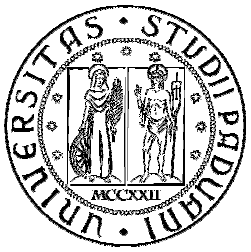

UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI INGEGNERIA

Applicazioni per Augmented Reality in ambito turistico:

Layar Reality Browser

Relatore: Federico Filira

Laureando: Michele Guarnaccia

Corso di laurea Magistrale in Ingegneria Informatica

Data di laurea: 07 Dicembre 2010

Anno Accademico 2010/2011

“It’s gonna happen, get ready for it”,

B. Sterling presso il Layar Launch Event

Indice

Parte I

Introduzione

Introduzione

Con il termine Augmented Reality (AR) si vuole indicare una visione del mondo reale, diretta o indiretta, i cui elementi sono "aumentati" tramite oggetti virtuali; quest'aumento è, quasi per convenzione, in tempo reale e all'interno di un contesto semantico predefinito: la sovraimpressione di risultati sportivi durante un match calcistico ne è un esempio. Con l'aiuto di tecnologie avanzate per AR (es. computer vision, riconoscimento di oggetti), l'informazione del mondo che circonda l'utente si piega di fronte ad esso e diventa interattiva e digitale: dati artificiali, riguardanti l'ambiente stesso e gli oggetti ivi contenuti, possono essere salvati, recuperati e visualizzati come un layer, uno strato, sovrapposto al mondo reale.

Ad oggi l'attenzione dei ricercatori di questo campo è andata maggiormente focalizzandosi sulle aree dell'intrattenimento e dell'advertising: il rapido sviluppo della tecnologia ha infatti prodotto dispositivi desktop e mobile dotati di tutto l'hardware necessario per supportare applicazioni di Realtà Aumentata nella vita quotidiana. Particolare interesse è rivolto al mondo degli smartphone dove le tecnologie mobili possono contare su una consistente e funzionale dotazione hardware, oramai contenuta in quasi tutti i telefoni prodotti per il mercato: CPU, sistemi operativi, fotocamere ad alte prestazioni, connessioni a banda larga, wireless, UMTS o HDSPA, ricevitori GPS, accelerometri, bussole digitali, ecc.; non può quindi sorprendere come la combinazione di questi elementi renda gli smartphone moderni ottime piattaforme per la Realtà Aumentata.

Ci troviamo di fronte alla nascita di un nuovo fenomeno di massa: l'Augmented Reality può essere classificata tra le "disruptive technologies", tecnologie innovative in grado di aprire nuovi mercati finanziari o alterare quelli già esistenti, fornendo beni e servizi attraverso inedite forme.

Parte di questo successo è certamente attribuibile a colossi industriali del calibro di Google ed Apple: le due aziende con i loro dispositivi mobili e sistemi operativi, Android e iOS, hanno il merito di aver contribuito per prime alla diffusione dell'AR ad un vasto pubblico e non solo ad una ristretta cerchia d'utenti dotati di hardware e conoscenze specifiche. Si pensi che l'advisor Gartner stima una crescita del mercato degli smartphone, tra il 2010 ed il 2013, da 180 milioni a 1,4 miliardi di unità. Si può ben capire come le applicazioni per Realtà Aumentata, distribuite ad un vasto pubblico, rappresenteranno i mezzi espressivi del futuro, i nuovi media: non sono solo semplici erogatori di informazioni, ma prodotti in grado di mutare i dati in loro possesso a seconda dell'ambiente che li circonda, rendendoli fruibili da coloro che vi interagiscono.

Gli approcci ed i contesti applicativi sono innumerevoli e, per ottenere una migliore comprensione di questa nuova tendenza e fenomeno, nel seguito, verrano discussi e presentati alcuni di questi software: esistono infatti diversi applicativi sia per iPhone che per Android appartenenti a questa categoria, molti dei quali devono ancora essere approvati o rilasciati.

In particolare, nel seguito della trattazione sarà analizzato Layar Reality Browser, il quale permette agli utenti di "scoprire" il mondo che li circonda, semplicemente puntando il proprio smartphone in una qualsiasi direzione: una volta selezionato il layer desiderato, le informazioni in esso contenute, e relative alla posizione dell'utente stesso, saranno sovraimposte al mondo reale tramite

lo schermo del dispositivo. La peculiarità di questo software è insita nella possibilità di discernere tra diversi strati di contenuto realizzati da una comunità di sviluppatori esterni: Layar mira infatti a realizzare un browser per navigare la Realtà Aumentata.

Nel seguito verrà quindi esaminato il settore dei beni culturali, il quale si presta in particolar modo all'impiego di Layar Reality Browser: grazie ad esso le informazioni relative a ciò che viene inquadrato come nomi di opere o monumenti, date storiche, foto, video, testi, curiosità, possibilità d'interazione coi social network e mappe satellitari possono essere offerte agli utenti con un'unica applicazione e in ogni momento.

L'obbiettivo principale di questo lavoro sarà la realizzazione di un layer a scopo turistico distribuito per i visitatori della città di Padova, e fruibile tramite il browser Layar. Il servizio, denominato Padova Tour, offrirà agli utenti la possibilità di partecipare a visite tematiche del capoluogo veneto, basate su percorsi prestabiliti, integrate con diversi strumenti interattivi, collaborativi e meccanismi di feedback. Questo strumento potrà quindi essere impiegato da uno o più enti turistici, per strutturare dei percorsi a tema, anche sulla base dei riscontri ricevuti. Accanto all'ambiente AR risulterà utile ed interessante fornire un servizio di content management system (CMS) che permetta a tali enti di inserire, aggiornare e modificare i contenuti del layer, in maniera semplice, veloce e soprattutto non tecnica, ovvero ignorando i dettagli di programmazione.

Obbiettivi

Gli obbiettivi prefissi per questo lavoro di tesi possono essere riassunti nei seguenti punti:

- approfondire la conoscenza dell'applicazione Layar Reality Browser, analizzando la struttura, i meccanismi di funzionamento ed esplorando le potenzialità d'impiego;
- eseguire una corretta contestualizzazione dell'applicazione nel panorama dell'Augmented Reality, ricercando anche soluzioni alternative a Layar Reality Browser;
- sviluppare un servizio per Layar Reality Browser, utilizzando le API per sviluppatori, e contestualizzarlo nell'ambito turistico. Il layer realizzato dovrà offrire agli utenti diverse funzionalità che permettano agli utenti di:
    - partecipare a visite guidate all'interno del territorio Padovano, seguendo itinerari storico-artistici in completa autonomia;
    - sussidiare l'esperienza di visita tramite elementi multimediali quali foto e file audiovisivi;
    - accedere ad un libro degli ospiti virtuale, dove poter lasciare un commento o conoscere le impressioni di altri utenti;
    - aumentare l'esperienza di visita tramite l'interazione con l'applicazione;
    - migliorare l'esperienza di visita tramite un semplice meccanismo di feedback.
- integrare il layer all'interno di un sistema per la gestione dei contenuti.

Parte II

Augmented Reality

Capitolo 1

Augmented Reality

1.1 Definizione

Con il termine **Augmented Reality** (AR) si vuole indicare una visione del mondo reale diretta o indiretta, i cui elementi sono "aumentati" tramite immagini virtuali. Esso è in relazione con un concetto più generale chiamato Mediated Reality, nel quale la visione della realtà è modificata in vari modi, magari diminuita (Diminished Reality) piuttosto che aumentata, da un calcolatore. Il risultato è quello di alterare la percezione del mondo reale da parte di uno o più individui. Nel caso della Realtà Aumentata l'aumento avviene per convenzione in tempo reale e all'interno di un contesto semantico predefinito: la sovra impressione di risultati sportivi durante un match calcistico ne è un esempio. Con l'aiuto di tecnologie avanzate per AR (es. computer vision, riconoscimento di oggetti) l'informazione del mondo circostante l'utente diventa interattiva e digitale: dati artificiali riguardo l'ambiente e gli oggetti in esso possono essere salvati, recuperati e visualizzati come un layer, uno strato, sovrapposto al mondo reale.

Realtà Aumentata è una variazione di quelli che vengono definiti Virtual Environment (VE), comunemente conosciuti come Virtual Reality, i quali permettono di "immergere" completamente l'utente in un ambiente sintetico: durante quest'operazione l'individuo non può che vedere il mondo virtuale intorno a lui, al contrario AR consente di vedere oggetti virtuali sovrapposti o mixati al mondo reale. Pertanto, AR integra la realtà piuttosto che sostituirla completamente: idealmente sembrerebbe, per l'utente, che gli elementi virtuali e reali convivano nello stesso spazio.

Augmented Reality aumenta quindi la percezione e l'interazione degli individui con il mondo reale. Gli oggetti virtuali permettono così di visualizzare informazioni che l'utente non potrebbe individuare direttamente con i propri sensi.

La ricerca nel campo della Realtà Aumentata esplora l'applicazione di immagini generate al computer in live-stream video, come un modo per espandere il mondo reale. La ricerca avanzata comprende l'uso di head mounted display (descritti in seguito) e virtual retinal display (apparecchiature che permettono di visualizzare un'immagine direttamente nella retina dell'utente) per scopi di visualizzazione, e la costruzione di ambienti controllati contenenti un numero qualsiasi di sensori ed attuatori.

Ad oggi esistono due definizioni comunemente accettate di Augmented Reality. Quella più adottata, però, è stata espressa da Ronald Azuma [9] nel 1997, con l'intento di evitare la limitazione della Realtà Aumentata a determinate piattaforme o strutture, e afferma che essa:

- combina realtà e virtualità;
- è interattiva in tempo reale;

• è tridimensionale.

Questa definizione permette di includere altre tecnologie oltre ai già citati head mounted display (HMDs), ma non comprende ad esempio i film (non sono media interattivi) o overlay bidimensionali (non sono tridimensionali).

La seconda definizione è dovuta a Paul Milgram e Fumio Kishino [8]: nel 1994, i due ricercatori definirono quello che viene detto Milgram's Reality-Virtuality Continuum (riportato in Figura 1.1). I due studiosi descrivono un continuum che spazia dalla realtà, così come la conosciamo, ad un ambiente puramente virtuale. Tra questi due estremi troviamo la Mixed Reality (MR), di cui fanno parte Augmented Reality (più vicina al reale) e Augmented Virtuality (più prossima al virtuale).

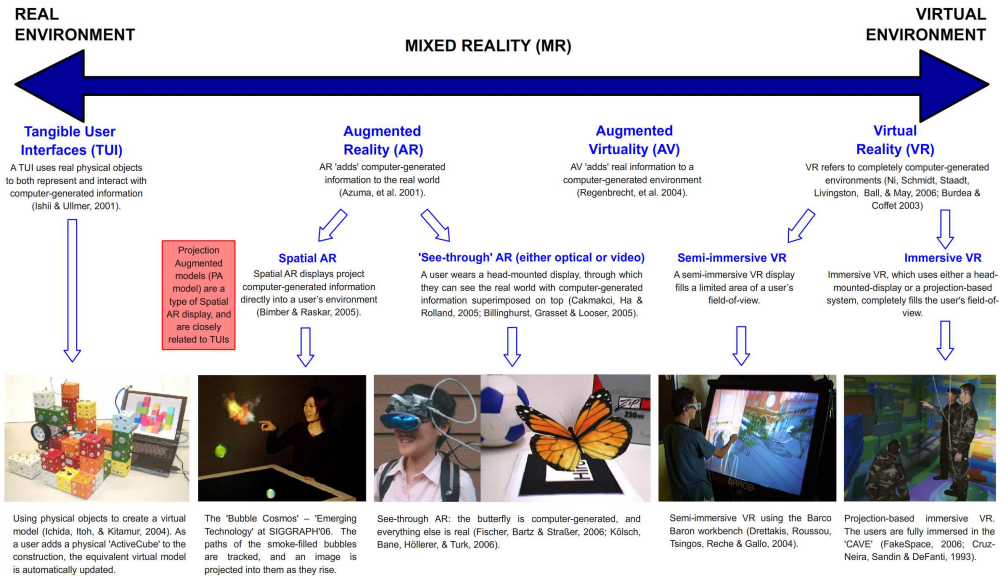

Figure 1.1: Il continuum di Milgram.

1.1.1 Augmentation

L'Augmented Reality si può estendere a tutti i sensi, non solo alla vista. Fino a poco tempo fa, i ricercatori si concentravano sulla fusione di immagini reali e virtuali, tuttavia l'AR può essere estesa per includere anche il sonoro: l'utente potrebbe infatti indossare delle cuffie (o ascoltare tramite alcuni diffusori) che aggiungerebbero un suono sintetico e direzionale.

Oltre ad inserire oggetti in un ambiente reale, è anche possibile rimuovere quelli già presenti nella scena: gli overlay grafici possono essere utilizzati anche per eliminare o nascondere parti dell'ambiente reale ad un utente. Ad esempio, per rimuovere una scrivania, si potrebbero disegnare le pareti ed il pavimento, rispettivamente, dietro e sotto di essa, effettuandone così l'occultamento. Facendo riferimento all'audio augmentation, dei microfoni esterni potrebbero rilevare i suoni generati nell'ambiente dando al sistema la possibilità di mascherarli o di coprirli, generando un segnale di mascheramento che annulli esattamente il suono in arrivo.

Anche il tatto può essere aumentato: si pensi ad esempio a dei guanti con dispositivi che forniscono un feedback tattile. Tali dispositivi potrebbero andare ad aumentare le forze all'interno

dell'ambiente e, se un utente scorresse la mano su di una superficie, degli effettori tattili nel guanto potrebbero aumentare le sensazioni della mano, magari facendola percepire più grezza in certi punti.

1.2 Tecnologia

La Realtà Aumentata non è basata su una tecnologia in particolare, ma piuttosto su un insieme di tecnologie, metodi ed algoritmi. Di seguito sono riportati gli strumenti che Azuma [9] riteneva essenziali per l'AR.

1.2.1 Hardware

Le principali componenti hardware utilizzate per produrre Realtà Aumentata sono essenzialmente un display, un sistema di tracking, dispositivi di input, ed un elaboratore.

1.2.1.1 Display

Un elemento cruciale nella modellazione di sistemi AR è come eseguire la combinazione tra ambiente reale e virtuale. Due scelte di base sono disponibili: l'impiego di tecnologie ottiche (see-through) e di quelle basate su video (video-through). Ognuna gode di particolari vantaggi e svantaggi.

- *Semplicità*: la miscelazione tra realtà e virtualità è più semplice e meno costosa se realizzata tramite tecnologia ottica; gli approcci ottici, infatti, hanno un solo flusso di video di cui preoccuparsi, ovvero le immagini grafiche realizzate, poiché il mondo reale è visto direttamente attraverso il dispositivo. La combinazione delle due viste avviene quasi in tempo reale (generalmente pochi nanosecondi di delay). Al contrario, le tecnologie video-through devono fare i conti con diversi flussi video per le immagini reali e virtuali. La digitalizzazione di queste ultime aggiunge, inoltre, un ritardo nel flusso video, così come la sincronizzazione che deve essere realizzata per non incorrere in risultati affetti da distorsioni temporali: un see-through HMD offre una visione del mondo reale priva di distorsione, mentre le tecnologie ottiche ne possiedono, quasi sempre, una certa quantità che deve essere compensata;

- *Flessibilità*: un problema di base con i dispositivi ottici è che gli oggetti virtuali non possono oscurare del tutto quelli del mondo reale poiché il combinatore ottico permette la coesistenza di luce sia da fonte virtuale che reale. Normalmente, gli oggetti sono progettati per essere a fuoco in un solo punto nel cammino ottico (l'occhio dell'utilizzatore), invece negli HMD display si presentano due punti in cui l'immagine è a fuoco: questa caratteristica rende il design ottico molto più difficile e complesso: gli oggetti virtuali spesso appaiono semi-trasparenti, danneggiando l'illusione di realtà. Al contrario, un display video-through è molto più flessibile poiché sia la componente reale che quella virtuale sono disponibili in formato digitale;

- *Visione*: le distorsioni nei sistemi ottici sono una funzione della distanza radiale dall'asse ottico, ovvero più un utente guarda lontano dal centro della vista, maggiore sarà la distorsione ottenuta; eventuali distorsioni devono essere corrette otticamente piuttosto che digitalmente, poiché generalmente il sistema non possiede l'immagine digitalizzata del mondo reale. Al contrario i dispositivi video non sono affetti da tale problematica.

Le due tecnologie sono alla base di diverse tipologie di display, tra i quali:

- *Head-mounted*: si dividono in see-through e closed-view; i primi consentono all'utente di vedere il mondo reale con oggetti virtuali sovrapposti, mentre i secondi non ne permettono alcun visione diretta del. Questi dispositivi, che come dice il nome sono "montati in testa", offrono agli utente un'esperienza molto coinvolgente poiché garantiscono completa libertà di movimento;

- *Hand-held*: si tratta di una tecnica che utilizza un piccolo dispositivo (ad esempio uno smartphone) che può essere tenuto nella mano di un utente. Le soluzioni di questa tipologia sfruttano tecniche video-through per sovrapporre le informazioni grafiche al mondo fisico. Inizialmente questi display venivano associati a sensori come bussole digitali e GPS, per realizzare un tracking a sei gradi di libertà (6 DOF). I due principali vantaggi di questa tecnologia sono da ricercarsi nella natura portatile dei dispositivi e nel loro carattere ubiquitario: accanto alla visione reale dell'utente vi è quella ottenuta tramite il terminale, entrambe fruibili contemporaneamente e generate da punti d'origine differenti;

- *Spatial*: rispetto delle precedenti, questa tecnologia fa uso di proiettori digitali per visualizzare le informazioni grafiche su oggetti fisici. La differenza fondamentale nei sistemi SAR (Spatial Augmented Reality) è che il display è separato dagli utenti del sistema. Poiché il display non sono associati ad ogni utente, questa tecnologia risulta scalabile naturalmente a diversi gruppi di utenti consentendo così la collaborazione tra di essi (Social Augmentation). SAR ha diversi vantaggi rispetto ai display head-mounted e a quelli hand-held: l'utente non è tenuto a trasportare o ad indossare la tecnologia di visualizzazione, rendendola effettivamente un buon candidato per il lavoro collaborativo, inoltre non soffre di problemi legati alla risoluzione e alle dimensioni del display poiché può semplicemente inserire più dispositivi per espandere l'area di visualizzazione. Le tecnologie precedentemente presentate lavorano su piccola finestra del mondo, tuttavia un sistema SAR può presentare diversi elementi grafici su un qualsiasi numero di superfici di un ambiente con un'unica operazione.

1.2.1.2 Tracking, dispositivi di input e CPU

I moderni sistemi di telefonia mobile che implementano applicazioni per Realtà Aumentata, tendono ad utilizzare una o più delle seguenti tecnologie di rilevamento: macchine fotografiche digitali, accelerometri, GPS, giroscopi, bussole a stato solido, RFID e sensori wireless. Ciascuna di queste ha diversi livelli di accuratezza e precisione, ma molto importante è la localizzazione della posizione dell'utente in base alla quale realizzare l'aumento della visione dell'utente.

Esistono invece diversi strumenti di input, da pinch gloves a bacchette "magiche". Nel caso di dispositivi smartphone, si impiegano principalmente fotocamere, ma ad esempio (poiché si tratta proprio di un campo di ricerca aperto) si potrebbe utilizzare un dispositivo di puntamento 3D, che ricostruisce la posizione tramite le immagini catturate dalla fotocamera.

Infine si noti che sistemi basati su fotocamere richiedono potenti CPU e una notevole quantità di RAM per l'elaborazione immagini acquisite. Ad esempio iPhone 3Gs (uno dei primi smartphone adatti all'AR) possiede un processore a 600 MHz e 256 MB di RAM.

1.2.2 Software

Per ottenere una coerente fusione delle immagini sul mondo reale, dalla fotocamera, con le immagini virtuali, queste ultime devono essere integrate all'ambiente in modo visivamente realistico. Questo significa che dovrebbe essere ricostruito un sistema di coordinate del mondo reale indipendente dalla fotocamera: tale processo è chiamato "registrazione di immagini" e fa parte

della definizione di Azuma di Augmented Reality. Questa tecnica utilizza diversi metodi di visione artificiale, per lo più legati al tracking di immagini, che solitamente consistono di due parti:

1. detection dei punti di primo interesse, o marcatori fiduciari, rilevati nel flusso ottico della telecamera. In questa prima fase si possono utilizzare metodi di feature detection come corner, blob, edge o thresholding, ma anche altri meccanismi di elaborazione delle immagini;

2. ricostruzione del sistema di coordinate reali. Alcuni metodi sfruttano oggetti di cui è nota la geometria (marcatori fiduciari), presenti nella scena, e ricostruiscono interamente la geometria del mondo; altri ancora, se non è stata fatta alcuna ipotesi sulla scena, ricostruiscono il mondo tramite metodologie basate sul movimento: geometria epipolare, filtri di Kalman e particolati.

[9].

1.2.2.1 Limitazioni

In realtà le tecnologie AR soffrono di diversi problemi [19], per esempio legati al sistema GPS: essi infatti hanno un grado di precisione di 9 metri e non funzionano correttamente all'interno di edifici; recentemente però sono state sviluppate tecnologia come XPS che sfrutta le reti wifi per offrire un servizio più affidabile e preciso. Altri problemi possono essere correlati alle dimensioni del display, che può essere troppo piccolo per gestire tutte le informazioni fornite; anche in questo caso l'evoluzione tecnologica porrà presto rimedio, grazie per esempio ai display flessibili AMOLED di Samsung.

1.3 Applicazioni per realtà aumentata

1.3.1 Scenario attuale

Quando Azuma pubblicò il suo survey [9] nel 1997 descrisse sei tipologie di applicazioni AR, di seguito riportate come nell'originale:

1. annotation

2. entertainment and advertising

3. manufacturing and repair

4. medical visualization

5. military aircraft

6. robot path planning.

Interessante è la distinzione che opera Gary Hayes [14], esperto del settore e, più in generale, della comunicazione mediatica, il quale riconosce cinque tipologie di software per Realtà Aumentata, non basandosi sul campo di applicazione, ma eseguendo una schematizzazione relativa alle tecnologie utilizzate:

- *surface*: la più semplice forma di AR, caratterizzata da schermi che rispondono all'interazione dell'utente visualizzando diverse informazioni;

- *pattern*: applicazioni che operano un riconoscimento di pattern specifici come marker o volti umani, sostituendoli o accostandovi nuovi dati (ad esempio un modello 3D);
- *outline*: sistemi che eseguono tracking del movimento, mixandolo con elementi virtuali all'interno di un ambiente virtuale (si pensi alle applicazioni di EyeToy di Sony);
- *location*: tecnologie che si basano su GPS e meccanismi di localizzazione per identificare oggetti, edifici e persone;
- *hologram*: simile ad Outline, ma in questo caso gli oggetti reali e/o virtuali vengono proiettati in uno spazio fisico in cui l'utente può interagirvi.

Secondo Azuma nel futuro sarebbero state le applicazioni industriali e militari a guidare lo sviluppo delle tecnologie AR. Oggigiorno, però, anche se il mondo della Realtà Aumentata può essere ancora suddiviso secondo la sua visione, l'attenzione è andata maggiormente a focalizzarsi sulle aree dell'intrattenimento e dell'advertising.

Sono molti i fattori attribuibili a questo spostamento, ma il principale è legato al manifestarsi della possibilità di utilizzare la Realtà Aumentata nella vita quotidiana: in special modo, il rapido sviluppo delle tecnologie smartphone ha prodotto telefoni cellulari dotati di tutto l'hardware necessario, ed anche se non progettati specificatamente per AR, questi sono divenuti una piattaforma ideale per le applicazioni di Realtà Aumentata.

Parte di questo successo è certamente dovuta a Google ed Apple: le due aziende con i loro dispositivi mobili e sistemi operativi, Android e iOS, hanno il merito di aver contribuito per prime alla diffusione dell'AR ad un vasto pubblico e non solo ad una ristretta cerchia d'utenti dotati di hardware e conoscenze specifiche.

I dati che possono essere sfruttati dall'AR nella quotidianità sono innumerevoli: basti pensare alle informazioni che esistono in Rete sotto forma di blog, feed, foto, video, mappe, ma anche post e profili su social network come Facebook o Twitter, dove l'informazione viene continuamente accresciuta. Alcune informazioni esistono sotto forma di geotag: il geotagging è un meccanismo che permette di associare a dati geografici, quali coordinate GPS, contenuti diversi che possono andare da semplici testi ad elementi multimediali. Ancora una volta, la moderna tecnologia per smartphone ci viene incontro, permettendo agli utenti di eseguire geotagging in maniera semplice e veloce: combinando questa tecnica con informazioni di vario genere e reti sociali, diviene chiaro come la realtà possa essere aumentata in ogni modo ed in ogni luogo.

Le applicazioni AR presto ci permetteranno di ottenere qualsiasi informazione desideriamo da qualsiasi luogo, aumentando e migliorando la nostra percezione della realtà in cui viviamo [12].

Nel seguito saranno presentati alcuni esempi di applicativi per Realtà Aumentata su dispositivi mobili.

1.3.2 Esempi

ARToolkit è la libreria open source per eccellenza, realizzata dall'Università di Washington, per lo sviluppo di applicazioni per Augmented Reality (`http://www.hitl.washington.edu/artoolkit/`); essa implementa diversi algoritmi per il tracking della posizione, dell'orientamento e della calibratura di videocamere, detection di marker, ed overlay a tre dimensioni. ARToolkit è distribuita per diverse piattaforme quali SGI IRIX, Linux, MacOS e Windows, ed è completa di API con esempi che ne illustrano le diverse funzionalità.

ARToolkit è stata rilasciata in diverse versioni e tra le più importanti troviamo: FLARToolkit, che permette l'integrazione con Adobe Flash e Adobe Flex, NyARToolkit per sistemi hostati su Virtual

Machine, mixare e AndAR per dispositivi Android. Tutte le estensioni della libreria, e maggiori informazioni, sono visualizzabili presso l'indirizzo `http://en.wikipedia.org/wiki/ARToolKit`.

SREngine , o Scene Recognition Engine (`http://www.srengine.com/`), è un'applicazione per iPhone (attualmente disponibile solo per il Giappone) basata su tecnologie GPS e di image detection, che permette agli utenti di investigare qualsiasi oggetto venga puntato dalla videocamera: il software esegue una scansione del contenuto della scena isolandone l'elemento predominante, dal quale estrae informazioni tramite riconoscimento di forme e caratteri (e quindi anche codici a barre e QR), ed operando una ricerca su database online. SREngine è un software open source ed è rilasciato insieme ad un SDK per sviluppatori, purtroppo la maggior parte della documentazione è disponibile in lingua giapponese.

TAT Augmented ID è un software in via di sviluppo da TAT e Polar Rose che implementa tecnologie per il riconoscimento facciale: ogni utente deve inizialmente registrare un account cui assocerà informazioni personali ed i propri profili online (ad esempio il proprio Twitter o YouTube). Completata quest'operazione, chiunque punti il proprio dispositivo mobile, con l'applicazione TAT Augmented ID attiva, verso un qualunque individuo registrato sulla piattaforma, potrà eseguirne il riconoscimento ed accedere alle informazioni suoi profili! Di certo questo tipo di software dovrà scontrarsi con la complessa gestione della privacy dell'utenza.

Unifeye , prodotto da Metaio (`http://www.metaio.com`), offre un piattaforma software per il developing di applicazioni AR di vario genere. Agli sviluppatori è data la possibilità di utilizzare un SDK, dotata di API di qualità e ben documentate, per diversi linguaggi di programmazione (C# e linguaggi di scripting) e piattaforme mobili (Android, iOS, Symbian e Windows Mobile). Una delle applicazioni di maggior successo per quest'azienda è Lego Digital Box: realizzata per Lego, permette ai clienti di visualizzare un modello tridimensionale e animato di un qualsiasi prodotto Lego, solamente avvicinandone la confezione ad un "box" dotato di webcam e schermo; il modello può essere esplorato in ogni direzione, anche al suo interno, semplicemente ruotando la confezione.

Ovviamente il mondo dell'AR non si limita a quest'ambito ma spazia in più settori: esistono software che permettono di utilizzare queste tecnologie in ogni ambito, come ZugStar di Zugara (`http://www.zugara.com/augmented-reality/video-conferencing`) che offre all'utenza la possibilità di partecipare a video conference aumentate o ancora Futuroscope (`http://www.lesanimauxdufutur.com/`), un parco divertimenti a tema dove i visitatori possono riscoprire la preistoria del nostro pianeta interagendo con ogni suo elemento.

Le applicazioni per Realtà Aumentata rappresentano i mezzi espressivi del futuro, i nuovi media: non sono solo semplici erogatori di informazioni, ma prodotti in grado di mutare i dati in loro possesso a seconda dell'ambiente che li circonda, rendendoli fruibili da coloro che vi interagiscono.

Parte III

Layar Reality Browser

Capitolo 2

Layar Reality Browser

Layar Reality Browser è un'applicazione per smartphone, effettivamente disponibile per sistemi Android e per Iphone (3Gs, 4), sviluppata nel 2009 dagli olandesi Raimo van der Klein, Claire Boonstra e Maarten Lens-FitzGeraldche. Essa mira a realizzare un browser per navigare la Realtà Aumentata recuperando informazioni relative al mondo degli utenti (suddivise in layer), prodotte e gestite da sviluppatori esterni: proprio come in un browser web, come ad esempio IExplorer od Opera, che consente di visualizzare pagine e documenti realizzati da terze parti. In Layar ogni dato è percepito sotto forma di un **point of interest** (POI) con cui è possibile interagire, sovrapposto alla vista reale.

Figura 2.1: Schermata di Layar Reality Browser. Particolare del layer Canal+.

Layar Reality Browser è diviso in due moduli principali: il browser (front-end) che permette di sperimentare i livelli di contenuto e la piattaforma per servirli e pubblicarli (back-end); quest'ultima è dove gli strati sono definiti e funge da collegamento con l'editore che ne fornisce i dati. La definizione dello strato è una componente chiave di ogni layer poiché ne contiene il titolo, il look&feel, i meta-tag e l'ubicazione del servizio web (ovvero dove i dati sono realmente ospitati ed i metodi con cui vengono elaborati): ogni strato è infatti basato su un web service che elabora e gestisce le informazioni da presentare all'utente finale.

Le principali caratteristiche e funzionalità che possono essere utilizzate, includono:

1. *punti d'interesse (POI)*: questi sono alla base del funzionamento di Layar e sono i dati che vengono serviti all'utente, sotto forma di informazioni relative a luoghi specifici, e che vengono quindi visualizzati dall'applicazione;

2. *elementi tridimensionali*: al posto di icone semplici per indicare un POI, è possibile utilizzare modelli 3D;

3. *trigger di prossimità*: essi definiscono un'azione che si verificherà quando l'utente giunge in prossimità di una località. Un esempio, può essere un uovo di Pasqua che compare su una piazza quando gli utenti si trovano a 10 metri da essa;

4. *multimedia*: tutti i punti di interesse possono avere uno o più elementi audio, audio/video assegnati; in questo modo, restando sull'esempio precedente, una canzone di accompagnamento potrebbe essere eseguita in combinazione con il trigger;

5. *autenticazione*: consente di sviluppare servizi ad-personam. Un esempio ne è lo strato Tweepsaround: solo dopo il login, è possibile accederne ai contenuti, vedere i propri amici Twitter nei dintorni e interagirvi.

Layar Reality Browser è un software gratuito e quindi non vi è nessun costo per l'utilizzo della piattaforma; recenti sviluppi permettono però lo sviluppo di commercial-layer, ovvero la possibilità di ottenere particolari contenuti sotto il pagamento di una piccola quota, ma ciò non incide sulla libertà di utilizzo del browser [16].

Nel seguito saranno presentate l'applicazione client, l'architettura di sistema e le API per lo sviluppo di livelli di contenuto.

2.1 L'applicazione client

Layar Reality Browser consente agli utenti di esplorare il mondo attraverso diversi strati; ognuno di questi contiene informazioni geo-codificate che possono essere visualizzate attraverso l'applicazione. La caratteristica più interessante e peculiare dell'applicazione è sicuramente la veduta Augmented Reality, in cui si sovrappongono i dati dello strato al mondo reale (ovvero la vista catturata dalla telecamera): puntando il dispositivo nella direzione desiderata, si vedranno comparire i punti di interesse e le informazioni addizionali associatevi.

Nel seguito verranno evidenziati gli elementi caratterizzanti della struttura dell'applicazione client: questi saranno presentati nell'ordine con cui vengono acceduti da un utente durante una sessione di lavoro.

2.1.1 Galleria

I diversi strati possono essere trovati sfogliando la Galleria Layer (un elenco organizzato di tutti i contenuti), attualmente suddivisa in quattro sezioni:

- *Cerca*: l'utente può cercare i livelli inserendo una parola chiave (tag);

- *Strati*: l'intera collezione di contenuti, suddivisa in quattro sottosezioni:

- locali: layer utilizzabili solo nelle vicinanze dell'utente. Si noti che se questi è in roaming in un paese diverso da quello del paese predefinito per il suo telefono, l'elenco mostrerà anche i livelli specifici per il paese in cui si trova;
- popolari: sulla base del numero di spettatori unici di un livello, viene stilata una classifica di popolarità in cui i 20 strati più popolari sono mostrati in essa. Tale insieme è aggiornato quotidianamente. Da notare che se uno strato è limitato ad una particolare un'area geografica, il suo rango di popolarità sarà probabilmente basso;
- consigliati: un piccolo insieme di layer scelti dal personale di Layar e dotati di funzionalità interessanti o recentemente rilasciati;
- categorie : suddivisione dei layer per categorie d'interesse.

- *Mia roba*: gli utenti possono inserire un bookmark per qualsiasi livello abbiano visitato; al fine di velocizzare ogni sessione gli strati bookmarked sono raccolti in quest'elenco (funzione "Preferiti");
- *Vicino*: una lista di POI nelle vicinanze, indicizzati da diversi strati.

2.1.2 Opzioni

Dopo che l'utente ha selezionato uno dei livelli, può essergli presentata una finestra di settaggi, dove potrà impostare diverse opzioni per il livello in esame. Uno sviluppatore può decidere di visualizzare queste impostazioni automaticamente la prima volta che lo strato viene acceduto. Altrimenti, tali impostazioni sono comunque raggiungibili direttamente attraverso una voce di menù.

Queste impostazioni solitamente contengono i filtri utilizzati per limitare il raggio della ricerca o per la selezione dei contenuti, i quali possono essere presentati sotto forma di:

- text box,
- check box,
- custom slider,
- range slider,
- radio button.

Queste opzioni per il livello, non sono da confondere con le impostazioni del browser le quali sono accessibili dal menù principale: queste ultime permettono di operare sulle impostazioni dei filtri e di visualizzazione, eseguire il login al proprio account Layar (funzionalità per sviluppatori) e ottenere informazioni sull'applicazione.

2.1.3 Visualizzazione Augmented Reality

La visualizzazione Augmented Reality viene attivata quando un utente seleziona uno strato. Una volta acceduta, l'applicazione selezionerà automaticamente il POI più vicino allo spettatore in linea d'aria e ne visualizzerà le informazioni. All'interno della vista AR un piccolo radar mostra tutti i POI individuati nel range di ricerca attuale i quali sono sovra imposti su di una griglia tridimensionale nella vista principale; per ciascuno di essi è indicata la posizione e, se possibile, la distanza dall'utente, mentre un'icona ne permette un'identificazione (spesso non univoca). Le icone

utilizzate possono essere progettate per simulare, in base alle loro dimensioni, la distanza reale tra il client ed il punto scelto (discusso in seguito).

L'utente può anche eseguire un lock su un POI: una volta bloccato, la scelta fatta non cambierà, lasciando i dati ad essa collegati in sovrimpressione sullo schermo fino a quando non verrà selezionato un POI altrove nella visualizzazione AR.

Nella parte inferiore dello schermo, vengono visualizzate le informazioni sul POI selezionato. Questa informazione, denominata Brief Info Widget (BIW), è generalmente costituita da un titolo, tre campi di testo, un'ulteriore stringa di piccole dimensioni, ed un'immagine.

Quando la BIW è selezionata può comparire un "foglio di azione": esso offre all'utente maggiori opzioni, per mezzo delle quali può ottenere informazioni aggiuntive o interagire con il POI, tramite diverse attività, quali ad esempio:

- raggiungere un URL (Uniform Resource Locator),
- chiamare un numero di telefono,
- inserire del testo,
- spedire messaggi email,
- ottenere le indicazioni per raggiungere un POI, utilizzando le mappe o le applicazione di navigazione del dispositivo (attualmente solo Google Maps).

Fatta eccezione per l'ultima opzione, tutte le altre azioni sono configurabili dallo sviluppatore per ciascun POI.

2.1.4 Circular Info Widget

Come impostazione di default, ciascun POI viene visualizzato come un disco colorato con una dimensione corrispondente alla sua posizione rispetto al suo stato: questi indicano se un punto è "selezionato", oppure a quale distanza dall'utente si trova (lontano, vicino, ...). I colori di questi possono essere selezionati dallo sviluppatore del livello e sono denominati Circular Info Widget (CIW). Layar attualmente offre la possibilità di specificare delle icone personalizzate tramite una procedura guidata: all'interno di un livello possono essere definiti fino a tre diversi tipi (ma dalla versione 3.6 non vi è limite) di punti di interesse, ognuno con il proprio insieme di icone. Ad esempio, uno strato di location per feste ed eventi può distinguere tra bar, club e ristoranti, assegnando un'icona diversa per ciascuno di essi. Ciascun icon set è composto da quattro icone, una per ogni possibile stato del POI. Una buona idea è quella di impiegare la stessa icona per tutti e quattro gli stati, variandone soltanto colore e dimensione, senza danneggiare l'esperienza dell'utente; in alternativa può essere definita solo l'icona di stato "selezionato", lasciando le restanti al valore di default. Infine, un altro aspetto su cui porre attenzione è di utilizzare le icone personalizzate con parsimonia per evitare di affollare lo schermo con una moltitudine di immagini e fornire all'utenza un'esperienza confusa.

2.1.5 Lista

La visualizzazione Lista, permette all'utente di ottenere un elenco di tutti i POI individuati dal livello ordinati per valori crescenti della distanza dal client. Ogni elemento di questa mostra le stesse informazioni del BIW della vista AR e può essere selezionato con le stesse modalità. Un utente può anche eseguire il lock su un POI: esso rimarrà bloccato passando alle altre visualizzazioni.

2.1.6 Mappa

Questa modalità mostra all'utente tutti POI della ricerca come nella vista AR salvo che vengono visualizzati su di una mappa. Come per la vista AR, i punti sono rappresentaati per mezzo di un'icona e la BIW del POI selezionato viene visualizzata nella parte inferiore dello schermo ed offre lo stesso foglio di azioni. Le tre viste sono tra loro interscambiabili per mezzo di un semplice menù.

2.2 Architettura

L'applicazione client costituisce l'interfaccia attraverso la quale gli utenti interagiscono con il sistema, inviando richieste e visualizzando i dati ricevuti. L'architettura alla base del funzionamento di Layar Reality Browser è costituita di cinque moduli:

1. il client sul dispositivo mobile dell'utente;
2. il server Layar, il cuore del servizio, che fornisce le interfacce ed i link alla piattforma di provisioning e a quella esterna Layar Service Providers;
3. il sito web Layar Provisioning dal quale gli sviluppatori possono presentare e gestire i propri livelli;
4. i Layar Service Provider creati ed utilizzati dagli sviluppatori;
5. le sorgenti di contenuto che forniscono di fatto le informazioni da visualizzare nel browser, come ad esempio immagini tratte da Flickr.com. Le fonti di contenuto non sono necessariamente separate dal Service Provider ma, in generale, possono essere entità logiche diverse, come un database di informazioni geo-codificate connesso a servizi web che non supportano le API per sviluppatori Layar.

Ciascun layer viene configurato dagli sviluppatori tramite il Layar Provisioning e la sua definizione viene inserita nel Layar Server. Quando un utente seleziona uno strato, l'applicazione client invierà alcune informazioni al server Layar, quali la posizione dell'utente, il nome dello strato ed altri dati (definiti nelle richiesta), che verrano qui utilizzate per reperirne i contenuti; da qui la richiesta verrà inoltrata verso un Service Provider che la elaborerà e la restiruirà attraverso il percorso.

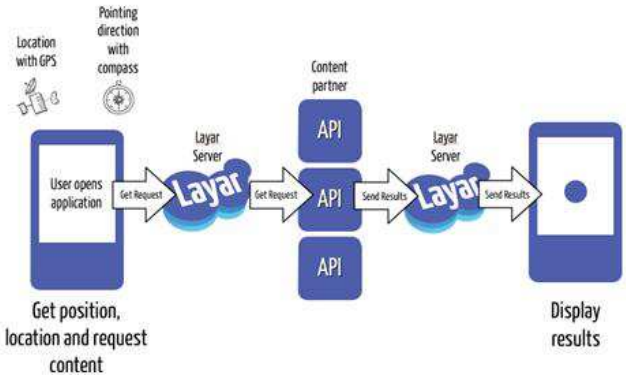

Figure 2.2: Rappresentazione dell'architettura di Layar Browser.

Le interfacce utilizzate per l'applicazione sono invece due:

1. client API: l'interfaccia tra il server e l'applicazione Layar, non accessibile agli sviluppatori;
2. developer API: l'interfaccia tra il server ed i fornitori di servizi. Tramite essa gli sviluppatori possono creare i propri livelli, e tramite il sito di Provisioning li trasmettono al Layar Server.

Un'applicazione è di fatti il risultato del processo di creazione di un nuovo strato, utilizzando il sito Layar Provisioning ed esponendo le informazioni del layer tramite l'API per gli sviluppatori [17].

Layar Reality Browser è stato realizzato utilizzando la piattaforma AppEngine di Google ed è basato sui servizi di cloud computing offerti da Amazon.com.

2.2.1 Representational State Transfer (REST)

2.2.1.1 Architettura REST

Prima di proseguire nella trattazione ed analizzare le API dell'applicazione sarà discusso lo stile architetturale REST, poichè interessa tutte le chiamate verso queste.

Representational State Transfer (REST) è uno stile di architettura software per sistemi distribuiti di hyper-media come il World Wide Web. Il termine Representational State Transfer è un concetto abbastanza recente: esso è stato introdotto e definito nel 2000 da Roy Fielding nella sua tesi di dottorato [10]. Fielding è anche uno dei principali autori del Hypertext Transfer Protocol (HTTP) versioni 1.0 e 1.1, il quale è conforme ai vincoli REST e vi è riferibile come "RESTful".

REST può essere considerata come la descrizione a post hoc delle caratteristiche del Web che ne hanno decretato il suo successo. Questo stile esemplifica come l'architettura del Web sia emersa a caratterizzare e condizionare la macro-interazioni delle maggiori quattro componenti di Internet, vale a dire i server di origine, i gateway, i proxy ed i client, senza imporre limitazioni sui singoli partecipanti: REST infatti gestisce essenzialmente il corretto comportamento dei suoi elementi.

Le architetture basate su REST consistono principalmente di client che inviano richieste ai server, i quali le processano e restituiscono risposte adeguate. Le richieste e le risposte sono definite in riferimento al trasferimento di *rappresentazioni* di *risorse*. Una risorsa può essere essenzialmente un qualsiasi concetto coerente e significativo che può essere indirizzato, mentre una rappresentazione è tipicamente un documento che coglie lo stato attuale o presunto di una risorsa.

In qualsiasi momento, un client può essere o in transizione tra gli stati dell'applicazione oppure a riposo: in questo stato un'applicazione è in grado di interagire con l'utente, ma non aumenta il carico di lavoro e non consuma spazio di memoria sul set di server o, più in generale, nella rete. Il client invia quindi le richieste quando è pronto ad operare la transizione verso un nuovo stato e viene considerato, appunto, in transizione. Ciascuna rappresentazione di stato dell'applicazione contiene link simbolici che possono essere riutilizzati la prossima volta che il client sceglie di avviare una transizione di stato (magari per recuperarne uno precedente). REST è stata inizialmente descritta nel contesto di HTTP, ma non si limita solo a tale protocollo: architetture RESTful possono essere basate su altri protocolli del livello applicativo, a patto che essi forniscano un vocabolario ricco ed uniforme per le operazioni basate sul trasferimento di stato di rappresentazione.

Le applicazioni RESTful tendono a massimizzare l'uso delle preesistenti e ben definite interfacce, e di altre funzionalità integrate, previste dal protocollo di rete scelto, nonchè a ridurre al minimo l'implementazione di nuovi metodi specifici. D'altra parte, protocolli come SOAP RPC su HTTP, che incoraggiano ogni designer a definire un nuovo vocabolario di sostantivi e verbi arbitrari (ad esempio `getUsers`(), `savePurchaseOrder`(...)), solitamente sovrascrivono i verbi HTTP, modificando la struttura di base del protocollo: ciò va a discapito delle capacità di autenticazione, caching,

negoziazione del contenuto, ecc., e lasciano all'application designer la possibilità di re-inventare molte di queste funzioni all'interno del nuovo vocabolario.

2.2.1.2 Vincoli

Un'architettura RESTful è sottoposta ai seguenti sei vincoli, legati in particolar modo alla trasparenza ed alle performance del sistema nonchè all'interazione delle sue parti, lasciando allo sviluppatore completa libertà nell'implementazione delle singole componenti:

1. **client-server**: i client sono separati dai server tramite un'interfaccia uniforme; questo implica che, ad esempio, i client non si preoccupano dell'archiviazione dei dati, che rimane interna a ogni server, di modo che la portabilità del codice sia migliorata. I server non sono interessati invece all'interfaccia dell'utente o allo stato di quest'ultimo, al fine di ottenere semplicità e scalabilità. Server e client possono essere sostituiti e sviluppati in maniera indipendente a condizione che l'interfaccia di comunicazione non venga alterata;

2. **stateless**: la comunicazione client-server è limitata dall'esigenza che nessun contesto client venga memorizzato su server. Ogni richiesta proveniente da qualsiasi utente contiene tutte le informazioni necessarie, e ogni stato è mantenuto da quest'ultimo. Il server può comunque essere stateful, ma tale vincolo implica solamente che lo stato del lato server sia indirizzabile tramite URL come una risorsa; ciò non solo rende i server più visibili per le operazioni di monitoraggio, ma li rende anche più affidabili di fronte a fallimenti, anche parziali, della rete e ne rafforza ulteriormente il livello di scalabilità;

3. **cacheable**: come accade per il WWW, i client sono in grado di salvare le risposte in cache. Le risposte devono, pertanto, implicitamente o esplicitamente, definire il loro essere "cacheable", per impedire il riutilizzo di dati non aggiornati o non appropriati in risposta ad ulteriori richieste. Un meccanismo di caching ben gestito, elimina parzialmente o completamente alcune interazioni client-server, migliorando ulteriormente la scalabilità e le prestazioni;

4. **sistema layered**: un client generalmente non è a conoscenza se esso sia collegato direttamente al server finale, o ad un intermediario lungo il path. Elementi intermedi possono difatti migliorare la scalabilità del sistema consentendo il bilanciamento del carico, alimentando anche un sistema di cache condivisa. Essi possono anche applicare policy di sicurezza. Questa proprietà è conosciuta anche come "stratificazione";

5. **interfaccia uniforme**: l'interfaccia uniforme tra client e server, discussa in parte in precedenza, semplifica e disaccoppia l'architettura, consentendo ad ogni parte di evolvere indipendentemente. I quattro principi guida di questa interfaccia sono illustrati nel seguito;

6. **codice a richiesta** (opzionale): i server sono in grado, temporaneamente, di estendere o personalizzare le funzionalità di un client, trasferendogli la "logica" che esso può eseguire. Esempi di questo possono includere componenti come applet Java e script lato client come JavaScript.

Se un servizio viola un qualsiasi vincolo (eccetto quello opzionale), non può assolutamente essere riferito come RESTful; il rispetto di questi, e quindi la conformità allo stile architettonico, consentirà a qualsiasi tipo di sistema distribuito per hyper media di avere, auspicabilmente, proprietà quali prestazioni, scalabilità, semplicità, modificabilità, visibilità, portabilità e affidabilità.

2.2.1.3 Principi guida dell'interfaccia

L'interfaccia uniforme, che un'architettura REST deve fornire, è un elemento fondamentale nella progettazione di ogni servizio. E' perciò utile che siano definiti degli standard, delle linee guida, che permettano di sviluppare interfacce facilmente integrabili tra loro:

1. *identificazione delle risorse*: le risorse sono individuate nelle richieste, ad esempio utilizzando gli URI (Uniform Resource Identifier) in sistemi REST web-based, e sono concettualmente distinte dalle rappresentazioni che vengono restituite al client. Ad esempio, il server non invia il suo database, ma, con linguaggio HTML, XML o JSON, rappresenta alcuni record del database, codificati in qualche formato a seconda dei dettagli della richiesta e dell'implementazione del server di attuazione;

2. *manipolazione delle risorse attraverso le rappresentazioni*: quando un client detiene una rappresentazione di risorsa, inclusi eventuali metadati allegati, dispone di informazioni sufficienti per modificarla o eliminarla sul server a patto che abbia i permessi per farlo;

3. *messaggi auto-descrittivi*: ogni messaggio contiene le informazioni necessarie e sufficienti per descrivere come elaborarlo. Per esempio, può essere il parser da invocare, specificato con un tipo di media su Internet (tipo MIME), o ancora, le risposte che esplicitamente indicare la loro cacheability;

4. *hypermedia come motore dello stato dell'applicazione*: se è probabile che il client desideri accedere a risorse correlate, queste devono essere individuate nella rappresentazione restituita, ad esempio fornendo gli URI, nel contesto adeguato, come i collegamenti ipertestuali.

Un altro concetto importante in REST è l'esistenza di risorse (fonti di informazioni specifiche), ciascuna delle quali viene riferita con un identificatore globale. Al fine di manipolare queste risorse, i componenti della rete (user agent e server di origine) devono comunicare attraverso un'interfaccia standardizzata per lo scambio delle rappresentazioni di queste, ovvero i documenti effettivi delle informazioni. Ad esempio, una risorsa che descrive una circonferenza può accettare e restituire una rappresentazione che specifica un punto di centro ed il raggio, formattati in formato SVG (Scalable Vector Graphics), ma può anche accettare e restituirne una che specifichi tre punti distinti lungo la curva come un elenco separato da virgole.

Un qualsiasi numero di connettori (ad esempio, i client, server, cache, ecc.) possono mediare la richiesta, ma ciascuno senza "vedere oltre": questo comportamento è denominato stratificazione, uno dei vincoli REST, ed è un comune principio in molte altre architetture di rete ottenuto tramite il requisito di trasparenza. Pertanto, un'applicazione in grado di interagire con una risorsa deve conoscere due informazioni basilari: l'identificatore della risorsa e l'azione richiesta; non c'è bisogno di sapere se ci siano cache, proxy, gateway, firewall, o qualsiasi altro elemento tra esso ed il server che realmente detiene le informazioni. L'applicazione, tuttavia risente della necessità di capire il formato delle informazioni restituite.

Un servizio web RESTful è un semplice web-service implementato utilizzando HTTP e i principi di REST. Si tratta di una raccolta di risorse con tre aspetti ben definiti quali l'URI di base per il servizio web, come `http://example.com/resources/`, il tipo MIME dei dati supportati dal servizio web (spesso JSON, XML o YAML ma può essere qualsiasi altro tipo valido) e l'insieme degli interventi realizzati dallo stesso, utilizzando metodi HTTP come ad esempio POST e DELETE.

Si noti infine che a differenza dei servizi web basati su SOAP, non esiste uno standard ufficiale per il servizio web RESTful: questo perché REST è una architettura, a differenza di SOAP, che è un protocollo.

La seguente tabella evidenzia come i verbi HTTP siano in genere utilizzati per implementare un servizio web [10],[20].

Risorse	GET	PUT	POST	DELETE
Collezione di URI, come `example.com/resources/`	Elenca gli URI ed altri dettagli della collezione.	Sostituisce l'intera collezione con un'altra.	Genera una nuova entry nell'insieme. L'URL è generalmente assegnato automaticamente e inserito nella risposta.	Elimina l'intero insieme di risorse.
Elemento URI, come `example.com/resources/1`	Recupera la rappresentazione di un oggetto utilizzando i MIME adeguati.	Aggiorna o genera l'elemento della collezione.	Tratta l'elemento come una nuova collezione e genera una nuova entry in esso.	Elimina l'elemento.

Tabella 2.1: Applicazione generica dei metodi del linguaggio HTTP.

2.3 API

In questa sezione saranno presentate le API che formalizzano il servizio offerto da Layar Reality Browser messe a disposizione degli sviluppatori; grazie a queste è infatti possibile interagire con l'applicazione e realizzare i propri livelli di contenuto.

2.3.1 Richiesta

2.3.1.1 Formato della richiesta

Tutte le richieste sono effettuate utilizzando i parametri di richiesta normalizzati e conformi alle seguenti regole:

1. i parametri devono essere ottenuti all'interno del servizio tramite GET;
2. i parametri vengono raccolti per nome e ordinati lessicograficamente. Se due o più parametri condividono lo stesso nome, essi saranno ordinati in base al loro valore;
3. i parametri sono concatenati in ordine alfabetico secondo le seguenti regole:
   (a) al loro valore per mezzo di un carattere "=" (codice ASCII 61);
   (b) con un "+" se contengono più di un valore;
   (c) per coppie tramite un simbolo "&" (codice ASCII 38).

4. se i parametri contengono stringhe in codifica UTF-8, queste devono essere sostituite dai loro corrispondenti valori *% HH*, dove HH rappresenta la codifica esadecimale del byte;

5. per gli URL sono utilizzate sequenze di escape.

Generalmente non è necessario conoscere la struttura di una richiesta a meno che il servizio elaborato non debba innescarne specifiche al layer in uso, o ad altri strati.

2.3.1.2 Chiamata GetPointOfInterest

La funzione `GetPointOfInterest` è invocata dal client ogni qualvolta debba inviare una richiesta per dei punti di interesse da visualizzare in un particolare livello. La chiamata è strutturata secondo le seguenti convenzioni:

- l'URI della richiesta deveessere strutturato come `http://<poiUrl>?<parameters>;`
- il campo header HTTP deve contenere:
  - user-agent: l'agente originario del client;
  - il formato: Layar/x.y [OS name]/x.y.z ([Brand] [Model]);
    Ad es. "Layar/3.1 Android/2.1 (Motorola Milestone)" o "Layar/3.1 iPhoneOS/3.1.2 (Apple iPhone3GS)".
- `poiURL`: quest'elemento è parte della definizione dello strato e può essere impostato utilizzando il Layar Provisioning site: esso identifica la pagina del servizio web;
- `parameters`: ogni richiesta contiene una sequenza di parametri che possono essere utilizzati dal servizio. I parametri, i loro formati di rappresentazione e una breve descrizione di ciascuno sono riportati nella seguente tabella:

Parametri	**Formato**	**Funzione**
userId	string	Identificativo unico ed "anonimo" dell'utente finale.
developerId	string	Id dello sviluppatore del layer
developerHash	string	Questo hash è utilizzato come un semplice strumento per verificarne l'autenticità. Per evitare la duplicazione delle richieste, può essere utilizzato anche il protocollo OAuth
layerName	string	Identificatore del layer
version	string	Questo indica la versione delle API in uso nel telefono, ma non può essere utilizzata per differenziare i dispositivi client (Android e iPhone possono utilizzare la stessa stringa di versione)
requestedPoiId	string	Stringa di testo che identifica il POI che ha attivato la richiesta per lo strato nel Layar Stream. Se possibile dovrebbe essere sempre inclusa nella risposta, indipendentemente dalle impostazioni del filtro
lang	string	Codice di due lettere che identifica il linguaggio nel client
timestamp	integer	Timestamp utilizzato come chiave hash
lat, lon	decimal	Latitudine e longitudine della posizione corrente (GPS)

RADIOLIST (opzionale)	string	L'optionid corrispondente al valore dell'opzione selezionata all'interno della radiolist (o il valore predefinito se non è stato cambiato)
SEARCHBOX, SEARCHBOX_n (opzionale)	string	Stringa contenente il termine di ricerca inserito. Blocchi di ricerca multipli saranno distinti dal prefisso _2, _3
CHECKBOXLIST (opzionale)	string	Contiene i valore selezionati nel checkbox: valori multipli vengono inviati al server utilizzando uan virgola come separatore.
CUSTOM_SLIDER, CUSTOM_SLIDER_n (opzionale)	float, integer	Il valore del custom slider selezionato dall'utente. Anche in questo caso slider multipli saranno distinti dal prefisso _2, _3
accuracy (opzionale)	integer	L'accuratezza della posizione corrente, come indicato dal dispositivo. Si noti che il valore accuracy non può essere inviato se è utilizzata una posizione fissa (developer feature)
radius (opzionale)	integer	Il valore per il raggio di ricerca selezionato dall'utente nel RANGE_SLIDER. Dalla versione 3 il raggio non è più obbligatorio, e può essere utilizzato un raggio flessibile (il client si adatterà automaticamente alla distanza restituita per i POI)
alt (opzionale)	integer	L'altitudine corrente dell'utente. Questo valore non è sempre conosciuto dal client e non sarà passato quando viene impiegata una posizione GPS fissa
pageKey (opzionale)	string	Se esistono molti risultati, il server può richiedere la pagian succesiva tramite ***pageKey***
oauth_consumer_key (opzionale)	string	Lo sviluppatore può scegliere di inviare questo valore
oauth_signature_method (opzionale)	string	L'algoritmo impiegato sarà HMAC-SHA1
oauth_nonce (opzionale)	string	Nonce di sicurezza
oauth_version (opzionale)	string	Versione OAuth impiegata
oauth_signature (opzionale)	string	Firma HMAC-SHA1 realizzata utilizzando la Signature Base String come testo ed il Consumer Secret (quindi nessun token) come chiave
oauth_timestamp (opzionale)	integer	Timestamp della richiesta

Tabella 2.3: Parametri della richiesta GetPointOfInterest.

Nota:

- Potrebbero essere aggiunti nuovi parametri della richiesta nelle future versioni di Layar Reality Browser. Pertanto, è consigliabile per lo sviluppatore ignorare parametri ignoti e non sollevare eccezioni, di modo che non debba modificare il proprio servizio dopo ogni nuova release.

- Per garantire la genuinità della richiesta è raccomandato utilizzare un controllo sulla stringa `developerHash` o effettuare una richiesta di autenticazione OAuth. In questo modo, solo le richieste dal server Layar vero e proprio possono essere filtrate e servite.

2.3.2 Risposta

2.3.2.1 Formato della riposta

L'applicazione accetta solo risposte in formato JSON: è perciò importante che le intestazioni della risposta HTTP dichiarino il tipo corretto di contenuto (`application/json` o `text/javascript`). Le risposte dovranno essere codificate in UTF-8 aggiungendo la stringa `charset = UTF-8` ottenendo un'intestazione HTTP come:

```
Content-type text / javascript; charset = UTF-8
```

JSON (JavaScript Object Notation) è un formato per lo scambio di dati che utilizza le convenzioni derivate dalla famiglia dei linguaggi di programmazione C.

JSON è costruito su due strutture di dati universali:

- una collezione di coppie (nome, valore) rappresentate come un oggetto definito dall'utente;
- un elenco ordinato di valori rappresentati da una matrice.

I separatori dati JSON sono identici a quelli utilizzati dai motori JavaScript per rappresentare strutture di dati quali stringhe e array. Questo consente un più semplice accesso ai dati di quello che potrebbe essere raggiunto con XML.

Convenzioni generali I parametri contenuti nella risposta devono sottostare ad alcune convenzioni, al fine di essere interpretati correttamente dall'applicazione client:

- formato delle date: tutti i timestamp sono numeri interi, rappresentati in millisecondi dal 01/01/1970 UTC. Sono attualmente utilizzati al solo fine di confrontare i valori temporali;
- formato dei colori: tutti i colori sono numeri interi (base-10), basati su RGB: 0xRRGGBB dove RR sono 2 byte di codice esadecimale per il rosso (0xFF massimo), GG è il codice a 2 byte per il verde e BB sono 2 byte di codice per il blu: così, un puro colore giallo è rappresentato come 0xFFFF00 che si traduce nel numero intero 16776960;
- formati delle stringhe: tutte le stringhe sono codificate secondo UTF-8: ogni byte o carattere che non sia ASCII o una cifra deve essere convertito in *%HH;*
- formato delle stringhe nella BIW:
  - fatta eccezione per il `title`, che può essere distribuito su più righe, tutte le altre stringhe devono occupare una singola riga nel client;
  - se c'è una possibilità che le stringhe restituite possano essere troppo lunghe, è necessario assicurarsi di ripetere le informazioni contenute in esse in una pagina di dettagli che l'utente può selezionare: in questo modo qualsiasi testo troncato può ancora essere letto;

- la sequenza dei caratteri speciali `%distance%` può essere utilizzata ovunque e sarà sostituita nel client con l'effettiva distanza tra l'utente ed il POI selezionato. Ad esempio, la stringa "%distance% da qui" verrà visualizzata "123km da qui" sul client, se l'utente ha scelto di utilizzare il sistema metrico, altrimenti sarà visualizzato il valore in miglia;
- Layar è stato sviluppato per lavorare su macchine client differenti, quindi non è possibile garantire una lunghezza delle stringhe che sia valida su tutte. Tutti i client possiedono attualmente, una schermata di larghezza di 320 pixel e una larghezza della casella di testo nel BIW di 180 pixel. Le lunghezze massime consigliate (compresi spazi e caratteri speciali), che garantiscono l'ottimalità della visualizzazione sulla maggior parte dei client sono:
  - `title`: 60 caratteri;
  - `line2`, `line3`, `line4`: 35 caratteri (senza wrapping);
  - `attribution`: 45 caratteri (senza wrapping).
- la formattazione HTML non è supportata: come detto in precedenza ogni stringa dovrà essere formattata secondo UTF-8.

Risposta JSON Il messaggio di risposta inviato dal web-service deve contenere i seguenti parametri:

- `hotspot` (array di punti di interesse): l'elenco dei POI per la posizione e strato corrente;
- `morePages` (boolean)(opzionale): indica se più pagine di risultati possono essere visualizzate (si veda in seguito);
- `radius` (integer)(opzionale): il valore del raggio deve essere restituito se la richiesta `getPOI()` non ne contiene un altro (raggio flessibile richiesto); non può invece essere utilizzato per sovrascrivere un valore di raggio che è stato fornito nella richiesta;
- `errorCode` (integer): nel caso si verifichi una condizione di errore, questo verrà notificato utilizzando il codice di errore. Si noti che in questo caso, il server risponde realmente, a differenza di una situazione di errore HTTP 500 (errore server). Il valore 0 = "OK" corrisponde ad un'operazione corretta. E' consigliato utilizzare un qualsiasi codice di errore tra 20-29 per la restituzione di un messaggio d'errore. La stringa di errore verrà visualizzata come messaggio sullo schermo dell'utente: questo però verrà visualizzato solo se è nella prima pagina nella risposta, gli errori inviati in qualsiasi pagina successiva verranno invece ignorati. Dalla terza versione del software, è stato inserito un `errorCode` speciale e non modificabile: "30: Questo livello richiede l'autenticazione.";
- `errorString` (stringa): la stringa che descrive l'`errorCode` associato. Questa deve contenere massimo 80 caratteri, oltre i quali verrà troncata;
- `nextPageKey` (stringa)(opzionale): la chiave per la pagina restituita. Essa viene passata come parametro per una richiesta successiva se l'utente richiede di visionare un'altra pagina di risultati (si veda in seguito);
- `layer` (stringa): il nome del livello per il quale i POI vengono restituiti.

Paginazione Quando l'utente accede ad uno strato attende che lo schermo sia popolato dai POI: è quindi importante restituire i risultati il più rapidamente possibile. L'API offre la possibilità di suddividere i risultati in pagine al fine di accelerare il processo di visualizzazione. In genere sono riportati circa 10 risultati per pagina.

Se il Layar Service Provider utilizza la paginazione per i risultati, esso può lasciare conoscere al server Layar l'esistenza di più risultati impostando il parametro `morePages` su `true` e passando un valore corretto per `nextPageKey` (questa è una stringa qualsiasi, può essere un numero di pagina o una chiave), ed il server la utilizzerà nella sua richiesta successiva se vuole recuperare una nuova pagina di risultati.

Nota:

- Su Android, solo i primi 90 POI restituiti sono visualizzato sul client. Su iPhone, tale valore è 100.
- E' consigliato restituire non più di 50 punti di interesse entro il raggio di ricerca; se questo è molto ampio, esso conterrà molti POI; a seconda della situazione, sta allo sviluppatore restituire solo i 50 POI più vicini o i 50 POI più rilevanti.
- L'impiego della paginazione è consigliato quando si hanno più di 15 risultati.

Visualizzazione Il client visualizzerà solo i POI restituiti nella risposta, che generalmente rientrano nel range specificato dall'utente (o in un raggio di 1500m dall'utenza). La distanza tra il client e il POI viene calcolata utilizzando le API fornite dal sistema operativo del client e può essere visualizzata tramite la stringa `%distance%`. Attualmente, su Android viene impiegato il metodo dell'ellissoide WGS 84, che può condurre a risultati leggermente diversi rispetto alla formula di Haversine che utilizza una sfera perfetta. La differenza tra i due metodi è però trascurabile per Layar, a causa dei valori ridotti normalmente utilizzati [17].

2.3.3 L'oggetto Point of Interest

Ogni POI è caratterizzato da una serie di parametri che lo identificano univocamente. La seguente tabella riporta ogni chiave descrivendone la funzione ed il tipo: ciascuna è obbligatoria e se non vengono utilizzate, deve essere passato un valore `NULL` per le variabili numeriche e di testo, un array vuoto per le azioni, una stringa vuota per le stringhe di testo della BIW.

Chiave	Valore	Descrizione
id	string	Un ID univoco per il POI all'interno del livello.
distance	decimal	La distanza del punto dalla posizione corrente.
title	string	Il testo per la prima riga della BIW.
type	integer	Il tipo di POI, da cui il client determina quali CIW visualizzare.
lat	integer	Il valore intero della latitudine del luogo di interesse; è necessario dividere per 10^6 per ottenere il valore della coordinata GPS in decimale.
lon	integer	Il valore intero della longitudine del luogo di interesse; è necessario dividere per 10^6 per ottenere il valore della coordinata GPS in decimale.
actions	array	Un elenco di URI che possono essere invocati dall'utente finale.
attribution	string	Ultima linea di testo della BIW.
line2	string	Seconda linea di testo nella BIW.
line3	string	Terza linea di testo nella BIW.
line4	string	Quarta linea di testo nella BIW.
imageURL	URL	L'indirizzo dell'immagine da visualizzare nella BIW. Le dimensioni massime sono di 100x100 px; le immagini con valori superiori verranno automaticamente riscalate ma ciò rallenterà il browser. Se la dimensione è superiore a 100 KB non saranno caricate.
dimension	integer	Tale valore sarà discusso in seguito.
object	json dictionary	Tale valore sarà discusso in seguito.
transform	json dictionary	Tale valore sarà discusso in seguito.
alt	integer	Tale valore sarà discusso in seguito.
relativeAlt	integer	Tale valore sarà discusso in seguito.
doNotIndex	boolean	Esclude il POI dall'essere indicizzato dall'applicazione.
inFocus	boolean	Imposta il focus sull'hotspot non appena viene visualizzato (massimo uno).

Tabella 2.5: Caratteristiche di un oggetto POI.

ImageURL e Action URL Rappresentano le caratteristiche più generiche visualizzabili ed eseguibili per ogni POI. E' opportuno assicurarsi che l'`imageURL` e l'action URL inizino con `http://` o `https://` (Hypertext Transfer Protocol over Secure Socket Layer) o che essi siano URI validi, in accordo con l'RFC 1738. A tal proposito è bene sottostare alle seguenti condizioni:

- gli spazi devono essere codificati utilizzando "%20" e non il simbolo "+" come in HTTP, dal momento che non viene accettato da Layar attualmente;

- i caratteri speciali, ad esempio caratteri cinesi o arabi, o i caratteri riservati come `;/?:@=&` dovrebbero essere codificati;
- caratteri alfanumerici, `$-_.+!*'()`, caratteri speciali e caratteri riservati utilizzati per fini riservati possono essere utilizzati in chiaro all'interno di un URL.

2.3.3.1 Azioni

Azioni per audio e video Gli URI `audio://` e `video:// sono` specificamente destinati per le azioni. Se uno sviluppatori desidera visualizzare un video da una pagina web dovrebbe inserire un normale link `http://`. L'impiego di questi URI serve per indicare all'applicazione che i file target dell'azione sono documenti dal contenuto audio/video o semplicemente audio. Il client Layar, effettivamente, sostituisce `audio://` e `video://` con `http://` quando esegue l'accesso alla risorsa, ma in questo caso conosce già l'applicazione da utilizzare per gestire il documento (non viene aperta la WebView).

Video: Tutti i formati supportati nativamente dal dispositivo lo sono anche dall'applicazione. Al fine di garantire la compatibilità cross-device, MPEG4 e 3GP sono i migliori formati indirizzabili (risoluzione 480x320). Assicurarsi inoltre che il video sia codificato per lo streaming HTTP.

Audio: Tutti i formati supportati nativamente dal dispositivo lo sono anche dall'applicazione. Per la compatibilità cross-device, sono raccomandati i file MP3. Per i frammenti audio sul telefono cellulare, uno stream rate di 96kbit dovrebbe essere sufficiente a garantire una buona qualità di riproduzione (massimo 128kbit, oltre tale valore non è garantito il funzionamento).

L'URI layar Qualora un'azione volesse aprire un nuovo strato, è disponibile un link speciale apposito per i servizi di Layar Reality Browser: `layar://`; un link di questa tipologia restituisce effettivamente il controllo all'applicazione omonima. L'uso di quest'URI permette di trasmettere messaggi al processo Layar:

- `layar://<layername>,` è lo schema URL supportato dalla versione 2 del browser e permette di aprire lo strato `<layername>`;
- `layar://<layername>/?action=refresh`, questo link riporta l'utente al livello nell'URL (vista AR) e aggiorna i POI (nuova richiesta `getPOI`);
- `layar://<layername>/?<filter>=<value>`, apre un livello impostando direttamente i filtri specifici definiti nei parametri dell'URL. I parametri dell'indirizzo possono essere concatenati tra loro: ad esempio si potrà eseguire un refresh e risettare alcune impostazioni usando `layar://<layername>/?action=refresh&CUSTOM_SLIDER_2=540`

Tali indirizzi possono essere richiamati da applicazioni di terzi o da un browser web (su dispotivi che supportano Layar).

Trigger Ogni strato può contenere POI con azioni auto-innescanti. Per comprendere meglio questo concetto, si può pensare ad un file audio che viene riprodotto automaticamente quando l'utente entra in un edificio una pagina web che viene chiamata quando l'utente raggiunge il suo obbiettivo in una caccia al tesoro. Si tratta quindi di comportamenti che innescano la presentazione automatica di

contenuti di varia natura. L'indirizzo `layar://` funziona anche su redirect: non necessita quindi di un'interazione diretta.

L'applicazione client rispetta le seguenti convenzioni per le trigger action:

- può capitare che l'utente entri in una regione che è coperta da due o più POI self-triggered: solo quello più vicino sarà attivato in tal caso; se entrambi si trovano alla stessa distanza verrà scelto quello con l'id inferiore.
- se l'azione è `autoTriggerOnly`, non sarà visibile quando l'utente seleziona il foglio d'azione del POI al di fuori del range specificato. Quando invece viene raggiunta la posizione del POI nel range, l'azione viene attivata automaticamente (potrà comparire un pop-up di conferma);
- una volta all'interno della regione definita, l'azione sarà visibile nell'apposita finestra, consentendo all'utente di innescare l'azione manualmente;
- quando il client lascia l'area attorno al POI, definita da `autoTriggerRange`, l'azione viene disattivata e torna invisibile nuovamente.

La seguente tabella riporta i parametri per la definizione delle azioni, il loro formato ed una breve descrizione.

[17].

Parametro	**Valore**	**Funzione**
uri	URL	Gli URI possibili sono 'tel://', 'sms://', 'mailto://', video://', 'audio://', 'http://', 'https://' e qualsiasi altro URI noto sul dispositivo (intent). Per ciascuno verrà eseguita la specifica applicazione client, fatta eccezione per quelli che possono essere visualizzati direttamente in Layar (ad esempio http, audio, video).
label	string	Se specificato, il testo verrà visualizzato nel pulsante corrispondente per l'azione. In caso contrario, l'applicazione visualizza un testo di default a seconda dello schema URI utilizzato.
autoTriggerRange (opzionale)	integer	La presenza di questo parametro indica se questa azione possa essere autotriggered raggiungendo il POI. Si noti che una sola azione può essere autotriggetred, il client infatti utilizzerà il primo elemento nell'elenco che contiene questo parametro. Il parametro riporta la distanza in metri entro il quale l'utente deve trovarsi dal POI in modo che il trigger sia attivato.
autoTriggerOnly (obbligatorio solo se autoTriggerRange è settato)	boolean	Indica se questa azione può essere invocata anche manualmente dall'utente al di fuori del range specificato. Se vero, l'azione non verrà mostrata quando si preme il pulsante per visualizzare le azioni possibili, a meno che l'utente non sia all'interno del range specificato dal POI.

Table 2.7: Parametri di definizione delle azioni.

Esempi

Layer interattivo Con il parametro di aggiornamento nell'URL `layar://` gli sviluppatori possono gestire l'aggiornamento dei POI di un layer in modo interattivo a seconda delle azioni selezionate. Per esempio, cliccando su un POI si potrebbe permettere di rispondere a un sondaggio e dopo aver cliccato su "Fatto" nel WebView si verrà riportati al browser Layar, che aggiornerà automaticamente i POI e rimuoverà l'azione di risposta mentre l'utente osserverà i risultati del sondaggio stesso. E' anche possibile utilizzare la `userID` nella richiesta `getPOI` per salvare i sondaggi che sono stati già completati da quel particolare utente.

Layer point-to-point Ancora una volta grazie al parametro di aggiornamento, è possibile guidare l'utente da un POI ad un altro. Per esempio, si può permettere che sul client si effettui un login (discusso in seguito) e, una volta completata l'autenticazione, si restituirà una certa lista di POI. Tramite una qualche azione si potrà quindi forzare il refresh del browser e mostrare il/i prossimo/i POI.

2.3.4 Autenticazione

Gli strati visualizzati da Layar Reality Browser permettono agli utenti di completare dei processi di autenticazione, garantendo agli sviluppatori la possibilità di fornire servizi personalizzati per l'utenza. La procedura di autenticazione può basarsi sul protocollo OAuth.

2.3.4.1 Il protocollo OAuth

Quando un client effettua un'autenticazione presso un servizio web, in genere, deve fornire delle credenziali quali username e password, che, una volta verificate, autorizzano l'utente finale ad accedere a particolari servizi. Lo schema generale di verifica delle credenziali si basa però su sistemi centralizzati e sulla condivisione di informazioni con servizi di terzi, meccanismo non sempre desiderabile: questo è particolarmente vero nei social network dove è spesso possibile collegare tra loro diversi profili condividendo però informazioni riservate da un server all'altro. Il protocollo OAuth risolve quest'inconveniente.

OAuth (Open Authentication, RFC 5849) è un protocollo aperto, che permette ad applicazioni web o desktop di invocare, in modo semplice, standardizzato, sicuro ed autorizzato, le API messe a disposizione da un servizio di terzi. Tale protocollo risulta essere molto efficace verso chiamate di tipo RESTful, ed è quindi facilmente applicabile a tutte quelle applicazioni scritte in linguaggi capaci di aprire una Socket.

Il protocollo OAuth si basa essenzialmente su due principi di semplicità e sicurezza enunciati nel sito ufficiale del progetto, la cui traduzione è riportata di seguito.

1. *"OAuth è un modo semplice per pubblicare e interagire con dati protetti. Allo stesso tempo è anche un modo più sicuro per consentirne l'accesso ad altri."*

2. *"Gli utenti non dovrebbero diffondere la propria password nel web per ottenere l'accesso ai propri dati. Utilizzando OAuth si può permettere l'accesso ai propri dati riservati, proteggendo però le loro credenziali di account."*

Prima dell'utilizzo di tale protocollo, deve essere instaurato un canale fiduciario tra il servizio e l'applicazione che vuole interagirvi. Ciò avviene in primo luogo tramite accordi out-of-band, ed in seguito attraverso lo scambio di un set chiavi elettroniche che rappresentano le credenziali per lo sviluppatore, denominate consumer key e consumer secret. Si noti che queste ultime non sono

vincolate a nessun algoritmo di crittografia in particolare e possono essere costruite in maniera del tutto arbitraria.

Ottenute le informazioni d'accesso, l'applicazione potrà richiedere un request token al server: quando un utente vorrà autorizzare l'applicazione ad utilizzare i propri dati protetti, quest'ultima reindirizzerà il client, tramite una chiamata request_auth, al server del servizio; una volta autenticatosi, l'utente convaliderà i permessi necessari alla consumer application ed ogni qualvolta l'applicativo vorrà accedere ai suoi dati, dovrà prime scambiare il proprio request token con un nuovo access token fornitogli dal servizio (il server riceverà una signature base string contenente tutte le informazioni necessarie).

OAuth definisce quindi tre tipologie di credenziali: *client*, *temporary* e *token*, ciascuno dei quali comprende un identificatore univoco e una chiave segreta (shared secret); le prime rappresentano quelle ottenute out-of-band tra client e server prima di qualsiasi istanziazione del protocollo, le credenziali di tipo temporary sono invece realizzate in fase di bootstrapping del protocollo mentre le token credential garantiscono l'accesso alle risorse protette una volta conclusa l'autenticazione OAuth.

Questo protocollo garantisce la sicurezza dei dati utente i quali non vengono mai trasmessi da e all'applicazione; inoltre è possibile utilizzare meccanismi di cifratura e autenticazione (come timestamp e nonce) dei messaggi, al fine di aumentare il livello d'integrità della comunicazione.

In ultima analisi è opportuno evidenziare che OAuth soffre di alcune problematiche tra cui la necessità di mantenere il consumer secret incorporato nel codice dell'applicazione, compromettendone la sua segretezza [4],[18].

2.3.4.2 Richieste firmate OAuth in Layar

Gli sviluppatori possono scegliere di ricevere richieste firmate dagli utenti del proprio strato. La firma deve essere conforme alle specifiche OAuth ed i parametri per questa possono essere impostati utilizzando il Layar Provisioning. Attualmente HMAC-SHA1 è il solo metodo di autenticazione supportato dall'applicazione.

La Signature Base String è generata tramite:

- il metodo HTTP GET;
- l'URL della richiesta;
- i parametri normalizzati di richiesta, compresi i parametri di richiesta OAuth, escluso `oauth_signature`.

I parametri della richiesta OAuth utilizzati dalle API (e riportati precedentemente in Tabella 2) sono: `oauth_consumer_key, oauth_signature_method, oauth_timestamp, oauth_nonce e oauth_version.`

Autenticazione e developer key Tutte le richieste di autenticazione di un layer conterranno come parametro un hashing della developer key ricevuta dallo sviluppatore quando ha generato il proprio account. La chiave di sviluppatore può essere rigenerata in qualsiasi momento sul sito Layar Provisioning: l'operazione non sarà istantanea in quanto esisterà un lasso di tempo tra il momento di revoca della vecchia chiave e il momento in cui sarà possibile configurare quella nuova sul server; la richiesta di una nuova chiave inoltre renderà il layer irraggiungibile agli utenti per un breve periodo.

La `developerKey` viene inserita insieme alla `developerId` nel campo corrispondente delle impostazioni del client. L'hash si ottiene nel seguente modo:

- si aggiunge il timestamp come stringa alla developer key;
- si esegue la codifica ASCII della stringa;
- si calcola l'hash SHA-1 della stringa così ottenuta.

Ad esempio:

```
developerKey=492ac3114,
timestamp=1242207092843
Key-to-hash: 492ac31141242207092843
developerHash: 9df7273b410761f74331bde746e5c2354b73b487
```

2.3.4.3 Il processo di autenticazione

Alcuni layer possono includere un campo `authURL` nella definizione per indicare il supporto per l'autenticazione: tale campo riporta il login URL verso cui l'applicazione dovrebbe indirizzare il WebView. Tale indirizzo deve essere nello stesso dominio del `getPOI` URL definito (il provider del layer dovrebbe impostare un cookie).

Si possono distinguere quattro metodologie per avviare il processo di autenticazione:

1. se il campo `authRequired` di un livello è impostato su un valore true, la pagina di autenticazione sarà richiamata automaticamente quando un utente apre il livello senza un cookie valido, mentre in caso contrario verrà inviata l'usuale richiesta `getPOI` con il cookie impostato e non sarà necessario alcun re indirizzamento alla pagina di autenticazione;
2. in risposta alla richiesta `getPOI`, è possibile indicare se l'autenticazione sia necessaria, tramite il parametro `authRequired`: se esso è falso, il client semplicemente invierà la richiesta. E' anche possibile scegliere di richiedere un login restituendo il codice d'errore #30. Anche nel caso in cui i cookie inviati non siano più validi, il provider dello strato può rispedire l'`errorCode` #30;
3. il fornitore dello strato può decidere di utilizzare il campo `AuthURL` anche all'interno di un'azione, indirizzando automaticamente l'utente alla pagina di login;
4. l'utente può avviare l'autenticazione utilizzando il pulsante speciale che punta allo `AuthURL` in un livello (se previsto). Quindi, anche se `authRequired` è falso, uno strato può ancora dare la possibilità di fornire le proprie credenziali manualmente. Quando nessun cookie è presente, il livello può restituire solo i POI pubblici per il layer d'interesse.

Una volta avviata la procedura di autenticazione l'utente finale viene reindirizzato ad una pagina nella WebView. I passaggi che seguono il redirect sono i seguenti:

1. la pagina di login viene aperta all'interno di un browser incorporato. Questa pagina permette di autenticare l'utente (ogni meccanismo può essere utilizzato, come self-hosted auth/OAuth, ecc.) e può reindirizzarlo ad una home page dinamica con le impostazioni da scegliere per il livello;
2. l'utente viene quindi riportato alla visualizzazione in cui si trovava quando il browser-embedded ha eseguito il redirect, tramite un link "`getbacktoapp`". Attraversare un link di questo tipo, significherà che l'autenticazione (e possibilmente la configurazione) dello strato è stata completata e l'utente può ritornare all'applicazione. Resta allo sviluppatore l'impostazione dei cookie che si applicano alla `getPOI` URL;

3. quando il link "Indietro" viene aperto nel browser incorporato, tutti i cookie applicati sono memorizzati nello strato (la scadenza dei cookie è gestita da Layar, secondo le specifiche cookie);

4. i cookie così memorizzati vengono inviati al Layar Service per gestire le richieste future, mentre tutti i dati ricevuti sono inoltrati al Layar Provider in modo che la richiesta possa essere identificata con una particolare sessione utente. Si noti che non è ammesso re indirizzare le richieste `getPOI`, poiché per motivi di sicurezza non è possibile inoltrare anche i cookie generati.

Per i livelli che utilizzano l'opzione di autenticazione (dal sito di Provisioning), viene visualizzato un pulsante nel livello in modo che gli utenti finali possano visitare una home page, quando sono già autenticati, per cambiare configurazione o effettuare un login [17].

Parametro	**Valore**	**Funzione**
authRequired (opzionale)	boolean	Indica se è richiesta l'autenticazione prima di aprire il livello . Se falso, il livello può essere avviato senza autenticazione. Altrimenti, il client deve verificare se esiste un cookie valido per quel livello e qualora non presente, aprire l'URL di autenticazione per consentire all'utente di autenticarsi.
authURL (opzionale)	URL	L'URL per autenticare l'utente (HTTPS preferibilmente); tale campo è obbligatorio se ***authRequired*** è vero, ma può essere presente anche se ***authRequired*** è falso: in tal caso, sarà possibile per l'utente caricare il livello senza autenticazione e avviare manualmente l'autenticazione una volta caricato il livello. Si noti che ***AuthURL*** deve risiedere sullo stesso dominio (o in un sotto dominio) del ***poiURL*** (poiché si opera sui cookie, vanno applicate tutte le regole per cookie del browser).

Table 2.9: Parametri per la definizione di servizi di autenticazione.

2.3.5 Strati tridimensionali

A partire dalla versione 3.0 di Layar Reality Browser, è possibile per gli sviluppatori aggiungere modelli tridimensionali ai propri livelli. A tal fine è stato definito il formato Layar3D (. l3d), che consente la memorizzazione di texture in un unico file. Questo formato è ottimizzato per l'analisi e la visualizzazione su dispositivi mobili.

In generale, si possono definire due tipologie di strati e le caratteristiche di ciascuna possono essere riassunti come segue:

1. *Generico*: i POI sono rappresentati da CIW, che possono essere definiti nella scheda "POI icons" nella definizione del layer; la tipologia di un POI viene definita da un numero che dovrebbe essere attribuito al campo `type` di un hotspot. Per un punto generico i valori `object` e `transform` (si veda in seguito) non sono necessari per la risposta JSON. Come opzione di default la dimensione del POI è fissata a 1. Gli strati generici rappresentano quelli realizzati sino alla versione 2 del client;

2. *Oggetti 2D e 3D nello spazio*: i POI sono rappresentati da immagini personali 2D o modelli a tre dimensioni; anche in questo caso possono essere definite CIW personalizzate: queste icone

verranno utilizzate nel caso gli oggetti non vengano caricati correttamente dall'applicazione. I parametri `transform` e `object` sono obbligatori nella risposta JSON; per un POI bidimensionale il valore del campo `dimension` deve essere impostato a 2 mentre a 3 per i modelli 3D.

Sorge spontaneo chiedersi quale sia la differenza tra un layer generico ed uno che utilizza solamente icone bidimensionali nello spazio. La prima sostanziale differenza è insita nella loro definizione: ogni set di CIW contiene quattro icone che vengono utilizzate per rappresentare i diversi stati dei punti di interesse ed ogni icona ha una dimensione fissa, diversamente inun oggetto 2D vi è una sola immagine, ma su di esso possono essere operate delle manipolazioni, come ridimensionamento o rotazione. La seconda differenza è relativa al loro impiego: i CIW semplici sono possono essere assegnati a vari gruppi di POI ed ognuno utilizzerà lo stesso set d'icone, mentre le immagini bidimensionali si adattano al momento rappresentativo, rispecchiando e personalizzando il contenuto e la rappresentazione di ciascun POI, ad esempio un layer turistico può visualizzare la foto di ciascuna opera o monumento.

Parametro	**Valore**	**Funzione**
dimension	integer	Icona POI semplice. Rappresenta il valore di default Utilizza un'immagine bidimensionale per il POI Utilizza un oggetto tridimensionale per il POI.
alt	integer	Altitudine per l'oggetto. Se assente verrà applicata l'altitudine del client.
relativeAlt	integer	Altitudine relativa rispetto a quella del client.
transform	json dictionary	Obbligatorio se ***dimension*** è 2 o 3. Determina come posizionare l'oggetto nello spazio.
object	json dictionary	Obbligatorio se ***dimension*** è 2 o 3.Determina la posizione dei file che definiscono l'oggetto.

Table 2.10: Parametri di definizione per oggetti tridimensionali.

2.3.5.1 Strati con oggetti

Ogni punto è associato ad una serie di parametri che ne determinano l'aspetto; se si utilizzano immagini bidimensionali o modelli a tre dimensioni è obbligatorio segnalare anche l'origine di tali elementi e le modalità di rappresentazione. Nel seguito saranno presentate brevemente le caratteristiche di questi.

Parametri POI tridimensionali I parametri riportati nella tabella sottostante, vanno inseriti opzionalmente all'interno della risposta del servizio, a patto che la chiave `dimension` sia impostata ad 1. Tali valori determineranno all'applicazione le modalità di gestione della rappresetnazione di ciascun punto.

Transform Nel caso lo strato serva contenuti bi/tri-dimensionali (e quindi se il valore `dimension` è superiore ad 1), i campi riportati nella seguente tabella sono obbligatori per la risposta: essi determinano i valori di rotazione e scalatura dei modelli associati ai diversi POI.

Il sistema di coordinate per gli oggetti bidimensionali è tale che l'immagine stessa appartenga al piano XZ (asse X orizzontale), mantenendo l'asse Y orientato verso l'esterno, sul retro dell'oggetto.

Object I parametri `object` determinano la sorgente dei dati da visualizzare, indicando una location per il modello intero ed una per quello scalato: l'applicazione infatti non esegue operazioni di resizing sugli oggetti mentre un utente si allontana/avvicina ad un hotspot,allo scopo di non rallentare l'esperienza offerta dal browser.

Ogni oggetto deve comunque indicare le dimensioni "reali" che deve possedere ed una immagine di riferimento.

Fallbacks (lato client): Se uno qualsiasi dei file dell'oggetto non può essere caricato, il client utilizza un meccanismo di fallback per recuperare e utilizzare le icone definite a livello globale per lo strato:

- se il `full` è assente, il client deve utilizzare una speciale icona che indica un file mancante. Dovrebbe però essere chiaro all'utente che il punto desiderato non è rappresentato come previsto, ovvero tramite un oggetto 3D. Il client riproverà a scaricare il file se l'errore è dovuto ad un time-out;
- mentre l'oggetto non è reperibile, è corretto utilizzare l'oggetto successivo a livello di file, se è disponibile (l'ordine prestabilito è `full - reduced - icon`);
- se nessuna delle rappresentazioni precedenti è presetabile, l'applicazione utilizzerà il CIW personalizzato del livello, se definito;
- come ultima risorsa, il client dovrà utilizzare le icone predefinite per lo strato.

Specifiche degli oggetti Di seguito sono enunciate alcune proprietà auspicabili per gli oggetti di un layer non generico.

Oggetti bidimensionali. Attualmente la dimensione di un oggetto 2D è determinata dalla sua risoluzione; questa rappresentazione è per certi versi problematica, in quanto è necessario che gli oggetti `full` e `reduced` abbiano la stessa risoluzione perché si adatti ad un contesto reale. Per la versione `full` dell'oggetto la risoluzione massima è di 640x480 pixel, mentre per la `reduced` è di 240x180 pixel. Inoltre tutte le immagini devono avere dimensioni inferiori a 75KB e preferibilmente proporzionali tra loro.

Oggetti tridimensionali. I modelli tridimensionali devono sottostare ai seguenti vincoli:

- `full`: tale modello può utilizzare al massimo 5.000 poligoni definiti da facce triangolari;
- `reduced`: quest'oggetto può utilizzare al massimo 500 poligoni definiti tramite facce triangolari;
- texture: allo scopo di elaborare correttamente le texture nel client, la larghezza e l'altezza di queste dovrebbero essere espresse come potenze di 2 (fino a 512) , in modo da ottenere valori validi come 256x256, 128x128, ma anche combinazioni come 512x256 sono consentite. Le texture possono essere in formato PNG o JPG;
- effetto trasparenza nelle strutture: è supportato (parzialmente) se si utilizzano texture in formato PNG.

Per le texture, in generale, gli sviluppatori sono liberi di utilizzare un qualsiasi software di modellazione 3D (ad esempio Blender) e realizzare i propri modelli, compresi di texturing. Ogni modello deve essere esportato nel formato wavefront: durante l'operazione verranno creati un file OBJ e uno MTL. Il file MTL fa riferimento ai materiali utilizzati nel modello e ogni materiale può riferirsi a diverse texture. Le coordinate UV saranno invece conservate nel file OBJ.

Durante la conversione al formato L3D, tutte le immagini delle texture vengono incorporate nel file, in modo da ottenere un unico modello che può essere utilizzato dal servizio; infine è consigliabile raggruppare gli elementi in base ai materiali che li compongono, se il software impiegato supporta questa funzione: ciò permette che l'oggetto venga disegnato utilizzando un minor numero di chiamate OpenGL, migliorando notevolmente le prestazioni del client.

2.3.5.2 Modellazione di oggetti 3D

In questa sezione si descrivono brevemente le basi per il rendering di un modello tridimensionale, e successivamente gli impieghi e le caratteristiche del Model Converter per Layar Reality Browser.

Prima di procedere, è però opportuno introdurre il sistema di coordinate, utilizzato da Layar, cui gli oggetti saranno riferiti; in esso la mappatura con il mondo reale è realizzata utilizzando le seguenti convenzioni:

- asse X: orientato da ovest a est,
- asse Y: orientato da sud a nord,
- asse Z: orientato dal basso verso l'alto.

Un modello a tre dimensioni è composto, come noto, da un insieme di vertici, spigoli e facce. Esistono diverse metodiche per descrivere un solido nello spazio, ad esempio si possono utilizzare i quadtree, b-rep, manifold o ancora la rappresentazione winged-edge; il motore grafico di Layar Reality Browser implementa un modello descrittivo "face-vertex mesh", in cui i modelli sono rappresentati dai propri vertici (con le loro coordinate) raggruppati in base alla faccia cui appartengono, così, per esempio, un cubo sarà descritto da 8 punti e una piramide da 5. Le facce, composte da 3 o più vertici, sono caratterizzate dalle proprie normali: i vettori normali indicano infatti la direzione di una faccia e quindi l'ordinamento dei vertici al suo interno.

Le facce sono gli elementi visibili di ogni modello, ma nell'applicazione solo quelle frontali sono renderizzate, limitando le operazioni di calcolo nel client. Se le normali non sono descritte nel modello, l'orientamento delle facce è determinato dall'ordinamento in senso antiorario dei vertici stessi.

Nella modellazione 3D, luci e colori possono essere applicati a ciascun modello, definendone i materiali di cui sono costituiti. Ogni ambiente di modellazione offre agli sviluppatori diversi tipi di luci, ad esempio OpenGL dispone di 7 varietà, nonchè numerosi effetti di colore. Il seguente elenco riporta gli effetti applicabili ai modelli renderizzati in Layar ed interpretati dall'applicazione:

- `diffuse`: il colore principale del materiale che viene utilizzato quando la luce si riflette su di esso;
- `ambient`: il colore del materiale che viene utilizzato per l'ambiente;
- `specular`: il colore della luce nel modello (spesso bianco);
- `shininess`: questo parametro controlla la dimensione della luce. Valori alti sono relazionati a piccoli highlights, viceversa valori bassi rendono superfici maggiori.

Una volta determinata la struttura di un elemento è possibile aggiungere, grazie al meccanismo di Texture Mapping, un ulteriore livello di dettaglio applicandovi un'immagine di superficie od un colore, detti texture; le texture rendono il modello più realistico, inoltre la sovrapposizione di queste è utilizzata per creare degli effetti speciali come riflessi o nebbia.

A tale proposito, il motore grafico di Layar supporta solamente texture semplici a colore diffuso, mentre le bump mapping e le normal map sono attualmente ignorate (e quindi non vengono visualizzate, impoverendo il modello).

Creare un modello tridimensionale

Requisiti del modello. La modellazione di elementi 3D compatibili con Layar Reality Browser, come già detto, può essere effettuata con qualsiasi applicazione a patto che questa permetta di esportare il modello in formato Wavefront (obj/mtl). Durante la fase di creazione è ben osservare di:

- mantenere la complessità del modello più bassa possibile, poichè gli attuali telefoni cellulari non possiedono le capacità computazionali per il rendering di modelli complessi ad un frame rate accettabile per l'utente;
- solo gli oggetti definiti tramite face-vertex mesh sono supportati;
- il modello dovrebbe consistere di sole facce triangolari;
- l'unità per le coordinate nel modello (la dimensione reale del versore) è impostata ad 1 metro;
- l'effeto trasparenza è supportato solo parzialmente (anteponendo diversi oggetti non sarà possibile vedere quelli retrostanti, bensì il mondo reale dietro ad essi).

Requisiti per le texture. Mediante l'applicazione di texture, può essere aggiunto ai modelli un maggior numero di dettagli senza aumentarne la complessità. Per una qualità ottimale, è bene realizzare/converite le texture in formato PNG, per ottenere un pieno controllo della qualità, mentre per motivi di prestazioni, è consigliabile ridurre la quantità delle immagini utilizzate e riunirle in un unico file di texture (MTL).

Layar non supporta pienamente l'effetto trasparenza nella visualizzazione AR, a causa di incompatibilità con l'alpha blending tuttavia, tramite la creazione di texture con parti trasparenti, è possibile ovviare a questa lacuna, realizzando elementi "see-through" nel modello: questo consente di aggiungere dettagli, senza incrementare ancora una volta la complessità del modello.

Va inoltre ricordato che il texturing è supportato solo utilizzando un colore `diffuse`, e che ciascuna texture deve essere ridimensionata in modo da avere larghezza ed altezza espresse tramite potenze di 2; risulta conveniente mantenere immagini di piccole dimensioni per le texture, per non incidere negativamente sui requisiti di banda e soprattutto sulle prestazioni del client.

Creare un modello con texture animate Utilizzando delle texture animate in un modello 3D, è possibile "renderlo vivo": ad esempio si può creare un insegna pubblicitaria con del testo scorrevole. Si noti però che a causa delle possibilità attuali dei telefoni e della connessione dati utilizzata, ci saranno dei limiti che devono essere considerati.

Requisiti delle texture animate. Poichè ogni fotogramma deve essere caricato nella memoria del telefono, non è possibile creare e visualizzare interi filmati con un'unica texture animata! Questi elementi infatti dovrebbero incorporare gif animate semplici o ad alta qualità. Nel seguito vengono riportate alcune linee guida utili alla realizzazione di modelli con animazioni:

- le dimensioni di ciascuna texture non devono superare i 256x256 pixel (anche se questo requisito non è implementato);
- la dimensione del file di ogni fotogramma dell'animazione determina lo spazio occupato in memoria e nella banda per l'utente, ed è quindi opportuno adottare un buono standard per la compressione dei file di immagine. Nel caso di file PNG, ad esempio, riducendo il numero di colori si riduce anche la dimensione del file stesso. Si noti inoltre che nel modello L3D la dimensione non può superare la soglia di 1MB;
- il numero di frame supportati da Layar Reality Browser per una texture animata, dipende dalle dimensioni (altezza e larghezza) e dalla taglia del file. Risultati ottimali possono essere raggiunti con un rate di 25 frame al secondo, ma per favorire l'esperienza dell'utente su un maggior numero di dispositivi è consigliabile mantener un rate di 10 fps. Questa considerazione è particolarmente importante se più modelli 3D (con animazione) sono visibili allo stesso tempo.

Tutti i file L3D con un'animazione, verranno visualizzati solo da client che utilizzano la versione 3.5, o superiore, del browser Layar. Se gli sviluppatori desiderano presentare un modello tridimensionale ad utenti che usufruiscono di versioni antecedenti alla 3.5, essi dovranno caricare un ulteriore file L3D di sole immagini statiche, e utilizzare il parametro `version` della richiesta, per determinare quali file del modello servire al client.

2.3.5.3 Model Converter

Uno strumento essenziale nella costruzione di layer tridimensionali è senz'altro il Model Converter di Layar: esso è implementato in linguaggio Java e può essere eseguito su qualsiasi sistema. Il vantaggio di questa soluzione è che permette di mantenere una singola build per diverse piattaforme, facendo impiego di librerie native per facilitare l'anteprima dei modelli utilizzando OpenGL nel convertitore.

Tra le funzionalità del software vi ne è anche una che permette la conversione da Wavefront a L3D. Infatti il Model Converter utilizza oggetti nel formato file Layar3D (discusso precedentemente): questo si basa su Wavefront, ma contiene una rappresentazione dell'oggetto ottimizzata (OBJ) per essere gestita in uno smartphone.

Solitamente i modelli consistono di un singolo file OBJ, contenente i vertici e le facce che lo descrivono, un file MTL, contenente i riferimenti a file delle texture, ed i file PNG o JPG contenenti le immagini di texture. Nel caso uno qualsiasi dei file riferiti sia mancante, il convertitore mostrerà un messaggio di errore.

L'applicativo offre agli sviluppatori anche funzioni di editing dei modelli:

1. *drop della normal*e: ricalcola le normali sui vertici smussati (smooth), utilizzando la media delle normali su tutte le facce cui il vertice appartiene;
2. *calcolo della normale*: assegna ai vertici una normale per ciascuna delle facce cui appartengono;
3. *flip delle facce*: cambia l'ordine dei vertici di ogni faccia, "rovesciando" il modello;

4. *ottimizzazione dei materiali*: riordina le facce per materiale, migliorando la velocità di rendering;

5. *rotazione*: ruota il modello intorno all'asse X, con un angolo compreso tra 90° e -90°;

6. *scalatura*: ridimensiona il modello secondo un fattore specificato.

Aggiungere animazioni. Per poter inserire le animazioni all'interno dei modelli, è necessaria l'ultima versione del Model Converter (v. 2.1.1). Il supporto iniziale per l'animazione texture è stato infatti aggiunto nella versione 2.0, ma esso si limitava a caricare un'immagine GIF; la versione 2.1 del software contiene invece numerosi miglioramenti nella creazione di animazioni, tra cui la possibilità di realizzare quest'ultime all'interno del Model Converter stesso, consentendo di aggiungere singoli fotogrammi senza dover prima generare una GIF animata.

L'elenco seguente riporta le funzionalità introdotte con la versione 2.1, le quali permettono di:

1. cambiare texture: sulle scheda "Material" del convertitore, è possibile vedere tutti i materiali contenuti nel modello e le texture che vi fanno riferimento. Per ciascuno si presentano tre opzioni:

   (a) -: nessuna texture viene assegnata al materiale. Durante il rendering, solo i colori `diffuse/ambient/specular` sono utilizzati ;

   (b) *static*: applica una texture fissa scelta dallo sviluppatore;

   (c) *animated*: applica un'animazione in loop, cui è permesso di aggiungere frame multipli.

   Quando si cambia il materiale di una texture, il convertitore cercherà di mantenere la struttura esistente: se, ad esempio, si cambia una texture da statica ad animata, la prima verrà impiegata come fotogramma iniziale dell'animazione. Nel caso in cui non vi fosse alcuna texture in precedenza, verrà visualizzato un dialog box dove potrà essere selezionare l'immagine da utilizzare nella trama. Si nnoti che quando un materiale non ha una trama assegnata, aggiungerne una nuova, non sempre conduce a risultati corretti: il file OBJ dovrebbe contenere le coordinate della nuova texture, altrimenti il convertitore non saprà come posizionarla nel modello, generando una situazione d'errore. Per evitare questo problema, è sufficiente aggiungere una texture statica ad un oggetto ed eventualmente modificarla nel convertitore.

2. aggiungere e rimuovere fotogrammi: quando si è impostato un materiale per contenere una texture animata, la finestra per l'anteprima di questa conterrà alcune opzioni di controllo che possono essere utilizzate per modificare l'animazione; tali funzionalità sono:

   (a) *add*: Aggiunge uno o più fotogrammi alla texture;

   (b) *up*: Muove il fotogramma selezionato in una posizione antecedente nell'elenco dei frame;

   (c) *down:* Muove il fotogramma selezionato in una posizione successiva nell'elenco dei frame;

   (d) *remove:* Elimina i frame selezionati dall'animazione corrente (e non l'animazione stessa);

   (e) *replace*: Sostituisce l'immagine utilizzata nella texture.

3. regolazione delay: la quantità di fotogrammi in una texture animata è limitata, ed esiste il rischio che sembri relativamente statica. Modificando il ritardo di visualizzazione tra i frame, è possibile controllarne il tempo di visibilità per ciascuno. Ogni valore è riportato in millisecondi;

4. anteprima: l'animazione può essere visualizzata in anteprima nel convertitore di modelli (per abilitarla, usare "View > Enable animation").

[11],[17].

Nota: All'inizio del terzo trimestre del 2010, Layar ha annunciato una collaborazione con Kooaba (`http://www.kooaba.com/`) una piattaforma per Realtà Aumentata incentrata sul riconoscimento di immagini. Questa partnership potrebbe portare in un futuro all'integrazione di sistemi per il riconoscimento di marker e codici QR, permettendo così agli utenti finali di utilizzare Layar Reality Browser anche in ambienti indoor eliminando la dipendenza assoluta dalla componente GPS.

Parametro	Valore	Funzione
baseURL	url	L'URL dove è conservato l'oggetto.
full (obbligatorio se la dimensione è 2 o 3)	string	Rappresenta l'oggetto visualizzato dall'utente, quando si trova entro 50m da questo. Il suo valore determina il percorso relativo al file, al fine di recuperare l'oggetto.
reduced (opzionale)	string	Percorso veros la rappresentazione dell'oggetto nel range di 50-100m dala posizione utente. (L'oggetto deve essere riscalato).
icon(opzionale)	string	La rappresentazione 2D dell'oggetto quando esso è al di fuori dei range previsti (o quando non è disponibile un modello 3D). Il file deve essere in formato PNG e di dimensioni 32x32 pixel. Può essere usato anche come alternativa al parametro type.
size (obbligatorio se la dimensione è 2 o 3)	float	Le dimensioni dell'oggetto in metri. Questa è la lunghezza dello spigolo del più piccolo cubo cui l'oggetto può utilizzare. Si noti che questa è la dimensione prima di qualsiasi trasformazione. Per oggetti 3D, questo parametro è utilizzato dal client per stabilire se può passare dalla visualizzazione a 'icona', a quella 'ridotta' o a quella 'completa'.
rel	boolean	Se questo valore è vero, la rotazione è calcolata rispetto alla posizione dell'utente: l'oggetto sarà sempre rivolto verso l'utente indipendentemente dalla direzione assoluta. Inoltre l'angolo sarà impostato a zero: l'utente non sarà quindi in grado di camminare intorno all'oggetto, al fine di visualizzarne il retro.
angle	decimal	Quando si progetta un oggetto, l'asse x è allineato nella direzione Est-Ovest, mentre l'asse y è allineato in direzione Nord-Sud. Questo parametro esprime l'angolo di rotazione, in gradi, per ruotare l'oggetto attorno al suo asse z. Il verso di rotazione è determinato dalla regola della mano destra, ovvero un angolo positivo implica una rotazione in senso orario nel piano xy. Così, per esempio ruotare di -90°, significa che l'oggetto sia orientato verso Est: un utente che guarda in questa direzione può vedere l'oggetto come è stato progettato.
scale	decimal	Le dimensioni degli oggetti è determinata dagli stessi (l'unità del sistema di coordinate è un metro). Per gli oggetti 2D, 200 pixel equivalgono ad 1m.

Table 2.11: Parametri per la visualizzazione dei modelli tridimensionali.

Capitolo 3

Location-based layer application

Oggigiorno le tecnologie mobili e di geo localizzazione possono contare su una consistente e funzionale dotazione hardware, ormai contenuta in quasi tutti i telefoni cellulari prodotti per il mercato: sistemi operativi, fotocamera ad alte prestazioni, larga banda wireless UMTS o HDSPA, ricevitore GPS assistito, accelerometro, bussola, ecc.; non può quindi sorprendere che software come Layar Reality Browser si stiano diffondendo nel mondo mobile. Esistono diverse applicazioni per iPhone e Android appartenenti a questa categoria, molte delle quali devono ancora essere approvate o rilasciate.

Gli ambienti, gli approcci ed i contesti applicativi sono innumerevoli e, per ottenere una migliore comprensione di questa nuova tendenza e fenomeno, nel seguito verrano discussi e presentati alcuni di questi software, detti location layer application, ponendoli a confronto tra loro. Tale panoramica sarà incentrata maggiormente sui livelli d'interazione dell'utente finale, sulle informazioni ed i contenuti ad essi proposti, e sul piano degli sviluppatori per quanto riguardo la facilità e le possibilità di programmazione, piuttosto che sulle performance: quest'ultimo confronto, oltre che dipendere dal terminale di prova, non è operabile in quanto non esiste un vero termine di paragone.

3.1 Applicazioni

Acrossair , come Layar Reality Browser, è un browser gratuito per Augmented Reality Location Based Application con numerose funzionalità. Questo software presenta un'interfaccia utente molto semplice ed intuitiva: una volta aperta l'applicazione, l'utente finale potrà subito accedere a diversi contenuti, dalla ricerca di locali e servizi (perfettamente suddivisi in categore come alberghi, ristoranti, cinema,...), alla visualizzazione di contenuti Foursquare, Panoramio, Twitter, Wikipedia e Youtube. La ricerca avviene utilizzando quattro tra i maggiori motori di ricerca attualmente disponibili: Google, Yelp!, Qype e Bing. L'utente sarà anche in grado di condividere la propria posizione o anche contenuti feed RSS. Infine è disponibile una funzionalità "Trova auto" (già utilizzata in altri software specifici come "cAR finder") che permette all'utente di salvare la posizione del proprio autoveicolo e successivamente ritrovarlo facilmente.

Ogni contenuto è associato ad un'azione prestabilita come la visualizzazione di una pagina YouTube o di una foto Panoramio, a cui vanno aggiunte funzionalità di sharing, geotagging, bookmarking e routing.

I punti possono essere visualizzati attraverso diverse modalità: visualizzazione AR, mappa e lista; nella visualizzazione AR gli utenti potranno interagire con ciascun POI, oltre che tramite azione, scorrendo l'insieme dei risultati della ricerca: i punti non sono difatti posizionati all'interno

della finestra utente secondo le loro coordinate GPS, ma piuttosto vengono raccolti e presentati in un'area circostante all'utente come un insieme di pannelli scorrevoli (mantenendo inizialmente un ordinamento geografico).

L'applicazione è sicuramente interessante disponendo di una vasta gamma di contenuti, purtroppo però non fornisce un supporto a possibili sviluppatori per servizi di contenuto.

AroundMe , è un'applicazione per iPhone gratuita che ha anticipato in qualche modo il concetto di Augmented Reality. Sfruttando la connettività dello smartphone, i servizi di localizzazione e un database molto vasto, AroundMe fornisce informazioni immediate su POI vicini all'utente. I punti d'interesse sono indicizzati e catalogati per categorie e, come per le altre applicazioni, AroundMe permette di osservare il tutto su una mappa e creare un itinerario dalla posizione corrente al luogo desiderato. Quest'opzione, unitamente con la possibilità di postare su Twitter e Facebook, rendono AroundMe un'applicazione davvero completa.

Con la versione 3.5 Around Me aggiunge alle sue funzioni anche la visualizzazione AR: per aprirla l'utente dovrà accedere alla mappa, eseguire un "tap" e ruotare il proprio dispositivo in modalità landscape (meccanismo poco intuitivo). La versione Augmented Reality di Around Me necessita ancora di qualche accorgimento per poter essere al livello di Layar, uno su tutti, il rendere sensibili i POI direttamente dalla visualizzazione AR: una volta individuato il POI l'utente non potrà interagirvi e dovrà ritornare sulla mappa o alla lista dei punti.

Sul piano dell'interattività invece quest'applicazione offre diverse possibilità, al pari di Layar Reality Browser: sarà possibile infatti telefonare, inviare sms ed email, aggiungere un luogo tra i preferiti e condividerlo con altri utenti, senza contare la già citata possibilità di postare sui propri social network. In aggiunta, AroundMe integra Navigon Mobile Navigator (se installato sulla macchina client), offrendo all'utenza la capacità di tracciare il proprio percorso non solo in Google Maps, ma direttamente sul navigatore.

Per quanto riguarda il lato developer, AroundMe non lascia possibilità di sviluppare strati personalizzati.

BionicEye , è un'applicazione a pagamento sviluppata per iOS che visualizza punti d'interesse (negozi, ristoranti, alberghi, hotspot, stazioni dei mezzi pubblici, ecc.) contenuti in un database offline. I POI sono corredati di informazioni testuali ed immagini (ad esempio l'insegna del negozio) ed integrati con le funzionalità Google Maps. Una caratteristica interessante di questo software, è quella di visualizzare graficamente, nella visualizzazione AR, il percorso verso un POI specifico tramite la sovraimposizione di immagini (frecce orientate).

Questo software non necessita di alcuna connessione di rete per reperire le informazioni da visualizzare all'utente, in quanto queste sono scaricate assieme all'applicazione durante la prima fase di setup. Tale caratteristica è in un certo qual modo, limitante per gli utenti sia dal punto di vista dello spazio su disco, che per quanto concerne l'affidabilità delle informazioni (il database sarà aggiornato solo ad intervalli periodici tramite il software updater del telefono). Inoltre la piattaforma è chiusa ai potenziali developer, andando ad influire negativamente sulla diffusione di questo prodotto che ad oggi è disponibile solo per Stati Uniti, Inghilterra, Francia e Giappone.

In contrasto Layar, nonostante richieda una connessione Internet aperta, offre un servizio sempre disponibile e adatto a diversi tipi di esigenze garantendo all'utente un alto livello d'interazione.

BionicEye non è però l'unica applicazione caratterizzata da un database interno al terminale, esistono infatti molti programmi per AR che si basano su questo meccanismo: come BionicEye, essi sono limitati ad aree ed utilizzi ristretti e molto spesso non sono rilasciati per uso gratuito.

Un esempio ne è Nearest Tube, un'app che visualizza informazioni dettagliate sulla metropolitana londinese, dalle fermate alla direzione della linea, o ancora Museum of London che funge da guida delle meraviglie artistiche di Londra.

Junaio , è un browser gratuito per Realtà Aumentata disponibile per Android ed iPhone (`http://www.junaio.com`) e basato sulle tecnologie AR di Metaio (`http://www.metaio.com`). L'utente aprendo l'applicazione viene subito portato alla visualizzazione AR da cui potrà accedere al menù "Channels": come Layar, Junaio offre diversi contenuti, chiamati channels, organizzati in un menù, secondo categorie (Featured, New, Banking, Culture,...). Ad ogni apertura dell'app viene caricato l'ultimo canale visitato. L'utenza potrà navigare tra diverse visualizzazioni (mappa e lista) ed aggiungere un canale tra i preferiti sottoscrivendo un feed RSS.

Sul piano dell'interattività Junaio è paragonabile a Layar Reality Browser, in quanto dispone di numerose azioni assegnabili a ciascun POI anche se queste si limitano ad immagini, file audio e video, mappe e web link, ed includono meccanismi di autenticazione. Inoltre Junaio permette di associare modelli tridimensionali ad ogni punto proprio come in Layar.

Per quanto riguarda il developing, l'applicazione è corredata di un'ottima documentazione e un da set di API online ben organizzate.

Due sono le caratteristiche peculiari di questo software: la creazione di un profilo utente e la localizzazione in ambienti indoor. La prima offre la possibilità per gli utenti finali di registrare un proprio profilo su Junaio che, una volta autenticati all'interno dell'applicazione, possonno condividere POI in tempo-reale tramite il browser stesso; ogni profilo inoltre potrà essere associato ad un gruppo di utenti, realizzando così una rete di contenuti. Un problema spinoso per le tecnologie di POI location, sta nella difficoltà di eseguire la localizzazioni di elementi all'interno di strutture ed edifici dove la copertura GPS è limitata o assente. Junaio risolve questo inconveniente fornendo agli sviluppatori la possibilità di eseguire una perfetta localizzazione tramite lettura di marker, detti LLA marker (soluzione proprietaria di Metaio), posizionati all'interno delle strutture. Questa soluzione permette agli utenti di usufruire del browser anche, per esempio, per orientarsi tra i padiglioni di una fiera.

RobotVision Augmented Reality , sviluppato solo per iPhone 3Gs/4, questo browser ricerca luoghi, ristoranti ed altri punti di interesse, il tutto utilizzando le potenzialità del motore di ricerca Bing: basta inquadrare con la fotocamera i luoghi nelle circostanze, perchè compaiano in sovraimpressione le informazioni su questi (sfruttando un database interno a Bing), associate ad immagini Flickr o post di Twitter. RobotVision permette all'utente di ricevere dati su zone di interesse turistico, ristoranti, locali ed altro, con tanto di recensioni e valutazioni. Inoltre orientando il telefono verso il basso, viene mostrata una cartografia con i vari punti di interesse.

A differenza di Layar questo browser è a pagamento ed è limitato al solo iOS di Apple. In aggiunta, poichè i contenuti sono visualizzati in tempo reale tramite il motore di ricerca Bing, non è possibile sviluppare strati che selezionino quali POI presentare all'utente: ciò rende l'applicazione chiusa ai possibili developer.

Per quel che riguarda l'interazione anche con quest'applicazione non sarà possibile intraprendere azioni il cui target sia diverso da un semplice web link.

Nello scenario delle location based layer application, esistono diversi software simili a RobotVision, tra cui è opportuno citare Cyclopedia; questo software permette agli utenti di visualizzare i luoghi presenti all'interno del database di Wikipedia che si trovano entro un certo raggio di ricerca (impostato manualmente). Le informazioni sono visualizzate tramite sovraimposizione con le im-

magini della videocamera o tramite mappa, e ad ogni POI è associato un web link alla pagina Wikipedia cui si riferisce.

Sekai Camera , pur non rientrando pienamente nel confronto tra browser AR, risulta un'applicazione davvero interessante. L'idea alla base di questa applicazione è quella di utilizzare iPhone come un terminale mobile d'informazione e collegare la realtà a dei contenuti realizzati dagli utenti, operando del vero e proprio social tagging; nello specifico, i POI visualizzati dall'applicazione non rappresentano luoghi d'interesse, bensì messaggi lasciati dall'utenza: Sekai Camera sfrutta infatti dei contenuti user-generated, per visualizzare brevi messaggi di testo associati ad una immagine, e chiamati "air-tags", all'interno della visualizzazione AR.

Il risultato è quello di una realtà condivisa, una forma di comunicazione aumentata che immerge l'utente in un grande social network. Ogni client potrà infatti aprire un proprio account e registrare un profilo personale, accessibile ad altri tramite l'applicazione. Altre feature includono bookmarking, logging e tweetting.

Un prodotto simile a Sekai Camera è In the Mood, un'applicativo che si integra con il music player del telefono, che utilizza un sistema di punti d'interesse per segnalare la posizione in cui sono stati riprodotti dei brani musicali.

Wikitude World Browser , per Android, Symbian e iOS (iPhone), consente di navigare in ambiente Augmented Reality attraverso dei "mondi personali" (un analogo dei layer) e rappresenta il principale competitor per Layar Reality Browser sia per funzionalità che per diffusione. Da `www.wikitude.me` è possibili aggiungere POI (Points Of Interest) attraverso un'interfaccia Google Maps, caricando tramite un web service i dati da un proprio server utilizzando anche i formati KML (Keyhole Markup Language di Google Earth) e ARML (Augmented Reality Markup Language, che permette maggior controllo sui dati quali, web link e indirizzi di specifici dei POI) entrambi basati su XML. Al pari di Layar, anche quest'applicazione è disponibile gratuitamente ed aperta agli sviluppatoro, i quali possono implementare i propri "mondi" utilizzando le API di Wikitude.

Per quanto riguarda i contenuti, entrambi i browser (Layar e Wikitude) presentano diversi mondi o layer compatibili, ad esempio in ciascuno è presente un livello Google Local Search, così come Youtube Local o Flickr; inoltre in ciascun caso i livelli vengono suddivisi per categorie: locali, preferiti e suggeriti. Le differenze tra i due browser sono però da ricercarsi sul piano dell'interazione con l'utenza. In primo luogo, Wikitude risulta leggermente meno intuitivo di Layar in fase di start-up: nonostante l'interfaccia sia simile non è subito chiaro come utilizzare l'applicazione. In secondo luogo, Wikitude pecca di interattività: nonostante l'interfaccia utente sia ben integrata con il sistema operativo permettendo una facile navigazione, una volta selezionato un POI, gli utenti potranno solamente visualizzarne le informazioni ed anche le immagini (con funzione di zoom integrata), perdendo ogni possibilità di interagirvi tramite azioni (le quali si limitano a semplici web link). Wikitude inoltre non presenta alcun meccanismo di autenticazione, ne metodi per la personalizzazione dell'interfaccia. Per quanto riguarda il developing, Wikitude offre un pacchetto API offline e ben organizzato, e permette l'integrazione con piattaforme come Wikipedia e Worldpress.

3.2 Confronto

Concludiamo la panoramica con una tabella riassuntiva in cui sono i riportati pregi e difetti delle diverse applicazioni. Nel confronto effettuato, Layar Reality Browser si classifica tra i primi posti,

dimostrando maggiori possibilità di interazione con l'utente finale, tramite una vasta gamma di comportamenti (azioni), e proponendo una moltitudine di contenuti grazie anche alla cooperazione con utenti developer, vantaggio ottenuto realizzando una piattaforma aperta e pubblicando un pacchetto API decisamente completo e ben documentato. Le uniche limitazioni all'utilizzo di Layar sono da ricercarsi nelle difficoltà d'integrazione con altri software e, parzialmente, nei requisiti di connessione (a differenza di programmi come BionicEye che operano solo in locale). Junaio offre un servizio superiore a Layar Reality Browser sotto certi aspetti, primo tra tutti la localizzazione indoor, peccando leggermente sul piano dell'interazione; in futuro potrà probabilmente affermarsi come miglior browser per AR. Al pari di Layar possiamo classificare AroundMe e Acrossair, apps che offrono agli utenti un alto grado di interattività e d'integrazione (nel caso di AroundMe) sacrificando però l'apertura della piattaforma, e Wikitude che propone molteplici contenuti, a volte esattamente gli stessi di Layar, rinunciando invece ad una forte interazione con l'utenza. Prodotti come BionicEye risultano certamente svantaggiati nel confronto poiché sacrificano numerose funzionalità, focalizzandosi sulla specificità dei contenuti.

Nome	**Pagamento**	**Interattività**	**Integrazioni**	**API**	**Database**
Acrossair	No	Elevata	Nessuna	No	Online
AroundMe	No	Elevata	Navigon Mobile Navigator	No	Online
BionicEye	Sì	Scarsa	Nessuna	No	Offline
Junaio	No	Elevata	LLA marker	Sì	Online
Layar Reality Browser	No	Elevata	Nessuna	Sì	Online
RobotVision	Sì	Limitata	Nessuna	No	Online
Sekai Camera	No	Limitata	Nessuna	No	Online
Wikitude World Browser	No	Limitata	Navigon Mobile Navigator, Wikipedia, WorldPress	Sì	Online

Tabella 3.1: Tabella di confronto per applicazioni AR.

Parte IV

Realizzare uno strato

Capitolo 4

Lo strato

Dopo aver esaminato le potenzialità del browser Layar, spostiamo l'analisi sulla strutturazione degli strati ovvero i contenuti proposti durante la navigazione. Gli elementi fondamentali di un layer sono essenzialmente tre:

1. definizione dello strato;
2. punti di interesse;
3. elementi di dettaglio e d'interazione.

La definizione di strato, ovvero del servizio, viene inserita sul sito di Publishing tramite un'interfaccia web e tutti i parametri ad essa connessi (impostazioni, look&feel, autenticazione e filtri) possono essere presentati in questo ambiente. I punti d'interesse forniti dal servizio sono possibilmente localizzati in un database e possono essere user-generated. Gli elementi di dettaglio (informazioni ed azioni) forniscono agli utenti un livello superiore d'esperienza.

Ciascun layer può essere in due stati: *test* o *published*. In entrambi i casi un utente può accedervi, ma un livello test può essere raggiunto solo dallo sviluppatore che l'ha definito (o da utenti autorizzati) utilizzando la chiave sviluppatore, mentre nello stato published può essere consultato da chiunque utilizzi il browser.

Una volta definito e pubblicato, uno strato viene elencato nella Layar Gallery: quando l'utente finale lo seleziona, i POI per quel livello saranno recuperati (relativamente alla posizione corrente) utilizzando i parametri di filtro eventualmente impostati.

Nel seguito vedremo nel dettaglio come realizzare uno strato personalizzato, analizzandone la definizione e l'implementazione.

Si noti che dal 3 ottobre 2010, gli ambienti di Publishing e Provisioning sono stati unificati, al fine di facilitare le operazioni di gestione per i developer: è quindi possibile riferirsi al contesto di definizione utilizzando entrambi i termini.

4.1 Definizione di uno strato

4.1.1 Interfaccia web

Lo sviluppatore può definire un nuovo livello cliccando sul pulsante "Create layer" sul sito di pubblicazione. Apparirà quindi un modulo web con le seguenti proprietà:

- *Available in country*: permette di selezionare i paesi in cui il layer sarà visibile, fino ad un massimo di 20. Scegliendo "international", il servizio sarà usufruibile ovunque (utile per strati con contenuti generici);
- *Name*: un nome unico per lo strato;
- *Non-commercial:* indica se gli utenti devono completare un pagamento per usufruire dei servizi dello strato;
- *poiURL*: l'URL per il servizio web che fornisce informazioni sui punti di interesse (POI) presenti nel database. Il "POI URL" dovrebbe puntare ad un file PHP, come ad esempio `http://your_site.net/my_layer.php`. Inoltre solo le porte 80 (HTTP) e 443 (HTTPS) possono essere utilizzate;
- *Publisher*: il nome dello sviluppatore o della compagnia che ha realizzato il layer;
- *Short description*: una breve descrizione del livello; è importante descrivere lo scopo dello strato e la localizzazione dei POI (ad esempio "Cerca gli hotel a Parigi") per favorire gli utenti;
- *Tag*: un elenco di termini che si applicano alla funzione di ricerca del browser;
- *Title*: il titolo che sarà visibile una volta pubblicato; il titolo non deve essere più lungo di 18 caratteri;
- *Type*: indica il tipo di livello ("General" o "2D/3D"); si ricordi che se un livello è generico, il campo `dimension` dei POI deve essere impostato a 0 o 1.

Dopo aver completato il modulo, è necessario selezionare "Salva e personalizza" per lavorare sui dettagli. Una nuova tabella verrà visualizzata nell'interfaccia contenente sei schede:

1. *AR View*: definisce l'aspetto dell'interfaccia sul client (discusso in seguito);
2. *Details*: qui è possibile inserire una descrizione dettagliata per il livello (l'obiettivo del livello, dove si trovano punti di interesse, se i dati sono generati dall'utente finale o se vi è possibilià di interazione con i dati, o anche se il livello è disponibile solo temporaneamente); le opzioni di autenticazione possono essere attivate e configurate in questa scheda;
3. *Filters*: definisce i filtri disponibili per le impostazionie del livello, ed i valori ammessi (discusso in seguito); di default un range slider viene impiegato per definire l'intervallo di ricerca;
4. *General*: questa scheda permette di modificare il tipo del livello, l'url del servizio (entrambi definiti in precedenza), e la selezione della regione di visibilità (discusso in seguito);
5. *List*: qui è possibile caricare una immagine di 64x64 pixel associata al livello (in modo che tutti siano in grado di riconoscerlo), impostare il titolo, la descrizione, l'editore, i tag e la categoria dello strato. Vi è anche la possibilità di evidenziare i requisiti per l'esecuzione del livello (ad esempio se utilizza modelli animati);
6. *POI icons*: per definire ogni CIW.

Per una migliore comprensione del significato di queste schede, saranno ora presentati alcuni dettagli utili alla corretta compilazione dei diversi campi informativi. Terminata la compilazione di questi, lo strato sarà definito.

4.1.1.1 Definire un bounding box

All'interno della Galleria Layer, vi è una scheda per strati locali che permette agli utenti di trovare ciò che è veramente rilevante per la posizione in cui si trovano. Sul sito di pubblicazione, durante la modifica di uno strato, gli sviluppatori possono specificare una o più aree di delimitazione (bounding box). Il seguente elenco riporta alcune note utili e problemi noti riguardanti tali elementi:

- utilizzare un sito come `http://mygeoposition.com/` per identificare le coordinate dell'area di delimitazione; le coordinate vanno inserite nell'ordine Nord, Est, Sud, Ovest;
- il bounding box non dovrebbe essere maggiore di 200 kmq (20x20 km per l'Equatore, 15x20 km circa a 45° Nord);
- è possibile definire più di un rettangolo di selezione, fino ad un massimo di 20 (come il numero di paesi in cui può essere impiegato un singolo layer).

4.1.1.2 Filtri

In ogni livello possono essere inserite cinque tipologie di filtri: `TEXT BOX`, `RANGE_SLIDER`, `CUSTOM_SLIDER`, `CHECKBOX`, `RADIOBUTTON`. I valori di ciascun filtro verranno inviati nella richiesta inserendoli in un apposito contenitore di dati. Alcuni filtri possono essere utilizzati solo una volta nella definizione del livello, ovvero `CHECKBOX` e `RANGE_SLIDER`.

La gestione dei filtri, che avviene attraverso il sito di definizione degli strati, caratterizza ogni servizio. Nelle future versioni del browser è prevista la possibilità di utilizzare filtri personalizzati: in questo modo sarà possibile creare pagine di impostazioni su misura e adatte ai propri contenuti (ad esempio un filtro che seleziona i POI in base all'editore che li ha realizzati).

4.1.1.3 Presentazione dell'interfaccia

Ogni strato è altamente personalizzabile sotto l'aspetto del look&feel, in modo che gli utenti possano facilmente distinguerli: sarà quindi possibile impostare i colori, il banner, il logo, così come stringhe di testo della BIW, e gli indirizzi delle immagini; lo sviluppatore sarà in grado di vedere il risultato, direttamente tramite un modulo on-line con schermate di mockup.

Gli sviluppatori possono aggiornare le informazioni di un livello in ogni momento successivo alla pubblicazione (l'aggiornameto viene completato nell'arco di un'ora circa), tuttavia, cambiare spesso i dati di presentazione, va a discapito degli utenti, i quali otterranno solamente un'esperienza confusa.

La seguente tabella riporta i parametri per la definzione dell'interfaccia utente.

Parametro	**Formato**	**Descrizione**
bannerTxtColor (opzionale)	RGB	Il colore del testo nel banner (il titolo del layer viene visualizzato qui)
bannerBgColor (opzionale)	RGB	Il colore di sfondo per il banner
bannerIcon (opzionale)	PNG	L'icona da utilizzare nel banner per il branding del livello (al massimo 60x26 pixel)
biwBgColor (opzionale)	RGB	Il colore di sfondo da utilizzare per la CIW a fuoco, sullo sfondo del BIW e sulla pagina delle impostazioni.

biwBg (opzionale)	PNG	L'immagine da usare come sfondo per il BIW. Le dimensioni devono essere 320x83 px. Il top-padding è di 4px.
countryCode	string	Identifica il paese per il quale il livello è pertinente. 'Intl' è attribuito ad uno strato internazionale.
customCiws (opzionale)	PNG	Un elenco di icone opzionali da utilizzare come CIWs. Se non definito, viene utilizzato il widget circolare standard.
detailDescription	URL	L'URL che viene prelevato quando il pulsante 'Dettagli' è selezionato.
filters (opzionale)	array	Un elenco di filtri che l'utente può impostare quando accede al livello.
icon (opzionale)	PNG	L'icona da utilizzare nella Galleria Layer. 64x64 pixel.
innerColor (opzionale)	RGB	Il colore dei punti nel cerchio interno nel cono radar e nella schermata principale.
middleColor (opzionale)	RGB	Il colore dei punti nel cerchio centrale nel cono radar e nella schermata principale.
outerColor (opzionale)	RGB	Il colore dei punti nel cerchio esterno nel cono radar e nella schermata principale.
publisher	string	Nome dello sviluppatore del livello.
name	string	Il nome del livello. Questo valore non è visualizzato all'utente finale, ma rappresenta la chiave che viene utilizzata per il recupero di informazioni all'interno del livello. Permette lo cambio d'informazioni tra il server Layar e lo sviluppatore.
shortDescription	string	Breve descrizione di un livello, riportata nell'elenco della Galleria Layer.
showFilterOnLaunch (opzionale)	boolean	Permette al'applicazione di mostrare o meno la pagina di impostazioni quando il livello è selezionato dall'utente, la prima volta durante una sessione.
spotColor (opzionale)	RGB	Il colore dei POI del radar che attualmente sono al di fuori del cono di visualizzazione.
tags	array	Un elenco di parole che descrivono il livello (massimo 10). I termini nel titolo del livello vengono automaticamente aggiunti alla lista di tag.
textColor (opzionale)	RGB	Il colore del testo all'interno del BIW.
title	string	Il testo da visualizzare nel banner sulla destra e nella lista dei livelli.
titleColor (opzionale)	RGB	Il colore della prima riga di testo nella BIW.
type	integer	Il tipo di layer, che stabilisce come il client deve visualizzare i POI (generico, 3D).
poiURL	URL	L'URL per òa chiamata GetPointsOfInterest. Si noti che solo le porte 80 (http) e 443 (https) sono supportate.

oauth_required (opzionale)	boolean	Indica se il server Layar dovrà firmare le richieste utilizzando OAuth.
oauth_consumer_key (opzionale)	string	La consumer_key da utilizzare al momento della firma.
oauth_consumer_secret (opzionale)	string	Il consumer_secret da utilizzare al momento della firma.

Table 4.2: Parametri di definizione dell'interfaccia di livello.

4.1.1.4 CIW personalizzate

Un CIW, presentato precedentemente tra le API di Layar, è un disco colorato associato a ciascun POI del livello, in cui colore e dimensione variano per ciascuno dei quattro stati del punto:

1. Focus (`biwBgColor`),
2. Inner (`innerColor`),
3. Middle (`middleColor`),
4. Outer(`outerColor`).

Invece della forma circolare di default gli sviluppatori hanno la possibilità di specificare immagini personalizzate in formato PNG: in totale è possibile specificare 3 CIW, realizzando così 3 diversi tipi di POI, ma dalla versione 3.6 di Layar Reality Browser è invece possibile definirne un numero arbitrario (il valore `type` potrà contenere valori da 1 a n). Ogni set d'icone sarà definito da un numero univoco all'interno del singolo livello, e conterrà quattro valori booleani per le rappresentazioni `focus`, `inner`, `middle` ed `outer`.

Parametro	**Valore**	**Funzione**
type	integer	Un intero che definisce il tipo di POI. Quando il client dovrà visualizzare un punto di una specifica tipologia, utilizzerà il set di icone associate.
focus	png	Definisce l'icona per lo stato 'focus' (ovvero selezionato), di dimensioni massime 55x55 pixel.
inner	png	Icona per lo stato 'inner', ovvero un POI molto vicino al client (range di 50 metri) ma non selezionato. Anche quest'icona ha dimensioni massime 55x55 pixel.
middle	png	Icona per i punti nello stato 'middle', ad uan distanza tra 50 e 100 metri dall'utente. Le dimensioni massime sono 38x38 pixel.
outer	png	Icona per lo stato 'outer' in cui si trovano tutti i punti al di fuori dei range precedenti. Le dimensioni concesse sono 28x28 pixel.

Table 4.3: Parametri di definizione per un CIW.

4.2 La formula di Haversine

Molto spesso all'interno di un layer sarà necessario calcolare la distanza tra due punti geografici (ad esempio l'utente ed un edificio), conoscendo solamente le loro posizioni GPS. Il metodo più adatto per raggiungere lo scopo richiede l'uso della trigonometria sferica, la quale, approssimando la Terra ad una forma sferoidale, permette di calcolare la distanza spaziale tra punti posti su una sfera di raggio R tramite la formula di Haversine.

La **formula di Haversine** è un caso particolare di una formula più generale, la legge del haversines, che mette in relazione lati e angoli di un elemento sferico; essa restituisce la distanza tra due punti posti su una sfera, conoscendo le loro coordinate (espresse come latitudine e longitudine) ed il raggio della sfera, ed è scritta in funzione della funzione $haversin(\vartheta) = sin^2(\vartheta/2)$.

La formula di Haversine è la seguente:

$$haversin(d/R) = haversin(\varphi_1 - \varphi_2) + cos(\varphi_1)cos(\varphi_2)haversin(\Delta\lambda)$$

dove abbiamo che:

- d è la distanza tra due punti lungo il "great circle" definito sulla sfera (ovvero l'intersezione tra un piano contenente i punti e la sfera stessa, che produce la circonferenza maggiore);
- R è il raggio della sfera;
- φ_1 rappresenta la latitudine del primo punto in radianti;
- φ_2 rappresenta la latitudine del secondo punto in radianti;
- $\Delta\lambda$ è la differenza tra le longitudini dei due punti, espressa in radianti.

Questa formula garantisce però risultati non particolarmente precisi sulle grandi distanze, ma solo un'approssimazione, poiché, come è risaputo, la Terra non è una sfera perfetta: il raggio del pianeta, R, varia infatti da 6356,78 km ai poli di 6378,14 km all'equatore [22].

Capitolo 5

Implementazione

Prima di creare un livello è necessaria una chiave sviluppatore la quale può essere ottenuta tramite la registrazione presso `http://dev.layar.com`. La procedura di registrazione richiede la compilazione di un classico form, in seguito alla quale, un link di attivazione viene inviato al developer entro pochi giorni; dopo aver attivato l'account, lo sviluppatore può accedere al sito di pubblicazione.

Gli strati possono essere realizzati utilizzando diversi linguaggi di programmazione, come PHP o Javascript. Lo sviluppo di nuovi servizi richiede che nel server dove essi risiedono, sia disponibile una versione di PHP (5.2 o superiore, preferibilmente) con supporto JSON, e opzionalmente una base di dati MySQL magari corredata da un gestore di database come phpMyAdmin.

5.1 Un semplice layer

Quando le applicazioni necessarie sono state installate, il passo successivo è quello di preparare il database che memorizza le informazioni dei POI. Si noti però che non è necessario predisporre un database per ogni servizio: ad esempio il layer Space Invader 3D legge la posizione del client contenuta nella richiesta e restituisce su questa un POI, rappresentato da un vascello spaziale (il cui modello sarà contenuto nel server che ospita il servizio) senza quindi interagire con una base di dati.

5.1.1 Database

Le posizioni dei diversi POI potrebbero essere memorizzate in un file XML o in qualsiasi formato simile, ma scegliere di utilizzare MySQL (o equivalentemente PostgreSQL) permette di eseguire una query in base al raggio di ricerca: ciò significa che il calcolo effettivo dei risultati da restituire, viene lasciato alla query. La seguente istruzione viene utilizzata per creare la tabella denominata POI_Table.

```
-- Table structure for table `POI_Table`
--
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
CREATE TABLE IF NOT EXISTS `POI_Table` (
`id` int(50) NOT NULL auto_increment,
`attribution` varchar(50) default NULL,
`title` varchar(50) default NULL,
`lat` decimal(20,10) default NULL,
`lon` decimal(20,10) default NULL,
`imageURL` varchar(255) default NULL,
```

```
‘line4‘ varchar(50) default NULL,
‘line3‘ varchar(50) default NULL,
‘line2‘ varchar(50) default NULL,
‘type‘ int(11) NOT NULL default ’0’,
‘actions‘ varchar(50) default NULL,
‘dimension‘ int(1) NOT NULL default ’1’,
‘alt‘ int(10) default NULL,
‘relativeAlt‘ int(10) default NULL,
‘transform‘ int(10) default NULL,
‘object‘ int(10) default NULL,
‘distance‘ decimal(20,10) default NULL,
PRIMARY KEY (‘id‘)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;
```

Listing 5.1: Istruzione SQL per la creazione della tabella POI_Table.

Quando la tabella POI_Table è pronta, è possibile popolarla inserendo i dati relativi ai diversi POI: Google Maps è un ottimo strumento per ottenere tali informazioni, ma gli sviluppatori possono utilizzare anche altri strumenti. Inseriamo ad esempio:

```
INSERT into POI_Table (title, lat, lng)
values (’Dipartimento di Ingegneria dell’Informazione’, ’45.4094177’, ’
   11.8938053’);
```

5.1.2 PHP web service

Passiamo ora alla definizione del servizio vero e proprio. Innanzitutto sarà necessario creare uno script PHP il quale sarà poi richiamato nella definizione di livello tramite il `poiURL`. Per iniziare si devono quindi impostare le variabili per la connessione al database MySQL utlizzando la seguente sintassi:

```
$dbhost = "localhost";
$dbdata = "database_name";
$dbuser = "database_username";
$dbpass = "database_password";
```

La definizione dei parametri di connessione dovrebbe essere inserita in un file apposito (`dbconfig.php`) non raggiungibile dall'esterno senza le adeguate credenziali.

Sulla base della definizione di livello una richiesta HTTP viene inviata all'URL del servizio: per recuperare i valori dei parametri in essa contenuti è possibile utilizzare la variabile `$_GET`. I parametri del metodo `getPOI`, esposti in precedenza, verranno quindi inseriti in un array associativo denominato `$value`, che utilizzerà come nome dei parametri la chiave di questi e come valore il valore del parametro. Il listanto sotto proposto prende in esame quattro parametri:

- `layerName`: il nome del livello,
- `lat`: la latitudine della posizione in cui si trova l'utente,
- `lon`: la longitudine della posizione in cui si trova l'utente,
- `radius`: il raggio di ricerca, all'interno del quale i POI devono essere individuati.

```
// Put needed parameter names from the GetPOI request
//in an array called $keys.
$keys = array( "layerName", "lat", "lon", "radius" );
// Initialize an empty associative array.
$value = array();
// Retrieve parameter values using $_GET and put them in
// $value array with parameter name as key.
foreach( $keys as $key ) {
        $value[$key] = $_GET[$key];
}
```

Listing 5.2: Lettura dei parametri della richiesta `getPointsOfInterest`().

Giunti a questo punto è opportuno configurare correttamente la connessione al database MySQL selezionando la base di dati che contiene la tabella POI_Table.

```
// Connect to predefined MySQL database.
$db = new PDO( "mysql:host=$dbhost; dbname=$dbdata", $dbuser, $dbpass );
// set the error reporting attribute to Exception.
$db->setAttribute( PDO::ATTR_ERRMODE , PDO::ERRMODE_EXCEPTION );
```

Listing 5.3: Istruzioni di connessione del servizio al database tramite PDO.

Come si può notare le istruzioni per la connessione al database utilizzano PDO, un'estensione di PHP che formalizza la connessione verso i database, tramite l'impiego di metodi e strutture predefinite che favoriscono la sicurezza dei sistemi aiutando a prevenire attacchi di tipo SQL injection. Tramite una combinazione dei metodi `PDO::prepare`() e `PDO::execute`(), per preparare ed eseguire le query SQL, si realizzerà la funzione che recupera i punti dalla base di dati; si ricordi che tutti i parametri definiti in oggetto POI sono obbligatori (se non si vuole assegnare un valore ad un campo questo verrà settato a NULL di default). L'istruzione SQL seleziona tutti gli elementi del database, fino ad un massimo di 50, la cui distanza calcolata tramite metodi di trigonometria sferica, risulta inferiore alla soglia scelta dall'utente.

Il calcolo della distanza (in metri) tra due località si basa sulla formula di Haversine ed ha una complessità pari ad *O(N)*, dove *N* è il numero di record nel database e considerando le operazioni di calcolo come *O(1)*. Il metodo richiede infatti di eseguire alcune computazioni in tempo costante per ciascuno degli *N* record e quindi *N*O(1)=O(N)*.

```
// Use PDO::prepare() to prepare SQL statement.
// This statement is used due to security reasons and will help
// prevent general SQL injection attacks.
// ":lat1", ":lat2", ":long" and ":radius" are named parameter
// markers for which real values will be substituted when the statement
// is executed.
// $sql is returned as a PDO statement object.
$sql = $db->prepare( " SELECT *, (((acos(sin((:lat1 * pi() / 180)) * sin((
   lat * pi() / 180)) + cos((:lat2 * pi() / 180)) * cos((lat * pi() / 180)
   ) * cos((:long - lon) * pi() / 180))) * 180 / pi()) * 60 * 1.1515 *
   1.609344 * 1000) as distance
```

```
                FROM POI_Table
//this command filters result by radius
HAVING distance < :radius
ORDER BY distance ASC
LIMIT 0, 50
" );
// PDOStatement::bindParam() binds the named parameter markers
// to the specified parameter values.
$sql->bindParam( ':lat1', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':lat2', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':long', $value['lon'], PDO::PARAM_STR );
$sql->bindParam( ':radius', $value['radius'], PDO::PARAM_INT );
// Use PDO::execute() to execute the prepared statement $sql.
$sql->execute();
```

Listing 5.4: Preparazione ed esecuzione della query di ricerca dei POI nel database.

Individuati i risultati da restituire all'utente, lo script dovrà adoperarsi per costruire una risposta JSON corretta, nella quale devono essere inseriti i valori delle chiavi richieste tra i quali ricordiamo:

- `layer` (string): il nome del layer,
- `errorCode` (integer): 0 per nessun errore,
- `errorString` (string): il messaggio d'errore,
- `hotspots` (array): la lista dei POI restituiti.

Per inserire i risultati della query in un array associativo (`$reponse["hotspots"]`) utilizzeremo una funzione che chiameremo `Gethotspots()`, alla quale sarà passato in input il valore `$sql` contenente i dati dell'interrogazione alla base di dati, mentre in output sarà ritornato l'array contenente punti di interesse. All'interno di `Gethotspots()` sarà impiegata una funzione `ChangetoIntLoc()` che convertirà i valori decimale di latitudine o longitudine (GPS) in interi, eseguendo una moltiplicazione per 10^6. Di seguito il codice proosto per questi due metodi.

```
function ChangetoIntLoc( $value_Dec ) {
        return $value_Dec * 1000000;
}//ChangetoIntLoc

//
function Gethotspots( $sql ) {
// Iterator for the response array.
        $i = 0;
        // Use PDO::FETCH_ASSOC to fetch $sql query results and
        // put each POI information into $poi associative array.
        while ( $poi = $sql->fetch(PDO::FETCH_ASSOC) ) {
                // Set the "actions" for this poi to an empty array for
                   now.
                $poi["actions"] = array();
                // Store the integer value of "lat" and "lon" using
                //predefined function ChangetoIntLoc.
```

```
                $poi["lat"] = ChangetoIntLoc( $poi["lat"] );
                $poi["lon"] = ChangetoIntLoc( $poi["lon"] );
                // Put the poi into the response array.
                $response["hotspots"][$i] = $poi;
                $i++;
        }//while
        return $response["hotspots"];
}//Gethotspots
```

Listing 5.5: Funzioni `Gethotspots`() e `ChangetoIntLoc`().

I risultati della funzione `GetHotspots`() andranno quindi inseriti nella risposta insieme alle chiavi ivi richieste: a questo punto tutti i dati necessari per l'applicazione client sono stati elaborati e possono essere restituiti insieme allo stato (codici d'errore) di questi; utilizzando la funzione nativa di PHP `json_encode`(), operiamo la conversione della risposta in formato JSON e restituiamola in output (dichiarando prima l'header corretto). Infine si dovrà chiudere la connessione al database.

```
// Create an empty array named $response.
$response = array();
// Assign corresponding values to mandatory JSON response keys.
$response["layer"] = $value["layerName"];
// Use Gethotspots() function to retrieve returned
// POIs from the SQL query $sql.
$response["hotspots"] = Gethotspots( $sql );
// if none POI is founded, return a custom error message.
if ( empty( $response["hotspots"] ) ) {
        $response["errorCode"] = 20;
        $response["errorString"] =
        "No POI found. Please adjust the range.";
}//if
else {
        $response["errorCode"] = 0;
        $response["errorString"] = "";
}//else
// Put the JSON representation of $response into $jsonresponse.
$jsonresponse = json_encode( $response );
// Declare the correct content type in HTTP
//response header (as declared before).
header( "Content-type: application/json; charset=utf-8" );
// Print out Json response.
echo $jsonresponse;
/* Close the MySQL connection.*/
// Set $db to NULL to close the database connection.
$db=null;
```

Listing 5.6: Preparazione della risposta del servizio. Definizione dei codici d'errore e conversione in formato JSON.

Lo script appena realizzato, basandosi su un'unica tabella di informazioni, permette agli utenti finali di usufruire di un semplice servizio di ricerca. Nelle sezioni successive vedremo come includere meccanismi di interattività e lavorare su strati con modelli tridimensionali.

Prima di proseguire si noti che è possibile applicare alcuni cambiamenti al servizio appena realizzato, in modo da ottenere una gestione più corretta delle informazioni, o per aggiungere semplici comportamenti:

1. per gestire i caratteri speciali (ad esempio caratteri cinesi), nonchè particolari URL, è possibile modificare l'istruzione per la connessione al database come segue:

   ```
   $db = new PDO(
   "mysql:host=$dbhost; dbname=$dbdata", $dbuser, $dbpass, array(
   PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8") );
   ```

2. Layar Reality Browser esegue un'indicizzazione di tutti i contenuti presenti nei vari layer (come Google che indicizza ogni sito web) riproponendoli nellasezione *Vicino* della Layar Gallery. Lo sviluppatore può però decidere quali elementi saranno catalogati e quali no, tramite l'inserimento di un parametro `doNotIndex`; aggiungendo un parametro `inFocus`, sarà possibile selezionare il POI le cui informazioni saranno visualizzate dal client una volta ottenuta la risposta. Applichiamo la seguente istruzione SQL per attivare le funzionalità appena illustrate:

   ```
   ALTER TABLE `POI_Table` ADD `doNotIndex` TINYINT( 1 ) NULL DEFAULT
      NULL ;
   ALTER TABLE `POI_Table` ADD `inFocus` TINYINT( 1 ) NULL DEFAULT NULL;
   ```

   Nelle implementazioni seguenti considereremo anche i parametri `inFocus` e `doNotIndex`.

5.2 Layer con azioni

Una delle caratteriste principali offerte dalle API di Layar Reality Browser è la possibilità di incorporare in uno strato funzionalità interattive accanto a ciascun POI. Come anticipato in precedenza, alcune azioni potranno essere attivate manualmente da parte degli utenti tramite un "foglio di azione" (es. guardare un video o fare una telefonata), mentre altre potranno essere collegate ad un trigger che le attiverà automaticamente non appena il client entrerà in una certa regione. Prima di esaminare lo script PHP che implementa il servizio, riproponiamo i principali tipi di azioni supportate in Layar (ulteriori azioni possono essere definite in relazione a diverse tipologie di MIME validi):

Per includere i comportamenti interattivi nel servizio precedentemente realizzato vi verranno apportate le seguenti modifiche:

- database: si creerà una nuova tabella denominata 'ACTION_Table',
- codice PHP: si aggiungerà una funzione chiamata `Getactions` per la lettura delle azioni.

Azione	Descrizione
http://, https://	Collegamento a pagine web.
tel://	Chiama un numero di telefono.
mailto://	Invia un messaggio email.
audio://	Esegue un file audio; gli mp3 con playback rate di 96kbit sono consigliati
video://	Esegue un file video; gli mp4 con risoluzione 480x320 px sono consigliati.
layar://	Richiama uno strato tramite il browser Layar.

Table 5.1: Esempi di URI validi per le azioni.

5.2.1 Database

All'interno del database in cui è contenuta la tabella POI_Table andremo ad inserire un nuovo insieme di dati: consideriamo infatti ogni azione come un'entità a sè stante, unica e dotata dei propri attributi; ciascuna sarà associata ad un proprio identificativo (chiave primaria) e ad un campo "poiID" dove sarà memorizzato l'"id" di un elemento POI, riferito a POI_Table, un'etichetta "label" che identifica l'azione all'interno del "foglio d'azione", un target "uri" e le impostazioni di triggering. Ogni interazione verrà poi caratterizzata da un target (campo "uri"), dalla presenza di un trigger e dalla regione di attivazione (se necessaria).

```
--
-- Table structure for table `ACTION_Table`
--
CREATE TABLE IF NOT EXISTS `ACTION_Table` (
`poiID` int(10) default NULL,
`label` varchar(30) default NULL,
`uri` varchar(255) default NULL,
`autoTriggerRange` int(10) default NULL,
`autoTriggerOnly` tinyint(1) default NULL,
`ID` int(10) NOT NULL,
PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Listing 5.7: Istruzione SQL per la creazione della tabella ACTION_Table.

5.2.2 PHP web service

Aggiungiamo ora una funzione `Getactions`() al nostro script PHP: questa leggerà il contenuto di ACTION_Table ed inserirà in un apposito array tutte le azioni associate ad un particolare POI. Il seguente codice propone un'implementazione del metodo: questo riceve in input il POI in esame ed il connettore per il database del servizio e restituisce in output una struttura contente i dati delle azioni (ritorna un insieme vuoto se non è stata rilevata alcuna corrispondenza tra l'elemento ed il contenuto dell'ACTION_Table).

Inseriamo anche un metodo `ChangeToBool`() per convertire i valori numerici contenuti nel campo "AutoTriggerOnly", in booleani.

```
// Put fetched actions for each POI into an associative array.
// The returned values are assigned to $poi[actions].
// Arguments:
// poi : The POI handler.
// db : The database connection handler.
//
// Returns:
// array : An array of received actions for this POI.
//
function Getactions( $poi, $db ) {
        // A new table called "ACTION_Table" is created to store
        // actions, each action has a field called
        // "poiID" which shows the POI id that this action belongs to.
        //The SQL statement returns actions which have the same
        // poiID as the id of $poi ($poi['id']).
        $sql_actions = $db->prepare( " SELECT * FROM ACTION_Table WHERE
           poiID = :id " );
        // Binds the named parameter markers ":id"
        // to the specified parameter values "$poi['id']".
        $sql_actions->bindParam( ':id', $poi['id'], PDO::PARAM_INT );
        // Use PDO::execute() to execute the prepared
        // statement $sql_actions.
        $sql_actions->execute();
        // Iterator for the $poi["actions"] array.
        $count = 0;
        // Fetch all the poi actions.
        $actions = $sql_actions->fetchAll( PDO::FETCH_ASSOC );
        /* Process the $actions result */
        // if $actions array is empty, return empty array.
        if ( empty( $actions ) ) {
                $poi["actions"] = array();
        }//if
        else {
                // Put each action information into $poi["actions"] array.
                foreach ( $actions as $action ) {
                        // Assign corresponding action key values 'uri', '
                           label'
                        $poi["actions"][$count]['uri'] = $action['uri'];
                        $poi["actions"][$count]['label'] = $action['label'
                           ];
                        // Return values for 'autoTriggerRange' and
                        // 'autoTriggerOnly' if 'autoTriggerRange' is not
                           NULL.
                        if ( strlen( trim( $action['autoTriggerRange'] ) )
                            != 0 ) {
                                $poi["actions"][$count]['autoTriggerRange'
                                   ]=
                                 $action['autoTriggerRange'];
                                // If 'autoTriggerOnly' value is 1, a
                                   boolean value
                                // "TRUE" is used; Otherwise, FALSE;
                                // Use function called "ChangetoBool"
                                $poi["actions"][$count]['autoTriggerOnly']
```

```
                                        =
                                    ChangetoBool( $action['autoTriggerOnly'] );
                            }
                            $count++;
                    }// foreach
            }//else
            return $poi["actions"];
}//Getactions

//Convert a TinyInt value to a boolean value TRUE or FALSE
//
// Arguments:
// value_Tinyint ; The Tinyint value (0 or 1)
//of a key in the database.
//
// Returns:
// value_Bool ; The boolean value, return 'TRUE'
// when Tinyint is 1. Return 'FALSE' when Tinyint is 0.
function ChangetoBool( $value_Tinyint ) {
            if ( $value_Tinyint == 1 ) $value_Bool = TRUE;
            else $value_Bool = FALSE;
            return $value_Bool;
}//ChangetoBool
```

Listing 5.8: Funzioni `Getactions()` e `ChangeToBool()`.

All'interno della chiamata `Gethotspots()` verrà quindi inserita una invocazione del metodo `GetActions()`, allo scopo d'includere le azioni nella risposta JSON: questo sarà richiamato durante l'esecuzione del ciclo interno della funzione, come proposto nel seguente sorgente:

```
// Put each POI information into $response["hotspots"] array.
foreach ( $pois as $poi ) {
            // Use function Getactions() to return an array of actions
            //asscociated with the current POI.
            $poi["actions"] = Getactions ( $poi, $db );
            // Store the integer value of "lat" and "lon" using predefined
            //function ChangetoInt.
            $poi["lat"] = ChangetoInt( $poi["lat"] );
            $poi["lon"] = ChangetoInt( $poi["lon"] );
            // Change the values of "doNotIndex" into boolean value,
            //if the value is not NULL.
            if ( strlen( trim( $poi["doNotIndex"] ) ) != 0 )
                    $poi["doNotIndex"] = ChangetoBool( $poi["doNotIndex"] );
            // Change the values of "inFocus" into boolean value,
            //if the value is not NULL.
            if ( strlen( trim( $poi["inFocus"] ) ) != 0 )
                    $poi["inFocus"] = ChangetoBool( $poi["inFocus"] );
            // Put the poi into the response array.
            $response["hotspots"][$i] = $poi;
            $i++;
}//foreach
```

Listing 5.9: Modifica del ciclo interno alla funzione `Gethotspots()`.

Tramite la creazione di una nuova tabella del database è l'implementazione di un metodo per il recupero dei dati in essa contenuti, è stato realizzato un servizio che permette agli utenti di interagire con gli elementi della vista AR.

5.3 Layer con oggetti

Per realizzare un layer con oggetti bidimensionali e tridimensonali nello spazio (quindi non più uno strato di tipo generico), sarà neccessario apportare le seguenti modifiche al codice precedentemente presentato:

- database: aggiungere due tabelle nel database, ovvero OBJECT_Table e TRANSFORM_Table,
- codice PHP: aggiungere due funzioni `GetObject` e `GetTransform` per recuperare oggetti e trasformazioni applicate ai POI.

5.3.1 Database

Accostiamo alle tabelle già realizzate, due nuove strutture che saranno utilizzate per memorizzare i diversi oggetti e le trasformazioni da applicarvi, vale a dire OBJECT_Table e TRANSFORM_Table. Come per punti ed azioni, considereremo anche oggetti e trasformazioni come entità separate ma collegate tra loro. Ogni tabella manterrà un proprio campo "id" utilizzato come chiave primaria e un campo "poiID" per memorizzare l'identificatore del corrispondente POI nella POI_Table. Ulteriori attributi per ciascun elemento sono definiti nella API di Layar Reality Browser.

```
--
-- Table structure for table `OBJECT_Table`
--
CREATE TABLE IF NOT EXISTS `OBJECT_Table` (
`ID` int(10) NOT NULL auto_increment,
`poiID` varchar(255) NOT NULL,
`baseURL` varchar(255) NOT NULL,
`full` varchar(255) NOT NULL,
`reduced` varchar(255) default NULL,
`icon` varchar(255) default NULL,
`size` float(15,5) NOT NULL,
PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;

--
-- Table structure for table `TRANSFORM_Table`
--
CREATE TABLE IF NOT EXISTS `TRANSFORM_Table` (
`ID` int(10) NOT NULL auto_increment,
`poiID` varchar(255) NOT NULL,
`rel` tinyint(1) NOT NULL default '1',
`angle` decimal(5,2) NOT NULL default '0.00',
`scale` decimal(12,2) NOT NULL default '1.00',
PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
```

Listing 5.10: Istruzione SQL per la creazione delle tabelle OBJECT_Table e TRANSFORM_Table.

5.3.2 PHP web service

Nel servizio contenuto nel `poiURL`, andiamo ad inserire due nuove funzioni che andranno ad interrogare OBJECT_Table e TRANSFORM_Table, inserendo i risultati all'interno della struttura che descrive ciascun punti d'interesse. Come nella precedente `GetActions`(), i dati d'ingresso saranno determinati dal connettore al database e dal POI in esame. Entrambe le chiamate dovranno essere eseguite all'interno della funzione `Gethotspots`(). Infine aggiungiamo un metodo `ChangeToFloat`() per la gestione dei valori non interi, restituiti dalla query sotto forma di stringhe. Si ricordi che, generalmente, ciascun POI è associato ad un solo oggetto, il quale è collegato ad una sola trasformazione.

Il sorgente seguente propone un'implementazione di questi metodi.

```
// Put fetched object related parameters for each POI
// into an associative array. The returned values are
//assigned to $poi[object].
//
// Arguments:
//   poi : The POI handler.
//   $db : The database connection handler.
//
// Returns:
//   array : An array of received object related parameters for this POI.

function Getobject( $poi, $db ) {
        // A new table called "OBJECT_Table" is created
        //to store object related parameters, namely
        //"baseURL", "full", "reduced", "icon" and "size".
        // "poiID" which shows the POI id that this object belongs to.
        // The SQL statement returns object which has the same poiID
        // as the id of $poi ($poi['id']).
        $sql_object = $db->prepare( "SELECT baseURL, full, reduced, icon,
           size FROM OBJECT_Table WHERE poiID = :id LIMIT 0,1 " );
        // Binds the named parameter markers ":id" to the specified
        // parameter values "$poi['id']".
        $sql_object->bindParam( ':id', $poi['id'], PDO::PARAM_INT );
        // Use PDO::execute() to execute the prepared statement
           $sql_object.
        $sql_object->execute();
        // Fetch the poi object.
        $object = $sql_object->fetchAll( PDO::FETCH_ASSOC );
        /* Process the $object result */
        // if $object array is empty, return empty array.
        if ( empty( $object ) ) {
                $poi["object"] = NULL;
        }//if
        else {
                // Since each POI only has one object.
                //Logically, only one object should be returned.
                //Assign the first object in the array to
```

```
                //$poi["object"]
                $poi["object"] = $object[0];
                // Change "size" type to float.
                $poi["object"]["size"] = ChangetoFloat( $poi["object"]["
                    size"] );
        }//else
        return $poi["object"];
}//Getobject
// Put fetched transform related parameters
// for each POI into an associative array.
// The returned values are assigned to $poi[transform].
//
// Arguments:
//   poi : The POI handler.
//   $db : The database connection handler.
// Returns:
//   array : An array of received transform related
//parameters for this POI.
function Gettransform( $poi, $db ) {
        // A new table called "TRANSFORM_Table" is created to store
        // transform related parameters, namely
        // "rel", "angle" and "scale"
        // "poiID" which shows the POI id that this transform belongs to.
        // The SQL statement returns transform which has the same
        //poiID as the id of $poi ($poi['id']).
        $sql_transform = $db->prepare(" SELECT rel, angle, scale FROM
            TRANSFORM_Table WHERE poiID = :id LIMIT 0,1 " );
        // Binds the named parameter markers ":id" to the specified
        // parameter values "$poi['id']".
        $sql_transform->bindParam( ':id', $poi['id'], PDO::PARAM_INT );
        // Use PDO::execute() to execute the prepared statement
            $sql_transform.
        $sql_transform->execute();
        // Fetch the poi transform.
        $transform = $sql_transform->fetchAll( PDO::FETCH_ASSOC );
        /* Process the $transform result */
        // if $transform array is empty, return empty array.
        if ( empty( $transform ) ) {
                $poi["transform"] = NULL;
        }//if
        else {
                // Since each POI only has one transform.
                // Logically, only one transform should be returned.
                // Assign the first transform in the array to
                //$poi["transform"]
                $poi["transform"] = $transform[0];
                // Change the value of "rel" into boolean value,
                //if the value is NULL, return NULL.
                $poi["transform"]["rel"] = ChangetoBool(
                $poi["transform"]["rel"] );
                // Change the values of "angle" and "scale" to demical.
                $poi["transform"]["angle"] =
                ChangetoFloat( $poi["transform"]["angle"] );
```

```
                $poi["transform"]["scale"] =
                ChangetoFloat( $poi["transform"]["scale"] );
        }//else
        return $poi["transform"];
}//Gettransform
```

Listing 5.11: Funzioni `Getobject`() e `Gettransform`().

Come anticipato, all'interno di `Gethotspots`() dovranno essere inserite due chiamate ai metodi appena esposti.

```
// Use function Getobject() to return an object
// associated with the current POI.
$poi["object"] = Getobject ( $poi, $db);
// Use function Gettransform() to return a transform
//dictionary associated with the current POI.
$poi["transform"] = Gettransform ( $poi, $db);
```

Listing 5.12: Modifica del ciclo interno alla funzione `Gethotspots`().

La combinazione dei sorgenti presentati in questa sezione, permette ora di utilizzare un servizio di localizzazione che impiega una ricerca basata sulla distanza dall'utente, oggetti interattivi e rappresentazioni tridimensionali, tramite poche interrogazioni presso un database opportunamente costruito [15],[17].

Infine vediamo la funzione `ChangeToFloat`() che converte una stringa in un valore in virgola mobile.

```
// Change a string value to float
// Arguments:
// string : A string value.
// Returns:
//   float : If the string is empty, return NULL.
function ChangetoFloat( $string ) {
if ( strlen( trim( $string ) ) != 0 ) {
        return (float)$string;
}else
        return NULL;
}//ChangetoFloat
```

Listing 5.13: Funzione `ChangetoFloat`().

Capitolo 6

Testing

Come per qualsiasi software, una volta realizzato il servizio è necessario eseguirne il testing per verificare che esso funzioni correttamente. Nello specifico sarà necessario controllare che i risultati restituiti dopo ogni richiesta siano stati correttamente elaborati. Ci sono due metodi per testare un livello:

1. utilizzando la pagina di test associata a ciascuno strato e disponibile nel sito di Publishing. Qui, tramite un'interfaccia web, viene visualizzata una mappa nella quale è presente un "individuo di prova", che può essere collocato in una qualsiasi posizione arbitraria, simulando il posizionamento di un ipotetico utente finale. Dopo aver regolato le impostazioni dei filtri è possibile inviare una richiesta alla pagina del servizio e recuperare i punti di interesse. I risultati contenuti nella risposta saranno poi visualizzati in un menù contestuale;

2. tramite la scheda "Test" dell'applicazione client. Qui gli sviluppatori possono testare, in via esclusiva, il proprio strato dal vivo.

Prima di richiedere la pubblicazione è molto importante che lo strato sia stato testato sia su API che su un telefono: gli sviluppatori saranno in grado di richiedere la pubblicazione solamente se il loro layer ritorna con successo una risposta sulla pagina di test API. Durante il test è consigliato seguire alcune linee guida:

1. ottenere una risposta JSON valida:

   - assicurarsi che tutte le chiavi JSON siano restituite nella risposta, anche se alcune sono NULL o non sono utilizzate. In particolare, i valori degli identificatori di ciascun POI devono essere univoci;
   - assicurarsi che i POI siano restituiti nella risposta o, qualora non vengano individuati dei risultati, venga visualizzato un messaggio d'errore;
   - un layer che visualizzi messaggi di errore come "layer provider not available", "layer name mismatch", "Error 404", ecc, saranno respinti durante il processo di approvazione.

2. impostazioni dei filtri:

   - text box: assicurarsi di aver testato non solo le parole chiave che generano i risultati;
   - range slider: assicurarsi che solo i POI che si trovano entro l'intervallo di ricerca definito possano essere restituiti;

- custom slider: verificare se i POI sono restituiti entro il range selezionato dal client;
- checkbox: eseguire il test per ciascuna opzione di controllo separatamente; eseguire un test anche nel caso in cui nessuna delle opzioni sia selezionata;
- radio button: testare ogni pulsante di opzione separatamente;
- testare alcune combinazioni di filtri: può accadere che i filtri funzionino correttamente se utilizzati separatamente, ma non in combinazione tra loro.

3. nessun risultato trovato:

   - restituire un messaggio personalizzato per il client quando non sono disponibili POI per la risposta;
   - in base alle diverse cause d'errore è preferibile restituire un messaggio personalizzato e auto descrittivo.

4. azioni: (alcune devono essere testate "sul campo")

   - numero di telefono: controllare che il formato sia corretto. Utilizzare `tel://`;
   - email: controllare che il formato sia corretto. Utilizzare `mailto://`;
   - sito Web: controllare che il formato sia corretto. Utilizzare `http://` o `https://`;
   - audio: controllare che il formato sia corretto. Utilizzare `audio://`;
   - video: controllare che il formato sia corretto. Utilizzare `video://`.

   In ciascun caso assicurarsi che vi sia un riferimento valido, altrimenti è consigliabile non visualizzare l'azione. Si noti inoltre che alcune azioni devono essere testate entro un particolare range dal punto a cui sono riferite.

5. look & feel: controllare la combinazione di colori del layer per assicurarsi che i testi possano essere letti facilmente e chiaramente sullo sfondo; controllare anche la visualizzazione di eventuali icone personalizzate;

6. modelli: assicurarsi che i diversi oggetti siano caricati con successo. Si provi a lavorare sulle impostazioni della distanza e ci si assicuri che siano visualizzati utilizzando la giusta dimensione ed il corretto angolo di presentazione. Controllare infine che i modelli `reduced` siano inferiori per taglia e dimensioni rispetto alle rispettive versioni `full`.

Dopo aver testato il livello, lo sviluppatore potrà pubblicarlo per renderlo visibile agli utenti Layar: tale operazione è indicata come richiesta o processo di pubblicazione e sarà analizzata nelle pagine successive [17].

Capitolo 7

Pubblicazione

Il processo di pubblicazione può essere schematizzato come in figura seguente: esso si compone di diverse fasi, anche se alcune non sono ancora disponibili. Per pubblicare un livello è sufficiente premere il tasto "Request publish" che è presente accanto ad ogni livello nella "scheda dei livelli". Prima della pubblicazione sarà inoltre richiesto di leggere ed accettare le condizioni imposte da Layar Reality Browser.

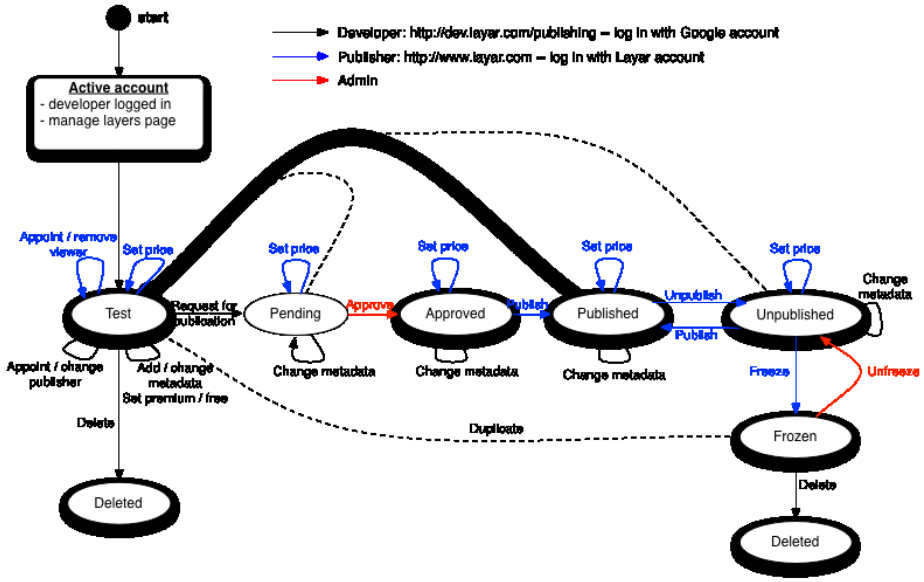

Figure 7.1: Il processo di pubblicazione.

Dopo aver richiesto la pubblicazione, lo staff Layar verificherà il funzionamento del servizio e deciderà se approvare o rifiutare la richiesta (il tempo richiesto è di circa 5 giorni lavorativi). In seguito ad ogni rifiuto di pubblicazione, gli sviluppatori riceveranno un'esauriente spiegazione, di modo che possano ricercare ed applicare delle soluzioni per risolvere i problemi riscontrati.

Si noti che durante il processo di approvazione, il layer non deve essere alterato, ma l'autore, sarà comunque in grado di visualizzarlo e modificarlo: può infatti esservi la necessità di apportare ulteriori cambiamenti; purtroppo quest'ultima operazione richiede agli sviluppatori di inoltrare una nuova richiesta di pubblicazione.

7.1 Criteri per la pubblicazione

Ogni strato, per essere pubblicato, deve superare la fase di testing (e quindi essere correttamente funzionante) e deve favorire una buona e chiara user-experience: questo requisito è auspicabile nella maggioranza degli strati, e può essere raggiunto osservando i seguenti criteri:

- titolo, logo e descrizione: la prima impressione è ciò che determina la quotazione di un livello, quindi è consigliato caricare un logo, scegliere un titolo chiaro e utilizzare una descrizione senza termini tecnici, adatta ad un vasto pubblico;
- descrizione dettagliata: utilizzare questa sezione per spiegare quali feature offre lo strato e se gli utenti hanno la possibilità di interagirvi, ad esempio, accedendo un profilo personale o restituendo un feedback;
- colori: assicurarsi che tutto il testo possa essere letto sullo sfondo selezionato;
- icone personalizzate: queste non sono obbligatorie, ma possono rendere maggiormente interessante lo strato;
- qualità: assicurarsi di utilizzare immagini ad alta qualità, per tutte le icone;
- codice paese: se il livello ricerca solo POI di un certo paese, è conveniente utilizzare il codice corretto, in modo da visualizzarlo coerentemente all'interno della Galleria Layer;
- tag: è possibile utilizzare fino a 30 tag; per ogni termine conviene usare entrambe le forme singolari e plurali, inoltre si ricordi che la ricerca non è case-sensitive. Sono infine permessi i caratteri Unicode;
- nessun POI da visualizzare: se non è possibile restituire alcun POI è preferibile utilizzare un messaggio d'errore personalizzato per informare l'utente, offrendogli un suggerimento che spieghi la ragione per cui non sono stati trovati risultati;
- 3D e 2D: si ricordi che è consigliabile fornire un'icona e una versione ridotta per tutti gli oggetti, e che siano differenti dall'oggetto `full`. Con oggetti 2D, la versione `reduced` dovrebbe mantenere la stessa risoluzione di quella completa, ma può comunque presentare una qualità inferiore.

7.2 Approvazione

Una volta che uno strato è approvato, apparirà una nuova opzione "Publish" all'interno della "scheda dei layer". Si noti come la possibilità di decidere il momento di pubblicazione sia lasciata allo sviluppatore; questa scelta non è arbitraria ma è dovuta, in quanto:

1. è possibile voler annunciare il lancio attraverso un comunicato stampa o un evento,

2. si desidera modificare l'URL del servizio, spostandolo da un server di testing ad uno pubblico.

In entrambi i casi è necessario assicurarsi che lo scopo principale del livello non venga alterato, ne che siano modificate ulteriori proprietà.

7.3 Post-pubblicazione

In seguito alla pubblicazione il livello sarà disponibile a qualunque utente. In questo stato sarà ancora possibile modificarlo e tali modifiche saranno visualizzate quasi in tempo reale all'utenza. Nel caso si desiderino apportare delle modifiche è opportuno considerare che:

- le informazioni di Layar restano nella cache del client per circa un'ora: le modifiche apportate all'interfaccia del livello potrebbero non apparire immediatamente agli utenti finali;
- se si modifica il `poiURL` si riceveranno, per qualche tempo, richieste per il vecchio servizio;
- è consigliato aggiungere alcuni tag al livello, se si scopre che l'utenza ha difficoltà a raggiungerlo;
- si raccomanda di creare un livello duplicato di prova, per verificare le modifiche, prima di rendere effettive le nuove impostazioni; una volta che un nuovo strato è stato approvato per la pubblicazione, gli sviluppatori possono sostituirlo con un altro: in questo modo, gli utenti non vedranno nè nuovo layer, nè alcun cambiamento.

Nota: Concludiamo questa parte dedicata alla realizzazione di uno strato riportando alcune note rilasciate dal team di Layar (per alcune di esse è necessario fare riferimento alla figura precedente).

- La funzionalità per sviluppatori di rinominare/cambiare editore non è ancora disponibile.
- Dopo che un layer viene ri-approvato, esso passa nello stato unpublished in cui:
  - non è più visibile nelle gallerie (ricerca, popolare, primo piano, ecc.);
  - è ancora visibile e può essere utilizzato dagli utenti che posto un bookmark su di esso;
  - può essere pubblicato nuovamente.

  La manutenzione su un livello avviene quindi sotto questi vincoli: i nuovi utenti non possono vedere il livello, ma un utente finale che lo avesse tra i preferiti può richiamarlo. E'conveniente che all'apertura dello strato venga restituito un messaggio informativo sullo stato dello strato.
- Lo status frozen può essere raggiunto solo dallo stato unpublished. In questa modalità, uno strato:
  - non è più visibile nelle gallerie;
  - può essere ancora visibile tra i preferiti e può essere acquistato, ma gli utenti non possono più accedervi (messaggio di errore: "'Layer non più disponibile");
  - può essere eliminato anche rimosso in modo definitivo, se lo sviluppatore è interessato.

[17].

Capitolo 8

Strumenti e servizi

Nonostante Layar sia un'applicazione ancora "giovane", si sono già sviluppati diversi servizi e tool intorno ad essa. Nel seguito saranno elencati gli strumenti gratuiti ed open source, più "promettenti". I tool di sviluppo non si limitano certo a questi elementi e sono in continuo sviluppo. Gli utenti possono venire a conoscenza di nuovi strumenti tramite il blog ufficiale di Layar Reality Browser (`http://site.layar.com/company/blog/`).

Hoppala! Augmentation , `http://www.hoppala.eu`, fornisce uno servizio semplice per utenti creativi, privi però di conoscenze di tipo tecnico, per iniziare a sperimentare la Realtà Aumentata e Layar: con solo alcuni clic del mouse è possibile creare il proprio strato e pubblicarlo. Il servizio viene fornito tramite un'interfaccia con una mappa a schermo intero, dalla quale è possibile selezionare i propri POI e modificarne le proprietà; è anche possibile caricare immagini e icone, file audio, video, contenuti multimediali e modelli 3D. Hoppala! Augmentation offre anche un servizio di hosting, ed è accessibile tramite browser, senza bisogno d'installare alcun software aggiuntivo.

LightRod , `http://www.lightrod.org/`, è un gestore di contenuti open source ottimizzato per la ricerca di POI: esso necessita di un database di punti, dotati di latitudine e longitudine, in input, e processa una richiesta di ricerca ad una determinata posizione restituendo i risultati trovati in ordine di distanza. Il software è scalabile per qualsiasi numero di record e utilizza le curve di riempimento bidimensionali, per trovare i risultati più vicini senza tener conto della distanza dall'origine della richiesta.

La soluzione LightRod non soffre di problemi di scalabilità e fa uso di codici di Peano, che convertono in modo efficace un piano bidimensionale in una linea monodimensionale: per qualsiasi insieme di record ritorna una risposta con un tempo costante. LightRod inoltre integra Google Maps e altri strumenti di cartografia online, permettendo la visualizzazione immediatamente dei punti del database (simile a ciò che accade nel sito di testing di Layar).

porPOIse , `http://code.google.com/p/porpoise`, è un framework aperto basato su PHP che permette di sviluppare layer facilmente: converte i punti di interesse in risposte al client e agisce come un server per esso, inoltre esegue la formattazione JSON e il calcolo della distanza. Esso include alcune migliorie tecniche per incrementare la sicurezza, la facilità di aggiornamento e le prestazioni per grandi insiemi di dati. Con questo framework gli sviluppatori possono costruire un semplice strato molto velocemente o estendere liberamente codice preesistente per creare esperienze ricche ed interattive. porPOIse supporta modelli 3D, azioni, filtri e autenticazione.

Parte V

Un layer turistico

Capitolo 9

Motivazioni

Il settore dei beni culturali si presta in particolar modo all'utilizzo dell'AR, specie per quanto riguarda le informazioni su monumenti, musei, edifici ed opere artistiche in generale. Una delle più riuscite applicazioni in quest'ambito è stata ideata dal Power House Museum di Sydney (`http://www.powerhousemuseum.com/`) facendo interagire la Realtà Aumentata con le risorse web 2.0 del museo: sfruttando una consistente galleria fotografica di oltre 400 immagini storiche di Sydney, ogni utente, puntando la videocamera del proprio device, può acquisire in tempo reale dati storici relativi ai luoghi selezionati, ricevendoli direttamente dalle schede del museo o addirittura da contenuti di Flickr. Il tutto è stato realizzato utilizzando proprio Layar.

Date le sue caratteristiche Layar Reality Browser si adatta molto bene a questo settore: le informazioni relative a ciò che viene inquadrato come nomi di opere, palazzi o dei monumenti di riferimento, date storiche, foto, video, testi, curiosità, unitamente alle possibilità d'interazione con social network e visualizzazione di mappe satellitari, possono essere offerte agli utenti con un'unica applicazione e in ogni momento.

Nel seguito verrà presentato un layer a scopo turistico realizzato per i visitatori della città di Padova. Il servizio è denominato Padova Tour e offre agli utenti la possibilità di partecipare a visite tematiche nel capoluogo veneto, basate su percorsi prestabiliti, integrate con diversi strumenti interattivi e collaborativi.

Capitolo 10

Scheda tecnica

Padova Tour offre all'utenza la possibilità di visitare oltre 90 luoghi d'interesse (POI) nella città di Padova, seguendo un percorso libero oppure attraversando cinque itinerari tematici:

1. Scrovegni, San Gaetano e Pedrocchi,
2. Palazzo della Ragione,
3. Ghetto ebraico e Duomo,
4. Prato della Valle e Sant'Antonio,
5. Mura di Padova.

Per ogni POI l'utente finale può visualizzarne una descrizione in diverse lingue accompagnata da un'immagine, l'itinerario a cui fa riferimento, la distanza dalla posizione attuale (aggiornata in tempo reale) ed un valore di rating assegnato.

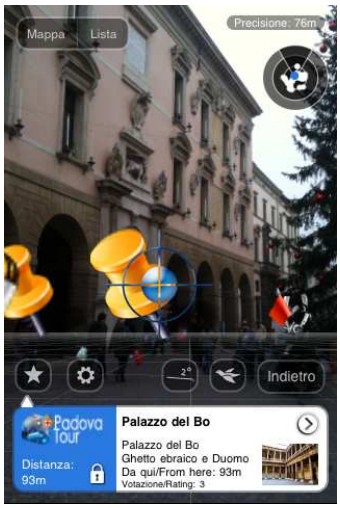

Figura 10.1: Interfaccia del layer Padova Tour.

Analizziamo quindi il funzionamento di PadovaTour: una volta effettuato l'accesso, gli utenti verranno reindirizzati ad una pagina di configurazione ove sarà possibile impostare diversi parametri, quali:

- *raggio di ricerca*: offre all'utente la possibilità di decidere l'area entro cui verrà effettuata la scansione delle informazioni, ovvero la regione da esaminare alla ricerca di punti d'interesse;
- *tipo di itinerario*: utilizzando un checkbox sarà invece possibile selezionare il/i percorso/i da visualizzare, e in questo modo si potrà scoprire solo i POI relativi a "Prato della Valle e Sant'Antonio" oppure a "Mura di Padova" o appartenenti ad entrambi;
- *visualizzazione di itinerari vicini*: quest'opzione permette di personalizzare maggiormente la propria esperienza con Layar Reality Browser. Mentre si segue il percorso "Palazzo della Ragione" si potranno vedere anche i POI appartenenti ad altri itinerari, a patto che questi si trovino nel raggio di 500 metri da un punto tra quelli selezionati; essenzialmente questa funzionalità offre all'utenza la possibilità di conoscere che a pochi passi dalla posizione attuale, si sviluppano altri percorsi tematici, dando così l'opportunità di decidere se proseguire nella visita oppure intraprenderne una nuova;
- *rating di ricerca*: aggiunge un ulteriore criterio di selezione alla ricerca. Attraverso uno slider, l'utente potrà impostare un valore tra 1 e 5 (ovvero l'intervallo di valutazione concesso) e richiedere di visualizzare tutti i POI con un punteggio superiore a quello prescelto (il valore di default 1 visualizza tutti i POI).

Gli utenti inoltre hanno la possibilità di interagire con gli elementi del layer tramite diversi comportamenti visualizzabili nella "scheda delle azioni":

- *Take me there*: visualizza una mappa interattiva che propone un percorso dalla posizione attuale al POI selezionato, utilizzando uno strumento Google Maps;
- *Audio/Listen:* permettono di utilizzare un audio guida in lingua italiana ed inglese, che accompagna l'utente nella sua visita. Il collegamento ai suddetti elementi multimediali non è visualizzabile ovunque poiché essi sono associati solo a determinati punti del layer;
- *Guestbook*: tramite quest'azione l'utente verrà reindirizzato verso una pagina web (aperta all'interno della WebView) ove sarà possibile acquisire informazioni relative al POI selezionato, compilare un form per inserire un proprio commento e visualizzare quelli dei visitatori precedenti, ma anche dare una valutazione personale con un voto da 1 a 5 (tale feedback verrà poi utilizzato per discriminare i diversi POI durante la ricerca). *Guestbook* sarà visualizzabile interagendo con un qualsiasi POI nello strato esclusi i punti che appartengono ad un itinerario non selezionato nella pagina delle impostazioni;
- *Cambio percorso*: permette di passare da un itinerario ad un altro. Quest'azione sarà visibile se l'utente ha attivato l'opzione "*Visualizzazione di itinerari vicini*", e solamente su POI che appartengono ad un itinerario "vicino".

All'interno dello strato si possono distinguere POI semplici, POI con una valutazione elevata (superiore a 4/5), POI di itinerari vicini e POI con contenuto multimediale: ciascuno di questi sarà associato ad un'icona diversa, di modo che si possano discernere facilmente. Inoltre può rivelarsi utile riconoscere anche i punti iniziali di ciascun percorso, permettendo agli utenti di organizzare la propria visita in maniera appropriata.

Capitolo 11

Il servizio Padova Tour

Nota: Tutto il materiale presentato nel seguito di questa trattazione è ospitato gratuitamente su un server remoto di proprietà di Altervista. Tale scelta è motivata dalla necessità di accedere ai diversi contenuti in ogni momento tramite un servizio affidabile, sia in fase di testing che nella successiva fase di pubblicazione, senza però possedere le infrastrutture adatte. I servizi di hosting Altervista supportano la realizzazione di layer per Layar Reality Browser fornendo un supporto a PHP 5 e un database MySQL.

Durante le interrogazioni alla base di dati in uso sarà impiegato il motore di query MyISAM: ciò è imposto dall'ambiente di lavoro poiché Altervista non fornisce altri motori per i propri database; questo vincolo comporta l'implementazione di maggiori controlli ed elaborazioni durante l'interazione con il database. Una scelta migliore è certamente quella di utilizzare un motore InnoDB che permette di usufruire delle proprietà di integrità referenziale. Quest'ultima considerazione è applicabile anche alla sezione precedente, in cui sono stati presentati e sviluppati alcuni servizi per la gestione di layer.

Infine si vuole evidenziare che il livello è definito come *generic*, ma il servizio elaborato è predisposto comunque per accogliere possibili modelli 2D e 3D in caso di eventuali e future estensioni (si vedano le funzioni `GetTransform` e `GetObject`).

11.1 Definizione

Il primo passo nella realizzazione dello strato Padova Tour è l'accesso al portale di gestione degli strati dove andranno riportate e definite le sue proprietà basilari. Vediamo quindi le più importanti, escludendo i dettagli relativi al look&feel che riguardano i custom CIW ed i colori utilizzati nel livello:

- il *poiURL* è localizzato presso l'indirizzo:
  `http://padovaar.altervista.org/layar/padovalayar.php`;
- il servizio è rilasciato ad uso gratuito: nel seguito sarà utilizzato del materiale coperto da licenza non-commercial;
- lo strato sarà erogato nell'intero territorio italiano: in questo modo sarà maggiormente visibile all'interno della Layar Gallery;
- non sarà definito alcun bounding box: tale scelta è stata elaborata al fine di favorire la conoscenza della presenza dello strato ai potenziali turisti;

- inseriamo tre nuovi filtri nella relativa area di gestione: oltre al `RANGE_SLIDER` di base, inseriamo un `CHECKBOX` per la selezione del percorso, una `RADIOLIST` per l'attivazione della funzionalità "*visualizzazione di itinerari vicini*" ed un `CUSTOM_SLIDER` per la gestione della ricerca basata sul rating.

Analizziamo ora i dettagli implementativi.

11.2 Implementazione

La seconda operazione è la strutturazione di un servizio personalizzato: lo strato Padova Tour sarà costituito da script realizzati in linguaggio PHP, organizzati sulla base del codice presentato precedentemente, opportunamente modificato per offrire le funzionalità descritte nella scheda tecnica. L'implementazione sarà integrata ad un database MySQL nel quale saranno contenute le tabelle POI_Table ed ACTION_Table: i punti di interesse elaborati dall'applicazione rappresenteranno infatti luoghi ed opere d'interesse artistico-culturale, ciascuno dotato di propri attributi; per ognuno di essi sarà visualizzato un titolo in lingua italiana (`title`), uno in lingua inglese (`line2`), l'itinerario di appartenenza (`line3`) e la distanza dal client (`line4`) tramite l'utilizzo della stringa `%distance%`. Inoltre alcuni POI saranno connessi ad azioni descritte tramite un titolo (`label`), un target (`URI`), dalla presenza di un trigger (`autoTriggerOnly`) e dal suo raggio di attivazione(`autoTriggerRange`).

Prima di procedere nella descrizione dei dettagli implementativi si dovrà modificare la tabella contenente le informazioni sui POI ed aggiungerne al database una nuova:

- POI_Table: inseriamo due nuovi campi "rating" e "tot", che assoceranno ciascun POI ad un proprio valore di rate (sottomesso dagli utenti) ed al numero di votanti che hanno partecipato alla valutazione dell'elemento; questi attributi verranno impiegati durante la ricerca dei POI e saranno integrati alle funzioni del guestbook.

```
ALTER TABLE `POI_Table` ADD `tot` INT( 11 ) NULL DEFAULT NULL;
ALTER TABLE `POI_Table` ADD `rating` DOUBLE( 9,5 ) NULL DEFAULT NULL;
```

- route: questa tabella conterrà i dati di ciascun itinerario quali nome e identificativo. Inseriamo un campo "id" come chiave primaria ed imponiamo un vincolo di unicità sul nominativo "route": se infatti esistessero due percorsi con lo stesso nome, un utente finale che utilizza lo strato riceverà le informazioni riguardanti entrambi senza possibilità di discernere, generando così una user-experience decisamente confusa.

```
--
-- Table structure for table `route`
--
CREATE TABLE IF NOT EXISTS `route` (
`ID` int(50) NOT NULL,
`route` varchar(50) NOT NULL,
PRIMARY KEY (`ID`),
UNIQUE (`route`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Listing 11.1: Istruzione SQL per la creazione della tabella route.

Possiamo quindi procedere alla realizzazione del servizio. La prima operazione consisterà nel modificare il main dello script, impostandolo perché gestisca i dati inseriti nella richiesta, la quale conterrà le impostazioni scelte dall'utente per i filtri del livello. La seguente stringa rappresenta la richiesta per il layer Padova Tour: tramite l'API test page per sviluppatori è possibile infatti catturare ogni richiesta, inoltrata da un client verso il servizio, e conoscerne così i valori realmente contenuti.

```
http://padovaar.altervista.org/layar/padovalayar.php?lang=en&countryCode=
    IT&lon=4.887339&userId=6f85d06929d160a7c8a3cc1ab4b54b87db99f74b&
    developerId=18330&developerHash=1172
    ba19e580e3e847168d7252a887a44be069dc&RADIOLIST=0&CHECKBOXLIST=2%2C3%2C4
    %2C5&version=4.0&radius=1500&CUSTOM_SLIDER=1&timestamp=1286433531629&
    lat=52.377544&layerName=padovalayer&SEARCHBOX=&accuracy=100
```

A differenza di quanto svolto con gli script precedenti, in questo caso è necessario raccogliere i valori dei diversi filtri, i quali verranno poi applicati ai parametri per la scansione della POI_Table; modifichiamo l'istruzione che realizza l'array per la lettura dei parametri della richiesta utilizzando la seguente:

```
$keys = array( "layerName", "lat", "lon", "radius", "CHECKBOXLIST", "
    RADIOLIST", "CUSTOM_SLIDER");
```

Una volta raccolti i dati della richiesta, sarà possibile procederne all'elaborazione, ed il primo valore che utilizzeremo sarà associato al filtro `CHECKBOXLIST`: il contenuto di questo elemento determina, come anticipato, il percorso di appartenenza per ciascuno dei risultati dell'interrogazione. Al fine di eseguire correttamente la selezione dei POI in base all'itinerario, applicheremo alla stringa che lo descrive (`line3`) un'espressione regolare nella quale si dovrà tenere in considerazione la possibilità che siano utilizzati anche più percorsi contemporaneamente. Estraiamo e modifichiamo il contenuto della variabile `$value['CHECKBOXLIST'],` separando ogni valore con un simbolo "|" (OR), facilmente interpretabile all'interno dell'espressione. Inseriamo il seguente codice prima della costruzione della query MySQL:

```
//Load the checkbox-options and place an OR symbol between values
$value['CHECKBOXLIST'] = str_replace(',','|',$value['CHECKBOXLIST']);
$value['CHECKBOXLIST'] = str_replace('%2C','|',$value['CHECKBOXLIST']);
//Generate a string for the regex
$regex='('.$value['CHECKBOXLIST'].')$';
```

Listing 11.2: Istruzione per la realizzazione della regex di selezione degli itinerari.

Le istruzioni appena proposte generano una stringa costituita da un elenco di valori numerici, poiché ciascuna opzione nel filtro `CHECKBOX` è associata ad un numero intero, separati da un simbolo di OR; un'espressione così formata permetterebbe di ricercare gli itinerari in base ad una corrispondenza tra questo ed un valore numerico. La soluzione più semplice e naturale, nonché efficace, è quella di associare i valori delle opzioni del filtro all'identificativo di ciascun percorso. Un'ulteriore soluzione potrebbe essere rappresentata dal ricercare ciascuna cifra all'interno del nome di un itinerario: questa porta però a comportamenti scorretti e non sempre desiderabili, in quanto alcuni percorsi condivideranno la stessa cifra o non ne presenteranno alcuna all'interno del proprio nome.

Utilizziamo quindi la stringa dell'espressione regolare, per interrogare la base di dati alla ricerca dei nomi di ciascun itinerario selezionato dall'utente; ottenuti tali nominativi, li sostituiremo ai corrispettivi numerici nella regular expression che riutilizzeremo successivamente. Di seguito è proposto il codice che realizza la funzione presentata.

```
//Get the names of the routes
$sql = $db->prepare( "
                SELECT route
                FROM route
                WHERE ID REGEXP :regex" );
//PDOStatement::bindParam() binds the named parameter markers to the
   specified parameter values.
$sql->bindParam( ':regex', $regex, PDO::PARAM_STR);
// Use PDO::execute() to execute the prepared statement $sql.
$sql->execute();
//Fetch data from query
$routes = $sql->fetchAll(PDO::FETCH_ASSOC);
//Rewrite the regex
$first = true;
foreach($routes as $elem){
        if($first) {
                $regex = '^('.$elem['route'];
                $first = false;
        }
        else $regex = $regex.'|'.$elem['route'];
}
$regex = $regex.')$';
```

Listing 11.3: Selezione degli itinerari dal database ed aggiornamento della regular expression.

La ricerca dei POI potrà ora valutare quali elementi appartengono all'itinerario prescelto sul client; per considerare questo parametro, è necessario apportare alcune modifiche alla ricerca dei POI sviluppata nei servizi di base: andremo infatti ad inserire nella query una clausola `WHERE`, che selezionerà solo le righe della tabella a cui è possibile applicare l'espressione regolare precedentemente realizzata. L'istruzione utilizzata è la seguente:

```
WHERE line3 REGEXP :regex
```

Il secondo requisito da esaminare, durante la selezione dei punti di interesse dal database, è il livello di valutazione che questi devono possedere. Per implementare la ricerca basata su rating è necessario aggiungere tale criterio all'interrogazione tramite l'istruzione:

```
WHERE rating >= :rate
```

All'interno della chiamata al metodo `prepare`, le due clausole `WHERE` possono essere unificate tramite `AND`, mentre le variabili `:regex` e `:rate`, possono essere collegate tramite binding alle rispettive variabili di scripting, come riportato nel codice seguente.

```
$sql = $db->prepare( "
                SELECT id, attribution, title, lat, lon, imageURL, line2,
                   line3, line4, dimension, alt, relativeAlt, doNotIndex,
                   inFocus, actions, object, transform, (((acos(sin((:lat1
                    * pi()/180)) * sin((lat * pi()/180)) + cos((:lat2 * pi
                   ()/180)) * cos((lat * pi()/180)) * cos((:long  - lon) *
                    pi()/180))) * 180/pi()) * 60 * 1.1515 * 1.609344 *
                   1000) as distance
```

```
                FROM POI_Table
                WHERE (line3 REGEXP :regex AND rating >= :rate)
                HAVING distance < :radius
                ORDER BY distance ASC
                LIMIT 0, 50 " );
//PDOStatement::bindParam() binds the named parameter markers to the
    specified parameter values.

$sql->bindParam( ':rate', $value['CUSTOM_SLIDER'], PDO::PARAM_INT);
$sql->bindParam( ':lat1', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':lat2', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':long', $value['lon'], PDO::PARAM_STR );
$sql->bindParam( ':radius', $value['radius'], PDO::PARAM_INT );
$sql->bindParam( ':regex', $regex, PDO::PARAM_STR );
```

Listing 11.4: Preparazione della query SQL.

Una volta eseguita la query, questa conterrà i risultati rispondenti ai parametri della richiesta, i quali saranno inviati alla funzione `Gethotspots` di modo che siano visualizzati all'utente: la procedura descritta in precedenza non permette però di conoscere il rating dei diversi elementi e dovrà essere ampliata. Per ottenere quest'informazione all'interno dell'applicazione client, si dovrà associare il valore della valutazione ad una delle stringhe della BIW, tramite le seguente istruzioni:

```
//Place the rating value inside the BIW
include_once 'rating.php';
$poi["attribution"] = getRating($poi["id"],$db);
```

Listing 11.5: Inserimento del valore di rating. Modifica della funzione `Gethotspots`().

Il codice proposto richiede di includere il file `rating.php`, nel quale è contenuta la definizione della funzione `getRating`(); quest'ultima riceve in input l'id dell'elemento in analisi ed il database in uso dal servizio (avente una connessione già aperta). Il funzionamento di `getRating()` è molto semplice in quanto esegue una query sulla tabella "POI_Table" , restituendo alla pagina principale una stringa formattata riportante la valutazione dell'elemento in formato numerico.

```
function getRating($id,$db){
        $rate = 3;
        $sql_rate= $db->prepare("SELECT rating FROM POI_Table WHERE id = :
            id " );
        $sql_rate->bindParam(':id', $id, PDO::PARAM_INT );
        $sql_rate->execute();
        $rates = $sql_rate->fetchAll( PDO::FETCH_ASSOC);
                foreach($rates as $row){
                $rate = "Votazione/Rating: ".round($row['rating'],3)."";}
        return $rate;}
```

Listing 11.6: Funzione `getRating`().

Il servizio sviluppato permette ora la selezione dei POI in base sia all'itinerario di appartenenza che alla valutazione espressa dagli utenti e associata a ciascun elemento.

Resta quindi l'elaborazione dei valori del filtro `RADIOLIST`, grazie a quali l'utente potrà attivare la feature che permetterà visualizzare punti appartenenti ad itinerari vicini a quelli prescelti. L'implementazione di tale funzionalità è ottenuta nel modo seguente: in primo luogo, inseriamo un controllo del valore `$value['RADIOLIST']`, dopo la chiamata a `Gethotspots`; se il valore `RADIOLIST` è 0 (modalità di visualizzazione attiva), viene invocata la funzione `Intersection`, definita nel file `route_intersection.php`, la quale restituirà un array di hotspot, appartenenti a percorsi differenti da quelli selezionati dall'utente: tali elementi andranno quindi aggiunti all'insieme di risultati già convalidati (attraverso un'operazione di push), per essere restituiti al client. Il seguente listato riporta la funzione di controllo, l'invocazione del metodo `Intersection` e l'inserimento dei risultati nella risposta.

```
if($value["RADIOLIST"]==0){
   //Find the intersections of the routes
   include_once'route_intersection.php';
   $intersect = Intersection($db, $response["hotspots"], $regex);
   //Add finded hotspots to the response
   $j= count($response["hotspots"]);
   foreach($intersect as $hotspot){
         array_push($response["hotspots"],$hotspot);
         $j++;
   }
}
```

Listing 11.7: Invocazione del metodo `Intersection`() per la ricerca di intersezioni.

Analizzeremo ora la funzione che esegue il calcolo delle "intersezioni" degli itinerari; nel seguito consideriamo un POI x appartenente all'intersezione di due percorsi $w \bigcap z$ se e solo se, presi $x \in w$ e $y \in z$ due elementi appartenenti a due itinerari distinti, x è posizionato (rispetto alle sue coordinate espresse in latitudine e longitudine) all'interno di un'area circolare di raggio r e centrata in y. `Intersection()` riceve in input un database, col quale è già instaurata una comunicazione, l'array associativo multidimensionale di elementi selezionati dall'utente, per i quali effettuare la ricerca di POI vicini, e l'espressione regolare (`$regex`) realizzata nella funzione main; l'output sarà invece costituito da un'insieme di POI realizzati ad hoc, in un certo senso "fittizi", le cui informazioni, azioni e rappresentazioni sono parzialmente alterate rispetto a quanto presente nel database dell'applicazione.

La chiamata esegue una query sul database, interrogandolo per tutti gli elementi non richiesti dall'utente: tale operazione è realizzata inserendo la clausola `WHERE line3 NOT REGEXP :regex`, la quale seleziona difatti gli elementi della tabella del database che non soddisfano il criterio imposto dall'espressione regolare. Prima di eseguire la query è però necessario riconvertire i valori di latitudine e longitudine presenti nell'array (in quanto vi è stata precedentemente applicata la funzione `ChangetoIntoLoc`()).

Per ogni elemento appartenente all'intersezione, verrà quindi estratto l'identificatore per l'itinerario di appartenenza, tramite una query alla tabella "route", grazie al quale potremo costruire un'azione che permetta al client di cambiare il percorso visualizzato: supponiamo che un utente stia percorrendo l'itinerario "Palazzo della Ragione", allora questi potrà visualizzare un POI appartenente a "Ghetto ebraico e Duomo" (avendo ovviamente attivato la funzionalità di ricerca delle intersezioni),

il quale concederà al client la possibilità di eseguire uno switch, caricando le informazioni relative a "Ghetto ebraico e Duomo". L'implementazione di questa funzionalità è ottenuta inserendo il seguente URI nel campo "action" di un'azione: `layar://padovalayer/?CHECKBOXLIST=$it` (in questo caso può essere utile impostare anche una funzionalità di auto trigger, in modo da avvisare l'utente non appena questi si trova ad una distanza massima di `$range` metri da un punto d'intersezione).

Completiamo infine la procedura formalizzando l'oggetto hotspot che sarà restituito per ciascuna intersezione: questo sarà associato ad un insieme di informazioni ad esso collegate (come nome, itinerario ed immagine) e all'azione appena realizzata; si noti che non vengono inseriti i comportamenti che autorizzano l'utilizzo del guestbook o l'ascolto dell'audio guida in quanto saranno disponibili solo nel percorso di pertinenza.

Il codice sottostante si riferisce alla funzione `Intersection()` appena descritta, imponendovi però un limite massimo di cinque risultati per ciascun punto originario.

```
function Intersection($db, $resp, $regex){
    //Set the area for the intersection search.
        $range= 500;
        // Create an empty array named response
        $result = array("hotspots");
        //index for scanning true-hotspots
        $i=0;
        //for each result of the main search, find other POIs that aren't
           in the same route but are within a route intersection
        foreach ( $resp as $poi)  {
        //convert data to a proper format
        $lat = $poi['lat']/ 1000000;
        $lon = $poi['lon']/ 1000000;
        $sql_int = $db->prepare("
                        SELECT *, (((acos(sin((:lat1 * pi()/180)) * sin((
                           lat * pi()/180)) + cos((:lat2 * pi()/180)) *
                           cos((lat * pi()/180)) * cos((:long  - lon) * pi
                           ()/180))) * 180/pi()) * 60 * 1.1515 * 1.609344
                           * 1000) as distance
                        FROM POI_Table
                        WHERE line3 NOT REGEXP :regex
                        HAVING distance < :radius
                        ORDER BY distance ASC
                        LIMIT 5 "  );
        // PDOStatement::bindParam() binds the named parameter markers to
           the specified parameter values.
        $sql_int->bindParam( ':lat1', $lat, PDO::PARAM_STR );
        $sql_int->bindParam( ':lat2', $lat, PDO::PARAM_STR );
        $sql_int->bindParam( ':long', $lon, PDO::PARAM_STR );
        $sql_int->bindParam( ':radius', $range, PDO::PARAM_INT);
        $sql_int->bindParam( ':regex', $regex, PDO::PARAM_STR );
        // Use PDO::execute() to execute the prepared statement $sql_int
        $sql_int->execute();
        $i = 0;
        // Use fetchAll to return an array containing all of the remaining
            rows in the result set
        // Use PDO::FETCH_ASSOC to fetch $sql query results and return
           each row as an array indexed by column name.
        $pois = $sql_int->fetchAll(PDO::FETCH_ASSOC);
```

```
        // Put each POI information into $response["hotspots"] array
        foreach ( $pois as $poi ) {
        //Retrieve the route-id
                $line_regex = '^'.$poi["line3"].'$';
                $sql_route = $db->prepare("SELECT ID FROM route WHERE
                   route REGEXP :name LIMIT 1");
                $sql_route->bindParam( ':name', $line_regex, PDO::
                   PARAM_STR );          $sql_route->execute();
                $pois_route = $sql_route->fetchAll(PDO::FETCH_ASSOC);
                foreach ($pois_route as $elem) {$it = $elem["ID"]; }
        //action for the uri
        $label = "Cambia/Change";
        $action ="layar://padovalayer/?CHECKBOXLIST=$it";
                $poi["actions"] = array(array('label' => ($label), 'uri'
                   => ($action), 'autoTriggerRange' => $range, '
                   autoTriggerOnly' => TRUE));
                // Store the integer value of "lat" and "lon" using
                   predefined function ChangetoIntLoc.
                $poi["lat"] = ChangetoIntLoc( $poi["lat"] );
                $poi["lon"] = ChangetoIntLoc( $poi["lon"] );
                // Change to Int with function ChangetoInt.
                $poi["type"] = ChangetoInt( $poi["type"] );
            $poi["dimension"] = ChangetoInt( $poi["dimension"] );
            $poi["alt"] = ChangetoInt( $poi["alt"] );
            $poi["relativeAlt"] = ChangetoInt( $poi["relativeAlt"] );
        // Change to demical value with function ChangetoFloat
                $poi["distance"] = ChangetoFloat( $poi["distance"] );
                // Change the values of "doNotIndex" into boolean value,if
                    the value is not NULL. Otherwise, return NULL.
                $poi["doNotIndex"] = ChangetoBool( $poi["doNotIndex"] );
                // Change the values of "inFocus" into boolean value,if
                   the value is not NULL. Otherwise, return NULL.
        $poi["inFocus"] = ChangetoBool( $poi["inFocus"] );
        // Put the poi into the response array.
        $result["hotspots"][$i] = $poi;
        $i++;
                }//foreach
        }//else
        return $result["hotspots"];
}
```

Listing 11.8: Funzione `Intersection()`.

Il servizio così costruito, non imponendo alcun limite sul numero di risultati individuati sia nel main che nella funzione `Intersection()`, risulta possedere una complessità quadratica, essendo questa pari a *O(N-k)*k*, dove $k>1$ è il numero di punti non appartenenti all'itinerario selezionato e *N-k* è il numero di POI selezionati dall'utente (si noti che un POI deve essere inserito in un solo itinerario). L'analisi del caso peggiore, $k=N/2$, dimostra una complessità eguale a $O(N^2)$.

11.2.1 Miglioramento delle performance

Software di localizzazione al pari di Layar Reality Browser devono rispondere alle esigenze del client in tempo reale, o comunque entro intervalli temporali molto ristretti: minore sarà il tempo di attesa, maggiori saranno le probabilità che un utente scelga quella determinata applicazione, o come in questo caso un layer, poiché più performante (un po' come accade per i motori di ricerca online). L'implementazione fin qui realizzata può (e deve) essere migliorata al fine di ridurre i tempi di risposta tra client e server.

La selezione dei POI all'interno del database avviene circoscrivendo la regione di ricerca ad un'area limitata: utilizzando la formula di Haversine viene calcolata la distanza di ciascun punto dall'utente, la quale sarà verificata essere contenuta all'interno del range specificato. Includendo una nuova condizione nella query sarà possibile risparmiare alcune operazioni di calcolo, selezionando solo le righe le cui coordinate "lat" e "lon" sono comprese entro un intervallo di ammissibilità precomputato. Attraverso quest'espediente non sarà infatti necessario applicare il calcolo della distanza ad ogni singolo elemento della tabella, ma solamente a quelli che andranno inseriti nella risposta.

I gradi di latitudine sono calcolati rispetto ai paralleli terrestri, nei quali la separazione tra elementi contigui rimane pressoché costante su tutta la sfera: essa è infatti approssimabile al valore di 69,1 miglia (~111 chilometri); per quanto concerne le longitudini, invece, la situazione varia notevolmente poiché i meridiani terrestri sono maggiormente distanziati sulla linea dell'equatore, con un valore di 69,172 miglia (111,321 km), e convergono gradualmente a zero in prossimità dei poli.

Dall'analisi appena svolta, possiamo approssimare la distanza coperta tra due gradi di latitudine, per l'intera superficie terrestre, al valore di 69 miglia. Per quanto riguarda il valore di separazione tra due gradi di longitudine, esso varia a seconda della latitudine dei punti stessi e per cui non v'è applicabile lo stesso ragionamento appena svolto; tuttavia, poiché la latitudine è pari all'angolo che la verticale di un punto sulla superficie della Terra forma con il piano equatoriale, possiamo valutare lo spazio tra due meridiani come il coseno della latitudine (alla quale viene operato il confronto), moltiplicato per il valore massimo di distanza. In questo modo, data una distanza ed un punto d'origine, sarà possibile calcolare le coordinate dei punti che si trovano entro il raggio di ricerca prestabilito tramite una semplice conversione [13].

Si noti che quest'operazione modifica il meccanismo di selezione dei punti: utilizzando la formula di Haversine la scansione del database restituiva solamente i POI contenuti in una regione di forma circolare; utilizzando il nuovo metodo proposto, si andrà invece a circoscrivere un poligono di forma (approssimativamente) quadrata alla precedente, ottenendo un ampliamento delle dimensioni dell'area di ricerca, restituendo tutti i punti, del database, situati al suo interno.

All'interno del servizio, una volta ottenuta la posizione dell'utente ed il raggio di di ricerca, andremo ora a calcolare le coordinate dei quattro vertici che delimitano il perimetro della regione di selezione, interrogando il database per i punti le cui coordinate sono contenute nell'area appena definita. Per semplificare ulteriormente le operazioni, sarà sufficiente calcolare solo due vertici (opposti) del poligono. Si noti inoltre che le distanze calcolate dovranno essere convertite da miglia a metri. Di seguito è riportato il codice proposto per il main del servizio principale.

```
//Use a bounding box for filtering query result
//Convert miles to meters
$m= 69*1.609344*1000;
$tmp =  cos(deg2rad($value['lat']))* $m;
//Calculate absolute value
if ($tmp<0) {
        $tmp*=-1;
```

```
}
$latA = $value['lat'] - $value['radius']/$m;
$lonA = $value['lon'] - ($value['radius']/$tmp);
$latB = $value['lat'] + $value['radius']/$m;
$lonB = $value['lon'] + ($value['radius']/$tmp);

...

//Query updated
$sql = $db->prepare("
                SELECT id, attribution, title, lat, lon, imageURL, line2,
                   line3, line4, dimension, alt, relativeAlt, doNotIndex,
                   inFocus, actions, object, transform, (((acos(sin((:lat1
                    * pi() / 180)) * sin((lat * pi() / 180)) + cos((:lat2
                   * pi() / 180)) *cos((lat * pi() / 180)) * cos((:long  -
                    lon) * pi() / 180))) * 180 / pi()) * 60 * 1.1515 *
                   1.609344 * 1000) as distance
                FROM POI_Table
                WHERE
                        line3 REGEXP :regex AND
                        rating >= :rate AND
                        lon BETWEEN :lonA AND :lonB AND
                        lat BETWEEN :latA AND :latB
                ORDER BY distance ASC
                LIMIT 0, 50 ");
//PDOStatement::bindParam() binds the named parameter markers to the
   specified parameter values.
$sql->bindParam( ':rate', $value['CUSTOM_SLIDER'], PDO::PARAM_INT);
$sql->bindParam( ':lat1', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':lat2', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':long', $value['lon'], PDO::PARAM_STR );
$sql->bindParam( ':latA', $latA, PDO::PARAM_STR );
$sql->bindParam( ':latB', $latB, PDO::PARAM_STR );
$sql->bindParam( ':lonA', $lonA, PDO::PARAM_STR );
$sql->bindParam( ':lonB', $lonB, PDO::PARAM_STR );
$sql->bindParam( ':regex', $regex, PDO::PARAM_STR );
```

Listing 11.9: Modifica del servizio: miglioramento della query di ricerca.

Questo metodo dovrà essere applicato sia alla ricerca principale del servizio, che al calcolo delle intersezioni tra i percorsi. Come si può notare, l'impiego della distanza ottenuta tramite la formula di Haversine non è stato eliminato, poiché consentirà di ordinare i risultati in base alla loro distanza dall'utente.

Effettuando alcuni rilievi, interrogando il database di Padova Tour contenente circa 100 elementi, tramite un terminale iPhone 3Gs all'interno di una rete wireless pubblica, è stato rilevato che il tempo medio di risposta, utilizzando quest'ultima implementazione, si riduce di circa l'11%. Per quanto riguarda l'analisi della complessità temporale, considerando *O(1)* le operazioni matematiche, essa diviene *O(m)*, dove m indica il numero di record le cui latitudini e longitudini rientrano negli intervalli prescelti; nel caso peggiore, ovvero $m=N$, otteniamo una complessità di *O(N)*. Prendendo in esame l'algoritmo completo (compreso del metodo `Intersection`()), la complessità diviene *O(m-k)*k*, dove

m-k indica il numero di POI appartenenti all'itinerario scelto dall'utente e compresi nella regione determinata, che nel caso peggiore (*m=N* e *k=N/2*) ritorna $O(N^2)$. L'analisi della complessità evidenzia quindi come il metodo sia maggiormente efficace limitatamente a ricerche "locali", ovvero il cui raggio d'azione non copra l'intera area su cui sono disposti gli elementi del sistema.

11.2.2 Visualizzazione

Per favorire l'esperienza dell'utente finale è utile identificare i POI con caratteristiche diverse in maniera univoca. Ad esempio i POI con un alto rate possono essere evidenziati, aiutando così l'utenza nel riconoscere a prima vista gli elementi più "apprezzati".

Come riportato nella scheda tecnica, si è scelto di evidenziare quattro diverse tipologie di punti d'interesse: punti contenenti materiale multimediale, punti di "intersezione", punti con rate elevato e punti "d'inzio percorso". A tal proposito è stato realizzato per ciascuno un differente custom icon set prelevando alcune icone da un icon set gratuito reperibile all'indirizzo `http://www.iconspedia.com`.

La prima operazione per usufruire dei custom CIW è di dichiararli all'interno della definizione del layer alla voce "Look&feel"; si ricordi che ogni set dovrà essere composto di un massimo di 4 icone, una per ciascuno stato di visualizzazione (`focus`, `inner`, `middle` e `outer`) . Successivamente è necessario attribuire il giusto icon set ad ogni punto, modificando all'interno del servizio il campo `type` di ogni hotspot: durante l'esecuzione del `GetHotspots()`, inseriamo alcune istruzioni per impostare il valore della variabile `$poi['type']` al valore corretto come riportato nel seguente listato.

```
//Create a var to check if the type is setted
$setType = false;
if($setType==false){
        include_once 'start.php';
        $start = isStart($poi["id"],$db);
        if($start == 1){
                $poi["type"] = 5;
                foreach($poi["actions"] as $action){
                        $uri = $action["uri"];
                if(preg_match("/^(audio|video)/",$uri)){
                                $poi["type"]=6;
                   break;
                }
                }
          $setType = true;
        }
}
//Set the custom icon for high rated pois
      if($setType==false){
      $rating = $poi["attribution"];
      preg_match("/(\d)\./", $rating, $matches);
      if($matches[0]>=4){
        $poi["type"]=4;
          $setType = true;
      }
}
//Set custom icon for audio pois
if($setType==false){
        foreach($poi["actions"] as $action){
                $uri = $action["uri"];
```

```
            if(preg_match("/^(audio|video)/",$uri)){
                        $poi["type"]=2;
            $setType=true;
             break;
             }
           }
}
//If the type is yet undefined
if($setType==false){
        $poi["type"] = 1;
}
```

Listing 11.10: Impostazione del parametro `type`. Modifica del ciclo internoalla funzione `Gethotspots`()

Osserviamo il codice appena esposto: la prima istruzione imposta una variabile booleana di controllo, utilizzata per verificare che il campo `type` non sia già stato impostato. Successivamente viene controllato se il punto determina la posizione iniziale di un itinerario, verificando anche la presenza di collegamenti a materiale multimediale, invocando la funzione `isStart`() che controlla un valore "isStart" associato a ciascun elemento della tabella POI_Table (si utilizzi l'istruzione ALTER per modificare la tabella). La struttura di `isStart`() è molto simile a quella della funzione `getRating`() e non verrà discussa. Se il primo controllo non ha avuto esito positivo, viene eseguita un'operazione di pattern matching utilizzando il metodo `preg_match`(), con la quale si catturerà il valore di rate dell'elemento in analisi contenuto nella stringa `attribution` (un'altra soluzione equivalente è la definizione di una nuova funzione all'interno dello script `rating.php`): se tale valore è maggiore o uguale a quattro, ci troviamo di fronte ad un punto ad alta valutazione e dobbiamo modificarne il tipo di icona. La detection di punti con collegamenti multimediali, avviene interrogando l'array delle azioni possedute: se il POI contiene un link con MIME di tipo `audio://`, cambiamo l'icona associata, indipendentemente dal suo rate. Infine se il campo type non è stato impostato applichiamo al punto in esame un'icona di default.

Assegnamo un'icona di riconoscimento anche agli hotspot che segnalano un'intersezione di percorsi: nel file `route_instersection.php`, andremo a modificare il valore di `$poi['type']` in fase di definizione dell'elemento, tramite l'istruzione seguente:

```
$poi["type"] = 3;
```

11.3 Guestbook

Tramite l'esecuzione dell'azione *Guestbook* gli utenti potranno visitare un libro degli ospiti virtuale, all'interno della WebView dello strato (la pagina è ugualmente accessibile dal web utilizzando lo stesso indirizzo).

Lo script che genera le pagine del guestbook si basa sul seguente principio: esso estrae l'id del POI da visualizzare, contenuto tra i parametri dell'indirizzo tramite cui vi si accede, e una volta identificato il target della richiesta ne visualizza le informazioni associate. Ogni pagina guestbook è strutturata in tre sotto menù:

- *Informazioni*: vengono reperite alcune informazioni (`title`, `line2` e `image`) che vengono visualizzate all'utente. Inoltre viene restituito il rating dell'elemento (in forma grafica) unitamente alla possibilità di sottoporre la propria valutazione sul POI;
- *Commenta*: tramite un form, è possibile aggiungere un commento personale alla pagina;
- *Visualizza commenti*: permette di leggere le opinioni di altri utenti, limitatamente all'elemento selezionato.

L'elemento target è riconosciuto utilizzando la variabile speciale `$_REQUEST` applicata alla stringa della richiesta: essa è costituita dall'URL della pagina seguito da un parametro `‘‘?id=X’’`, dove X indica un identificatore. Per la realizzazione di queste feature, è stata implementata una classe aggiuntiva "guestbook" , la quale incorpora tutte le funzioni necessarie alla raccolta e alla visualizzazione dei dati, nonché alla sottomissione dei feedback inviati dall'utente.

```
//Include the guestbook.class file
include_once "guestbook.class.php";
//Create a new guestbook object
$guestbook = new Guestbook();
```

Listing 11.11: Creazione di un oggetto `Guestbook`.

Il costruttore di quest'oggetto, che andrà incluso nello script che genera la pagina guestbook, esegue una connessione al database del servizio che verrà utilizzata da ciascuna delle procedure al suo interno, esposte nel seguito della trattazione.

11.3.1 Visualizzazione immagini

La prima funzione che discuteremo riguarda la visualizzazione di immagini, all'interno del guestbook, dotate di risoluzione e dimensioni maggiori rispetto a quelle disponibili nello strato, dove le rappresentazioni grafiche sono vincolate alle dimensioni della BIW. A supporto di quest'operazione verrà inserito un nuovo attributo "image" nella tabella "POI_Table" del tutto identico al campo "imageURL".

```
ALTER TABLE ‘POI_Table‘ ADD ‘image‘ VARCHAR( 255 ) NULL DEFAULT NULL;
```

Listing 11.12: Istruzione SQL per la modifica della tabell POI_Table.

In seguito la visualizzazione dell'immagine relativa al POI viene operata, attraverso il metodo `getOriginal` e `getImage`, che esegue un'interrogazione alla tabella "POI_Table" per ottenere gli indirizzi delle immagini. Le stringhe così ottenute riportano l'indirizzo del file immagine originale e della versione thumbnail.

Durante la realizzazione dello strato si è scelto di utilizzare nella visualizzazione dei POI dei file immagine prelevati dal sito Panoramio.com.

11.3.2 Rating

Una delle peculiarità del servizio Padova Tour è dato dalla possibilità per gli utenti di sottomettere un feedback qualitativo sui POI che essi visitano sotto forma di un punteggio tra 1 e 5. La visualizzazione e l'inserimento di questi valori vengono realizzate tramite la chiamata alla funzione `rating_form`:

```
include_once "rating/rating.php";
rating_form($poi);
```

Listing 11.13: Chiamata alla funzione `rating_form`() nella pagina guestbook.

Essa riceve in input l'id del hotspot target e restituisce in output la visualizzazione grafica del rating, espressa come un numero di stelle proporzionale alla valutazione.

```
if(!isset($_GET['update'])){
        echo "<div class='rating_wrapper'>";
        }
?>
<div class="sp_rating" >
        <div class="rating">Voto - Rating:</div>
                <div class="base">
                        <div class="average" style="width:<?php echo
                           $rating->average; ?>%">
                        <?php echo $rating->average; ?>
                        </div>
                </div>
                <div class="votes">#<?php echo $rating->votes; ?>.
                </div>
                <div class="status">
                        <?php echo $status; ?>
                </div>
        </div>
<?php
if(!isset($_GET['update'])){
        echo "
        </div>
        ";
}
```

Listing 11.14: Funzione `rating_form`().

L'implementazione richiede, come si vede, l'utilizzo di oggetti di tipo `rating`, caratterizzati da diverse proprietà quali ad esempio il rate ed il numero di votanti, ma anche se l'utente ha già espresso una valutazione.

```
public $average;
public $votes;
public $status;
public $dbh;
```

Al client è permesso assegnare un punteggio al POI: per far ciò viene eseguita una chiamata alla funzione `set_score`, la quale aggiorna il database.

```
function set_score($score, $id){
        $sql= "SELECT rating, tot FROM POI_Table WHERE id=$id";
        $statement = mysql_query($sql, $this->dbh)or die(mysql_error());
        //for the single row
        while($row =  mysql_fetch_array($statement)){
                $this->votes = $row['tot'];
                $this->average = (($row['rating']*$this->votes+$score)/(
                   $this->votes+1));
                $this->votes++;
                }
        $sql = "UPDATE POI_Table SET tot=$this->votes, rating=$this->
           average WHERE id=$id";
        $statement =mysql_query($sql,$this->dbh);
        //average expressed in parts of 100
        $this->average=$this->average*20;
        $this->status = ':)';
        $this->dbh = NULL;
        }
```

Listing 11.15: Funzione `set_score()`.

11.3.3 Inserimento commenti

La gestione dei commenti è coadiuvata dall'impiego di una nuova tabella "messages", la quale andrà inserita all'interno della base di dati del servizio tramite la seguente istruzione.

```
--
-- Table structure for table `messages`
--
CREATE TABLE IF NOT EXISTS `messages` (
`poi_ID` int(50) NOT NULL,
`id` int(50) NOT NULL,
`message` tinytext NOT NULL,
`name` varchar(30) NOT NULL,
`time` int(40) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Listing 11.16: Istruzione SQL per la creazione della tabella messages.

Il form per l'inserimento dei commenti sarà composto da due campi: una casella di input testuale per il nome utente ed un textbox per il commento. Entrambi i valori sono obbligatori per la corretta sottomissione di questo tipo di feedback: l'assenza di uno od entrambi verrà indicata al client tramite un messaggio d'errore. Entrambi i campi, prima dell'inserimento nel database, verranno sottoposti ad una validazione: il nome utente dovrà infatti avere una lunghezza tra i 2 e i 30 caratteri, mentre il testo del commento dovrà essere compreso tra i 2 e i 140 caratteri (quest'ultimo valore è stato

selezionato in quanto rappresenta la massima lunghezza concessa nei post di Twitter). La verifica viene operata tramite una chiamata ad una funzione Javascript, `lengthCheck`(), passando il nome dell'elemento e l'intervallo di valori ammessi in fase di sottomissione.

```
onclick="lengthCheck(document.getElementById('name'),2,30,document.
    getElementById('message'),2,140)"
```

Una volta che un messaggio sarà stato convalidato, esso verrà inserito nella tabella dei messaggi. Nel contempo si vuole però visualizzare all'utente il corretto inserimento del commento nel sistema: nella tale operazione verrà eseguita utilizzando Javascript e AJAX, grazie ai quali si potrà ricaricare l'insieme dei messaggi senza dover richiedere nuovamente l'intera pagina al server. Il sorgente proposto nel seguito, riporta una funzione realizzata tramite jQuery che realizza quest'ultimo comportamento.

```
$("form#input_form").submit(function(){
        var message = $('#message').attr('value');
        var name = $('#name').attr('value');
        var date = '<?php echo date("d/m/Y"); ?>';
        var hr = '<?php echo date("H:i"); ?>';
        var id = '<?php echo $_REQUEST['id'];?>';
        $.ajax({
                type:"POST",
        url:"insert.php",
        data:"message="+message+"&name="+name+"&id="+id,
                success: function(){
                if(document.getElementById("message").value!="" &&
                   document.getElementById("name").value!=""){
                        var txt = 'Grazie per il commento.<br /> Thank you
                            for your comment.';
                        $.prompt(txt);
                        $("div#show").prepend("
                        <div class='hcomh'>Nome/Name: <b>"+name+"</b> | "+
                           date+" @ "+hr+"</div><div class='bcomh'>"+
                           message+"</div><br>");
                        //Show the new message
                        $("div#show div:all").fadeIn();
            //Reset the form fields
                        document.getElementById("message").value="";
                                          document.getElementById("
                           message").style.background='white';
                                  document.getElementById("name").value="
                           ";                          document.
                           getElementById("name").style.background='white'
                           ;
                        }
                }
        });
        return false;
});
```

Listing 11.17: Codice JavaScript per l'inserimento dei commenti e l'aggiornamento della pagina guestbook.

Il metodo sopra esposto opera l'inserimento dei commenti inviando una richiesta al file `insert.php`, all'interno del quale sono contenute le istruzioni per la connessione al database. Solo in caso di successo dell'operazione aggiornerà la pagina. Si noti che il metodo sottomette anche i valori di data e ora dell'inserimento: questi saranno impiegati in seguito in fase di visualizzazione.

```
<?php
include_once "guestbook.class.php";
$guestbook = new Guestbook();
$id=$_REQUEST['id'];
$guestbook->InsertMessage($id);
?>
```

Listing 11.18: Contenuto della pagina `insert.php`.

Il seguente listato si riferisce invece alla funzione `InsertMessage`, la quale legge il contenuto inserito nel form tramite la variabile speciale `$_POST`, ne esegue un parse (per la gestione dei caratteri speciali), ed inserisce le informazioni ottenute nella tabella "messages".

```
public function InsertMessage($id){
        if($_POST['message'] != "" AND $_POST['name'] != ""){
                $message = htmlspecialchars($_POST['message']);
                $name = htmlspecialchars($_POST['name']);
                $time = time();
                $sql = "INSERT INTO messages (message,time,name,poi_ID)
                   VALUES ('$message', $time,'$name','$id')";
                mysql_query($sql, $this->conn);
        }
}
```

Listing 11.19: Funzione `InsertMessage`() della classe `Guestbook`.

11.3.4 Visualizzazione commenti

La visualizzazione dei commenti all'interno della pagina guestbook, è realizzata utilizzando i metodi forniti dalla classe "guestbook": realizziamo quindi l'inclusione di tale classe è invochiamone il costruttore. L'oggetto implementato servirà per invocare la procedura `ShowMessages`().

```
<?php
include_once "guestbook.class.php";
$id= $_REQUEST['id'];
$guestbook = new Guestbook();
$guestbook->ShowMessages($id);
?>
```

Listing 11.20: Istruzioni per la visualizzazione dei messaggi nella pagina guestbook.

Il codice della funzione `ShowMessages`, riceve in input un intero che descrive un identificatore di POI ed interroga il database, a cui è collegato l'oggetto guestbook, restituendo tutti i messaggi relativi al punto richiesto, ordinati in ordine di tempo d'inserimento decrescente, fino ad un massimo di 20.

```
public function ShowMessages($id){
        $sql = "SELECT * FROM messages HAVING poi_ID=".$id." ORDER BY time
            DESC LIMIT 20";
        $res = mysql_query($sql,$this->conn);
        while($row = mysql_fetch_array($res)){
                echo '
<div class="hcom">Scritto da: <b>' . $row['name'] . '</b> il ' . date("d/m
   /Y", $row['time']) . ' alle ore ' . date("H:i", $row['time']) . '</div>
      ';  echo '<div class="bcom">' . $row['message'] . '</div>   <br>';
        }
}
```

Listing 11.21: Funzione `ShowMessages`().

11.3.5 Grafica

La gestione della grafica e degli effetti visivi, non sarà discussa in dettaglio, in quanto definisce solo la presentazione del servizio del libro degli ospiti. L'implementazione attuale è ottenuta tramite la libreria jQuery `http://jquery.com/`, la quale mette a disposizione dello sviluppatore diversi oggetti.

La visualizzazione dei messaggi d'errore relativi al form di inserimento dei commenti sono gestite utilizzando Impromptu (`http://trentrichardson.com/Impromptu/index.php`), un plugin di jQuery che mette a disposizione diversi effetti per presentare finestre di dialogo agli utenti, mentre la gestione delle immagini avviene utilizzando Thickbox (`http://jquery.com/demo/thickbox/`).

Supporto multilingua. Si noti che è possibile inserire facilmente un semplice supporto multilingua, inserendo un controllo per il parametro `lang` della richiesta; in base a quest'ultimo si potrà interrogare la base di date per le corrette espressioni linguistiche desiderate (nome del punto, nome itinerario, ...) ed assegnarle a ciascun POI in fase di costruzione durante la chiamata `getPointOfInterest`().

11.4 Riepilogo

Le operazioni svolte sinora sono state rivolte a produrre uno strato, Padova Tour, a scopo turistico: prima tramite la definizione degli ambiti d'interesse e delle caratteristiche tecniche che si desiderava proporre all'utenza, ed in seguito attraverso l'implementazione di un servizio che rispondesse a tutti i requisiti definiti.

Il layer permette agli utenti di effettuare visite turistiche della città di Padova solo tramite l'ausilio del proprio smartphone: essi potranno esplorare una serie di luoghi preselezionati, presenti nel database di consultazione, ed interagirvi per mezzo di azioni che permettono l'accesso a file multimediali, ipertesti e mappe interattive, rilasciando anche feedback di diverso tipo, quali la sottomissione di rating e messaggi.

La prima operazione è stata la definizione del servizio presso il sito di definizione degli strati, dove sono state inserite le caratteristiche basilari dello stesso. Il passo successivo è stato la strutturazione di un servizio che permettesse di ricercare i POI appartenenti al sistema sulla base delle impostazioni definite dall'utente, quali raggio di ricerca, itinerario prescelto e valore di rating, e che restituisse i dati in un formato corretto e comprensibile. In seguito è stata inserita la possibilità di dialogare con i risultati attraverso delle azioni: è stato quindi approntato un meccanismo di associazione tra punti e azioni. In particolare si è analizzata l'implementazione dell'elemento guestbook per mezzo del quale gli utenti possono condividere tra loro, e con il gestore del livello, informazioni utili. Infine si è inserita una funzionalità che permette all'utenza di conoscere la presenza di elementi d'interesse, presenti all'interno dell'area di ricerca, ma che non fanno parte delle reali preferenze contenute nella richiesta.

Parte VI

Content Management System

Capitolo 12

Motivazioni

12.1 Gestione dei contenuti dello strato

Date le sue caratteristiche e peculiarità, Padova Tour può essere impiegato da uno o più enti turistici per strutturare delle visite tematiche, anche sulla base dei feedback ricevuti. Risulta quindi interessante fornire uno strumento che permetta a tali enti di inserire, aggiornare e modificare i contenuti dello strato in maniera semplice, veloce e soprattutto non tecnica, ovvero ignorando i dettagli di programmazione.

A tale proposito si rivela necessaria la creazione e strutturazione di un Content Management System (CMS), letteralmente sistema di gestione dei contenuti ovvero di uno strumento software, presente su un server web, studiato per facilitare la gestione di risorse di varia natura (solitamente ipertesti), svincolando l'utente editor dalle conoscenze tecniche di programmazione. Un CMS è quindi un'applicazione lato server, la quale si appoggia su un database preesistente per lo stoccaggio dei contenuti, suddivisa in due parti:

1. la sezione di amministrazione (**back end**), che serve ad organizzare e supervisionare la produzione dei contenuti;

2. la sezione applicativa (**front end**), che l'editore utilizza per fruire dei dati e delle applicazioni del portale.

Un utente del CMS gestisce dal proprio terminale, tramite un pannello di interfaccia e controllo, gli elementi da inserire o modificare e inviandoli al database del server.

Lo scenario in esame risulta affrontabile tramite lo sviluppo di un CMS altamente specializzato, focalizzato sui contenuti offerti a Layar Reality Browser per lo strato Padova Tour. Strumenti esistenti come Joomla (`www.joomla.org`) risulterebbero troppo generici rispetto alle attuali esigenze di lavoro, mentre l'integrazione di tool specifici come i già citati porPOIse e LightRod andrebbero a limitare la personalizzazione dei servizi offerti.

In un approccio sistematizzato al problema della gestione dell'informazione, si affrontano generalmente le seguenti fasi:

- *identificazione degli utenti di back-end e dei relativi ruoli di produzione o fruizione dell'informazione.* Nel nostro caso andremo ad identificare come utenti per il back-end, tutti gli operatori turistici, o in generale, tutti coloro che desiderino usufruire dello strato realizzato per la strutturazione di visite turistiche. Tali utenti avranno un ruolo di editori e potranno inserire contenuti di vario genere ed agire su di questi. Un'altra tipologia di utente sarà

quella di amministratore, ovvero uno user dotato di privilegi particolari che potrà visionare i contenuti prodotti dalla totalità dell'utenza, apportandovi delle modifiche qualora sia ritenuto necessario. Infine si riconosce una categoria di utenti generici, i quali potranno solamente visualizzare i contenuti offerti dal livello;

- *assegnazione di responsabilità e permessi a differenti categorie di utenti per distinti tipi di contenuti.* Una volta identificati i ruoli di produzione e fruizione, si dovrà realizzare un meccanismo per la corretta attribuzione di questi ai diversi utenti. All'interno del CMS dovremo offrire la possibilità agli utenti di registrarsi di modo che essi possano ottenere l'abilitazione a produrre e modificare contenuti personali. Agendo invece a livello di database, e solo in un primo momento, verrà assegnato ad un utente il ruolo di amministratore: egli potrà in seguito estendere tale privilegio ad altri utilizzando delle funzioni appositamente elaborate;

- *definizione delle attività di workflow.* Si tratta di una formalizzazione delle operazioni offerte ai vari utenti. Nello specifico verranno realizzate delle procedure di registrazione, login/logout, inserimento e modifica dei punti d'interesse e delle relative caratteristiche (azioni, messaggi, rating) e d'inserimento e modifica degli itinerari;

- *pubblicazione del contenuto.* Determina le modalità con cui vengono pubblicati i contenuti. In questo caso, la pubblicazione del contenuto avverrà in tempo reale all'interno del portale tramite una visualizzazione grafica, e solo per l'autore di questo, mentre per gli altri utenti del CMS, nonché per i fruitori dello strato, sarà richiesto di avviare un refresh dell'applicazione;

- *definizione del palinsesto editoriale.* Una volta strutturate le diverse fasi, è opportuno organizzare anche quello che sarà l'ambiente di lavoro dei vari utenti. Nello specifico verranno suddivise tra loro le aree di registrazione degli utenti, login, logout, visualizzazione dei punti, visualizzazione dei contenuti del guestbook, inserimento di nuovi punti, modifica di punti preesistenti, inserimento e modifica di itinerari, gestione dell'utenza.

[23].

12.2 Sessioni PHP

Il protocollo che regola il Web è, come detto in precedenza, un protocollo senza stato (stateless). Ciò vuol dire che browser e Web server comunicano e si scambiano dati pagina per pagina, senza che quest'ultimo conosca i dettagli delle comunicazioni passate avviate con lo stesso utente, né avendo modo di sapere esattamente quali client si stiano connettendovi. Nella fattispecie, il server potrà solamente conoscere l'indirizzo IP del client (o del suo gateway) e il tipo di browser/sistema operativo; tali informazioni non sono però sempre affidabili. PHP 4 e successivi, nell'intento di colmare tale lacuna, includono un sistema di sessioni.

Con il termine **sessione** si intende l'arco di tempo dal momento in cui un client esegue la prima request al server, fino al momento in cui questo invia la sua ultima risposta al client. La logica che sta alla base di tale artificio è la seguente: quando si intende realizzare una sessione, per mantenere alcune informazioni ritenute importanti e al fine di riconoscere successivamente l'utente loggato (in genere dopo una procedura di login tramite username/password), il server utilizzando PHP salva un file di testo contenente tali dati. Esso rappresenta il file di sessione relativo all'utente autenticato. Il nome del file di testo è una stringa alfanumerica casuale composta di 32 caratteri, la quale viene inviata dal browser al server (in maniera del tutto trasparente all'utente finale ed al programmatore,

e sempre secondo il protocollo usato) non appena verrà richiesta una nuova pagina Web, facendo in modo che PHP recuperi il file di sessione ed i dati da associare alla nuova richiesta. La stringa identificativa della sessione, scambiata tra le due parti e che punta al file di testo sul server, prende il nome di *SID* o *Session IDentifier*. Le metodologie con le quali il browser può inviare il SID al server sono tramite cookie (header HTTP) o query string.

Una sessione è dotata di un proprio ciclo di vita delimitato da tre fasi:

1. inizializzazione: si tratta di una fase di avvio in cui iniziano le operazioni di scambio dei dati tra un client, e un'applicazione che gestisce un determinato servizio, questo avviene per esempio nel momento in cui si accede ad un'area riservata basata su procedura di autenticazione;

2. comunicazione (sessione di lavoro): in questa fase la sessione consente l'interscambio di dati tra un client e un servizio, ciò è reso possibile dal fatto che durante l'inizializzazione il server alloca in memoria alcune informazioni, i parametri di sessione, grazie a cui l'applicazione è in grado di riconoscere l'utente corrente permettendogli di accedere ai dati scambiati ed eventualmente di manipolarli;

3. terminazione: ovvero la chiusura di una sessione che può avvenire anche su richiesta dell'utente come per esempio nelle procedure di "logout", essa consiste nella cancellazione delle informazioni di sessione; se l'applicazione lo prevede, una sessione può essere terminata anche da un periodo di latenza in cui non vengono scambiati dati tra utente e servizio.

I dati rimarranno associati all'utente remoto, per tutte le pagine richieste in future comunicazioni e fino alla distruzione della sessione stessa (da parte del client o in seguito ad un timeout) o alla chiusura del browser.

Il motivo per cui le sessioni vengono memorizzate in remoto è molto semplice in quanto un server Web è certamente uno strumento con un livello di sicurezza superiore con cui gestire informazioni (spesso riservate), rispetto ad un comune browser per la navigazione su Internet.

Per gestire le sessioni il motore PHP registra quindi sul server un array associativo che può essere letto dalle pagine della sessione e che è associato ad un ID univoco sul server stesso; per quanto riguarda il client, crea sul computer dell'utente un cookie contenente lo stesso ID alfanumerico. Quando avviene così la chiamata HTTP, il server può verificare sul terminale dell'utente la presenza di un cookie contenente un ID valido sul server e associare quindi ad esso i dati della sessione. In questo modo esisterà sempre un collegamento univoco tra server e client. Nel caso quest'ultimo abbia disabilitato i cookie, PHP consente al client di inviare l'ID della sessione effettuando un *append* alla stringa di query oppure ai parametri di un form.

PHP permette di associare ad una sessione diverse informazioni tramite l'utilizzo della variabile `$_SESSION`, la quale rappresenta un array associativo contenente le variabili attive e valorizzate per la sessione in corso.

Per usufruire di tale strumento, all'interno di ogni pagina interessata, si dovrà impostare un collegamento tra server e client e inizializzare quindi la sessione tramite la funzione `session_start()`, messa a disposizione da PHP. Una volta aperta la sessione, si potrà accedere all'array `$_SESSION`, potendo così leggerne e modificarne il contenuto facilmente. Il salvataggio dei dati di sessione all'interno del relativo file, avviene attraverso un processo denominato *serializzazione*, la quale evita la gestione dei singoli valori che risulterebbe un procedimento complesso e oneroso in termini di risorse e di tempo; per questo motivo è necessario adottare una soluzione che consiste nella conversione dell'insieme di valori in un'unica stringa da allocare in memoria; nel momento in cui le informazioni

contenute all'interno del file di sessione devono essere messe a disposizione dei soggetti in comunicazione, queste potranno essere recuperate attraverso l'ID di sessione e rese leggibili grazie ad una procedura di de-serializzazione.

Nota: il codice presentato nel proseguo dell'elaborato, è stato realizzato utilizzando come riferimento i testi [6],[7]. Inoltre è stato utilizzando, a fini di debug, il software Firebug (`http://getfirebug.com/`): essa è un'estensione del browser Firefox che mette a disposizione dello sviluppatore una varietà di strumenti per lo sviluppo web disponibili durante la navigazione. Grazie ad esso è possibile modificare, eseguire debug e monitorare elementi CSS, HTML, JavaScript, DOM dal vivo ed in qualsiasi pagina web.

Nel seguito verranno discussi i dettagli implementativi riguardanti codici HTML, DHTML, PHP e JavaScript, ma non verranno menzionati quelli relativi ai fogli di stile CSS, in quanto la definizione di questi ultimi elementi non preclude il corretto funzionamento dell'applicazione.

Capitolo 13

L'interfaccia principale

Procediamo ora ad analizzare l'implementazione del CMS. In primo luogo è necessario impostare un'interfaccia tramite la quale gli utenti potranno interagire con le diverse funzionalità offerte dal content management system, osservando quelli che sono i privilegi di accesso di ognuno. Si realizzerà quindi una pagina PHP (`index.php`), dove verrà inizializzata la sessione utente per mezzo del codice seguente:

```
<?php session_start();
        if (!session_is_registered('sessionStartTime')) {
                $_SESSION['sessionStartTime']=time();}
?>
```

Listing 13.1: Inizializzazione della sessione PHP.

Inseriamo quindi un menù che permetta agli utenti di accedere alle aree del CMS: tali sezioni saranno successivamente visualizzate all'interno di un elemento `iframe` (in tal modo dovrà essere caricato solo quest'elemento e non la pagina intera, ogni qualvolta si desideri accedere ad un'area differente). Ogni elemento della lista dovrà contenere un riferimento ad una pagina web e dovrà impostare il suddetto `iframe` come target per il collegamento. L'assegnazione del target corretto viene effettuata tramite l'implementazione di un breve codice JavaScript che utilizza la libreria jQuery e che sarà incluso nel corpo della pagina.

```
$(document).ready(function() {
        var hash = window.location.hash.substr(1);
        var href = $('#nav li a').each(function(){
       var href = $(this).attr('href');
       if(hash==href.substr(0,href.length-5)){
            var toLoad = hash+'.html';
            $('#content').attr('src', toLoad)
       }else{
            var toLoad = hash+'blank.html';
            $('#content').attr('src', toLoad);
       }
});
```

Listing 13.2: Caricamento dinamico dei contenuti del portale.

Nota:

- Per impostare il file `index.php` come pagina index del nostro dominio, è necessario impostare il file `.htaccess` del nostro server con i seguenti parametri:

```
RewriteEngine On RewriteBase /
RewriteCond %{REQUEST_URI} ^$
RewriteCond %{HTTP_HOST} ^domain.com$
RewriteRule ^$ index.php [L,R=301]
```

Listing 13.3: Ridefinizione del file `htaccess`.

- E' opportuno rendere il portale di accesso un ambiente user-friendly in questo modo sarà maggiore la facilità d'uso per gli utenti; ad esempio tramite l'impiego della libreria jQuery e di un opportuno template si potrà presentare un'interfaccia chiara e comprensibile (dichiarato tramite file css).

Capitolo 14

Registrazione

Per concedere agli utenti la possibilità di identificarsi come editori di contenuti, si dovrà realizzare un modulo di registrazione. In primo luogo è opportuno preparare il database per accogliere i dati relativi ai diversi user: inseriamo una nuova tabella nella quale verranno memorizzati nome utente, email, password (codificata in md5), una chiave di controllo (challenge), un valore di check della registrazione ed il livello di autorizzazione.

```
CREATE TABLE ‘users‘ (
  ‘id‘ int(4) UNSIGNED NOT NULL AUTO_INCREMENT,
  ‘username‘ varchar(20) NOT NULL,
  ‘email‘ varchar(60) NOT NULL,
  ‘password‘ varchar(50) NOT NULL,
  ‘key_control‘ varchar(50) NOT NULL,
  ‘ver‘ int(1) NOT NULL DEFAULT ’0’,
  ‘superuser‘ tinyint(1) NOT NULL DEFAULT ’0’,
  PRIMARY KEY  (‘id‘),
  UNIQUE  (‘username‘))
ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

Listing 14.1: Istruzione SQL per la creazione della tabella users.

Inizializziamo un nuovo file, che chiameremo `form.php`, all'interno del quale realizzeremo il modulo di registrazione: creiamo un elemento `form`, impostandovi il campo action con il file `register.php` (che vedremo in seguito) ed il campo `method` su POST. Dichiariamo quindi tutti i parametri necessari alla registrazione (ovvero gli attributi della tabella "users").

```
<form action="register.php" method="POST">
        Username:<br> <input type="text" name="username" maxlength="20">
        <br><br>
        Email:<br> <input type="text" name="email" maxlength="60">
        <br><br>
        Password:<br> <input type="password" name="password" maxlength="20
           ">
        <br><br>
        Ripeti password:<br> <input type="password" name="password2"
           maxlength="20">
        <br><br>
```

```
        <input type="submit" value="Registrati">
</form>
```

Listing 14.2: Strutturazione del form di registrazione.

Una volta compilati, i campi del form dovranno essere inviati tramite `submit` all'indirizzo indicato nel campo `action`: questo punta al file `register.php` all'interno del quale viene attivata la procedura di registrazione, creando prima un oggetto di tipo `NewUser` ed in seguito invocando una procedura `AddUser`; al termine del processo l'utente verrà avvisato del corretto completamento. L'oggetto `NewUser` è dichiarato, assieme ai suoi metodi, all'interno del file `newuser.class.php` (tale file deve essere incluso all'interno della pagina `register.php`). Prima di analizzare il file contenente la classe è necessario impostare i parametri di connessione al nostro database in un file apposito che verrà poi incluso nella dichiarazione della classe stessa (come già fatto per il guestbook):

```
class NewUser {
        public $conn;
        protected function DbConnect() {
                include "db_config.php";
                $this->conn = mysql_connect($host,$user,$password) OR die(
                   "Impossibile connettersi al database");
                mysql_select_db($db, $this->conn);
        }
}
```

Listing 14.3: Costruttore per la classe `NewUser`.

Realizziamo quindi alcune funzioni di controllo per i campi del form, le quali ritorneranno un valore booleano relativo all'esito del controllo; tali procedure verificano, ad esempio, che nessuno dei campi sia vuoto, o che la password inserita coincida con quella di verifica (al fine di evitare una registrazione con una password non digitata correttamente). Altre funzioni come `UsernameExists`() verificano che il nome utente non sia già presente: questo controllo permette di avvisare di tale eventualità, riportando un messaggio d'errore (la query fallirebbe comunque, ma senza segnalare nulla poiché il campo "username" è impostato come UNIQUE). Allo stesso modo impostiamo la procedura `EmailExists`() per verificare che anche il campo email non sia già presente nel nostro database.

```
protected function UsernameExists() {
        $this->DbConnect();
        $sql = "
        SELECT *
        FROM users
        WHERE username='$_POST[username]'";
        $res = mysql_query($sql, $this->conn);
        if($row = mysql_fetch_array($res))      {
                mysql_close($this->conn);
                return TRUE;
        }else{
                mysql_close($this->conn);
```

```
                return FALSE;
        }
}
```

Listing 14.4: Funzione `UsernameExists`().

Un altro controllo da eseguire è quello sulla correttezza dell'email inserita, operato tramite l'utilizzo di un'espressione regolare e della funzione `ereg`().

```
protected function VerifyEmail() {
        $pattern =  "^([a-zA-Z0-9])+([a-zA-Z0-9]+[-_\.]?)*([a-zA-Z0-9])+(@
           )([a-zA-Z0-9])+([a-zA-Z0-9]+[-_\.]?)*([a-zA-Z0-9])+(\.[a-z
           ]{2,4})$";
        if(ereg($pattern,$_POST['email'])){
                return TRUE;
        }else{
                return FALSE;
        }
}
```

Listing 14.5: Funzione `VerifyEmail`().

Infine è necessario verificare che il nome utente sia privo di simboli che possano interferire con future interrogazioni: ovvero eliminiamo ogni possibilità di eseguire SQL Injection o di interferire con le espressioni regolari, permettendo solamente username composti di caratteri alfanumerici.

```
protected function IsValidName(){
        $pattern = "^[a-zA-Z0-9]+$";
        if(!ereg($pattern ,$_POST['username'])){
                return TRUE;
        }else{
                return FALSE;
        }
}
```

Listing 14.6: Funzione `IsValidName`().

A questo punto è importante validare i dati inoltrati e valutare tutti i possibili errori riscontrabili, come pure le conseguenze che avrà sul flusso dei dati il verificarsi di uno di questi; in caso di un errore reindirizziamo l'utente alla pagina del form e accodiamo a tale riferimento il tipo di errore che si è verificato che verrà poi recuperato utilizzando un GET e stampato a schermo tramite la visualizzazione di un alert dialog. Il metodo che si occuperà di svolgere questo compito è `ErrorResult`(), che riceve come argomento un numero indicante il tipo di problema riscontrato e reindirizza il client alla pagina del form, passando questo valore come parametro di "alert". In seguito interrompe il flusso della comunicazione invocando la funzione `die`(). Vedremo solo successivamente come saranno utilizzati questi valori.

```
public function ErrorResult($num) {
        header("Location: form.php?alert=".$num);
        die;
}
```

Listing 14.7: Funzione `ErrorResult`().

`ErrorResult` verrà invocata al termine di ogni controllo (in caso di esito negativo), passandole il valore assegnato al relativo errore, mentre nel caso in cui i dati inseriti nel form rispettino ogni vincolo verrà intrapresa una procedura per l'inserimento del nuovo utente. Quest'impostazione ci permette inoltre di rendere il metodo molto più leggibile.

Prima di passare all'analisi della procedura di inserimento, è necessario realizzare due ulteriori funzioni: la prima, `GetKey`(), si occuperà della generazione di un stringa challenge che permetterà di all' utente di attivare del proprio account, mentre la seconda, `SendUserMail`(), eseguirà l'invio di una email di conferma contenente il link per l'attivazione. Grazie a questi due metodi sarà possibile verificare correttamente l'identità degli user sia come legittimi proprietari dell'indirizzo di posta indicato, sia come autori della richiesta di registrazione. Vediamo quindi i due metodi.

```
protected function GetKey() {
        $car = "aAbBcCdDeEfFgGhHiIlLjJkKmMnNoOpPqQrRsStTu
           UvVwWxXyYzZ0123456789";
        $dim = 40;
        srand((double)microtime()*1000000);
        $string = '' ;
        for($inc=0; $inc<$dim; $inc++){
                $rand = rand(0, strlen($car)-1);
                $scar = substr($car, $rand, 1);
                $string = $string . $scar;
        }
        return $string;
}
```

Listing 14.8: Funzione `GetKey`().

La funzione `GetKey`() costruirà una stringa di 40 caratteri, presi casualmente da un alfabeto di input, restituendola in output alla funzione `SendUserMail`(), la quale la inserirà all'interno dell'url di verifica, che sarà poi trasmesso all'utente tramite l'invocazione del metodo `mail`(), realizzando una stringa simile alla seguente:

"`'verify_user.php?key='.$key`". Il listato, sotto esposto, riporta il codice per la funzione `SendUserMail`().

```
protected function SendUserMail($key)   {
        $content = "Benvenuto $_POST[username] su PadovaAR,\r\n";
        $content .= "Per confermare la tua iscrizione devi cliccare sul
           seguente link:\r\n\r\n";
        $content .= "http://padovaar.altervista.org/register/verify_user.
           php?key=" . $key;
```

```
        mail($_POST['email'], "Iscrizione PadovaTour CMS", $content, "From
            : PadovaAR<padovaar@altervista.org>");
        return;
}
```

Listing 14.9: Funzione `SendUserMail`().

Una volta verificata la corretta compilazione del form d'iscrizione e realizzata una funzione per la gestione degli errori, possiamo quindi procedere all'implementazione della funzione per inserimento del nuovo utente.

```
protected function InsertNewUser() {
    //Encrypt the password
        $password = md5($_POST['password']);
    //Get the challenge code
        $key_control = $this->GetKey();
    //Insert user data
        $sql = "INSERT INTO users (username,email,password,key_control)
            VALUES ('$_POST[username]','$_POST[email]','$password','
            $key_control')";
        $this->DbConnect();
        mysql_query($sql,$this->conn);
        mysql_close($this->conn);
    //Send an activation mail
    $this->SendUserMail($key_control);
}
```

Listing 14.10: Funzione `InsertNewUser`().

Grazie a quest'ultima procedura l'utente può ora essere registrato correttamente: non resta che realizzare la pagina di verifica dell'iscrizione, ovvero l'elemento il cui url è stato inviato nel corpo della mail inviata tramite `SendUserMail`() sotto forma di web link e che, una volta aperto, confermerà all'utente la corretta terminazione del processo di registrazione. Questa pagina si appoggerà ad un metodo `VerifyUser`(), appartenente alla classe `NewUser`, il quale utilizzerà il challenge contenuto nella richiesta di verifica per convalidare l'utente nel sistema. Il file `verify_user.php`, realizzato tramite il codice riportato nel listato sottostante, implementa la pagina di verifica.

```
<?php
        include 'newuser.class.php';
        $newuser = new NewUser();
        $newuser->VerifyUser();
?>
```

Listing 14.11: Invocazione del metodo di verifica dell'utente.

Di seguito invece il codice della suddetta funzione `VerifyUser`():

```
public function VerifyUser()    {
    $sql = "SELECT id FROM users WHERE key_control='$_GET[key]'";
    $this->DbConnect();
    $res = mysql_query($sql,$this->conn);
    if($row = mysql_fetch_array($res))      {
        $query = "UPDATE users SET ver=1,key_control='0' WHERE id='$row[id
           ]'";
        mysql_query($query,$this->conn);
        mysql_close($this->conn);
        echo "Account attivato!";
    }       else            {
        echo "Impossibile verificare l'account!";
    }
}
```

Listing 14.12: Funzione `VerifyUser()`.

Rimane un ultimo elemento da impostare: il file per la gestione degli errori, da includere all'interno della pagina `form.php`. All'interno di questo file, che chiameremo `error_definition.php`, utilizzeremo l'ambiente PHP per invocare una funzione JavaScript di tipo `alert()`, tramite la lettura del contenuto della variabile `$_GET['alert']` e un costrutto di tipo switch-case-break. Il codice sottostante riporta un estratto di tale funzione PHP.

```
if(isset($_GET['alert'])) {
    //Write JS code on the client page
        echo '<script type="text/javascript">';
        switch($_GET['alert'])  {
                case 1:
                echo 'alert("Tutti i campi devono essere compilati")';
                break;
(...)
        }
        echo '</script>';
}
```

Listing 14.13: Script per la sottomissione degli errori di compilazione del form.

14.0.1 Riepilogo

L'implementazione di una procedura di registrazione ha richiesto la preparazione di una tabella nella base di dati e l'instaurazione di una connessione con questo, la creazione di una pagina web dove l'utenza può inserire i propri dati ed inviarli al sistema, un meccanismo per la verifica e l'elaborazione di questi, un meccanismo di convalida della richiesta di registrazione sicuro ed un metodo per la gestione degli errori.

Capitolo 15

Controllo dell'utente

15.1 Login

Una volta che gli utenti hanno effettuato la registrazione presso il CMS, è necessario adottare una procedura di autenticazione al fine di permettere l'accesso alle aree riservate dello stesso.

Come primo passo, realizziamo un semplice form all'interno del quale l'utente potrà fornire le proprie credenziali, ed inseriamolo un nuovo file di script che chiameremo `login.php`.

```
<form action="verify_login.php" method="POST" >
        username:<br><input type="text" name="username" /><br><br>
        password:<br> <input type="password" name="password" /><br><br>
        <input type="submit" value="login" />
</form>
```

Listing 15.1: Strutturazione del form di login.

Il campo `action` di tale form si riferisce al file `verify_login.php` (si veda il listato successivo), all'interno del quale verrà gestita la chiamata alla procedura di login, la quale si appoggia alla classe `UserAuthentication` e ai suoi metodi; vediamo quindi come viene creato un nuovo oggetto di tipo `UserAuthentication`, sul quale sarà poi invocato il metodo `VerifyLogin`.

```
<?php
        include 'authentication.class.php';
        $auth = new UserAuthentication();
        $auth->VerifyLogin();
?>
```

Listing 15.2: Invocazione del metodo di verifica dell'autenticazione.

Analizziamo ora la classe `UserAuthentication`. Come per `NewUser` è necessario introdurre un metodo per la connessione al database, attraverso il quale potremo recuperare le informazioni di ciascun utente. Inoltre aggiungiamo il comando `session_start()` nell'apertura della classe: in tal modo potremo accedere alla variabile di sessione ed al suo contenuto.

Il metodo di verifica dell'accesso è di per se molto semplice: sarà sufficiente catturare i dati forniti dall'utente tramite il verbo POST (eseguendo la codifica md5 della password) ed utilizzarli

per interrogare il database. Se i dati inseriti risulteranno corretti, la query ritornerà una serie di valori associati all'utente stesso che saranno utilizzati per aggiornare la variabile di sessione, reindirizzandlo poi ad una pagina dove verrà avvisato del corretto completamento dell'operazione di login. In caso contrario, l'interrogazione al database fallirà e l'utente sarà riportato alla pagina di autenticazione e, come per la procedura di registrazione, sarà informato dell'errore tramite un alert dialog. Di seguito il codice per `VerifyLogin`().

```
public function VerifyLogin()          {
        $username = $_POST['username'];
        $password = md5($_POST['password']);
        $this->DbConnect();
        $sql = "SELECT id,superuser,username FROM users WHERE username='
           $username' AND password='$password' AND ver=1";
        $res = mysql_query($sql,$this->conn);
        if($row = mysql_fetch_array($res))                    {
                session_start();
                $_SESSION['auth'] = 1;
                $_SESSION['user_id'] = $row['id'];
        $_SESSION['username'] = $row['username'];
                $_SESSION['super_user'] = $row['superuser'];
                header("Location: login_success.php");
                mysql_close($this->conn);
                die;
        }else{
                header("Location: login.php?alert=1");
                mysql_close($this->conn);
                die;
        }
}
```

Listing 15.3: Funzione `VerifyLogin`().

Allo scopo facilitare la gestione dell'autenticazione dell'utente (soprattutto in caso di accesso ad aree riservate), possiamo inserire un nuovo metodo `IsAuth`() nella classe, il quale si occuperà di verificare il valore della variabile `$_SESSION['auth']` e di restituire un booleano al termine del controllo.

```
public function IsAuth(){
        if((!isset($_SESSION['auth'])||($_SESSION['auth']!=1)){
                return false;
        }
    return true;
}
```

Listing 15.4: Funzione `IsAuth`().

Torniamo ora alla funzione `VerifyLogin`: in caso di successo la variabile di sessione verrà aggiornata e l'utente sarà reindirizzato alla pagina `login_success.php`. Qui, il server verificherà il

completamento dell'operazione di accesso tramite la funzione `IsAuth`() appena esposta e, se anche questa verifica verrà terminata correttamente, il client sarà informato tramite un messaggio di benvenuto personalizzato; tuttavia in caso di fallimento l'utente sarà riportato alla pagina di login dove dovrà ripetere il processo di autenticazione. Il seguente listato riporta il sorgente relativo a `login_success.php`.

```
<?php
        include 'authentication.class.php';
        $auth = new UserAuthentication();
        if($auth->IsAuth()){
                echo "Benvenuto " . $auth->ShowUsername() . "! ";
        } else {
                header("Location: login.php");
                die;
        }
?>
```

Listing 15.5: Contenuto della pagina `login_success.php`.

Terminiamo l'esposizione della procedura di login presentando due ultime funzioni: la prima, `ShowUsername`, recupera dal database lo username di un utente, e può essere utilizzata in più occasioni come nella visualizzazione di un messaggio di benvenuto personale. Anche questa funzione viene richiamata su oggetti di tipo `UserAuthentication`.

```
public function ShowUsername(){
        $this->DbConnect();
        $sql = "SELECT username FROM users WHERE id=$_SESSION[user_id]";
        $res = mysql_query($sql,$this->conn);
        $row = mysql_fetch_array($res);
        mysql_close($this->conn);
        return $row['username'];
}
```

Listing 15.6: Funzione `ShowUsername`().

La seconda si occupa invece della gestione dell'unico errore segnalato ed inviato da `VerifyLogin`() nella pagina `login.php`: come già fatto nel caso della registrazione il messaggi d'errore sarà presentato operando un controllo sulla stringa dell'url tramite il verbo GET.

```
if(isset($_GET['alert']) AND $_GET['alert'] == 1) {
        echo '<script type="text/javascript">
        alert("Nome utente o password errati oppure account non attivato
           !")
        </script>';
}
```

Listing 15.7: Gestione degli errori in fase di login.

15.2 Logout

Per completamento è opportuno inserire anche una procedura di logout, realizzata tramite la funzione `session_destroy()`, che elimina tutte le informazioni relative alla sessione in corso e forza la chiusura della sessione attiva. Prima di eseguire tale chiamata, verifichiamo che l'utente si sia autenticato tramite il metodo `IsAuth()`. Le seguenti istruzioni implementano il logout del client e sono contenute in un file `logout.php`.

```
<?php
        session_start();
        include 'authentication.class.php';
        $auth = new UserAuthentication();
        if($auth->IsAuth()){
                $_SESSION = array();
                session_destroy();
                print "Logout completato!";
        } else{
                print "Login necessario!";
        }
        exit();
?>
```

Listing 15.8: Contenuto della pagina `logout.php`.

15.3 Verifica dei privilegi utente

Grazie alla classe `UserAuthentication` possiamo verificare che l'utente abbia eseguito il login e permettergli l'accesso alle aree riservate del CMS includendo il seguente codice:

```
include 'authentication.class.php';
$auth = new UserAuthentication();
if($auth->IsAuth()){...}
```

Listing 15.9: Istruzioni per la verifica dei privilegi utente.

La verifica può essere utilizzata per impedire che l'utente esegua dei comportamenti scorretti, ad esempio nella pagina di login possiamo aggiungere il codice sottostante per avvisare il client di aver già effettuato l'accesso al proprio account personale e quindi escludere così login multipli.

```
session_start();
include 'authentication.class.php';
$auth = new UserAuthentication();
if($auth->IsAuth()){
        print "Login già effettuato!";
        exit();}
```

Listing 15.10: Verifica dei privilegi utente. Prevenzione di login multipli.

Abbiamo quindi realizzato una struttura semplice e funzionale per il controllo degli accessi operati, da parte dei diversi client, nel content management system. Come ultimo accorgimento, non resta che corredare la classe `UserAuthentication` con un metodo `IsSAuth`() che si preoccuperà di verificare se l'utente goda di particolari privilegi, ovvero se si tratti di un superuser (utilizzeremo più avanti questa funzionalità). L'implementazione ricalca quella del precedente metodo `IsAuth`.

```
public function IsSAuth()                {
        if((!isset($_SESSION['super_user']))||($_SESSION['super_user']!=1)
           ){
                return false;
        }
    return true;
}
```

Listing 15.11: Funzione `IsSAuth`().

15.4 Riepilogo

Il controllo dell'utente si suddivide in due fasi:

- l'autenticazione, che permette al client di essere riconosciuto nel sistema in seguito ad una procedura di login corredata da metodi per il recupero dei dati utente, memorizzati all'interno di una sessione, e la gestione degli errori. Accanto a questo è stato poi proposto un metodo di de-autenticazione;
- l'autorizzazione, che garantisce all'utente l'accesso alle diverse aree del CMS e realizzata tramite il controllo dei dati di sessione. Le verifiche riguardano lo stato di autenticazione ed il controllo dei privilegi super utente.

Capitolo 16

Visualizzazione del guestbook

Passiamo ora alla realizzazione del primo vero contenuto del nostro CMS: la visualizzazione delle pagine guestbook realizzate ad hoc per ciascun POI, e raggiungibili tramite l'applicazione di una determinata azione all'interno dello strato. Un utente però, può ritenere utile visionare i contenuti del guestbook per rivedere i propri commenti, o per conoscere le opinioni altrui, senza dover accedere al livello Padova Tour (e quindi all'applicazione Layar Reality Browser): per questo motivo risulta doveroso fornire un accesso alternativo a questa risorsa; inoltre, anche un utente generico può essere interessato a queste informazioni senza però voler contribuire ai contenuti dello strato: a fronte di ciò e è opportuno non effettuare il controllo dell'autenticazione per questo tipo di contenuto.

Con queste premesse andiamo a realizzare una nuova pagina, che chiameremo `gb_index.php`, nella quale andremo ad inserire un semplice menù, di tipo drop-down list, contente tutti i link per le diverse pagine del libro degli ospiti; nella costruzione di questo strumento sarà utilizzato un breve codice PHP (presentato nel seguente listato) che genererà in automatico gli elementi della lista, eliminando così il bisogno di dover aggiornare manualmente i contenuti della pagina.

```
//$site variable sets up a "get back" link in every guestbook page
$site=false;
if(isset($_GET['site'])){
        $site=$_GET['site'];
}
include_once 'db_config.php';
$conn = mysql_connect($host,$user,$password) OR die("Impossibile
   connettersi al database");
mysql_select_db($db, $conn) OR die ("Impossibile selezionare database");
$sql="SELECT * FROM POI_Table";
$res=mysql_query($sql,$conn);
//For each poi place a list element
while($row = mysql_fetch_array($res)){
                $id=$row['id'];
                $title=$row['title'];
                $link="http://padovaar.altervista.org/gbook/index.php?site
                   =".$site."&id=".$id."";
        echo '<li><a href='.$link.'>'.$title.'</a></li>';
}
```

Listing 16.1: Strutturazione del menù per la selezione delle pagine del guestbook.

Il codice appena presentato deve esser inserito all'interno della dichiarazione degli elementi del menù. In questo caso si è optato per utilizzare una lista di tipo `ul`, opportunamente modificata tramite CSS.

```
<ul class="menu">
        <li class="top">
        <a href="#" class="top_link"><span>Seleziona - Select</span></a>
        <ul class="sub">
...
        </ul>
        </li>
</ul>
```

Listing 16.2: Strutturazione del meù di selezione.

Analizziamo ora il sorgente PHP appena presentato. Inizialmente viene aperta una connessione con il database contente le informazioni di interesse: per far questo è necessario includere il file contenente le credenziali di accesso e, una volta che la connessione è stata instaurata, si potrà eseguire una query nella quale verranno selezionati tutti i POI che si desidera elencare nel menù (in questo caso l'intero insieme); per ognuno di essi sarà recuperato l'id, necessario per la costruzione del link corretto, ed il titolo utilizzato per distinguere i diversi elementi all'interno della lista. Ottenuti questi valori è possibile stampare il codice all'interno della pagina del client. Prima dell'ingresso nel ciclo while, verrà però istanziata una variabile `$site`, la quale sarà utilizzata per informare lo script che genera le pagine guestbook, che l'utente ne sta visionando il contenuto senza l'impiego di Layar Reality Browser: questo dato verrà poi sfruttato per presentare a questi, un link che lo riporterà al menù iniziale. Per usufruire di questa funzione è necessario elaborare il codice del file `index.php` del guestbook, inserendo all'interno del body uno script PHP che legga il parametro `$site` tramite GET, e che sulla base di tale valore visualizzi, o meno, un eventuale link di ritorno.

```
<?php
        if($_GET['site']){
                print"<A HREF=\"../gb_index.php?site=true\">Indietro</A>";
        }
?>
```

Listing 16.3: Istruzione per la visualizzazione di un link al menù di selezione.

16.1 Riepilogo

Grazie al codice presentato gli utenti saranno ora in grado di accedere agevolmente al libro degli ospiti, usufruendo di tutte le sue funzionalità ma senza dover utilizzare la piattaforma Layar. Questo contenuto è stato realizzato raccogliendo in un'unica pagina, tramite una procedura automatica, tutti i link di accesso alle pagine del libro degli ospiti.

Capitolo 17

Visualizzazione di una mappa interattiva

17.1 Mash-up

L'iscrizione al content management system comporta la possibilità di visualizzare ed arricchire i contenuti del layer Padova Tour, in modo semplice e veloce. Fino ad ora sono stati presentati il modulo di registrazione, quello di login/logout e quello per la visualizzazione del guestbook. Il prossimo elemento sviluppato riguarderà la visualizzazione dei POI all'interno di un ambiente grafico.

L'iscritto, che desidera contribuire ai contenuti dello strato, deve poter conoscere quali punti d'interesse sono presenti e dove essi siano ubicati nonché le informazioni ad essi relative, come ad esempio le possibili azioni che un visitatore può intraprendere; ogni POI è inoltre caratterizzato da una coppia di coordinate espresse sotto forma di latitudine e longitudine. L'insieme di queste proprietà permette di utilizzare una carta geografica, e in questo caso un prodotto Google Maps, come strumento per la localizzazione grafica dei punti. La mappa così realizzata verrà integrata con le informazioni contenute nel database, contestualizzate opportunamente: così facendo i diversi POI saranno collocati in maniera consona ed associati ai propri contenuti. Questo risultato è ottenibile realizzando un mash-up.

Mash-up (letteralmente "poltiglia") indica un sito o un'applicazione web di tipo ibrido che utilizza contenuti provenienti da più sorgenti per strutturare un servizio completamente nuovo. Il contenuto dei mash-up è normalmente prelevato da terze parti tramite API, feed (es. RSS e Atom) o Javascript: questi elementi stanno aprendo nuove frontiere allo sviluppo web, permettendo a chiunque di combinare dati provenienti da siti come Amazon.com, eBay, Google, Windows Live e Yahoo! in modi completamente innovativi.

I Web service, una raccolta di standard e formati di dati che consentono alle applicazioni web di comunicare tra loro e di condividere informazioni, sono il fulcro della tecnologia che sta alla base dei mash-up. Proprio grazie all'utilizzo di questi, applicazioni sviluppate in diversi linguaggi di programmazione e implementate su diverse piattaforme, possono diventare un unico grande contenitore per lo scambio o la visualizzazione di dati provenienti da fonti differenti.

Generalmente si possono distinguere tre categorie principali:

1. *consumer Mash-up.* Un consumer mash-up nasce dalla semplice unione di due o più servizi, combinati insieme in un unico ambiente grafico. Un tipico esempio di consumer mash-up sono le applicazioni che integrano le cartografie di Google Maps con dati provenienti da altre sorgenti di informazione;

2. *data Mash-up.* Un data mash-up si realizza tramite l'unione di diverse sorgenti di informazione tra loro eterogenee, al fine di creare un unico contenitore di dati;

3. *business Mash-up.* Un business mash-up consente di combinare dati (sia interni sia esterni all'applicazione interessata) per la creazione di un framework integrato. Solitamente esso viene realizzato per il monitoraggio delle vendite, del personale o per la gestione dei costi all'interno ad un'azienda.

Rispetto a quelle che sono le esigenze richieste dal content management system di PadovaTour, sarà sufficiente lo sviluppo di un'applicazione di tipo consumer: l'implementazione avverrà integrando la API di Google Maps, con alcune funzioni JavaScript e PHP. Al fine di presentare all'utente un ambiente di lavoro chiaro e funzionale, sarà inoltre necessario impostare correttamente la presentazione degli elementi tramite un corretto styling, ad esempio utilizzando i fogli di stile CSS.

Nel seguito verranno presentati i diversi servizi offerti dal CMS tramite mash-up: visualizzazione, inserimento, modifica ed itinerari. Essi si basano su due script JavaScript reperibili presso l'indirizzo `http://www.geocodezip.com/scripts`: `downloadxml.js` e `gxml.js`; si tratta di due collezioni di metodi che si occuperanno di reperire e decodificare (tramite parsing dei valori) informazioni codificate in formato XML. Tali elementi dovranno essere inclusi nell'intestazione di ogni pagina:

```
<script type="text/javascript" src="js/downloadxml.js"></script>
<script type="text/javascript" src="js/gxml.js"></script>
```

A livello di presentazione saranno utilizzate le API per JavaScript di Google Maps, aggiornate alla versione 3.2: l'interfaccia di programmazione JavaScript consente infatti, di incorporare Google Maps all'interno delle proprie pagine web. La versione 3 di quest'API, è stata appositamente progettata per essere più veloce e più applicabile rispetto a dispositivi mobili, così come ai tradizionali browser per desktop. L'API fornisce una serie di utility per mappe che consentono di manipolarne e aggiungervi contenuti attraverso una varietà di servizi, consentendo così di sviluppare solide applicazioni basate su cartografie interattive. Le JavaScript Google Maps API Version 3 sono un servizio gratuito, disponibile per qualsiasi sito web, a patto che siano utilizzate per fini non-commercial [26].

Accanto a questo strumento verrà utilizzata la già presentata libreria jQuery `http://jquery.com/`, la quale permette di effettuare chiamate AJAX, intervenire sulla struttura DOM della pagina e realizzare script funzionali in maniera semplice e concisa. Ogni elemento web dovrà quindi includere questi script nella propria intestazione.

```
<script type="text/javascript"  src="http://maps.google.com/maps/api/js?
   sensor=false"></script>
<script type="text/javascript" src="jquery-1.3.2.min.js"></script>
```

La visualizzazione delle informazioni di ogni POI sarà fornita tramite InfoBox, uno script per Google Maps che offre una facile personalizzazione degli elementi grafici, unitamente ad un plugin jQuery che permette di organizzare in maniera semplice ed elegante i diversi contenuti informativi. Anche questi elementi dovranno essere inclusi nelle pagine che ne fanno uso.

```
<script type="text/javascript" src="infobox.js"></script>
<script src="jquery.tools.min.js"></script>
```

Infine realizziamo un file `map.js` nel quale verranno inserite tutte le funzioni adibite all'interazione con la mappa ed i suoi elementi, e come per i precedenti script, anche questo dovrà essere incluso nell'header di ciascuna pagina.

```
<script type="text/javascript" src="map.js"></script>
```

17.2 Il servizio

Il servizio di visualizzazione dovrà presentare all'utenza i diversi POI, correttamente posizionati all'interno della mappa, e corredati con le proprie informazioni, di modo da permetterne un facile riconoscimento; esso sarà inoltre la base di partenza per i successivi servizi offerti tramite CMS.

In primo luogo, è necessario inizializzare la mappa: ciascuna verrà trattata come un oggetto con delle proprietà che ne caratterizzeranno, ad esempio, il tipo (classica, satellite, ibrida, ...) o le funzioni di navigazione (drag&drop, zoom, ...). A tale scopo implementiamo una funzione `initialize()`, la quale sarà richiamata in ogni pagina che utilizza una cartografia all'interno del costrutto `<body onload="">`, nella quale richiameremo un costruttore per l'oggetto mappa impostandovi le proprietà che più interessano come il livello di zoom, la posizione sulla quale verrà centrata, la tipologia ed i controlli di navigazione. Istanziamo quindi una variabile `map`, che verrà utilizzata a livello globale (in quanto sarà poi richiamata in diversi altri punti dello script) ed una variabile globale `padova` cui assegneremo la posizione della città di Padova, su cui poi centrare la nostra cartografia, espressa sotto forma di un oggetto di tipo `LatLng`.

```
//Makes a call to the map-object constructor
var myOptions = {
    zoom: 8,
    center: padova,
    mapTypeControl: true,
    mapTypeControlOptions: {style: google.maps.MapTypeControlStyle.
       DROPDOWN_MENU},
    navigationControl: true,
        mapTypeId: google.maps.MapTypeId.ROADMAP
}
map = new google.maps.Map(document.getElementById(mappa), myOptions);
```

Listing 17.1: Costruzione di un oggetto `Map`.

Come si può notare dal precedente listato, l'oggetto `Map` richiede, oltre ai valori di setup, un elemento HTML nel quale essere inserito: quest'ultimo può essere ottenuto tramite la chiamata della funzione o, come riportato, all'interno dell stessa tramite l'invocazione del metodo `getElementById()` che restituisce un elemento in base al proprio identificativo `id` (in questo caso essa sarà visualizzata in un elemento `div` con `id = mappa`, dove mappa è un parametro passato in ingresso alla procedura). All'interno della funzione `initialize()`, implementiamo inoltre un gestore degli eventi per la nostra mappa, grazie al quale forzeremo la chiusura delle finestre contenti le informazioni visualizzate. Il seguente listato propone l'invocazione del gestore di eventi, ma esclude i dettagli implementativi al suo interno legati principalmente alla presentazione dei dati che, come anticipato, verranno tralasciati.

```
google.maps.event.addListener(map, 'click', function() {
        //Code for hiding windows
    ...
});
```

Listing 17.2: Strutturazione di un oggetto `Listener` per l'evento 'click'.

La mappa è ora pronta per essere visualizzata all'interno delle nostre pagine, ma non contiene alcuna informazione utile: dovremo quindi realizzare una funzione che leggerà i contenuti del nostro database, e li inserirà correttamente nella pagina del servizio. Tale requisito esprime una nuova condizione, ovvero la necessità di convertire le informazioni contenute nella base di dati, in un formato più accessibile dall'applicazione web, come ad esempio un modello XML.

17.2.1 XMLificazione

Prima di proseguire, è bene sottolineare che non esiste un modo univoco per presentare i dati e, per questa ragione, nel seguito la struttura di questo script sarà analizzata limitatamente agli elementi strutturali evitando quelli di contenuto.

Realizzeremo ora uno script PHP che estrarrà i dati dal nostro database e li restituirà utilizzando il formato XML: in questo modo le informazioni di ogni POI risulteranno organizzate secondo uno schema uniforme e saranno di volta in volta aggiornate. A supporto delle operazioni che verranno svolte, utilizzeremo le sessioni e due script realizzati in precedenza: `db_config.php`, il quale contiene le informazioni di connessione, e `authentication.class.php`, che consente un'agevole gestione della sessione utente.

```
session_start();
include 'db_config.php';
include 'authentication.class.php';
```

La lettura dei dati avverrà limitatamente ai contenuti prodotti dal singolo utente: questa condizione si realizzerà richiedendo che si invii il proprio username assieme alla richiesta per la pagina; tale nominativo sarà poi catturato tramite la direttiva GET ed assegnato ad una stringa da utilizzare come pattern di espressioni regolari.

```
$author = $_GET['username'];
$regex= "^".$author."$";
```

Listing 17.3: Realizzazione di un'espressione regolare in base ai parametri della richiesta HTTP.

Inizializziamo quindi la struttura del documento XML creando un primo nodo radice "markers" i cui figli rappresenteranno i vari POI analizzati, e saranno indicati con il tag "marker".

```
//create the xml document
$xmlDoc = new DOMDocument();
//create the root element
$root = $xmlDoc->appendChild(
                $xmlDoc->createElement("markers"));
```

Listing 17.4: Inizializzazione della struttura del documento XML.

Il passo successivo è l'interrogazione della tabella POI_Auth al fine di conoscere i POI realizzati dall'utente che ha inoltrato la richiesta, e per ognuno di essi andremo ad inserire le informazioni che lo caratterizzano, quali:

- coordinate GPS (latitudine e longitudine), per il posizionamento;

- titolo e identificativo, per il riconoscimento.

Il seguente listato esprime la sintassi PHP da utilizzare durante l'assegnazione dei valori ai diversi elementi.

```
//create marker element
$markerTag = $root->appendChild(
                        $xmlDoc->createElement("marker"));
//create the attributes
$markerTag->appendChild(
                       $xmlDoc->createAttribute("lat"))->appendChild(
                       $xmlDoc->createTextNode($lat));
//add other attributes
...
```

Listing 17.5: Creazione dei nodi "attributo" per il documento XML.

Aggiungiamo quindi un nuovo tag "tabs" a cui verranno assegnati i dati da visualizzare nella finestra informativa :

- autore (in modalità super utente), coordinate, titolo, id, itinerario e immagine,
- azioni,
- dati feedback .

Questi potranno già essere preparati per la presentazione, ad esempio organizzandoli in una tabella o tramite schede.

L'insieme di tutte le informazioni correttamente formattate dovrà essere inserito all'interno di una variabile, che verrà poi utilizzata per completare la struttura del nodo "marker" come riportato nel codice seguente.

```
$contentsTag->appendChild(
                       $xmlDoc->createTextNode($text));
```

Una volta scanditi tutti i POI per l'utente di sessione, il documento XML potrà essere chiuso e preparato per essere stampato in output.

```
header("Content-Type: text/plain");
//Prepare output for printing
$xmlDoc->formatOutput = true;
echo $xmlDoc->saveXML();
```

Listing 17.6: Preparazione dell'header del documento XML e stampa in output.

17.2.2 Lettura dei dati

Proseguiamo la trattazione andando ad analizzare l'implementazione della lettura dei dati, ovvero la funzione `getXmlFile()`. Questa procedura riceve in input un file XML (in questo caso sarà usato l'output dello script PHP) ed un indice (che verrà utilizzato per il retrieving dei dati di un elemento specifico) e sarà richiamata nelle pagine web assieme alla funzione `initialize()` precedentemente discussa.

All'interno di `getXmlFile()` verrà innanzitutto invocata la funzione `downloadUrl()`, contenuta nello script `downloadXml`, la quale recupererà il contenuto XML e ne eseguirà il parsing (`xmlParse(doc)`). Successivamente saranno estratti i dati relativi ai vari "marker" ed inseriti in un array tramite l'istruzione:

```
var markers = xmlDoc.documentElement.getElementsByTagName("marker");
```

Utilizzando un ciclo for andremo quindi ad indagare il contenuto della variabile `markers`, estraendo da ogni suo elemento le informazioni associatevi, attraverso la direttiva `getAttribute()`. Una volta collezionati i dati per il posizionamento, eseguiremo la scansione delle informazioni contenute nel tag "tabs". Il seguente sorgente si riferisce alla procedura appena esposta.

```
var xmlDoc = xmlParse(doc);
//Obtains the array of markers and loops through it
var markers = xmlDoc.documentElement.getElementsByTagName("marker");
for (var i = 0; i < markers.length; i++) {
        //Obtains the attributes of each marker
        var marker_num = gmarkers.length;
        var lat = parseFloat(markers[i].getAttribute("lat"));
        var lng = parseFloat(markers[i].getAttribute("lng"));
        var label = markers[i].getAttribute("label");
    var poi_id = markers[i].getAttribute("id");
    var icon = markers[i].getAttribute("icon");
    var contentStringTabs = "";
    var contentStringBody = "";
    if (isNaN(lat) || isNaN(lng)) {
              alert("bad point "+i);
              continue;
    }
    var point = new google.maps.LatLng(lat,lng);
    //Gets the tabs informations
    var tabInfo = markers[i].getElementsByTagName("tab");
    if ((tabInfo) && (tabInfo.length > 0)) {
    for (var j = 0; j < tabInfo.length; j++) {
                var tabLabel = GXml.value(tabInfo[j].getElementsByTagName(
                   "label")[0]);
        var tabHtml = GXml.value(tabInfo[j].getElementsByTagName("contents
           ")[0]);
        contentStringTabs += tabLabel;
        contentStringBody += tabHtml;
        }
        var contentString = contentStringTabs+contentStringBody;
}
```

Listing 17.7: Estratto della funzione getXmlFile(). Lettura e formattazione dei valori di input.

Ogni punto, per essere visualizzato nella mappa, dovrà essere associato ad un elemento grafico, ovvero un'icona, che lo renda riconoscibile. A tale scopo realizziamo una struttura a scope globale contenente i dati relativi alle diverse icone applicate (variabile `gicons`), la quale verrà esplorata tramite la chiamata `getMarkerImage()`, di seguito presentata. Si noti che in questo caso sono state utilizzate delle icone fornite da Google (gratuitamente) e disponibili in 8 colori.

```
function getMarkerImage(iconColor) {
   if ((typeof(iconColor)=="undefined") || (iconColor==null)) {
       iconColor = "red";
        }
        if (!gicons[iconColor]) {
   gicons[iconColor] = new google.maps.MarkerImage("http://labs.google.com
      /ridefinder/images/mm_20_"+ iconColor +".png",
    // This marker is 20 pixels wide by 34 pixels tall
    new google.maps.Size(12, 20),
    // The origin for this image is 0,0
    new google.maps.Point(0,0),
    // The anchor for this image is at 6,20
    new google.maps.Point(6, 20));
   }
   return gicons[iconColor];
}
```

Listing 17.8: Funzione `getMarkerImage()`.

Ultimata la raccolta delle informazioni effettueremo la chiamata alla funzione che realizza l'inserimento del POI nell'elemento `Map`.

```
var marker = createMarker(point, label, contentString, icon, poi_id, start
   , reset);
```

Listing 17.9: Invocazione del metodo `createMarker()`.

17.2.3 Marker

La funzione `createMarker()` riceve in ingresso le coordinate di un punto, un'etichetta da associarvi, del contenuto informativo, il tipo di icona prescelto, l'identificativo (relativo alla tabella POI_Table) del POI rappresentato e due parametri d'aggiornamento.

```
var marker = new google.maps.Marker({
    position: latlng,
    icon: getMarkerImage(icon),
    shadow: iconShadow,
    map: map,
    title: name,
    zIndex: Math.round(latlng.lat()*-100000)<<5
});
```

Listing 17.10: Creazione di un oggetto `Marker`.

L'istruzione sopra riportata richiama il costruttore per l'oggetto `Marker`, cui verranno associate la posizione geografica, l'icona, il titolo di riconoscimento e l'oggetto `Map` nel quale essere inserito, ricevuti in input. Una volta completata tale operazione verrà richiamata la procedura `bindInfoWindow`(), nella quale si eseguirà un linkage tra il marker e una finestra (InfoBox) nella quale verranno inserite le informazioni distintive del punto: ad ogni oggetto marker verrà difatti associato un `Listener` di eventi, che eseguirà il bind ad un evento "click". Nel seguito questa funzione verrà utilizzata anche su oggetti non di tipo `Marker`, motivo per cui sarà opportuno inserire un ulteriore controllo grazie al quale andremo a catturare l'esatta posizione dell'evento click all'interno della mappa.

```
google.maps.event.addListener(object, 'click', function(event) {
        if (event) {
                point = event.latLng;
        }
//Code to use onclick for showing data
...
```

Listing 17.11: Creazione di un oggetto `Listener`.

A questo punto l'oggetto POI è stato realizzato e correttamente inserito nella mappa: possiamo quindi eseguire un push del marker in questione all'interno di un array `gmarkers[]` (con scope globale) per potervi accedere in futuro; inoltre risulterà utile mantenere un corretta corrispondenza tra l'id di ciascun marker ed il suo posizionamento all'interno dell'array `gmarkers`: per ogni POI salveremo la coppia *(id, indice)*.

17.2.4 Accessibilità

Un altro requisito da esprimere, risiede nella facile accessibilità ai punti da parte dell'utente. Infatti, osservando i marker posizionati sulla mappa, risulterebbe difficile eseguirne una corretta distinzione: un primo approccio potrebbe essere quello di visualizzare contemporaneamente le informazioni di ciascuno, oppure un'etichetta contenente un identificativo; purtroppo tale approccio influirebbe negativamente sull'esperienza dell'utente presentando a schermo una situazione decisamente caotica. Accanto alla mappa andremo perciò a posizionare un elenco dei punti in essa contenuti.

All'interno della funzione `createMarker`() aggiungeremo il seguente codice, che andrà ad eseguire un "append" di un nuovo elemento `li` ad una lista presente all'interno del nostra pagina web; quest'elemento verrà inoltre associato un controllo dell'evento click.

```
//Add the new marker to a global array structure
gmarkers.push(marker);
var index = gmarkers.length-1;
var temp = new Array();
temp[0]=id;
temp[1]=index;
mindex.push(temp);
```

```
//Create a list element connected to the marker
$("<li id=\""+id+"\"/>")
        .html(name)
        .click(function(){
        zoomToPOI(gmarkers[index]);
        myclick(index);
        })
        .appendTo("#list");
```

Listing 17.12: Salvataggio del riferimento all'oggetto `Marker` e aggiornamento dell'elemento `list`.

Il sorgente appena presentato rappresenta la prima componente del mash-up, realizzata attraverso l'interconnessione tra un elemento `google.maps` e un JavaScript jQuery (le funzioni `html()`, `click()` e `appendTo()` appartengono a quest'ultima libreria).

Esaminiamo infine le funzioni `zoomToPOI()` e `myclick()`, le quali realizzano la centratura della mappa sul punto selezionato ed il triggering dell'evento click sullo stesso; `myclick()` riceverà in input un indice relativo all'array `gmarkers[]`, mentre `zoomToPOI()` utilizzerà il marker in esame e ne estrarrà la posizione espressa come oggetto `LatLng`.

```
//This function triggers the click event on a marker
function myclick(i) {
        google.maps.event.trigger(gmarkers[i], "click");
}

//This function zooms on a particular POI
function zoomToPOI(point) {
        map.setCenter(point.position);
    map.setZoom(17);
}
```

Listing 17.13: Funzioni `myclick()` e `zoomToPOI()`.

17.2.5 Pagina web service

Giunti a questo punto, le funzionalità base del servizio sono state implementate e non rimane che preparare il codice per la pagina web da cui esso sarà acceduto. Apriremo quindi un nuovo documento `view_map.php` in cui andranno incluse tutte le classi presentate in precedenza. La struttura HTML sarà molto semplice, poiché sarà sufficiente inserire un `div` dove presentare la mappa ed un elemento `ul` per la lista dei marker.

Per invocare i metodi contenuti nel file `map.js` realizzeremo un breve script, richiamato tramite l'evento `onload` del `body`, contenente le chiamate ai metodi `initialize()` e `getXmlFile()`. Poiché il servizio sarà disponibile pubblicamente (in questo caso) possiamo decidere di visualizzare tutti i POI acquisibili dallo strato PadovaTour, realizzando così una panoramica dei suoi contenuti. All'interno del tag `script`, implementeremo una chiamata della procedura `getXmlFile()`, inviando l'indirizzo del servizio di parsing XML ed assegnando come username la stringa "`.*`" (interpretata nelle regular expression come "qualsiasi carattere") di modo da selezionare tutte le righe contenute nella tabella POI_Auth.

17.2.6 Riepilogo

Il meccanismo alla base dei servizi di geo localizzazione del CMS è stato sviluppato inserendo in una pagina web un contenitore per un elemento `Map` ed uno per la lista dei luoghi visualizzabili, approntando degli script per la gestione dei contenuti in essi presentati. In primo luogo è stata sviluppata una funzione per l'inizializzazione dell'ambiente `Map`; successivamente è stata elaborata una procedura per la rappresentazione dei dati in formato XML, contenuti in una base di dati, associata ad una per la lettura di questi all'interno delle pagine del CMS. Una volta interpretate, le informazioni su ciascun punto d'interesse sono quindi state inserite nella mappa realizzata e rese disponibili ed interattive (tramite l'istanziazione di gestori di eventi).

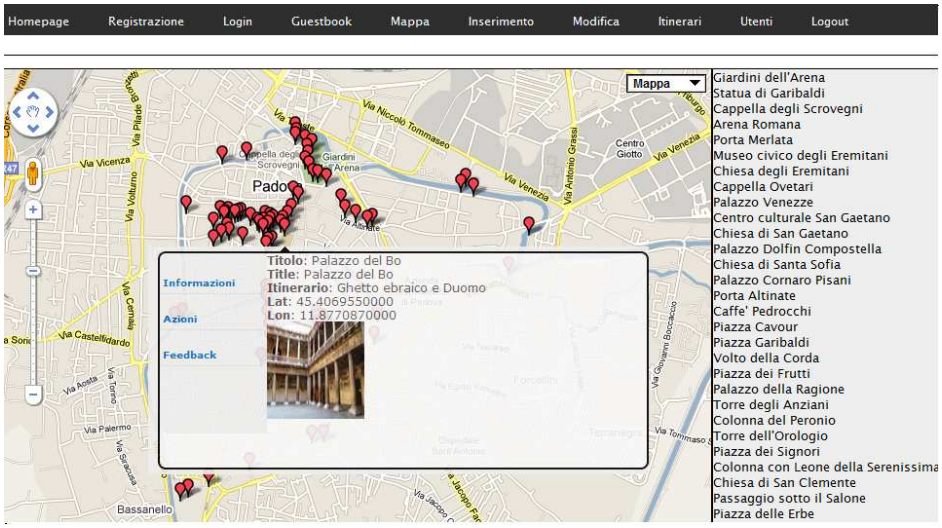

Figura 17.1: Visualizzazione della mappa.

Capitolo 18

Gestione dei contenuti

18.1 Inserimento

Passiamo quindi al primo servizio per utenti registrati: l'inserimento di nuovi POI nel sistema. Il proponimento, su cui si fonda tale web service, è quello di sfruttare la visualizzazione grafica per favorire l'interazione con l'ambiente da parte del fruitore. Accanto al servizio di base (in cui verranno visualizzati solo i POI realizzati dal client) sarà posto un form tramite il quale l'utente potrà sottoporre i dati relativi ai nuovi elementi.

18.1.1 Inserimento nel database: il form

Ogni singolo POI risulta caratterizzato da una serie di informazioni che lo contraddistinguono dagli altri. Per eseguire un inserimento nel database di un nuovo hotspot è necessario conoscere tali dati: accanto alla mappa andremo a posizionare un form dove questi potranno essere inseriti dall'utente. Le informazioni da sottomettere al database dovranno riguardare:

- titolo del POI, sia in italiano che in inglese;
- posizione geografica, espressa come latitudine/longitudine o come indirizzo (si veda in seguito);
- URL dell'immagine per l'applicazione Layar (100x100 px) e per il Guestbook;
- itinerario d'appartenenza (nuovo o preesistente);
- punto iniziale del percorso;
- azioni:
    - nome azione;
    - URI azione;
    - tipologia e range del trigger.

Oltre a questi, inseriremo anche un campo nascosto (`<input type=''hidden'' />`) ove verrà mantenuto un contatore delle azioni aggiunte per ciascun inserimento: ogni punto infatti potrà avere un numero arbitrario di azioni ad esso collegate (esclusa la *Guestbook* action che è inserita di default).

Poiché tra gli scopi del CMS vi è la realizzazione di un ambiente che favorisca la semplicità d'uso, è opportuno inserire una serie di controlli che gestiscano la visualizzazione dei campi del form

a seconda dell'input immesso dall'utente. Nell'implementazione discussa si è scelto, ad esempio, di nascondere i dettagli di creazione per nuovi itinerari se viene selezionato uno preesistente, oppure di nascondere i campi azione quando non richiesti. Vediamo ad esempio la funzione, da includere nella pagina del servizio, entro il tag `script`, che andrà ad aggiungere al form un elemento contenente i dati di un'azione, agendo sulla struttura DOM della pagina attraverso un'operazione di clonazione.

```
//On page loading, set counter to zero
var counter = 0;
//This function add a new field to the form
function moreFields() {
        counter++;
        var newFields = document.getElementById('azione').cloneNode(true);
        newFields.id = '';
        newFields.style.display = 'inline';
        var newField = newFields.childNodes;
        for (var i=0;i<newField.length;i++) {
                var nome = newField[i].name;
                if (nome)
                        newField[i].name = nome + counter;
        }
        var insertHere = document.getElementById('actions');
        insertHere.parentNode.insertBefore(newFields,insertHere.
           nextSibling);
        document.getElementById("num_actions").value=counter;
}
```

Listing 18.1: Script per la gestione del campo "azione" nel form d'inserimento.

`moreFields`() richiede che nella pagina sia presente un elemento HTML "azione" contenente i campi da clonare, ed un ulteriore elemento "actions" in cui posizionare il nodo clonato; durante la copia i campi del nodo vengono rinominati (in una pagina HTML gli identificativi devono essere unici) ed il valore del contatore `counter` viene aggiornato assieme al relativo campo nascosto(`num_actions`).

18.1.2 Draggable marker

Per favorire il corretto collocamento dei POI, senza conoscere le coordinate GPS di questi, è conveniente inserire un marker di tipo draggable, ovvero un oggetto che l'utente può spostare a piacimento all'interno della mappa. Realizzeremo una nuova funzione `createDraggableMarker`() che verrà invocata nella pagina del servizio, tramite la chiamata alla procedura `draggableMarker`().

```
function draggableMarker(label, point){
        if(((typeof(point)=="undefined") || (point==null))){
        point=padova;
    }
        createDraggableMarker(point,label,"blue");
}
```

Listing 18.2: Funzione `draggableMarker`().

`createDraggableMarker`() ricalca in parte la funzione `createMarker`(), ma riceverà in ingresso solamente un elemento `LatLng`, una stringa di testo ed un'icona; anche in quest'occasione sarà utilizzato un oggetto `Marker`, ma nella sua dichiarazione verrà impostato su `true` il valore della variabile `draggable`; inoltre sarà importate impostare correttamente il valore `zIndex`, di modo che l'elemento sia sempre visibile all'interno della mappa e non venga occultato da altri punti.

```
var marker = new google.maps.Marker({
        position: latlng,
        icon: getMarkerImage(icon),
        shadow: iconShadow,
        map: map,
        title: name,
        draggable: true,
        zIndex: -9999999999
});
```

Listing 18.3: Istanzazione di un oggetto `Marker` con proprietà di drag&drop.

Ancora una volta eseguiamo un push in `gmarkers`[] e `mindex`[], e conserviamo in una variabile globale `d_mark` l'indice della cella in cui è conservato il marker. A questo punto non resta che gestire l'interazione dell'utente: impostiamo quindi un nuovo gestore degli eventi.

```
google.maps.event.addListener(marker, "dragend", function() {
        var point = marker.position;
        map.panTo(point);
        google.maps.event.trigger(point, "position_changed");
});
google.maps.event.addListener(marker, "position_changed", function() {
        var point = marker.position;
        document.getElementById("latitude").value = point.lat();
        document.getElementById("longitude").value = point.lng();
});
```

Listing 18.4: Istanziazione di oggetti `Listener` per gli eventi "dragend" e "position_changed".

Il listato proposto fornirà il controllo per l'evento "dragend" (ovvero la terminazione dell'interazione) leggendo il valore della posizione corrente sulla mappa ed innescando un ulteriore evento "position_changed"; quest'ultimo riceverà in input la posizione del marker passandola al form d'ingresso in tempo reale. Il gestore dell'evento "dragend" inoltre eseguirà la centratura della mappa nella nuova posizione (`panTo()`). In questo modo si potrà esplorare la cartografia digitale di Google Maps tramite lo spostamento del nuovo marker. Si noti infine che non è stata associata alcuna finestra informativa a questo tipo di marker.

L'elemento così realizzato potrà quindi essere trascinato all'interno della mappa fino alla località desiderata: non tutti gli utenti però vorranno dover muovere il marker dalla sua posizione originaria al luogo prescelto. Implementeremo quindi una breve funzione che posizionerà il draggable marker in un punto della mappa solamente eseguendo un "doppio click" su di essa. La funzione `enableDoubleClick`() riceverà in ingresso un oggetto mappa e, se vi è un draggable marker settato, vi aggiungerà un gestore per l'evento `dblclick`: in risposta all'input dell'utente, questo leggerà la posizione del cursore assegnandola al parametro `position` del marker, ed infine invocherà il gestore dell'evento "position_changed" per aggiornare i campi del form con le nuove coordinate.

```
function enableDoubleClick(map){
        var point = null;
        google.maps.event.addListener(map, 'dblclick', function(event) {
                if (event) {
                        point = event.latLng;
                }
        if(d_mark!=null){
                        gmarkers[d_mark].position = point;
                        gmarkers[d_mark].setMap(map);
                        google.maps.event.trigger(gmarkers[d_mark], "
                           position_changed");
                        });
        }
        return;
}
```

Listing 18.5: Funzione `enableDoubleClick()`.

La chiamata a questo metodo sarà inserita nell'inizializzazione del servizio.

18.1.3 Geocoding

Le API di Google Maps si prestano particolarmente alle funzioni d'inserimento di nuovi punti nella mappa: è infatti possibile usufruire della funzionalità di *geocoding*, un servizio di traduzione degli indirizzi grazie al quale diventa possibile identificare una corrispondenza tra un indirizzo civico ed una posizione espressa in coordinate terrestri. Perciò, tramite geocoding, l'utente potrà aggiungere un nuovo POI collocato ad esempio all'indirizzo "Padova, Via Gradenigo, 6/B" senza dover conoscere i suoi valori di latitudine e longitudine. Le API di Google Maps presentano questo servizio attraverso la classe `Geocoder`.

All'interno del form d'inserimento, aggiungeremo un nuovo campo di testo "address" che conterrà l'indirizzo da tradurre. Nella pagina del servizio apporremo un nuovo inserto `script`, nel quale sarà in primo luogo inizializzato un oggetto `Geocoder` tramite la seguente istruzione:

```
var geocoded = new google.maps.Geocoder();
```

Aggiungiamo quindi una funzione `codeAddress()`, la quale riceverà in ingresso tre parametri che rappresenteranno gli identificativi di alcuni elementi del form: nello specifico si tratterà dell'indirizzo da codificare (`address`), della latitudine (`latitude`) e della longitudine (`longitude`) da ritornare. La funzione leggerà il contenuto del campo `address` e lo codificherà utilizzando l'oggetto `Geocoder`; in seguito, se la codifica andrà a buon fine, aggiornerà la mappa modificandone la posizione centrale e spostando il draggable marker (se presente), e modificherà il valore contenuto nei campi del form `latitude` e `longitude`. Nel caso in cui l'indirizzo riportato dall'utente sia scorretto o non sia geo codificabile, verrà invece restituito un messaggio d'errore. Il listato seguente propone il codice per la funzione `codeAddress()`.

```
function codeAddress(address, latitude, longitude){
    geocoded = new google.maps.Geocoder();
        var add = document.getElementById(address).value;
    geocoded.geocode( { 'address': add}, function(results, status) {
            if (status == google.maps.GeocoderStatus.OK) {
```

```
                var point= results[0].geometry.location;
                //center map, and zoom to poi
                        if(d_mark!=null){
                        gmarkers[d_mark].position = point;
                        gmarkers[d_mark].setMap(map);
                }
                map.panTo(point);
                //Adds the GPS coordinates to the form elements
                document.getElementById(latitude).value = point.lat();
                document.getElementById(longitude).value = point.lng();
                } else {
                        alert(status);
                        geocoded = false;
                }
        });
}
```

Listing 18.6: Funzione `codeAddress`().

18.1.4 Inserimento nel database: PHP

Una volta completati i campi del form, questi saranno inviati al server dove verranno elaborati da una pagina PHP che identificheremo come `insert.php` (da riportare come valore del campo `action` del form stesso); qui i dati verranno sottoposti ad un checking, al termine del quale saranno elaborati ed inviati al database, altrimenti verrà riportato un messaggio d'errore se il controllo avrà avuto esito negativo. Il codice relativo ai controlli dei valori in ingresso e alle procedure d'inserimento nel database sarà omesso in questa trattazione in quanto questo concerne una lunga serie di semplici istruzioni if-else implementate all'interno di pagine PHP, utilizzando metodologie simili a quelle già affrontate nell'analisi delle procedure di registrazione ed autenticazione degli utenti e nell'interazione con le risorse guestbook. Sarà invece esaminata la struttura della chiamata al metodo d'inserimento delle informazioni presso la base di dati: questa dovrà prima ottenere i parametri d'inserimento tramite l'uso di `$_POST` ed applicarvi una procedura di checking, al termine della quale (se non si riscontreranno anomalie) verrà invocata una funzione `success`(); quest'ultima restituirà al client un messaggio, in formato JSON, contenente alcune informazioni sul punto (latitudine, longitudine e id) ed un messaggio di avviso del corretto completamento delle operazioni.

```
function success($data, $message, $id) {
        die(json_encode(array('status' => 'success', 'data' => $data, '
           message' => $message, 'id'=>$id)));
}
```

Listing 18.7: Funzione `success`().

Nel caso in cui i valori di input non siano conformi al modello prestabilito sarà invocata una funzione `fail`(): i dati verranno respinti e l'utente ne sarà informato tramite un messaggio d'errore.

```
function fail($message) {
        die(json_encode(array('status' => 'fail', 'message' => $message)))
           ;
}
```

Listing 18.8: Funzione `fail()`.

18.1.5 Inserimento nel database: AJAX

La sottomissione dei dati del form al database sarà effettuata non attraverso un classico pulsante di tipo `submit`, ma utilizzando una funzione `submitForm()`, contenuta in uno script nella pagina del servizio; questa funzione riceverà in input quattro parametri `(pointForm, address, latitude` e `longitude)` contenenti gli identificativi dei campi del form, che verranno impiegati per eseguire il retrieving dei dati. Il comportamento della funzione varia a seconda che l'utente abbia già eseguito, o meno, il geocoding della posizione del nuovo POI.

- *Posizione non codificata*: Verrà invocata la funzione `geoEncode()`, simile a `codeAddress()` vista in precedenza, la quale tradurrà l'indirizzo inserito ed in caso di successo passerà il controllo a `savePoint()` (che sarà presentata successivamente), altrimenti restituirà un messaggio di errore.

```
function geoEncode(pointForm, field) {
    geocoded = new google.maps.Geocoder();
        var address = $(pointForm +" input[name="+ field+"]").val();
        geocoded.geocode( { 'address': address}, function(result,
           status) {
                if (status == google.maps.GeocoderStatus.OK) {
                        var point= result[0].geometry.location;
                        var data = new Array();
                        data[0]= point.lng();
                        data[1]=point.lat();
                        savePoint(data, pointForm);
                } else {
                        alert(status);
                        geocoded = false;
                }
        });
}
```

Listing 18.9: Funzione `geoEncode()`.

- *Posizione codificata*: I dati presenti nel form non richiedono ulteriori elaborazioni e potranno quindi essere estratti ed inviati alla procedura di salvataggio delle informazioni, ovvero `savePoint()`.

```
var geocode = new Array();
var long = document.getElementById(longitude).value;
var lat = document.getElementById(latitude).value;
geocode[0] = long;
geocode[1] = lat;
savePoint(geocode, pointForm);
```

Listing 18.10: Estratto della funzione `submitForm()`.

Analizziamo ora il codice per la funzione `savePoint()`: in primo luogo si eseguirà una serializzazione del contenuto del form e successivamente verranno aggiornati i valori di latitudine e longitudine in esso presenti, di modo che contengano i valori di posizione geo codificati. In questo caso, in fase d'implementazione, la policy utilizzata è stata quella di assegnare una priorità superiore all'indirizzo del punto, se inserito, e non alle sue coordinate. Il processo di serializzazione si basa su un metodo della libreria jQuery, `serializeArray()`, che crea un array di oggetti JavaScript adatti ad essere codificati in una stringa JSON.

```
var data = $(pointForm+" :input").serializeArray();
data[data.length] = { name: "lng", value: geocode[0] };
data[data.length] = { name: "lat", value: geocode[1] };
```

Listing 18.11: Serializzazione dell'input e lettura dei valori d'ingresso.

Il client sarà quindi pronto per inviare i dati al server tramite `POST`; il codice riportato di seguito propone la struttura della funzione che si occuperà di caricare le nuove informazioni nel database e di gestire il comportamento dell'applicazione al termine di quest'azione, facendo uso della libreria jQuery (metodo `post()`). Nella successiva sezione sarà esposta l'analisi di questa procedura.

```
$.post($(pointForm).attr('action'), data, function(json){
        if (json.status == "fail") {
                //In case of a failure inform the user
                alert(json.message);
        }
        if (json.status == "success") {
                //In case of a success update the page and inform the user
                alert(json.message);
                clearForm();
                //Create a valid username
                $user = ".*"
        //Load the informations for the new entry into the page
        getXmlFile('XML/poi.php?username='+user+'&pid='+json.id,-1);
        //Reset the draggable marker position on the map
        if(d_mark!=null){
                gmarkers[d_mark].position = padova;
                gmarkers[d_mark].setMap(map);
        }
        }
        }, "json");
}
```

Listing 18.12: Estratto della funzione `savePoint()`.

18.1.6 Aggiornamento della pagina

Una volta che il posting delle informazioni ha avuto successo, non resta che informare l'utente del corretto svolgimento delle operazioni (anche tramite una semplice finestra di dialogo), ed aggiornare la pagina del servizio: quest'ultima operazione avverrà, come riportato nel precedente listato, in seguito alla verifica del contenuto della variabile `json.status`, e può essere suddivisa in due fasi: aggiornamento dell'ambiente Google Maps e reloading della pagina web.

18.1.6.1 Updating della mappa e della lista marker

L'aggiornamento della mappa sarà eseguito inserendovi un marker da associare al nuovo POI, mentre per la lista si dovrà semplicemente aggiungere una nuova voce; queste operazioni vengono già realizzate in fase di inizializzazione dei contenuti dalla procedura `getXmlFile`(). Si noti che tale funzione andrebbe però a rielaborare tutto il contenuto dell'elemento `Map`, generando un considerevole carico di lavoro. La soluzione migliore sarebbe quella poter di scaricare nella pagina solo i dati associati al nuovo punto: andremo quindi a modificare il file `poi.php`, assegnato alle operazioni di XMLificazione, perché possa fornire le informazioni relative ad un particolare POI; l'implementazione avverrà alterando la query di ricerca utilizzando al suo intero un identificatore specifico.

```
//If there's a request for one POI element
if(isset($_GET['pid'])) {
        $pid = $_GET['pid'];
        $query = "SELECT * FROM POI_Auth WHERE poi_ID = $pid";
}else{
        $query = "SELECT * FROM POI_Auth WHERE username REGEXP '$regex'
           ORDER BY poi_ID";
}
```

Listing 18.13: Modifica della funzione di XMLificazione.

Si noti che per eseguire una chiamata a `getXmlFile`() tra i parametri richiesti vi sarà anche uno username: poiché la selezione del POI avverrà in base al suo id e non in base al proprio autore, sarà possibile utilizzare qualsiasi stringa valida.

Per non generare una user-experience confusa sarà opportuno segnalare, utilizzando un'icona speciale, il nuovo POI inserito: per implementare questa funzionalità dovremo dichiarare una variabile globale `insertion` che verrà letta in fase di generazione del marker (ovvero all'interno di `createMarker`()), e se il suo valore sarà impostato su `true` modificherà l'icona per l'elemento in esame.

```
if(insertion){
//Change icon color
        icon="green";
//Reset this condition
        insertion=false;
}
```

Listing 18.14: Modifica della funzione `createMarker`().

Sempre in termini di user-experience, dovremo reimpostare la posizione del marker mobile che potrà occultare il nuovo marker. Il codice seguente riporta le operazioni da eseguire per effettuare l'aggiornamento.

```
//Get last element data
var user=".*";
insertion = true;
```

```
getXmlFile('XML/poi.php?username='+user+'&pid='+json.id,-1);
//Reset draggable marker position, if present
if(d_mark!=null){
        gmarkers[d_mark].position = padova;
    gmarkers[d_mark].setMap(map);
}
```

Listing 18.15: Script di aggiornamento della pagina del servizio.

18.1.6.2 Update dei dati del form

Una volta che il nuovo dato è stato inserito, l'utente dovrà mantenere la possibilità di ripetere l'operazione in questione: perché ciò accada, spetta allo sviluppatore reimpostare i campi del form ai valori di default. I passi da svolgere in questo caso includeranno la pulizia del form e dei campi azione aggiunti dall'utente ed l'azzeramento della variabile `$counter` e saranno implementati attraverso la funzione `clearForm`(), contenuta nella pagina del servizio.

```
//This function clears the form for a new POI insertion
function clearForm(){
        var del = document.getElementById('poi');
        var i = 0;
    while (i<=counter) {
                del.removeChild(del.lastChild);
                i++;
        }
        counter = 0;
        document.getElementById("num_actions").value = counter;
}
```

Listing 18.16: Funzione `clearForm`().

Si noti come procedura sopra riportata, agisca sulla struttura DOM della pagina per eliminare tutti gli elementi inseriti tramite `moreFields`(), applicando alla pagina web un comportamento dinamico.

18.1.7 Pagina web service

Non appena il servizio d'inserimento sarà approntato, non resterà che implementarlo in una pagina web, la quale dovrà contenere la mappa, la lista dei marker ed il form, unitamente ad alcune delle funzioni precedentemente esposte. Si ricordi che tale servizio non sarà pubblico e dovrà contenere un controllo dell'utente, già presentato in precedenza, nella sua intestazione.

Durante il caricamento della pagina dovranno essere invocate le funzioni `initialize`() e `getXmlFile`(): quest'ultima dovrà essere inserita all'interno di un'ulteriore chiamata, di modo che possano essere prima raccolti i dati di sessione.

```
function pois(){
        var user = "<?php
                session_start();
```

```
                $auth = new UserAuthentication();
                if ($auth->IsSAuth()) {
            //Get all data
                        print ".*";
                }else{
                        print $_SESSION['username'];
                }
        ?>";
        draggableMarker("Spostami", null);
    var map = getMap();
    enableDoubleClick(map);
        getXmlFile('XML/poi.php?username='+user,-1);
}
```

Listing 18.17: Funzione `pois()`.

Come si può notare dal codice appena presentato, un superuser disporrà, in questa sezione del content management system, del privilegio di poter visualizzare tutti i POI del sistema.

All'interno dell'applicazione vi è però un altro vantaggio riconoscibile ad un super utente: poter inserire nuovi elementi all'interno degli itinerari realizzati dai diversi autori. Potrà infatti presentarsi l'eventualità che un percorso non risulti strutturato correttamente e che un utente privilegiato ritenga opportuno porvi delle modifiche: tramite questa feature l'amministratore potrà inserire un nuovo POI all'interno di un itinerario di un altro autore; durante la stesura del progetto si è scelto inoltre di assegnare al proprietario del percorso la paternità del nuovo elemento, al fine di non sviluppare comportamenti indesiderati e non impedire la libera gestione dei propri contenuti da parte dei diversi autori (la modifica dei dati avverrà infatti solo per le creazioni del singolo utente). Per implementare quest'estensione del servizio è sufficiente modificare il file `insert.php` (precedentemente analizzato) inserendovi un controllo dei privilegi operato tramite `IsSAuth()`.

18.1.8 Riepilogo

La procedura di inserimento si compone di diverse fasi e parti. Innanzitutto l'utente dovrà autenticarsi presso il CMS: solo in seguito a quest'operazione il servizio potrà caricare i dati corretti appartenenti all'autore. Tramite la compilazione di un form potranno essere inviati i dati relativi ad un nuovo elemento: l'utente potrà però trovare delle difficoltà nel conoscerne la latitudine e longitudine. Grazie alle API di Google Maps è però possibile superare quest'inconveniente utilizzando funzionalità di dragging e geocoding.

Una volta ottenuti i dati essi dovranno essere validati dal server che in caso di incoerenze nella forma di questi restituirà all'utente un errore, altrimenti porterà a termine la procedura d'inserimento e aggiornerà la pagina del servizio.

18.2 Modifica

Le funzionalità di modifica estendono il servizio di base inserendovi dei controlli dei campi d'informazione. Nello specifico durante la formattazione delle informazioni in XML, verrà aggiunto un form HTML tramite il quale l'utente potrà sottomettere al server le nuove informazioni. I controlli dei dati riguarderanno:

- modifica delle informazioni dei POI (titoli, immagini, itinerario),
- modifica dati guestbook ,
- reset informazioni di rating,
- eliminazione dei POI,
- modifica dati azioni (titolo, uri, trigger, range),
- eliminazione delle azioni.

Ciascuna tipologia di editing sarà associata ad una precisa funzione JavaScript che leggerà i contenuti da sottoporre al sistema, associandoli a delle variabili temporanee, per poi inviarli tramite la funzione `ajax`() della libreria jQuery; quest'ultima permetterà di inoltrare una richiesta di tipo asincrono al server, dove saranno eseguiti alcuni script PHP che completeranno l'operazione di aggiornamento della base di dati. Una volta che le modifiche saranno state applicate con successo, lo script del servizio si preoccuperà di aggiornare la mappa, la lista dei marker e le finestre d'informazione, mentre in caso contrario ritornerà all'utente un messaggio d'errore.

Data la presenza di una struttura comune, nel seguito alcune funzioni saranno esaminate solo attraverso le loro caratteristiche distintive; allo stesso modo saranno esclusi i dettagli degli script PHP che realizzano l'interazione con il server MySQL: tali procedure ricevono infatti i dati modificati tramite POST e, dopo averne verificato la validità, eseguono operazioni di INSERT, UPDATE o DELETE; qualora non fosse possibile completare tali operazioni l'utente ne verrà informato attraverso un alert dialog.

Prima di proseguire, si noti che è necessario che le informazioni associate a ciascun punto siano visibili e modificabili dall'utente: nell'implementazione corrente si è optato per inserire nello script di XMLificazione dei dati, un controllo per un parametro `edit` grazie al quale il server saprà se aggiungere alla risposta anche un form di modifica.

18.2.1 Modifica POI

La prima funzione esaminata sarà quella di editing delle informazioni di un POI: `edit_poi`(); questa riceverà in input un singolo parametro, l'id dell'elemento (riferito alla tabella POI_Table) e leggerà il contenuto dei campi del form di modifica conservandone i valori; verrà quindi richiamata la funzione `ajax`() impostando il tipo di operazione (POST), l'url di destinazione, i dati da consegnare, il formato della risposta e la procedura da eseguire in seguito al completamento. In caso di successo verrà aggiornato l'elemento della lista, relativo al POI in esame, ed il contenuto della mappa tramite la chiamata `getXmlFile`(). Di seguito il codice proposto per questa funzione.

```
function edit_poi(id){
    //Gets data from the form
        var title= $('#poi_title').attr('value');
        var line = $('#poi_line').attr('value');
        var url = $('#poi_image').attr('value');
        var urlB = $('#poi_imageBig').attr('value');
        var route= $('#route').attr('value');
        var nuovo= $('#iter').attr('value');
        var color = $('#color').attr('value');
    var start = $('#poi_start').attr('value');
    //Sends data to the server
```

```
        $. ajax ({
        type:"POST",
        url:"php/poi_edit.php?edit=1",
        data:"poi_id="+id+"&poi_title="+title+"&poi_line2="+line+"&
           poi_image="+url+"&poi_imgB="+urlB+"&poi_route="+route+"&poi_new
           ="+nuovo+"&color="+color+"&start="+start,
        dataType: "json",
        success : function (json){
                if (json.status == "fail"){
                        alert(json.message);
                }else{//Success
                        //Refresh the list
                        var elem= document.getElementById(id);
                        elem.innerHTML=title;
                 //Refresh the information window
                        getXmlFile('XML/poi.php?edit=true&username='+user+
                           '&pid='+id,id);
                alert(json.message);
                }
        }
        });
}
```

Listing 18.18: Funzione `edit_poi`().

Si noti che è opportuno eseguire una procedura di validazione dei dati inseriti dall'utente, ad esempio per evitare che inserisca due punti iniziali per lo stesso percorso.

Poiché la modifica avverrà sempre su un singolo POI, in fase di aggiornamento dovremo richiedere la lettura delle informazioni di un unico dato (come nel caso di update dovuto ad inserimento) altrimenti l'invocazione di `getXmlFile`() andrebbe ad incidere sul carico di lavoro: dovendo il sistema leggere l'intero insieme di dati, il tempo sarebbe proporzionale alle dimensioni dei contenuti sottomessi da un autore; inoltre verrebbe creato un nuovo elemento marker per la mappa, nonostante questo sia già presente. Risulta necessario apportare alcune modifiche al metodo `getXmlFile`(), inserendovi dei controlli aggiuntivi.

Assieme all'indirizzo del file XML, verrà quindi passato un ulteriore parametro che conterrà l'identificativo del hotspot modificato (valore indispensabile per l'aggiornamento rapido dei contenuti) e determinerà il comportamento del metodo; a livello implementativo si è optato per utilizzare questo come unico valore di controllo: se esso rappresenterà un id valido allora sarà richiesto solo un aggiornamento dei contenuti, altrimenti (ad esempio passando un valore negativo) il metodo proseguirà utilizzando le procedura precedentemente esaminate.

```
function getXmlFile(filename,start) {
        var resetAll = true;
        if(start<0){
                start=0;
        }else{
        //Selects the correct element in the gmarkers[] array
                start=searchIndex(start);
                resetAll = false;
        }
...
```

Listing 18.19: Modifica della funzione `getXmlFile()`.

Per modificare il marker corretto si dovrà conoscere la sua posizione all'interno dell'array `gmarkers[]`: ciò potrà essere ottenuto eseguendo una scansione dell'array `mindex[]`, realizzato in fase di caricamento dei marker e che contiene le corrispondenze *(indice, id)*, utilizzando un metodo `searchIndex()`, di seguito presentato.

```
//This functions get the index for the marker of a certain POI having
    its id
function searchIndex(id){
for(var j=0;j<mindex.length;j++){
     if (mindex[j][0]==id) return mindex[j][1];
        }
}
```

Listing 18.20: Funzione `searchIndex()`.

Infine si dovrà modificare la funzione `createMarker()` di modo che, leggendo il parametro `resetAll`, non venga ne realizzato un nuovo elemento marker ne operato un update della lista.

```
if(resetAll) {
        gmarkers.push(marker);
}else {
        //Use only one element
        gmarkers[start]=marker;
        gmarkers[start].setMap(map);
}
if(resetAll){
        var index = gmarkers.length-1;
        var temp = new Array();
        temp[0]=id;
        temp[1]=index;
        mindex.push(temp);
        $("<li id=\""+id+"\"/>")
        .html(name)
        .click(function(){
                zoomToPOI(gmarkers[index]);
                 myclick(index);
        })
        .appendTo("#list");
}
```

Listing 18.21: Modifica della funzione `createMarker()`.

A questo punto sono state approntate le operazioni per l'aggiornamento delle informazioni associate a ciascun POI. Gli utenti dovranno poter interagire ad un livello più alto con il sistema: inseriamo quindi la possibilità di gestire i contenuti di feedback, quali rating e messaggi guestbook.

18.2.1.1 Reset rate

Il reset della valutazione associata ad un punto avverrà inoltrando una richiesta di tipo POST, inviando come unico riferimento l'id di quest'ultimo. Una volta elaborata la richiesta, il server eseguirà un'operazione UPDATE sulla tabella "rating".

```
//Reset the POI rating
$sql="UPDATE POI_Table SET tot=1, rating=3 WHERE id=$id";
mysql_query($sql);
success("Rate resettato");
```

Listing 18.22: Procedura per il reset del valore di rating.

18.2.1.2 Gestione messaggi

La gestione dei messaggi guestbook può rivelarsi molto utile qualora non vi siano meccanismi efficaci per la prevenzione di attacchi di tipo "flood" (come un controllo CAPTCHA) o di blacklisting.

Inseriremo un nuovo metodo `del_msg`() che, una volta invocato, reindirizzerà l'utente verso una pagina web (`message.php`) ove verranno stampati tutti i messaggi connessi al POI in esame. Accanto a ciascun messaggio inseriremo un elemento HTML che permetterà all'utente di cancellare il post. Oltre all'interazione con la base di dati, inseriremo anche un controllo della struttura DOM al fine di eliminare il nodo contenente le informazioni del messaggio eliminato: la funzione dovrà quindi ricevere come parametri d'ingresso sia l'identificativo del messaggio nella tabella "messages", che un indice di posizione all'interno della struttura HTML. Ciò significa che lo script che si preoccuperà di eseguire la stampa a schermo dei messaggi, dovrà anche caratterizzare ogni feedback testuale con un valore univoco. Vediamo quindi la procedura di creazione per la pagina in questione:

```
$query = "SELECT * FROM messages WHERE poi_ID=$id";
$res = mysql_query($query);
$index=0;
while($row=mysql_fetch_array($res)){
        $name = $row['name'];
        $msg = $row['message'];
        $msg_id= $row['id'];
        print "
    <table id=\"table_$index\">
        /
/Insert message data
    ...
        </table>        ";
        $index++;
}
```

Listing 18.23: Realizzazione della pagina di gestione del guestbook.

e la funzione di cancellazione per i commenti degli utenti:

```
function del_msg(id, index){
        var answer = confirm("Vuoi eliminare il messaggio?");
        if(answer){
                $. ajax ({
                type:"POST",
                url:"msg_edit.php?id="+id,
                success : function (){
                        var tmp = document.getElementById('table_'+index);
                tmp.parentNode.removeChild(tmp);
                tmp = document.getElementById('counter');
                var val = parseInt(tmp.innerHTML)-1;
                document.getElementById('counter').innerHTML = val;
                        }
                });
        }
}
```

Listing 18.24: Funzione `del_msg()`.

18.2.2 Eliminazione POI

Spesso potrà accadere che un utente inserisca un punto erroneamente o semplicemente desideri rimuoverlo dai propri itinerari. Si dovrà quindi fornire un metodo che elimini tutti i dati, relativi all'elemento specifico, dal database. Questo dovrà ricevere in input l'identificativo del POI e dovrà interagire con il server MySQL tramite l'istruzione DELETE. Questa funzione, molto simile a quelle di inserimento, non sarà però presentata, ma verrà invece esaminato il codice per l'aggiornamento in seguito all'eliminazione dei POI dal sistema.

Il metodo agirà sia a livello DOM (rimuovendo l'elemento `li` collegato al punto eliminato), sia a livello di scripting tramite la funzione `delclick()`, la quale riceverà in input l'identificatore del hotspot e si preoccuperà di occultare all'utente il marker associatovi (reperito tramite la funzione `searchIndex()`). Vediamone l'implementazione.

```
function del_poi(id){
        var answer = confirm("Vuoi eliminare l'elemento?");
        if(answer){
                $. ajax ({
                type:"POST",
                url:"php/poi_edit.php?edit=0",
                data:"poi_id="+id,
                dataType: "json",
                success : function (json){
                        //refresh
                        if(json.status=="fail"){
                                alert(json.message);
                        }else{
                //Remove list element
                                var lista = document.getElementById('list'
                                    );
                                var elem = document.getElementById(id);
                lista.removeChild(elem);
```

```
                //Hide marker
                                delclick(id);
                //Close information window (if opened)
                mapclick();
                //Alert user
                                alert(json.message);
                        }
                        }
                });
        }
}
```

Listing 18.25: Funzione `del_poi`().

18.2.3 Azioni

18.2.3.1 Inserimento azioni

La gestione delle azioni è un elemento focale dello strato PadovaTour, e in generale di un layer, poiché determina il grado d'interazione dell'utente con l'ambiente virtuale: ciascun autore potrà quindi voler aggiungere, modificare o rimuovere questi comportamenti. Si noti che nel seguito non saranno proposti gli script PHP delegati all'interazione con il database.

Per assistere l'inserimento di nuove azioni sarà necessario realizzare un nuovo form i cui campi rappresenteranno i parametri di un'azione: una volta compilato questo modulo l'utente potrà inviare i dati al server cosicché possano essere elaborati e inseriti nella base di dati.

Nella pagina del servizio verrà implementato un metodo `save_action`() che, ricevendo in input un identificativo di un POI, assocerà una nuova azione al punto d'interesse selezionato e inoltrerà una richiesta di aggiornamento per questo.

```
function save_action(id){
        var label = $('#label').attr('value');
        var uri = $('#uri').attr('value');
        var only= $('#triggeronly').attr('value');
        var range= $('#range').attr('value');
        $. ajax ({
        type:"POST",
        url:"php/action.php?edit=2",
        data:"act_id="+id+"&act_label="+label+"&act_uri="+uri+"&act_only="
           +only+"&act_range="+range,
        dataType: "json",
        success : function (json){
                if(json.status=="fail"){
                        alert(json.message);
                }else{
                        //Refresh informations
                        getXmlFile('XML/poi.php?edit=true&username='+user+
                           '&pid='+id,id);
                        alert(json.message);
                }
                }
```

```
        });
}
```

Listing 18.26: Funzione `save_action`().

18.2.3.2 Modifica azioni

Similmente a quanto fatto con il metodo per l modifica dei POI, `edit_action`() invierà una richiesta al server contenente le informazioni da inserire nella tabella ACTION_Table quali l'id del punto d'interesse (necessario per il refresh dei dati) e quello dell'azione stessa. Una volta eseguito il POST dei dati il metodo invocherà la funzione `getXmlFile`() richiedendo un aggiornamento dei contenuti della pagina (limitatamente ad un singolo POI). Di seguito viene riportata l'implementazione proposta per questa procedura.

```
function edit_action(id,pid){
        var label = $('#act_label_'+id).attr('value');
        var uri = $('#act_uri_'+id).attr('value');
        var only= $('#act_only_'+id).attr('value');
        var range= $('#act_range_'+id).attr('value');
        $. ajax ({
        type:"POST",
        url:"php/action.php?edit=1",
        data:"act_id="+id+"&act_label="+label+"&act_uri="+uri+"&act_only="
           +only+"&act_range="+range,
        dataType: "json",
        success : function (json){
                if(json.status=="fail"){
                        alert(json.message);
                }else{
                        getXmlFile('XML/poi.php?edit=true&username='+user+
                           '&pid='+pid,pid);
                        alert(json.message);
        }
        }
        });
}
```

Listing 18.27: Funzione `edit_action`().

18.2.3.3 Eliminazione azioni

Come per i POI, anche le azioni possono dover essere rimosse dal sistema in quanto non più fruibili o ritenute inappropriate. La funzione di cancellazione in questo caso ricalcherà la precedente `edit_action`(), modificando solamente i parametri inviati tramite la richiesta di post. Ancora una volta, in caso di corretto completamento, la finestra dell'utente verrà aggiornata.

```
function delete_action(id,pid){
        var answer = confirm("Vuoi eliminare l'azione?");
        if(answer){
```

```
                $. ajax ({
                type:"POST",
                url:"php/action.php?edit=0",
                data:"act_id="+id,
                dataType: "json",
                success : function (json){
                        //Refresh informations
                        if(json.status=="fail"){
                                alert(json.message);
                        }else{
                                                getXmlFile('XML/poi.php?edit=
                           true&username='+user+'&pid='+pid,pid);
                alert(json.message);
                        }
                }
                });
        }
}
```

Listing 18.28: Funzione `delete_action()`.

18.2.4 Pagina web service

Oltre alle funzioni sopra analizzate, la pagina del servizio dovrà contenere dei metodi per la gestione dei form, come ad esempio per nascondere all'utente campi che non è tenuto a compilare, la verifica dell'autenticazione e il reperimento dei dati di sessioni (già presentate nella sezione precedente).

18.2.5 Riepilogo

Il servizio di modifica è stato costruito sulla base dei precedenti applicando delle funzioni di controllo, realizzate in linguaggio JavaScript, a dei form di compilazione inseriti nella struttura informativa associata a ciascun POI; queste comprendono la modifica e l'eliminazione delle informazioni di un hotspot nonchè l'elaborazione dei contenuti attivi. Infine si è aggiornato il metodo `getXmlFile()` di modo che elabori i dati di un singolo elemento senza andare ad incidere in maniera consistente sulle prestazioni del sistema.

18.3 Itinerari

I POI contenuti nel database, sono organizzati tra diversi itinerari: attraverso questa proprietà si realizza una suddivisione dei punti che permette all'utente finale, del layer PadovaTour, di selezionare quale sottoinsieme di questi visualizzare. Un autore di contenuti dovrà quindi avere la possibilità di modificare le informazioni di ciascun percorso e di verificare quali elementi li costituiscono.

Il servizio per la gestione degli itinerari sarà basato sulle funzionalità Google Maps ma aggiungerà, a quanto fin qui sviluppato, la possibilità di interagire con degli oggetti `Polyline`. La classe `Polyline` definisce sulla mappa un overlay lineare di segmenti tra loro connessi: ogni oggetto consiste di un array di punti `LatLng`,ì e realizza un insieme di linee che li uniscono secondo una sequenza ordinata.

Prima di addentrarci nell'analisi dovremo modificare la tabella "route" contenuta nella base di dati: essa dovrà ospitare le informazioni sui diversi itinerari quali id, autore, nome, e colore (che lo identificherà visivamente all'interno del CMS).

```
ALTER TABLE `route` ADD `username` varchar(255) NOT NULL ;
ALTER TABLE `route` ADD `color` varchar(7) NOT NULL;
```

Listing 18.29: Istruzioni SQL per la modifica della tabella route.

18.3.1 XMLificazione

Come per la visualizzazione dei marker all'interno della mappa, anche per gli itinerari sarà necessario creare un documento XML contenente le caratteristiche di ciascuno di essi. Tale file sarà poi passato in input ad una funzione `getXmlFileRoute`() (presentata in seguito), la quale si occuperà di tradurre il contenuto informativo in elementi interattivi.

Lo script reperirà lo username di sessione ed interrogherà il server MySQL per i percorsi realizzati dall'utente; per ognuno di questi verranno selezionate informazioni riguardanti l'autore, il nome e l'identificativo dello stesso e saranno impostati alcuni attributi grafici (come lo spessore della linea che rappresenterà il percorso, il colore e l'opacità): tali valori saranno successivamente inseriti all'interno di un nodo "lines" che si svilupperà dalla radice del documento. All'interno di ogni tag "lines" saranno inoltre inseriti dei nodi "marker", contenenti l'identificativo e le coordinate dei POI appartenenti all'itinerario in analisi, ed un nodo "tabs" cui saranno associate le informazioni da visualizzare all'interno della finestra informativa.

18.3.2 Oggetti Polyline

L'implementazione degli elementi `Polyline` avverrà prima istanziando un oggetto della classe omonima, utilizzando le impostazioni ricevute tramite il file XML, secondariamente inserendovi un comportamento interattivo ed infine aggiornando la pagina del servizio.

Realizzeremo quindi una funzione `getXmlFileRoute`(), del tutto simile a `getXmlFile`(), nella quale, all'interno di un ciclo iterativo, verranno letti i diversi parametri che caratterizzano il percorso quali il titolo, l'autore, l'identificativo e le informazioni da visualizzare nella finestra informazioni, nonchè alcune proprietà come il colore, l'opacità e lo spessore del tratto. Successivamente, per ogni itinerario dovranno essere raccolti i dati dei punti che lo compongono e i quali andranno conservati in un array `pts`[] come oggetti `LatLng`: grazie ai metodi di questa classe sarà possibile infatti calcolare la lunghezza del percorso tramite la direttiva `distanceFrom`(); il valore così ottenuto non esprimerà necessariamente la reale lunghezza di questo, poiché questo è calcolato come la distanza tra i vari punti senza tenere in considerazione il cammino reale dell'utente. Una buona convenzione da adottare per gli utenti editor, dovrebbe essere quella di inserire, oltre ai POI relativi ai luoghi artistici, anche POI "guida" che indirizzino lungo l'itinerario chi usufruisce del servizio.

Terminata la raccolta delle informazioni sarà possibile istanziare correttamente un nuovo oggetto `Polyline` attribuendovi l'oggetto `Map` a cui essere applicato, l'insieme di punti che vi apparterranno, alcune variabili grafiche ed impostando la possibilità di rilevare eventi "click": quest'ultima proprietà permetterà all'utente di visualizzare una finestra informativa associata al percorso, contestualmente all'ambiente Google Maps. Il seguente listato visualizza l'istanziazione dell'oggetto `Polyline`.

```
var poly = new google.maps.Polyline({
                                    map: map,
                                    path: pts,
                                    strokeColor: colour,
                                    strokeOpacity: opacity,
                                    strokeWeight: width,
                                    clickable: true
                                    });
```

Listing 18.30: Creazione di un oggetto `Polyline`.

Quindi inseriremo, al termine del ciclo interno a `getXmlFileRoute`(), una chiamata alla funzione `createClickablePolyline`(): questa riceverà in ingresso un oggetto `Polyline`, tre stringhe contenenti le informazioni associate al percorso, il titolo, la lunghezza e l'identificativo associati al percorso (nella tabella "route"), ed infine un array `linepoint`[] contenente gli identificativi dei POI che ne definiscono la struttura. Come per la generazione dei marker, inseriremo in un array `gpolys`[] la coppia (`Polyline`, `linepoint`[]) e salveremo la corrispondenza tra l'identificativo di ogni percorso (riferito al database) e l'indice della cella i-esima in cui verrà conservato.

Il prossimo passo sarà quello di associare una finestra InfoBox ad ogni percorso: per far questo si dovrà eseguire una chiamata alla funzione `bindInfoWindow`() (analizzata in precedenza), la quale eseguirà il bind tra un gestore per eventi di tipo "click" e l'oggetto `Polyline`. Tale funzione richiederà però che tra i parametri vi sia un elemento `Marker` a cui ancorare la finestra informativa: prima della chiamata istanziamo quindi un nuovo marker, il quale non verrà visualizzato all'interno della mappa, assegnandovi una posizione GPS arbitraria (ad esempio le coordinate del punto centrale del percorso). Il seguente listato riporta le istruzioni per la realizzazione della procedura `createClickablePolyline`().

```
//Links points to lines
var tmp = new Array();
var tot = linepoint.length;
tmp[0] = poly;
tmp[1] = linepoint;
gpolys.push(tmp);
var poly_num = gpolys.length - 1;

//Links routes to their database-id
tmp = new Array();
tmp[0] = poly_num;
tmp[1] = id;
polyindex.push(tmp);

//Create an "anchor" marker
var marker = new google.maps.Marker({
             position: point    });
bindInfoWindow(poly, marker, map, contentString);
```

Listing 18.31: Contenuto della funzione `createClickablePolyline`().

18.3.3 Funzioni interattive

Il servizio permette ora di visualizzare i percorsi ed i relativi InfoBox contestualmente all'ambiente Google Maps. Nel rispetto degli obbiettivi del content management system, diviene necessario inserire un'interfaccia di controllo che consenta all'utente un'intuitiva e semplice gestione degli itinerari: l'ambiente della mappa si è infatti arricchito di diversi elementi tra cui è necessario poter distinguere, senza però sacrificarne l'accessibilità.

Il requisito di accessibilità verrà soddisfatto realizzando, come per i marker, una lista di tutti gli itinerari realizzati dall'utente in cui ogni elemento sarà collegato ad una funzione di trigger per l'evento "click" associato all'oggetto `Polyline` corrispondente: in questo modo selezionando un elemento dalla lista, verrà operata una nuova centratura della mappa e sarà visualizzato un InfoBox. Nello specifico, all'interno del metodo `createClickablePolyline`(), si andrà ad accodare ad una stringa `side_bar_html` il codice HTML da inserire nella pagina del servizio.

Allo scopo di distinguere efficacemente gli elementi della mappa, implementeremo una nuova funzione `togglePoly`() che visualizzerà od occulterà un itinerario dalla mappa e, se richiesto dall'utente, anche i marker dei punti che la costituiscono. Aggiungeremo la chiamata per questa funzione agli elementi della lista utilizzando ad esempio un elemento di tipo `checkbox`, come proposto nel listato seguente:

```
//Set the label
if (!label) {
        label = "polyline #"+poly_num;
}
//Add the click trigger
label = "<a href='javascript:google.maps.event.trigger(gpolys[" + poly_num
    + "][0],\"click\");'>" + label + "</a>";
//Add a line to the sidebar html variabile
//Insert the toggle trigger
side_bar_html += '<div id="line_' + id + '"><input type="checkbox" id="
   poly' + poly_num + '" checked="checked" onclick="togglePoly(' +
   poly_num + ');" style="width:10px;">' + label + '</div><br />';
```

Listing 18.32: Assegnazione dinamica di un comportamento 'onclick' agli elementi della side bar.

Realizziamo poi un metodo `togglePoly`() (presentato nel successivo listato) che si appoggerà sulle funzioni `viewclick`() e `delclick`(), le quali gestiranno la visualizzazione su mappa dei diversi marker tramite il metodo `setMap`() contenuto nelle API Google Maps. Come già detto, l'utente sarà in grado di scegliere se nascondere o meno i punti che formano la linea stessa: tale funzionalità richiederà l'aggiunta di un campo di selezione all'interno della pagina del servizio ("poiView").

```
function togglePoly(poly_num) {
        //Get the map and the polys obects
        var map = getMap();
        var gpolys = getPolylines();
        //Depending on the checkbox value...
        if (document.getElementById('poly'+poly_num)) {
        //Finds out if POIs must be hidden
                var checked = document.getElementById('poiView').value;
                //Retrieve the list of POIs for the route
                var point = gpolys[poly_num][1];
```

```
                if (document.getElementById('poly'+poly_num).checked) {
                        //Display the polyline
                        gpolys[poly_num][0].setMap(map);
                        //Display POIs marker
                        for(var j=0;j<point.length;j++){
                                viewclick(point[j]);
                        }
                } else {
                        //Hide infobox
                        mapclick();
                        gpolys[poly_num][0].setMap(null);
                        //Hide markers if requested
                        if(checked==0){
                                for(var j=0;j<point.length;j++){
                                        delclick(point[j]);
                                }
                        }
                }
        }
}
```

Listing 18.33: Funzione `togglePoly()`.

Vediamo quindi il codice delle funzioni `delclick()`, `mapclick()`, `viewclick()`, `getMap()` e `getPolylines()` utilizzate nel metodo `togglePoly()` e contenute nel file `map.js` grazie alle quali si potranno realizzare diversi tipologie di interazione all'interno del mash-up.

```
//This function hides a marker
function delclick(id) {
        var i = searchIndex(id);
        gmarkers[i].setMap(null);
}
//This function triggers the map-click event
function mapclick() {
        google.maps.event.trigger(map, "click");
}
//This function shows a marker
function viewclick(id){
        var i = searchIndex(id);
        gmarkers[i].setMap(map);
}
//This function returns the map object
function getMap(){
        return map;
}
//This function returns the polylines array
function getPolylines(){
        return gpolys;
}
```

Listing 18.34: Funzioni `delclick()`, `mapclick()`, `viewclick()`, `getPolylines()`.

18.3.4 Modifica degli itinerari

Tramite l'elemento InfoBox l'utente sarà in grado di modificare alcune semplici caratteristiche dell'itinerario; tali modifiche verranno poi apportate seguendo lo schema già utilizzato per i marker: lettura delle informazioni tramite form, invio di una richiesta di tipo POST al server (utilizzando il metodo `ajax`()) e aggiornamento della pagina. Si osservi che sia nel caso della modifica che in quello della cancellazione di un itinerario, la funzione riceverà in ingresso l'identificativo di riferimento per la tabella 'route'. L'aggiornamento della pagina verrà poi operato eseguendo una pulizia dell'overlay realizzato (`clearPoly`()) e della lista dei percorsi e successivamente richiamando la funzione `getXmlFileRoute`(), che richiederà al server le informazioni aggiornate da visualizzare nella pagina.

Vediamo di seguito il codice della funzione `it_del`() per la cancellazione: questa richiederà come parametro di input l'identificativo del percorso utilizzato per modificare la tabella 'route'. Completata l'operazione di DELETE, il metodo invocherà `mapclick`() permettendo così di chiudere un'eventuale finestra InfoBox, `del_poly`() che nasconderà l'overlay per l'elemento selezionato, `clearSideBar`() e `clearPoly`() che reimposteranno le variabili dello script ai valori iniziali ed infine `routes`() che eseguirà una chiamata a `getXmlFileRoute`() utilizzando lo username di sessione. Imposteremo inoltre, all'interno dello script che si preoccuperà di interagire col database, un controllo che impedisca di eliminare percorsi che contengono POI: ogni punto d'interesse è infatti associato ad un itinerario e se quest'ultimo venisse eliminato, senza prima modificare l'appartenenza dei punti che vi sono contenuti, impedirebbe la visualizzazione degli hotspot agli utenti Layar poiché nello strato questi vengono presentati in base all'itinerario scelto dall'utente.

```
function it_del(id){
        var answer = confirm("Vuoi eliminare l'itinerario?");
        if(answer){
                $. ajax ({
                type:"POST",
                url:"php/route_edit.php?edit=1",
                data:"id="+id,
                dataType: "json",
                success : function (json){
                        if (json.status == "fail") {
                                alert(json.message);
                        }else {
                //Clear map
                                mapclick();
                                del_poly(id);
                                clearSideBar();
                                clearPoly();
                //Refresh data
                                routes();
                        }
                }
                });
        }
}
```

Listing 18.35: Funzione `it_del`().

Nel seguito sono invece riportate le funzioni `clearPoly()`, `clearSideBar()` e `del_poly()`; quest'ultima in particolare utilizzerà la corrispondenza indice-id memorizzata in fase di creazione dell'oggetto `Polyline` a fini di gestione.

```
//Removes a polyline overlay
function del_poly(id){
        var i = 0;
        for(j=0;j<polyindex.length;j++){
                if (polyindex[j][0]==id)
                        i = polyindex[j][1];
        }
        gpolys[i][0].setMap(null);
}

//Clears side_bar_html
 function clearSideBar(){
        side_bar_html="";
}

//Resets polylines (on the map and in the array)
function clearPoly(){
        for(var i=0;i<gpolys.length;i++){
                //Clears the map
                gpolys[i][0].setMap(null);
        }
        //Clears the polylines array
        gpolys=[];
}
```

Listing 18.36: Funzioni `del_poly()`, `clearSideBar()` e `clearPoly()`.

Infine inseriremo un semplice modulo che permetterà all'utente di creare un nuovo itinerario: tale elemento, che potrà essere costituito da un breve form, andrà inserito nella pagina del servizio e dovrà essere accompagnato da uno script che invii una richiesta di POST al server (come per modifica ed eliminazione).

Un ultimo accorgimento nell'implementare le funzioni di editing è quello di controllare i campi di testo: poiché il retrieving dei dati si basa su espressioni regolari, sarà opportuno inserire dei controlli che verifichino la presenza di caratteri riservati segnalandolo all'utente.

18.3.5 Pagina web service

La pagina del servizio conterrà diversi elementi di scripting che realizzano l'interazione con la mappa, gli elementi `Polyline` ed i form. Oltre a questo la pagina dovrà contenere una procedura che verifichi l'autenticazione dell'utente come per i servizi di inserimento e modifica. A differenza dei precedenti servizi, però,si dovrà inserire un nuovo elemento HTML ("side_bar") in cui inserire i link alle funzioni dedicate ai percorsi: questo renderà il framework di lavoro semplice e modulare e soprattutto user-friendly.

18.3.6 Riepilogo

La gestione degli itinerari si basa sul servizio di visualizzazione dei contenuti, estendendolo per la manipolazione di elementi `Polyline`. Dopo aver approntato una procedura per l'XMLificazione delle informazioni di ciascun itinerario, si è realizzata una funzione per la lettura di questi e l'inserimento nell'ambiente Google Maps; per ciascun itinerario è stato quindi assegnato un comportamento interattivo tramite il quale si permette agli utenti interagire con le rappresentazioni dei dati modificandole. Infine è stata implementata una procedura di aggiornamento dei contenuti.

18.4 Gestione utenti

L'ultima sezione del sistema di gestione dei contenuti, riguarda il controllo dell'account utente: l'amministratore del sistema deve infatti avere la possibilità di estendere o revocare i privilegi ad uno o più utenti, o di rimuovere gli account registrati dal server (assieme a tutti i contenuti a questi associati).

Quest'area del CMS sarà costituita da una pagina web dove risiederà il servizio (e quindi accessibile all'utenza), e due script PHP i quali si occuperanno di reperire le informazioni riguardanti i diversi account e contattare il server per applicare le modifiche. In particolare lo script `userlist.php` effettuerà, in fase di inizializzazione, una connessione alla base di dati ed estrarrà dalla tabella 'user' alcuni valori per ogni account quali lo username, l'email di riferimento ed il tipo di privilegi associati; in seguito verranno eseguite delle interrogazioni alle tabelle 'POI_Auth' e 'route' al fine di conteggiare il numero di POI ed itinerari realizzati dall'utente: in questo modo si potrà conoscere il grado di partecipazione di uno user al sistema PadovaTour (CMS e strato). Durante la stampa dei dati (cui è conveniente organizzare in una tabella al fine di favorirne la leggibilità), ad ogni account saranno allegate delle funzioni di modifica, ovvero la concessione/revoca dei privilegi e la cancellazione dell'account. La pagina del servizio dovrà quindi invocare lo script ed inserire l'output di quest'ultimo in un elemento HTML visualizzabile dall'utente, come riportato nel seguente listato:

```
function print_users(){
        $.get("
                php/userlist.php?edit=0",
            function (data){
                        var elem = document.getElementById('list');
                        elem.innerHTML = data;
                }
        );
}
```

Listing 18.37: Funzione `print_users`().

Le procedure di editing saranno eseguite utilizzando lo schema già adoperato in precedenza: tuttavia in quest'occasione non sarà necessario eseguirà la validazione dei dati, ma semplicemente inviare l'identificativo di ogni account al server e, utilizzando questo valore, modificare l'attributo corretto. Vediamo quindi il codice d'implementazione per la funzione `set_super`() che altera i privilegi utente:

```
function set_super(id){
        var answer = confirm("Attenzione! Verranno modificati i privilegi
           utente");     if(answer){
```

```
                $. ajax ({
                type:"POST",
                url:"php/user_edit.php?edit=1",
                data:"id="+id,
                dataType: "json",
                success : function (json){
                        if(json.status=="fail"){
                                alert(json.message);
                        }else{
                    alert(json.message);
                }
                }
                });
        }
}
```

Listing 18.38: Funzione `set_super()`.

Sia in caso che la richiesta POST vada a buon fine, che nel caso sia rifiutata, l'utente verrà a conoscenza del risultato dell'operazione tramite una finestra di dialogo. La pagina del servizio dovrà quindi contenere un elemento HTML in cui posizionare il risultato della funzione `print_users()` e gli script JavaScript che operano l'interazione con il server. Inoltre, a differenza dei servizi precedenti, dovrà essere inserito un ulteriore controllo nell'header della pagina: gli utenti che accedono a questa sezione dovranno infatti, oltre ad essere autenticati, godere dei privilegi di amministratore; per realizzare tale verifica implementeremo una chiamata al metodo `IsSAuth()` della classe `authentication`.

```
<?php
include '../login/authentication.class.php';
  $auth = new UserAuthentication();
if(!$auth->IsAuth()){
        print "Devi eseguire il login per accedere a quest'area!";
        exit();
}
if(!$auth->IsSAuth()){
        print"Non hai i privilegi necessari";
        exit();
}
?>
```

Listing 18.39: Implementazione del controllo dei privilegi di superuser.

18.4.1 Riepilogo

La gestione dell'utenza permette agli amministratori dell'applicazione sia di controllare il livello di partecipazione degli editori (e quindi il numero di contenuti proposti) che di modificarne le impostazioni di account. Questo servizio è stato realizzato impostando una connessione tra il CMS ed il database contenente le informazioni d'interesse, inserendo quindi queste ultime all'interno di una pagina web dotata di semplici funzioni di controllo quali alterazione dei privilegi ed eliminazione dell'account.

Parte VII

Conclusioni

Conclusioni

Al termine di questa trattazione, possiamo affermare che gli obbiettivi inizialmente prefissi sono stati pienamente raggiunti. L'analisi del contesto in cui si colloca l'Augmented Reality ha messo in evidenza come le location-based layer application stiano assumendo un ruolo di rilievo in questo campo, sia sul piano della ricerca che per quanto concerne la diffusione di queste tecnologie verso un pubblico sempre più vasto; allo stesso modo si è messo in luce come Layar Reality Browser, un browser per contenuti d'informazione georeferenziati, colga in pieno lo stato dell'arte per questa tipologia di software, riuscendo ad integrare diverse tecnologie e forme d'informazione in un'unica applicazione. Le location-based layer application generalmente propongono all'utenza un'esperienza innovativa ed immersiva, ma sono prive di una vera e propria differenziazione dei servizi offerti: Layar Reality Browser riesce invece a superare i canoni esistenti e raggiunge un grado superiore di coinvolgimento dell'utente, esplorando nuovi ambiti di interazione, senza focalizzarsi su un singolo ambito d'interesse; difatti, intorno a quest'applicazione si è andata progressivamente formando una community di sviluppatori in grado di realizzare i propri livelli di contenuto attraverso un'interfaccia ben formata, caratterizzati da un contesto di fruizione e da servizi differenti e personalizzati, i quali forniscono un valore aggiunto all'applicazione stessa.

Una volta individuati i limiti e le possibilità del browser è stato quindi realizzato un layer d'informazione in ambito turistico usufruendo delle API per sviluppatori messe a disposizione, identificandone prima il target d'utilizzo e le caratteristiche desiderate. Il livello di contenuto sviluppato si colloca perfettamente nel settore del turismo, costituendo un utile strumento attraverso il quale i potenziali turisti della città di Padova potranno conoscere l'ubicazione di diversi punti d'interesse; ciascuno di questi siti è inoltre dotato di elementi multimediali che sussidiano l'esperienza dell'utente, meccanismi d'interazione come la sottomissione di feedback qualitativi, quali rating e messaggi, e mappe digitali. Il servizio è stato implementato tramite la costruzione di script PHP in grado di soddisfare le diverse richieste dei client in base ai parametri in esse contenute, ed infine personalizzato approntando un'interfaccia user-friendly semplice ed intuitiva.

L'applicazione è stata quindi arricchita di un sistema per la gestione dei contenuti che permette ad un'utenza ben identificata (enti del turismo, promotori di viaggi, ecc.) di inserire i propri elementi d'informazione in maniera aspecifica, ossia senza conoscere i dettagli di programmazione. Questo servizio è stato realizzato tramite un mash-up ottenuto, utilizzando le apposite API, integrando i contenuti del livello con gli strumenti Google Maps; tale prodotto è stato corredato di diverse funzionalità per il controllo dei dati (inserimento, modifica, eliminazione) ed introdotto all'interno di un portale web dinamico. Attraverso il content management system lo strato potrà quindi essere esteso ed aggiornato da parte degli editori di contenuti utilizzando una browser-based application, quindi non vincolata ad alcun sistema operativo specifico, e soprattutto senza l'intervento dello sviluppatore. Il risultato complessivo consiste di un ambiente multi piattaforma per la creazione, gestione e fruizione di dati user-generated, dotata di un meccanismo per l'interscambio di informazioni, nella quale gli ambiti di produzione sono tra loro ben definiti e separati.

Il layer realizzato nell'ambito di questo lavoro non ha certamente esaurito le proprie potenzialità, potendo indubbiamente essere ampliato tramite l'implementazione di nuove funzionalità atte a migliorare l'esperienza del visitatore: si potrebbero difatti prevedere meccanismi per il salvataggio di una cronologia di viaggio, per rivedere i luoghi visitati di recente, o un sistema di pianificazione, per organizzare la visita di un POI per volta, o ancora l'integrazione del guestbook nel livello sotto forma di "air tags" (come in Sekai Camera) o il semplice adattamento a realtà più estese non limitate all'area padovana. Anche il content management system potrà essere arricchito per ottenere funzionalità caratteristiche del Web 2.0 quali la personalizzazione dei profili utente, la condivisione dei contenuti e l'integrazione con servizi di terze parti.

Questa trattazione si colloca quindi in un contesto certamente promettente e ricco di sviluppi sia per il futuro del Web sia per quanto riguarda l'AR; l'universo dell'Augmented Reality è infatti molto vasto ed in continua espansione: basti pensare al già citato ZugStar, il cui sistema di augmented video conferencing potrà ben presto essere esteso a sistemi di messaggistica istantanea e per comunicazioni VoIP, oppure a Junaio, un altro browser per AR, che ha da poco annunciato l'integrazione con meccanismi di object recognition basati su tecnologie Kooaba (http://www.kooaba.com), pratica che a breve sarà implementata anche in Layar Reality Browser. L'era tecnologica in cui ci troviamo permette il rapido proliferare di numerosi software i cui ambiti di applicazione non sono ancora del tutto definiti ed esplorati e lasciano ampi margini di approfondimento alla ricerca; inoltre, poiché molti dei programmi presenti sulla scena sono completamente o parzialmente Open Source, gli sviluppatori possono beneficiare della "cross-pollination" realizzando così servizi sempre più ricchi e performanti.

Layar Reality Browser ha sicuramente conquistato un ruolo di rilievo tra le tecnologie moderne per location based layer application non solo grazie ad un'infrastruttura ben organizzata e funzionale e ad una comunità di sviluppatori in continua espansione: Layar ha il pregio di essere riuscito a portare l'Augmented Reality nella cosiddetta "mainstream", promuovendone la diffusione presso un vasto pubblico e non solo verso ristrette cerchie di utenti dotate di hardware e conoscenze specifiche. Risulta sempre più evidente come la Realtà Aumentata stia affermandosi quale nuovo fenomeno di massa e quanto sia importante il contributo apportato da questo browser: basti pensare alla recente notizia (16 Novembre 2010) comparsa nel blog ufficiale di Layar, nella quale è stato annunciato l'investimento di una cifra pari a14M$ (circa 10M€) da parte di Intel Capital nella più grande piattaforma di Realtà Aumentata del mondo ad oggi esistente. Nonostante ciò, Layar non rappresenta ancora un'applicazione completa e matura ed offre numerose possibilità di crescita in particolar modo legate alla gestione di un numero di POI e strati in continuo aumento o alla localizzazione in ambienti indoor: tali sfide andranno affrontate senza perdere di vista gli obbiettivi originari e nel rispetto dell'utente finale, ma soprattutto adottando un'efficace strategia di sviluppo come hanno suggerito Larry Page (fondatore di Google) ed Eric Schmidt (CEO di Google) in occasione del Google Zeitgeist 2010: *"Make your products work very well, develop a culture focused on the user and iterate quickly"*.

Il mondo tecnologico si sta muovendo in una nuova direzione e, come già avvenuto per l'introduzione della telefonia mobile (1973), del primo PC (1977) e di Internet (1991), possiamo supporre che in un futuro non molto lontano giungeremo ad un rinnovato cambiamento epocale dovuto a questa nuova "tecnologia dirompente": una svolta che modificherà le nostre abitudini quotidiane promuovendo nuovi veicoli d'informazione e nella quale Layar Reality Browser giocherà un ruolo importante.

Parte VIII

Appendice

Appendice A

Formato dei dati

A.1 Il formato JSON

JSON, acronimo di JavaScript Object Notation, è uno standard aperto basato su testo, progettato per l'interscambio di dati in formato human-readable. JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C/C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

Il formato JSON è stato originariamente indicato da Douglas Crockford ed è descritto nel RFC 4627. Il tipo di media ufficiale per JSON è `application/json` e l'estensione del file è `.json`.

JSON è basato su due strutture dati universali:

- un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo;
- un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Virtualmente, tutti i linguaggi di programmazione moderni supportano entrambe le forme, le quali in JSON assumono queste forme:

- un oggetto è una serie non ordinata di nomi/valori. Esso inizia con "{" e termina con }. Ogni nome è seguito da ":" e la coppia di nome/valore sono separata da "," ;
- un array è una raccolta ordinata di valori. Un array comincia con "[" e finisce con "]". I valori sono separati da ",";
- un valore può essere una stringa tra virgolette, o un numero, o vero o falso o nullo, o un oggetto o un array. Queste strutture possono essere annidate;
- una stringa è una raccolta di zero o più caratteri Unicode, tra virgolette; per le sequenze di escape utilizza la barra rovesciata. Un singolo carattere è rappresentato come una stringa di caratteri di lunghezza uno. Una stringa è molto simile ad una stringa C o Java;
- un numero è molto simile ad un numero C o Java, a parte il fatto che i formati ottali e esadecimali non sono utilizzati;
- i caratteri di spaziatura possono essere inseriti in mezzo a qualsiasi coppia di token.

Questo formato viene spesso utilizzato per la serializzazione e la trasmissione di dati strutturati su una connessione di rete, principalmente per trasmettere dati tra un server e una web application, in alternativa a XML. JSON sta diventando sempre più uno standard de facto come formato per lo scambio di dati nelle applicazioni AJAX.

Anche se JSON è concepito come un formato di serializzazione dei dati, il suo design pone diversi problemi di sicurezza, tra cui l'esposizione a malicious script se utilizzato in combinazione con la funzione JavaScript `eval`(), in particolar modo quando i dati sono reperiti dalla Rete, poiché la stringa potrebbe contenere istruzioni potenzialmente dannose, operando un tipico attacco di cross-site scripting. Il **cross-site scripting** (XSS) è una vulnerabilità che affligge siti web dinamici che impiegano un insufficiente controllo dell'input (parametri di richieste HTTP GET o contenuto di richieste HTTP POST). Un XSS permette ad un attaccante di inserire codice al fine di modificare il contenuto della pagina web visitata: in questo modo si potranno sottrarre dati sensibili presenti nel browser degli utenti che visiteranno successivamente quella pagina (ad esempio tramite l'invio del contenuto di cookie a terze parti). Tale problematica è in parte risolta da una libreria di gestione JSON (reperibile all'indirizzo `http://www.json.org/json.js`) che offre un metodo `parseJSON`() per il parsing sicuro di stringhe che, si presuppone, contengano solo dati nel formato d'interscambio (tale metodo sarà persino incluso tra le funzioni predefinite del linguaggio JavaScript, in una prossima versione).

JSON è meno dettagliato rispetto a XML e la conversione da XML a JSON può presentare diverse sfide, soprattutto quando la distinzione tra il valore effettivo di attributi e il testo tra tag, in quanto le assegnazioni JSON sono eseguite con i due punti [3], [21],[27].

A.2 Il formato XML

XML (sigla di eXtensible Markup Language) è un framework per la definizione di linguaggi di markup, ovvero un ambiente che definisce un meccanismo sintattico che consente di estendere o controllare il significato di linguaggi marcatori. Il nome, anche se molto fuorviante, non indica un singolo linguaggio estensibile verso altri usi, ma piuttosto una notazione che permette di creare linguaggi caratterizzati da tag ad hoc. Ogni linguaggio XML è rivolto ad un particolare dominio applicativo anche se utilizzano la stessa sintassi di base e gli stessi strumenti di elaborazione dell'informazione.

Rispetto all'HTML, l'XML ha uno scopo ben diverso: mentre il primo definisce una grammatica per la descrizione e la formattazione di pagine web e, più in generale, di ipertesti, il secondo è un metalinguaggio utilizzato per creare nuovi linguaggi, atti a descrivere documenti ed informazioni strutturate. Mentre l'HTML ha un insieme ben definito e ristretto di tag, con l'XML è invece possibile definirne di propri a seconda delle esigenze, motivo per cui XML è oggi molto utilizzato come mezzo per l'esportazione di dati tra diversi DBMS. Le tecnologie XML generalizzano il concetto di dati sul Web, pubblicando non più solo documenti ipertestuali, ma informazioni arbitrarie incluse quelle tradizionalmente riservate alle basi di dati.

Ogni documento XML è costituito da una struttura gerarchica ad albero, chiamata appunto albero XML, ed organizzata in nodi. Esistono diversi tipi di nodi: nodi radice, nodi di testo, nodi elemento, nodi attributo, nodi commento e nodi d'istruzione per l'elaborazione.

Notiamo che mentre XML è un linguaggio di markup, JSON è un formato di interscambio di dati e non realizza quindi dei documenti strutturati. Entrambi però non hanno un sistema di rappresentazione dei dati binari, per cui è compito del programmatore adottare delle convenzioni appropriate (es. Base64) per convertire i dati binari in forma testuale.

[24],[2].

Appendice B

PHP Data Objects (PDO)

L'estensione **PHP Data Objects** (PDO) definisce un'interfaccia leggera e coerente per accedere ai database tramite PHP e formalizzare le connessioni verso questi. Grazie a driver specifici per ogni tipo di database, PDO è in grado di offrire funzionalità ed interfacce standard a prescindere dal tipo del database con cui si interagisce.

PDO fornisce un data-access abstraction layer, cioè un livello di astrazione per l'accesso ai dati; si tratta infatti di una classe, che mette a disposizione un insieme di sotto-classi derivate che agiscono in modo trasparente rispetto all'utente; il suo utilizzo permetterà di creare applicazioni dotate della massima portabilità, e l'utilizzatore potrà scegliere il DBMS di riferimento sulla base delle proprie esigenze, senza particolari costrizioni relative alla tipologia di accesso ai dati dell'applicazione. Naturalmente questa caratteristica presenta qualche limite, seppur di importanza marginale, poiché non offre retro compatibilità con versioni di PHP precedenti alla 5°: si tratta infatti di un'estensione che necessita della presenza nel core del linguaggio delle funzionalità relative alla programmazione ad oggetti. Quest'estensione non mette però a disposizione una database abstraction, quindi non è in grado di riscrivere istruzioni in linguaggio SQL o di emulare funzioni che non vengano fornite dal DBMS stesso.

Il sistema PHP Data Objects è strutturato in due moduli (classi): una per interfacciarsi al database e per interrogarlo (PDO), un'altra per accedere al record set restituito dalle query (PDOStatement). Grazie a PDO è quindi possibile separare gli ambiti di interrogazione del database di elaborazione dei dati e mantenere il codice ad un alto livello di portabilità.

Appendice C

Document Object Model (DOM)

Il **Document Object Model** (DOM) è uno standard ufficiale dell'istituto W3C, per la rappresentazione dei documenti strutturati tramite un modello orientato agli oggetti. DOM è costituito da un'interfaccia neutrale sia a piattaforme che a linguaggi, che consente a programmi e script di accedere e aggiornare il contenuto, la struttura e lo stile di documenti web in modo dinamico.

Nativamente supportato dai browser, DOM è un metodo per accedere e aggiornare dinamicamente gli elementi di un documento HTML, ed una visualizzazione alternativa dei contenuti. Il modello DOM contiene sia gli oggetti specifici del browser, che i contenuti delle pagine HTML, ma a causa delle numerose incompatibilità dovute alle diversità di gestione di DOM da parte dei vari browser, il W3C ne ha stabilito delle specifiche standard. Resta però il problema che ogni browser è libero o meno di aderirvi.

DOM non pone limitazioni alla struttura dei dati di un documento e permette di visualizzare ogni documento ben formato, sotto forma di albero. Questa implementazione richiede però che l'intero contenuto di un documento venga prima analizzato e salvato in memoria. DOM è utilizzato principalmente per recuperare informazioni da documenti con una strutturazione non standard, cioè dove gli elementi devono essere trovati in modo casuale (si escludono di conseguenza documenti XML).

Come per XML, gli elementi sono organizzati in nodi rappresentati a loro volta da oggetti, la cui accessibilità è garantita dai metodi DOM, che "navigano" le relazioni padre-figlio instauratesi nella struttura: si noti che, però, DOM non è un linguaggio di programmazione, ma fornisce semplicemente nozioni sull'organizzazione dei contenuti a linguaggi come JavaScript, che altrimenti ne sarebbero sprovvisti.

DOM diventa uno strumento essenziale nella realizzazione di pagine DHTML, che associano contenuti statici a comportamenti dinamici, senza dover eseguire il refreshing del documento. [3], [25].

Appendice D

Crittografia

Nei sistemi di crittografia vengono spesso utilizzate una classe di funzioni basate su digest message. Il **digest** (o impronta) di un messaggio M è un numero h(M), calcolato tramite hashing a partire da quest'ultimo, cui sono associate le seguenti proprietà:

- ha lunghezza fissa, il che ne permette una facile trasmissione e manipolazione (tipicamente 128 bit);
- è estremamente improbabile che due messaggi diversi abbiano lo stesso digest;
- non è invertibile, ovvero non esiste un algoritmo noto che, dato un digest, sia in grado di generare un messaggio che gli corrisponde; in altri termini, è estremamente difficile produrre un messaggio che abbia un digest predeterminato.

Le funzioni crittografiche di hashing godono in genere di diversi vantaggi quali:

- sono a senso unico (l'output, come detto, non è invertibile);
- hanno una bassa resistenza alla collisione, ovvero dati M e h(M), è computazionalmente impossibile trovare un messaggio $N \neq M$ tale che $h(N) = h(M)$;
- hanno un'alta resistenza alla collisione, il che implica che data solo h(), è computazionalmente impossibile trovare due diversi input M,N tali che $h(M) = h(N)$.

Tra gli algoritmi di questa classe di funzioni possiamo identificare MD5 e SHA, [1],[5].

D.1 MD5

, **Message Digest algorithm 5**, indica un algoritmo crittografico di hashing realizzato da Ronald Rivest nel 1991 (uno degli autori di RSA) e standardizzato con la RFC 1321. Questo tipo di codifica prende in input una stringa di lunghezza arbitraria e ne produce in output un'altra a 128 bit, che può essere usata per calcolare la firma digitale dell'input. MD5 presenta purtroppo diversi problemi di sicurezza, tra cui la presenza di codificatori e decodificatori.

D.2 SHA

, **Secure Hash Algorithm**, indica una famiglia di cinque diverse funzioni crittografiche di hash sviluppate a partire dal 1993 dalla National Security Agency (NSA) e pubblicate dal NIST come standard federale dal governo degli USA. Gli algoritmi della famiglia sono denominati SHA-1, SHA-224, SHA-256, SHA-384 e SHA-512: le ultime 4 varianti sono spesso indicate genericamente come SHA-2, per distinguerle dal primo. SHA-1 produce un digest del messaggio di soli 160 bit, mentre gli altri algoritmi producono digest di lunghezza in bit pari al numero indicato nella loro sigla (SHA-256 produce un digest di 256 bit). L'SHA-1 è il più diffuso algoritmo della famiglia SHA ed è utilizzato in numerose applicazioni e protocolli, purtroppo anch'esso, come MD5, soffre di problemi di sicurezza.

L'utilizzo dei digest e delle funzioni di hashing è applicato in diversi campi della crittografia dal controllo dell'integrità dei messaggi alla verifica dell'autenticità di questi, dall'autenticazione utente tramite challenge alla firma digitale.

Verificare l'integrità di un messaggio trasferito. Le reti di telecomunicazione prevedono già dei meccanismi per assicurare l'integrità dei dati trasmessi, ma in generale si tratta di semplici algoritmi di checksum. I checksum sono adatti per rilevare gravi guasti della rete, ma non danno le stesse garanzie di un digest. Il messaggio M viene trasferito da A verso B; insieme al messaggio, A trasmette anche il suo digest D(M) detto MAC (message authentication code). Il ricevente B ripete il calcolo del digest del messaggio e verifica che questo digest coincida con il MAC. Se i due digest differiscono, qualcosa è andato storto nella comunicazione e il messaggio deve essere scartato.

Verificare l'integrità e l'autenticità di un messaggio. Un intruso può intercettare il messaggio e scambiarlo con un altro messaggio artefatto. Il controllo di autenticità previene proprio questo pericolo e coinvolge un chiave segreta K. L'utente A invia il messaggio M e il digest D(K+M) calcolato sulla chiave segreta K, la quale non viene inviata, concatenata con il messaggio M. Il ricevente B conosce la chiave segreta e ripete il calcolo del digest per verificare sia che il messaggio non sia stato corrotto. Se un intruso I si inserisce nella comunicazione e intercetta il messaggio M, può modificarlo e trasmettere il messaggio modificato N verso B. Tuttavia l'intruso non sarà in grado di calcolare il corretto digest D(K+N) poiché non conosce la chiave segreta K.

Questa versione specializzata del MAC viene detta anche HMAC (hashed MAC), ed in base all'algoritmo di hashing usato si avrà ad esempio HMAC-SHA1 o HMAC-MD5.

L'autenticazione di un utente con la tecnica del challenge. Permette l'autenticazione di un client senza che la sua password venga mai trasmessa su un canale non sicuro. Il server invia al client una stringa casuale CHL (il challenge), in seguito il client concatena questa stringa con la password PASS e ne calcola il digest D(PASS+CHL), inviando il tutto al server; il server, che è a conoscenza della password dell'utente, ripete il calcolo e lo confronta col digest ritornato dal client. In questo modo un eventuale interceptor non vedrà mai passare la password; inoltre, per impedire che l'attaccante ripeta l'autenticazione, il server dovrà modificare continuamente il challenge.

Parte IX

Bibliografia

Bibliografia

[1] Bruce Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, 1996, Wiley

[2] Anders Møller, Micheal I. Schwartzbach, *Introduzione a XML*, 2007, Pearson Addison-Wesley, versione italiana pubblicata da Paravia Bruno Mondadori Editori

[3] Ivan Venuti, *Programmare con JavaScript. Dalle basi ad Ajax*, 2008, Edizioni FAG Milano

[4] Jim Webber, Savas Parastatidis, Ian Robinson, *REST in Practice: Hypermedia and Systems Architecture*, 2010, O'Reilly Media Inc.

[5] Andrew S. Tanenbaum, Maarten Van Steen, *Sistemi distribuiti. Principi e paradigmi*, 2007, Pearson Addison-Wesley, versione italiana pubblicata da Paravia Bruno Mondadori Editori

[6] Quentin Zervaas, *Sviluppare applicazioni Web 2.0 con PHP*, 2008, Apress, versione italiana pubblicata da APOGEO

[7] Ramez A. Elmasri, Shamkant B. Navathe, *Sistemi di basi di dati, Fondamenti*, 2007, Pearson Addison-Wesley, versione italiana pubblicata da Paravia Bruno Mondadori Editori

[8] Paul Milgram, Fumio Kishino, *Taxonomy of Mixed Reality Visual Displays*, tratto da IEICE Transactions on Information Systems, Vol E77-D, No.12 Dicembre 1994

[9] Ronald T. Azuma, *A Survey of Augmented Reality*, Hughes Research Laboratories, Malibu, tratto da Presence: Teleoperators and Virtual Environments, Agosto 1997

[10] Roy T. Fielding, Richard N. Taylor, *Principled Design of the Modern Web Architecture*, Day Software, University of California, Irvine, 2002

[11] Stephan Wassink, *3D in Layar. Manual version 1.0*, 2009

[12] Tom Bestebreurtje, *Augmented Reality. A hype today, a reality tomorrow?*, Vrije Universiteit, Amsterdam, 2010

[13] Alexander Rubin, *Geo (proximity) search with MySQL*, MySQL AB, 2006

[14] 16 Top Augmented Reality Business Models, `http://www.personalizemedia.com/16-top-augmented-reality-business-models`

[15] Augmented Reality: create your own Layar-layer `http://teknograd.wordpress.com/2009/10/19/augmented-reality-create-your-own-layar-layer`

[16] Layar Reality Browser overview, `http://site.layar.com/create/platform-overview`

[17] Layar Reality Browser APIs page, `http://layar.pbworks.com`

[18] OAuth, `http://oauth.net`

[19] Wikipedia, *Augmented Reality*, Wikipedia - The free encyclopedia, `http://en.wikipedia.org/wiki/Augmented_reality`

[20] Wikipedia, *REST*, Wikipedia - The free encyclopedia, `http://en.wikipedia.org/wiki/REST`

[21] Wikipedia, *JSON*, Wikipedia - The free encyclopedia, `http://en.wikipedia.org/wiki/Json`

[22] Wikipedia, *Haversine formula*, Wikipedia - The free encyclopedia, `http://en.wikipedia.org/wiki/Haversine_formula`

[23] Wikipedia, *Content management system*, Wikipedia - L'enciclopedia libera `http://it.wikipedia.org/wiki/Content_management_system`

[24] Wikipedia, *XML*, Wikipedia - L'enciclopedia libera, `http://it.wikipedia.org/wiki/XML`

[25] Document Object Model, `http://www.w3.org/DOM/`

[26] Google Maps API, `http://code.google.com/intl/it-IT/apis/maps/documentation/javascript/`

[27] Introduzione a JSON, `http://www.json.org/json-it.html`