per implementarla si è scelto di utilizzare una *WebView* che renderizzasse le pagine HTML. La *WebView* viene aperta all'avvio dell'applicazione ed è inserita all'interno di un `SwipeRefreshLayout` che permette il ricaricamento della pagina *web* con la classica *gesture* del trascinamento verso il basso. Inoltre il pulsante *back* del sistema operativo viene associato alla navigazione all'indietro nella cronologia delle pagine visitate. Giunti alla prima pagina visualizzata l'applicazione viene chiusa.

Di seguito viene descritto cosa è stato finora aggiunto nell'app *mobile* rispetto all'interfaccia *web*.

3.6.1 Integrazione di Google Awareness

Per poter utilizzare la Google Awareness API all'interno dell'applicazione *mobile* è stato necessario collegare il codice Java, che recupera i dati, con la *WebView*. Questo è stato possibile realizzando una classe Android di interfaccia fra codice JavaScript e codice Java. I metodi di questa classe possono essere resi accessibili (processo chiamato *binding*) all'interno degli *script* eseguiti dal *browser web*. È quindi possibile chiamare delle funzioni Java come se fossero comuni funzioni JavaScript. La funzione per catturare l'*awareness snapshot* viene invocata quando viene aperta la schermata per la creazione di un nuovo contenuto. Al termine del processo di acquisizione le informazioni ottenute vengono passate alla *WebView* che le inserisce, figura 3.8, nel *form* che verrà inviato al *server*.

L'acquisizione dello *snapshot* è delegata alla classe `AwarenessSnapshotController`. Questa richiede l'informazione su tutti i segnali disponibili nella Awareness API, eccetto i *beacon*. Viene fornito

un *timeout* di 10 secondi all'acquisizione dei dati al termine dei quali vengono ritornate solo le informazioni che è stato possibile ottenere. In condizioni normali questo tempo è sufficiente: l'API elabora le informazioni generalmente in pochi secondi.

3.6.2 Processing contenuti multimediali

Per limitare il consumo di banda degli utenti e lo spazio occupato dai contenuti multimediali si è deciso di processare i *file* prima che questi siano caricati sul *server*. Ciò consente di ridurre le dimensioni dei contenuti di qualità eccessivamente alta, ad esempio elevata risoluzione di video e immagini, e di ricodificare quelli poco compressi.

Si è scelto di utilizzare il programma FFmpeg compilato per il sistema operativo Android grazie ad Android NDK, *Native Development Kit*. In questo modo è possibile usare il *tool* e le sue opzioni anche all'interno di un'app Android.

Come metrica per stabilire quanto un *file*, audio o video, sia dispersivo in termini di spazio occupato è stato scelto il *bitrate*. Questo individua quanti dati vengono impiegati per riprodurre un secondo del flusso multimediale. Un *file* video, o audio, di qualità eccessivamente elevata o con codifica troppo debole avrà un *bitrate* più alto rispetto a un valore soglia da individuare. Per i video questo valore è stato scelto osservando la tabella[2] dei *bitrate* consigliati della famosa piattaforma Youtube.

Decisa la risoluzione massima per un video di 1280 *pixel* per 720 (720p) si ricodificano, a risoluzione eventualmente minore, tutti i *file* che hanno un *bitrate* superiore a 5 Mbit/s. La risoluzione viene limitata all'interno di un *bounding box* di dimensioni 1280 e 720 *pixel*, mantenendo l'*aspect ratio* originale. Lo *standard* video utilizzato per la ricodifica è H.264.

I *file* audio vengono ricodificati se il loro *bitrate* supera il valore di 128 Kbit/s, lo *standard* di codifica scelto è il formato MP3. I parametri scelti rispecchiano i valori dei comuni file MP3 che riducono del 90% la dimensione delle tracce audio in qualità CD.

Sia per i contenuti audio che per i video viene mantenuto solamente il primo *stream* di dati per ciascun tipo: il primo flusso video e il primo flusso audio. Eventuali flussi multimediali ulteriori o tracce di sottotitoli vengono eliminati.

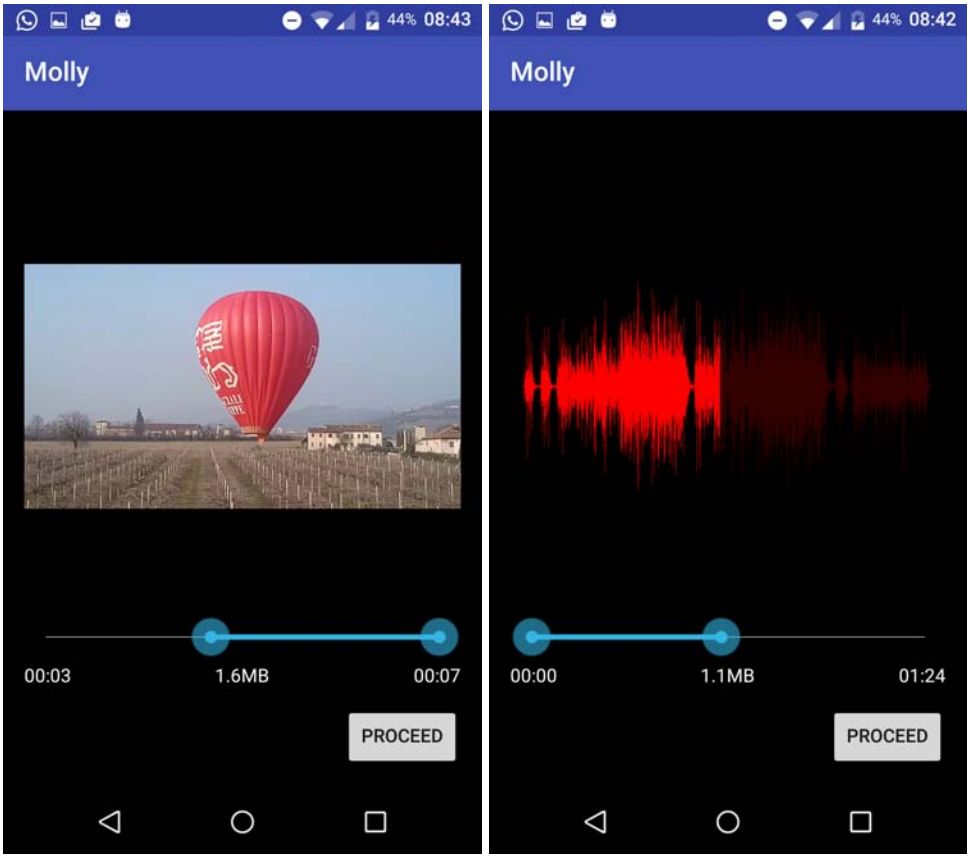
Le immagini sono scalate per limitarne la dimensione massima a 2000 *pixel*, il loro *aspect ratio* viene mantenuto e vengono ricodificate in formato JPEG.

Oltre alla ricodifica viene fornita all'utente la possibilità di tagliare una traccia video o audio. Se l'utente vuole allegare un contenuto multimediale viene aperto un *file chooser* di sistema. Alla selezione di un *file* viene quindi lanciata l'*activity* `VideoCutActivity`, per i video, e `AudioCutActivity` per gli audio. Entrambe consentono tramite una *range bar* di impostare i limiti della traccia che si vuole ottenere. `VideoCutActivity`, figura 3.9(a), mostra il *frame* del video all'istante corrispondente all'ultimo limite impostato. `AudioCutActivity`, figura 3.9(b), mostra la forma d'onda della traccia audio e ne evidenzia la porzione selezionata.

3.7 Allineamento audio

In previsione di una futura funzionalità di *application intelligence* all'interno della piattaforma, si è sviluppato un *tool* per l'allineamento di tracce audio. L'idea è quella di poter riconoscere i contenuti audio e video che contengono suoni registrati dello stesso evento. Per questo è necessario riuscire a individuare

[2] https://support.google.com/youtube/answer/1722171?hl=it



(a) (b)

Figura 3.9: *VideoCutActivity e AudioCutActivity*

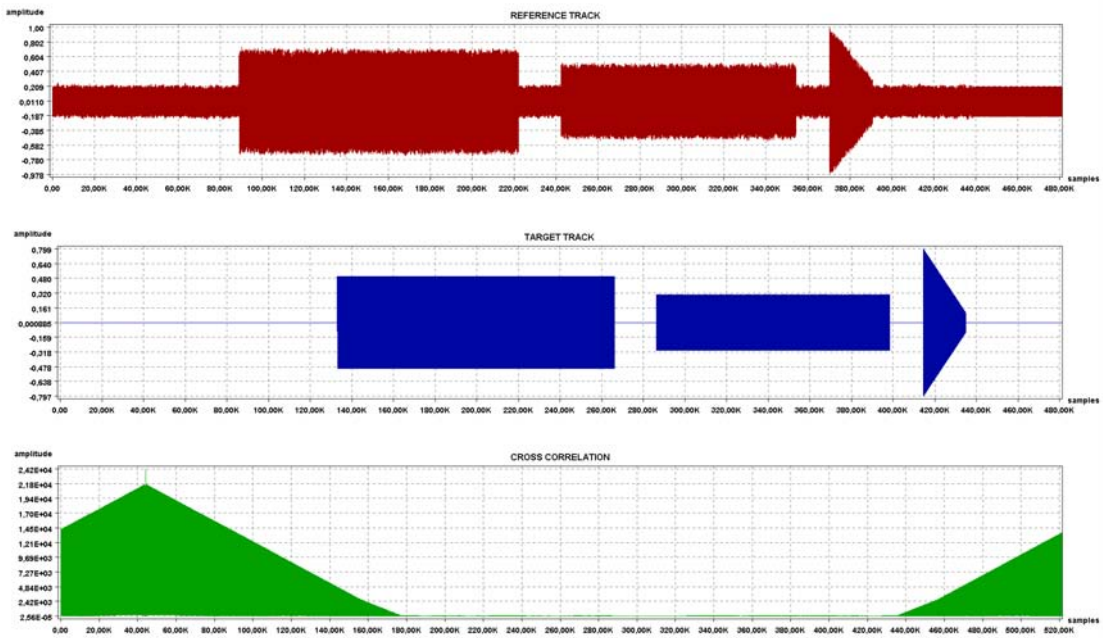


Figura 3.10: *Esempio di cross-correlazione fra tracce audio*

se due o più tracce audio riproducono gli stessi suoni, anche se catturati da diversi dispositivi, con diversi disturbi e ritardi.

È stata svolta una breve ricerca per individuare il metodo più opportuno per svolgere questa operazione. Partendo da un caso di *test*, formato da due registrazioni prese da due diversi, *smartphone* si è usato il *software* Matlab provando diversi algoritmi. È stato riscontrato che il metodo della cross-correlazione fra i campioni audio delle tracce è sufficientemente accurato per tale scopo. La cross-correlazione, nel caso di due segnali, $f[n]$ e $g[n]$ a tempo discreto, è definita come:

$$(f \star g)[n] = \sum_{m=-\infty}^{+\infty} f^*[m]g[m+n]$$

dove $f^*[m]$ rappresenta il segnale complesso coniugato di f. Due segnali fra loro simili, ma ritardati l'uno rispetto all'altro, presentano una funzione di cross-correlazione con un picco positivo in corrispondenza del punto coincidente con il ritardo, in termini di numero di campioni. Questo picco può però non essere l'unico presente nel grafico della funzione e può non essere il più elevato. Un esempio di cross-correlazione fra due tracce è mostrato in figura 3.10.

Si è implementato un programma in linguaggio Java che realizza la cross-correlazione fra le tracce sonore di due *file* audio o video. La funzione è stata calcolata trasformando i segnali nel dominio della frequenza. Dal teorema della convoluzione si ottiene che:

$$\mathcal{F}\{(f \star g)[n]\} = (\mathcal{F}\{f\})^* \cdot \mathcal{F}\{g\}$$

dove $\mathcal{F}\{x\}$ rappresenta la trasformata di Fourier del segnale x. Questo permette di diminuire il tempo di computazione della cross-correlazione. Trasformando i due segnali attraverso l'algoritmo FFT, *Fast Fourier Transform*, e ritrasformando il prodotto fra le due trasformate si ottiene un costo in termini di

tempo $\mathcal{O}(n \log n)$ a fronte di un costo $\mathcal{O}(n^2)$ applicando direttamente la cross-correlazione. Per il calcolo della FFT e della FFT inversa è stata utilizzata la libreria Java Apache Commons Math [3].

I campioni audio vengono letti dai *file*, audio o video, sfruttando i programmi FFprobe e FFmpeg. Questi consentono di prelevare le prime tracce audio di ciascun *file* ricampionandole in maniera che abbiano la stessa frequenza.

Il *tool* è stato successivamente testato su un campione di 45 coppie di *file*, audio e video, registrati presso il CSC usando due diversi dispositivi. Essendo i *file* particolarmente lunghi si è deciso di tentare l'allineamento utilizzando solamente i primi 30 secondi di ciascuna registrazione. Da un confronto visivo e uditivo delle tracce audio con i valori calcolati dal *tool* si sono osservati i seguenti risultati.

Su 45 coppie di registrazioni 43 sono risultate correttamente allineate, ovvero la differenza fra il valore calcolato e lo scostamento osservabile è minore di 5 centesimi di secondo, mentre 2 sono risultate erroneamente allineate. L'accuratezza sul campione è quindi del 95.6%. Dei due allineamenti errati in uno si è riscontrato un errore di 21 centesimi di secondo mentre nell'altro caso le tracce non sono risultate confrontabili nei primi 30 secondi.

Il *tool* Java viene lanciato in esecuzione all'interno della *web application* quando si richiede la sincronizzazione fra due o più contenuti.

[3] http://commons.apache.org/proper/commons-math/

Capitolo 4

Conclusione e sviluppi futuri

L'evoluzione tecnologica ha permesso lo sviluppo di strumenti informatici per la conservazione e la valorizzazione del patrimonio culturale. Nell'ambito dell'insegnamento, l'ICT è stata indispensabile per ampliare e migliorare i metodi di divulgazione e apprendimento delle conoscenze. La migrazione da metodi di insegnamento tradizionali verso altri più interattivi permette agli studenti di arrivare a risultati migliori[28].

In particolare la crescita del *mobile computing* e il suo impiego nelle scuole ha evidenziato la possibilità di realizzare una tipologia di studio individuale, più autonomo e autoregolato nei tempi. La recente diffusione dei dispositivi portatili fa intravedere un grande potenziale del loro impiego nel campo culturale e nell'insegnamento.

Il progetto di una piattaforma *mobile* di *digital learning* per la promozione dell'alfabetizzazione digitale presentato in questa tesi si muove in questa direzione. È stato delineato lo stato dell'arte per individuare strumenti simili a quello immaginato. A partire da questi si sono identificati i requisiti funzionali e tecnici necessari alla piattaforma. Si è quindi passati alla ricerca degli strumenti utili per la realizzazione del progetto, osservando anche le scelte effettuate da piattaforme tecnicamente simili a quella progettata.

Scelti gli strumenti idonei si è passati alla fase implementativa che ha seguito un approccio definibile come *bottom-up*. Dal problema del singolo caricamento e visualizzazione di una traccia audio si è via via passati allo sviluppo del *software* a livello più alto: contenuti di diversa tipologia, organizzazione in *dossier*, utenti,...

L'architettura della piattaforma è stata modellata con una componente *server side* che si fa carico della gestione delle richieste pervenute dagli utenti e della memorizzazione dei loro dati. La *web application* si basa sul *web server* Nginx, il *framework* PHP Yii e un *database* gestito dal RDBMS MySql. Sono stati adottati i principi del *design pattern* MVC: la *web application*, attraverso dei *controller*, espone all'esterno degli *entry point* che consentono di operare sui dati. In risposta alle richieste vengono restituite delle *view*, che possono essere documenti in formato HTML o JSON.

Gli utenti interagiscono con la piattaforma tramite un *client* che può essere un'interfaccia *web* oppure un'*app mobile* sviluppata per il sistema operativo Android. L'interfaccia *web* è stata realizzata seguendo i principi del *responsive design* grazie a tecnologie come Bootstrap e JQuery.

L'*app mobile* si basa sul *client web*, ma ne estende le funzionalità. L'utente può catturare i *file* multimediali, foto e video, nel momento in cui decide di creare un nuovo contenuto e questo può essere corredato con un insieme di informazioni sul contesto dell'utente, sfruttando Google Awareness API. Inoltre l'*app mobile* consente di effettuare una elaborazione dei contenuti multimediali per diminuirne la dimensione ottimizzando così lo spreco di risorse: spazio occupato e traffico dati.

Per la gestione dei *file* audio, video e immagini si è utilizzato, sia a livello *client* che *server*, il progetto FFmpeg, che permette l'elaborazione di molti formati multimediali.

Il progetto vuole realizzare alcune funzionalità di *application intelligence* implementando delle procedure per il riconoscimento automatico di immagini simili, o l'individuazione di registrazioni audio dello stesso evento. Per quest'ultimo obiettivo è stato creato un *tool* in linguaggio Java in grado di individuare il ritardo di una traccia audio rispetto a un'altra, registrata da un diverso dispositivo. Di tale strumento è stato effettuato un *test* presso il CSC che ne ha mostrato l'efficacia.

Durante il lavoro di questa tesi sono state gettate le basi del progetto che dovrà essere ulteriormente sviluppato per essere completato. Le scelte effettuate permettono di continuare lo sviluppo per raggiungere il soddisfacimento di tutti i requisiti funzionali prefissati.

La problematica più grande riscontrata risulta l'elaborazione di contenuti video su dispositivi *mobile*. La codifica di *file* può essere una operazione molto dispendiosa in termini computazionali. Non sono state individuate librerie Android disponibili per tale scopo e, quindi, si è ricorso alla libreria FFmpeg compilata sul sistema operativo *mobile*. Nonostante questa sia molto performante in ambiente *desktop*, operazioni relativamente veloci (un paio di secondi su PC fisso) risultano richiedere molto più tempo in Android. Inoltre, come era atteso, la soluzione di basare l'*app mobile* sull'interfaccia *web* comporta delle prestazioni grafiche peggiori rispetto a quelle normali del codice nativo, anche se accettabili.

Data la natura della piattaforma e la sua pubblica esposizione sulla rete *web*, è indispensabile, prima di qualsiasi rilascio al pubblico (anche per scopi di *test*), proteggere le comunicazioni tra *client* e *server* tramite protocollo HTTPS. Questo, aggiungendo ad HTTP la crittografia del protocollo SSL, *Secure Sockets Layer*, rende la trasmissione di dati al riparo da possibili attacchi di tipo MITM, *Man In The Middle*.

Oltre a soddisfare i requisiti funzionali primari e secondari individuati, ulteriori sviluppi futuri del progetto possono essere:

- estensione dell'*app mobile* affinché si possano catturare immagini, video e registrare audio all'interno dell'applicazione. Ora un contenuto multimediale può essere caricato dalla memoria del dispositivo oppure creato da una applicazione di sistema già installata. Una soluzione integrata permetterebbe di avere maggior controllo sulle impostazioni dei dispositivi *hardware* e quindi di creare contenuti multimediali secondo i requisiti della piattaforma, senza dover successivamente elaborarli. Inoltre una soluzione integrata renderebbe l'esperienza d'uso più rapida e omogenea.

- Possibilità di ottenere dal *server* le immagini e i video con diverse qualità secondo le necessità del *client*. Immagini e video a risoluzione troppo elevata rispetto ai dispositivi sui quali vengono visualizzati comportano uno spreco di risorse. Inoltre, specialmente nel caso dei video, contenuti troppo pesanti per la larghezza di banda dei dispositivi *client* possono rendere problematica la loro fruizione. La diversificazione delle qualità disponibili può prevedere una strategia *on demand*, dove il *server* genera secondo le necessità i contenuti a partire dal *file* sorgente, oppure statica, per la quale, quando il contenuto viene caricato per la prima volta, se ne memorizzano diverse copie con diversa qualità.

- Possibilità per gli insegnanti di aderire a giochi creati all'interno della piattaforma. Gli studenti di diverse classi potrebbero concorrere con altri per la realizzazione di *dossier* che vengono valutati da un'apposita commissione. In questo modo si stimolerebbe ulteriormente il lavoro degli alunni e si renderebbe più accattivante l'uso della piattaforma.

Appendice

Appendice A

Implementazione di RbacController

Di seguito viene riportata l'implementazione di RbacController. La funzione `init()` lega insieme le regole, i permessi e i ruoli secondo la gerarchia desiderata.

```
class RbacController extends \yii\console\Controller{

    public function actionInit(){
        $auth = Yii::$app->authManager;
        $auth->removeAll();

        //$authorRule definisce se l'utente e' l'autore di un commento
        $authorRule = new \app\rbac\AuthorRule;
        $auth->add($authorRule);

        //permesso di cancellare un commento
        $deleteComment = $auth->createPermission('deleteComment');
        $deleteComment->description = 'Delete comment';
        $auth->add($deleteComment);

        //permesso di cancellare un commento se l'utente e' l'autore
        $deleteOwnComment = $auth->createPermission('deleteOwnComment');
        $deleteOwnComment->description = 'Delete own comment';
        $deleteOwnComment->ruleName = $authorRule->name;
        $auth->add($deleteOwnComment);

        //$deleteComment usa anche il permesso $deleteOwnComment
        $auth->addChild($deleteOwnComment, $deleteComment);

        //definisco i ruoli all'interno dell'applicazione
        //$rule e' la regola che definisce se l'utente
        //appartiene a un certo gruppo
        $rule = new \app\rbac\UserRoleRule;
        $auth->add($rule);

        $user = $auth->createRole('user');
        $user->ruleName = $rule->name;
```

```
        $auth->add($user);

        $teacher = $auth->createRole('teacher');
        $teacher->ruleName = $rule->name;
        $auth->add($teacher);

        //lego tutto associando permessi e ruoli
        $auth->addChild($user, $deleteOwnComment);
        $auth->addChild($teacher, $deleteComment);

        //teacher ha anche i permessi di user
        $auth->addChild($teacher, $user);
    }
}
```

Bibliografia

[1] 3D printing applied to Cultural Heritage.Low-cost conservation techniques using 3D modelling and printers. *www.digitalmeetsculture.net*, Accessed Sat Mar 11 2017.

[2] Smith Aaron. Older Adults and Technology Use, Aprile 2014.

[3] C. Angeli and N. Valanides. Preservice elementary teachers as information and communication technology designers: an instructional systems design model based on an expanded view of pedagogical content knowledge. *Department of Education, University of Cyprus*, Luglio 2005.

[4] Matt Asay. PHP and Perl crashing the enterprise party. *www.cnet.com*, 2010.

[5] Luca Bianconi. Progetto e sviluppo di una soluzione software per la simulazione dell'ascolto quadrifonico di un documento sonoro. Bachelor's thesis, Università degli Studi di Padova, 2014.

[6] Elizabeth Cohen. Preservation of audio in folk heritage collections in crisis. *Proceedings of Council on Library and Information Resources*, 2001.

[7] Bebell Damian and Kay Rachel. One to one computing: A summary of the quantitative results from the berkshire wireless learning initiative. *The Journal of Technology, Learning, and Assessment*.

[8] Minkovska Daniela and Ivanova Malinka. Didactic principles of elearning - design and implementation of an interactive adaptive learning system. *Technical University of Sofia*, 2016.

[9] Cardinal David. How the smartphone changed everything; or, the rise of BYOD in the workplace, Dicembre 2015.

[10] europa.eu. Piano d'azione eLearning. Pensare all'istruzione di domani, Marzo 2001.

[11] The Apache Software Foundation. Apache License, Version 2.0. www.apache.org.

[12] Samantha Gartner. How Moodle is driven by user and community feedback, Aprile 2015.

[13] Galluzzo Gary. How do toddlers use tablets?, Giugno 2015.

[14] Burton Graeme. PC sales down in 2016 and will continue declining in 2017, say analysts, Gennaio 2017.

[15] Sterling Greg. It's Official: Google Says More Searches Now On Mobile Than On Desktop, Maggio 2015.

[16] G. Guidi, M. Russo, and D. Angheleddu. 3d survey and virtual reconstruction of archeological sites. *Digital Applications in Archaeology and Cultural Heritage*, 2014.

[17] A-C Haugstvedt and J. Krogstie. Mobile augmented reality for cultural heritage: A technology acceptance study. *IEEE*, 2012.

[18] iSpring. Moodle vs BlackBoard – That is the Question, Agosto 2015.

[19] Ristevski John. Laser Scanning for Cultural Heritage Applications, 2006.

[20] Sandip Kar. 20 pros and cons of implementing byod in schools. *securegenetworks.com*, Luglio 2015.

[21] James Lentz. User interface design for the mobile web. *www.ibm.com*, Luglio 2011.

[22] Mandanici Marcella, Rodà Antonio, and Canazza Sergio. The "harmonic walk": an interactive educational environment to discover musical chords. *Dipartimento di Ingegneria dell'Informazione, Università di Padova*, 2014.

[23] Bradford1 Peter, Porciello Margaret, Balkon Nancy, and Backus Debra. The blackboard learning system, 2007.

[24] Daniela Petrelli, Luigina Ciolfi, Dick van Dijk, Eva Hornecker, Elena Not, and Albrecht Schmidt. Integrating material and digital: A new way for cultural heritage. *interactions*.

[25] Canazza Sergio, Fantozzi Carlo, and Pretto Niccolò. Accessing tape music documents on mobile devices. *Università di Padova*, 2015.

[26] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[27] S. Somasegar and Daniel Li. The intelligent app ecosystem (is more than just bots!), Maggio 2016.

[28] Yao-Ting Sung, Kuo-En Chang b, and Tzu-Chien Liu. The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. *Department of Educational Psychology and Counseling, National Taiwan Normal University*, Agosto 2015.

[29] V. Vlahakis, M. Ioannidis, J. Karigiannis, M. Tsotros, M. Gounaris, D. Stricker, T. Gleue, P. Daehne, and L. Almeida. Archeoguide: an augmented reality guide for archaeological sites. *IEEE Computer Graphics and Applications*, 2002.

[30] Fred Vogelstein. The Day Google Had to 'Start Over' on Android. *The Atlantic*, accessed 6 November 2014. www.theatlantic.com.

[31] Joe Wolf. Responsive HTML5 Apps: Write Once, Run Anywhere? Where is Anywhere?, 2013.