Alma Mater Studiorum
UNIVERSITÁ DI BOLOGNA

Facoltà di **Scienze Matematiche, Fisiche, Naturali**
Corso di **LAUREA MAGISTRALE in INFORMATICA**

GEOLOCALIZZAZIONE E QRCODE

COME STRUMENTI A SUPPORTO

DI APPLICAZIONI MOBILE TURISTICO-CULTURALI

Relatore:
Prof. Vittorio Ghini

Laureando:
Emanuele Sanapo

Correlatore:
Dott. Giuseppe Frangiamone

Anno Accademico 2010 – 2011
II SESSIONE

INDICE

INTRODUZIONE

In questi ultimi anni, in Italia, la vendita e l'utilizzo di dispositivi mobili con supporti multimediali, è considerevolmente aumentato. In particolare, una delle categorie maggiormente in voga è quella degli smartphone.

Gli smartphone, sono dispositivi per la telefonia mobile che oramai includono una serie di funzionalità a supporto della multimedialità e non solo. Divenuti quasi dei notebook in miniatura, ne esistono di vari tipi e dimensioni e anche se la loro potenza di calcolo rimane di gran lunga inferiore a quella di un PC, riescono comunque a effettuare operazioni abbastanza evolute e con prestazioni accettabili.

Quello che si vuole realizzare con questo lavoro non è uno studio su questi dispositivi (anche se in parte sarà affrontato), ma un approfondimento sulla possibilità di utilizzare una buona fetta di essi, a supporto dello sviluppo turistico nel nostro paese. In particolare si vuole realizzare una piattaforma mobile che metta in comunicazione individui che vogliono fornire informazione di tipo turistico (beni culturali, eventi, attività commerciali, ecc.) con individui che vogliono fruire di tali contenuti (un qualsiasi turista che sia in grado di utilizzare uno smartphone).

Tale applicazione dovrà raccogliere informazioni su una determinata città e riuscire in maniera chiara e intuitiva a dare risposte alle principali domande che un turista si pone:

- dove sono?
- Cosa c'è d'interessante da visitare nei dintorni?
- Interessante questa statua! Di cosa si tratta?
- Quali sono gli eventi più interessanti ai quali potrei partecipare?
- Ho un leggero appetito dove posso andare a mangiare?
- Dove posso fare un po' di shopping?

CAPITOLO 1
IL PROGETTO IN SINTESI

In questo capitolo faremo una veloce introduzione di cosa andremo a realizzare in questa tesi. In particolare verrà fatta una sintetica panoramica per ogni capitolo.

Capitolo 2

In questo capitolo si cercherà di fornire una descrizione macroscopica della problematica, cercando di capire che tipo d'informazione andremo a trattare, i tipi di dispositivi per i quali vogliamo sviluppare, la tipologia di interazione con l'utente, le politiche che verranno adottate per sviluppare l'applicazione, ed infine un breve sguardo su ciò che è già presente al momento sul mercato.

Capitolo 3

In questo capitolo verranno presentati tutti gli strumenti necessari per realizzare il progetto. In particolare verranno analizzate le mappe digitali, le principali tecniche di geolocalizzazione (gps, localizzazione cell-based) e il QR-code.

Capitolo 4

In questo capitolo analizzeremo le risorse disponibili per gestire all'interno della nostra applicazione gli strumenti trattati nel capitolo precedente.

Capitolo 5

In questo capitolo verrà invece analizzata la web application, la quale pur non essendo stata sviluppata all'interno di questa tesi, sarà comunque il fulcro della nostra applicazione. Di quest'ultima si fornirà una

descrizione generica delle sue funzionalità principali, e se ne analizzerà brevemente la sua architettura.

Capitolo 6

In questo capitolo infine, si fornirà un metodo generale per mettere insieme le risorse descritte nei due capitoli precedenti, integrandole all'interno di un'unica applicazione per un generico sistema operativo mobile.

CAPITOLO 2

DESCRIZIONE MACROSCOPICA DEL PROGETTO

2.1 Tipologia d'informazioni da trattare

Alla luce di quanto detto nell'introduzione cerchiamo di capire che tipo di informazioni andremo a manipolare e rappresentare.
Ci riferiremo in particolare a tre categorie:

- beni culturali e naturali;
- eventi;
- business.

Beni culturali e Naturali

Per beni culturali intendiamo tutti i beni designati dallo Stato come importanti per l'archeologia, la letteratura, l'arte, la scienza, la demologia, l'etnologia o l'antropologia; si contrappongono, per definizione, ai "beni naturali" in quanto questi ultimi ci sono offerti dalla natura, mentre i primi sono il prodotto della cultura dell'essere umano. Ad esempio monumenti, chiese, opere d'arte, ecc[1].

Eventi

Per eventi intendiamo qualsiasi manifestazione pubblica o privata di particolare rilevanza per una determinata città con uno scope sia locale che nazionale. Ad esempio fiere, feste, esibizioni, concerti, ecc

Business

In quest'ultima categoria si include l'insieme d'informazioni turistiche prettamente commerciali, come ristoranti, trattorie, negozi ecc

[1] Definizione da tratta da Wikipedia

Per comodità, utilizziamo la parola *oggetto* per intendere un'informazione generica, dimenticandoci cioè della sua categoria di appartenenza.

In particolare ogni oggetto è composto da due parti quella *descrittiva* e quella *cartografica*.

Per parte *descrittiva* intendiamo quella porzione d'informazione che racchiude il nome dell'oggetto, la sua descrizione testuale e i contenuti aggiuntivi come audio video e immagini.

Per parte *cartografica* invece intendiamo quella porzione di informazione necessaria per associare ad un determinato oggetto una posizione all'interno di una mappa. Nella figura sottostante viene resa l'idea con un esempio.

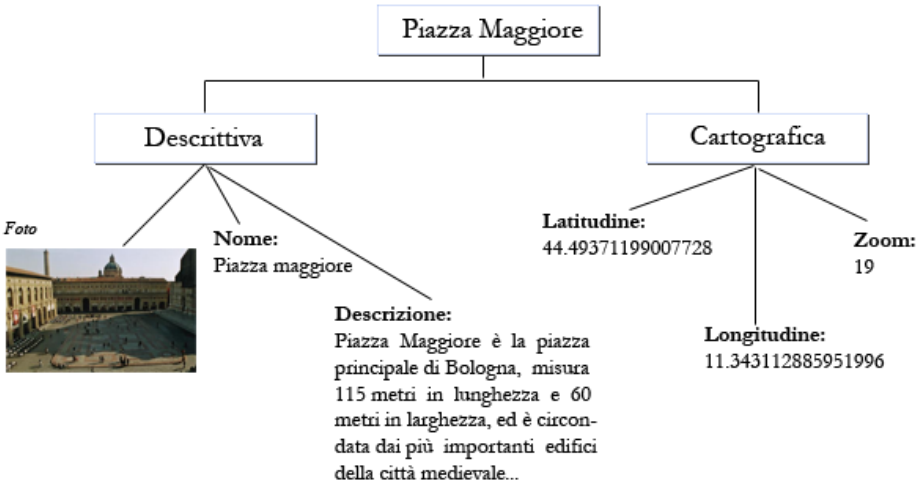

Figura 2.1: Scomposizione di un oggetto nelle due sottoparti: Descrittiva e Cartografica

2.2 Tipi di device

Come accennato nell'introduzione, l'idea di base è quella di fornire un'applicazione per dispositivi mobile (in particolar modo smartphone) che aiuti il turista a muoversi in una città a lui sconosciuta.

In questi ultimi anni il mondo degli smartphone ha subito una crescita esponenziale sia nelle vendite che nella varietà di modelli messi in commercio. Quest'ultimo aspetto è un punto cruciale se si vuole distribuire la propria applicazione sul maggior numero di dispositivi in commercio.

Allo stato attuale esistono molti dispositivi con caratteristiche hardware e software differenti. Il sistema operativo di uno smartphone e il modo in cui tale sistema si interfaccia con l'utente sono aspetti importanti dei quali non si può non tener conto.
Alla luce di ciò cerchiamo di fare un'analisi delle caratteristiche minime che un dispositivo deve avere per poter utilizzare la nostra applicazione.

Dimensioni dello schermo

Le dimensioni dello schermo giocano un ruolo importante, soprattutto in device come quelle mobili che dispongono solitamente di schermi ridotti. Applicazioni come quella che andremo ad implementare, richiedo la visualizzazione di mappe, immagini, descrizioni abbastanza verbose, che se riprodotte su schermi con dimensioni non appropriate, porterebbero l'utente ad avere un'esperienza negativa e soprattutto snervante, che scoraggerebbe l'utilizzo dell'applicazione. Perciò si presuppone che un dispositivo adeguato a questo compito dovrebbe avere uno schermo con dimensioni minime di 3 pollici.

Accesso a internet

L'accesso a internet, è un requisito fondamentale per la realizzazione di applicazioni come quella che andremo a realizzare. Esse infatti necessitano di un collegamento con un server esterno dal quale ottenere sempre informazioni aggiornate.

Capacità di riprodurre audio video e immagini

La capacità di riprodurre audio, video e immagini, è presente su ogni smartphone, il problema in questo caso diventa più il formato delle singole componenti. Quest'ultimo aspetto sarà trattato nei prossimi capitoli.

Fotocamera

La fotocamera, anche se non sarà utilizzata nel modo classico (questo punto verrà approfondito più avanti) rappresenterà una componente necessaria per l'utilizzo di alcune funzionalità della nostra applicazione. Essa ci permetterà con un semplice scatto di ricevere informazioni relative all'oggetto d'interesse. Anche se non sono richieste prestazioni eccellenti, sarebbe opportuno l'utilizzo di una fotocamera non inferiore ai 2 megapixel per rendere più agevole la nostra esperienza con l'applicazione.

Touch screen

La possibilità di un touch screen non è un requisito fondamentale, anche se sarebbe preferibile per la navigazione all'interno di un'applicazione fortemente basata su cartografia.

GPS

La possibilità di usufruire del GPS è auspicabile ma non necessaria, infatti, nel caso in cui tale opzione fosse disponibile, si avrebbe una precisione maggiore circa il rilevamento della posizione dell'utente, in caso contrario si opterebbe per la localizzazione tramite rete che fornirebbe nella maggior parte dei nostri casi, un' approssimazione accettabile. È ovvio che almeno una delle due funzionalità debba essere presente.

2.3 Politiche adottate

Prima di addentrarci nella progettazione di un applicazione, bisogna stabilire quelle che saranno le linee guida da tener presente durante tutto il ciclo di produzione del software. Volendo distribuire la nostra applicazione su più dispositivi l'aspetto principale che si è tenuto in considerazione, è stato quello di rendere il codice della nostra applicazione il più portatile possibile. La scelta di dover riscrivere l'intero codice per ogni sistema operativo sul quale intendiamo distribuire la nostra applicazione, non è la migliore per un'azienda che vuole ottimizzare sia i costi che i tempi.

Pensiamo a quanti sistemi operativi per mobile esistono:
iOS, Android, Symbian, Windows Mobile, Windows Phone 7, BlackBerry OS, Bada e questi sono solo i più conosciuti.
Per ovviare a questo problema la prima soluzione che potrebbe essere presa in considerazione, sarebbe quella di creare una web-application, pensata per essere eseguita su un browser mobile. Una soluzione del genere risolverebbe pienamente la nostra problematica, tutt'al più richiederebbe soltanto alcuni adattamenti per rendere il tutto compatibile sui vari browser, ma niente di paragonabile all'eventualità di riscrivere l'applicazione da capo. Purtroppo nel nostro caso questa soluzione avrebbe un piccolo inconveniente, infatti una web application non ha il controllo sull'hardware di uno smarphone, perciò utilizzando questa soluzione si dovrebbe rinunciare all'utilizzo di fotocamera, gps, e tutta una serie di utility per noi indispensabili.

Una seconda soluzione sarebbe quella di utilizzare alcuni framework come *PhoneGap*[1] o *Rhomobile*[2] i quali permettono di scrivere una web applicationche con una semplice importazione di file javascript riuscirebbe a controllare l'hardware dello smartphone e inoltre convertirebbe la web application in una app nativa per i principali sistemi operativi mobili.

Quest'ultima perciò sarebbe la soluzione ideale. Risolveremmo i problemi di riscrittura del codice in quanto potremmo convertire la nostra applicazione in iOS, Android, BlackBerry, Web OS, Symbian, certo non sono tutti ma sarebbe un'ottima soluzione.
Dov'è allora il problema? A quanto pare Apple ha intenzione a breve di non accettare più applicazioni sviluppate con tali framework.
Perciò o si decide di non implementare per iPhone o si decide di riscrivere il codice solo per iOS, oppure si deve trovare un'altra soluzione.

[1] Phonegap sito ufficiale http://www.phonegap.com
[2] Rhomobile sito ufficiale http://rhomobile.com

smartphone Platform	Share(%) of EU5 smartphone Users		
	Jul-10	Jul-11	Point Change
Total smartphone Users	100%	100%	0.0
Symbian	53.9%	37.8%	-16.1
Google	6.0%	22.3%	16.3
Apple	19.0%	20.3%	1.3
RIM	8.0%	9.4%	1.5
Microsoft	11.5%	6.7%	-4.8

Tabella 2.1: Variazione in un anno del market share delle principali piattaforme per smartphone nella zona Euro5. Fonte: ***comSource***[1]

Vista la stima delle vendite riassunte nella tabella 2.1, la decisione di escludere lo sviluppo della nostra applicazione per iOS non è di sicuro la migliore, dato che la piattaforma di Apple dagli ultimi sondaggi occupa il venti percento circa del mercato degli smartphone dei quali tutti supportano le caratteristiche base del nostro sistema.

Quindi l'unica soluzione sembrerebbe quella di scrivere un'applicazione per iOS e una per tutti gli altri. Una scelta di questo tipo però ha i suoi pro e suoi contro. È vero che in questo modo si passa da n versioni a circa due, ma comunque mantenere aggiornate le due applicazioni potrebbe essere fastidioso, soprattutto nei primi periodi nei quali le modifiche sono all'ordine del giorno. Ma allora cosa fare?

Esaminando più a fondo la faccenda si è cercato di optare per una soluzione ibrida tra creare una web application e crearne n native.

[1] comScore è una società di ricerca marketing via internet.

Il problema della soluzione web era quella dell'impossibilità di controllare l'hardware del dispositivo. Analizzando nello specifico la nostra problematica però si è notato che l'unico componente hardware che non si riuscirebbe realmente ad utilizzare sarebbe solo la fotocamera la cui gestione, richiederebbe una piccola porzione di codice rispetto a quello totale.

Alla luce di quest'ultima analisi perciò si è scelto una soluzione ibrida che prevede la realizzazione di una web application con tutte le funzionalità eccetto la gestione della fotocamera, la quale verrà eseguita all'interno di un'applicazione nativa che la gestirà.

Per quanto riguarda questa tesi si è deciso di analizzare la web application (creata da NSI Nier soluzioni informatiche) e di integrarla con un applicazione nativa (app) per la gestione della fotocamera. In particolare a si forniranno due versioni di esempio di tale app: quella per Android e quella per iPhone.

Si sono scelti per il codice d'esempio queste due piattaforme, in quanto dai risultati della *tabella 1* risultano i due sistemi più informa considerando diffusione e tendenza alla crescita di quest'ultima. Inoltre la maggior parte degli smartphone che rispecchiano la configurazione ideale descritta nel paragrafo 2.2, utilizzano principalmente queste due piattaforme.

2.4 Tipologia di interazione con l'utente

Come detto nella parte introduttiva con questa applicazione intendiamo rivolgerci ad un'utenza in grado di utilizzare uno smartphone, in prevalenza turisti che sono interessati a scoprire le bellezze di una città a loro sconosciuta, ma anche a persone del posto che vogliono utilizzare uno strumento pratico per scoprire e soprattutto vivere tramite eventi e manifestazioni la propria città.

Possiamo dividere l'utenza tipo del nostro sistema in due categorie che chiameremo *Producer* e *Consumer.*

Con *producer* intendiamo qualunque individuo o gruppi di essi interessati a produrre contenuti multimediali di tipo turistico.

Con *consumer* invece si intende qualsiasi individuo interessato a consultare i contenuti forniti dai producer.

Per quanto concerne l'utente di tipo producer, si è pensato a un'unica modalità d'interazione, formata da una semplice maschera nella quale inserire per un oggetto informazioni cartografiche, contatti, descrizioni sia sintetiche che dettagliate, immagini e contenui audio e video. Ovviamente l'accesso a tali funzionalità sarà possibile solo previa registrazione.

Per l'utenza di tipo consumer sono state pensate due principali modalità d'interazione con la piattaforma.

Nella prima l'utente potrà visualizzare in base alla propria posizione e ad appositi filtri configurabili (come raggio d'azione, categorie d'informazione) i vari punti d'interesse attorno a lui, e una volta scelta la propria destinazione, potrà visualizzare tutti i contenuti multimediali relativi ad essa, commenti di altri utenti, altri punti d'interesse nei dintorni e relative informazioni cartografiche.

Il secondo tipo di interazione invece prevede l'utilizzo di una fotocamera per identificare direttamente un oggetto. Immaginiamo che un utente stia passeggiando liberamente senza una meta precisa, ad un certo punto nota una fontana sulla quale vuole avere informazioni dettagliate, allora apre l'applicazione usa la funzione camera e scatta una foto ad un codice posizionato su una targa nei pressi della fontana. L'applicazione a questo punto recupera i contenuti multimediali e li presenta all'utente.

Quest'ultima modalità d'interazione, potrebbe essere molto utile anche all'interno di un museo dove il gps non sarebbe di nessun aiuto, dove

orientarsi non è l'esigenza principale (di solito all'interno di un museo il percorso è già tracciato e non c'è la possibilità di perdersi) ma lo è invece avere la possibilità di avere informazioni su ogni singolo oggetto.

Per utilizzare tali interazioni l'utente non dovrà necessariamente essere registrato.

2.5 Progetti simili

Nel momento in cui si scrive esiste un discreto numero di applicazioni mobile che cercano di fornire dell'assistenza turistica adeguata.

Di tali applicazioni una grossa fetta si limita a fornire un servizio simile a quello offerto dalle comuni guide cartacee, semplicemente rendendo più comoda e gradevole la consultazione dei contenuti informativi permettendo solo in alcuni casi l'utilizzo del gps per una presentazione più adeguata. Una parte decisamente inferiore invece prevede anche ulteriori funzionalità aggiuntive come la realtà aumentata[1] o l'utilizzo di codici come il Qr-Code (cos'è e come funziona il QR Code verrà chiarito nel capitolo successivo) per identificare univocamente un oggetto.

Losna[2] per esempio ha realizzato una serie di guide turistiche multimediali gratuite per Android riguardanti alcune delle principali città italiane. Tali guide però non fanno uso di nessun tipo di localizzazione.

Hydrusa.com[1] ha realizzato in collaborazione con il comune di Otranto una guida turistica che utilizza anche il supporto di codici a barre bidimensionali (QR-Code) per fornire informazioni ai turisti.

---

[1] La realtà aumentata (augmented reality, abbreviato AR) è la sovrapposizione di livelli informativi (elementi virtuali e multimediali, dati geolocalizzati, ecc.) all'esperienza reale di tutti i giorni. Gli elementi che "aumentano" la realtà possono essere aggiunti attraverso un dispositivo mobile, come un telefonino di ultima generazione, con l'uso di un PC dotato di webcam, con dispositivi di visione (p. es. occhiali VR), di ascolto (auricolari) e di manipolazione (guanti VR) che aggiungono informazioni multimediali alla realtà già percepita "in sé"

[2] Losna è una web agency, che opera in vari ambiti tra cui la realizzazione di applicazioni informatiche a supporto dei beni culturali. Sito web ufficiale: http://www.losnaweb.com/

Wigoo[2] invece è un'applicazione più completa la quale utilizza sia la localizzazione attraverso la rete, che attraverso il gps, inoltre, fornisce anche la possibilità di utilizzare la realtà aumentata ma non fa uso di Qr Code.

Un ultimo progetto che si cita solo per completezza ma che in realtà è poco attinente con la tipologia di applicazione che andremo a realizzare è Whai WhaiWhai[3]. Quest'ultimo progetto non è una guida turistica vera è propria ma un modo innovativo ed originale per conoscere una città partecipando a un gioco che permette all'utente mediante l'invio di semplici messaggi ad un sistema centralizzato, di vivere un'avventura fantastica attraverso le ricchezze artistico culturali della città.

Il fattore comune di queste guide però è la presenza di un'entità centrale che distribuisce l'informazione e di un variabile numero di utenti che consultano tali informazioni.

La novità introdotta da questo progetto invece è quella di voler mettere in comunicazione "produttori" di informazione (quelli che precedentemente abbiamo chiamato producers) e "consumatori" di informazione (consumers). Qualsiasi ente, organizzazione culturale, museo, commerciante, ecc potrà fornire i propri contenuti informativi e contribuire a distribuire informazione nella propria città. Inoltre tutte le funzionalità appena descritte nelle precedenti applicazioni (tralasciando per questa prima implementazione la realtà aumentata) verranno integrate in un unico ambiente.

---

[1] Hydrusa.com è una piccola casa di produzione che si occupa di editoria, editoria multimediale e progetti di comunicazione integrata.

[2] WIGO è un'applicazione mobile che mette a disposizione contenuti di interesse turistico, educational o aziendale a diverse tipologie di utenti tramite smartphone o tablet
Sito Web ufficiale: http://www.wigoapp.com/

[3] Sito ufficiale: http://www.whaiwhai.com/

CAPITOLO 3

GLI STRUMENTI NECESSARI

3.1 Panoramica

Dopo aver descritto brevemente il progetto e i suoi principali obbiettivi, passiamo ora ad esaminare a livello macroscopico quelli che saranno gli strumenti principali che verranno utilizzati nel progetto.

Per creare un sistema adeguato a risolvere le problematiche introdotte nel capitolo precedente, abbiamo bisogno di tre tipi di strumenti fondamentali: un supporto per la gestione delle mappe, un supporto per la localizzazione della posizione dell'utente su di una mappa e infine la possibilità di identificare in maniera univoca un singolo oggetto.

3.2 Le mappe

Gli strumenti principali per rappresentare la parte di informazione cartografica sono ovviamente le mappe.

Ma cos'è una mappa digitale?

In un contesto di cartografia informatica vera e propria una mappa digitale può essere vista come una sovrapposizione potenzialmente infinita di livelli (layer). Utilizziamo un esempio molto elementare per capire meglio la logica che sta dietro ad una mappa (la realtà è ben più complessa).

Figura 3.1: Mappa finale Bologna

Figura 3.2: Livello stradale

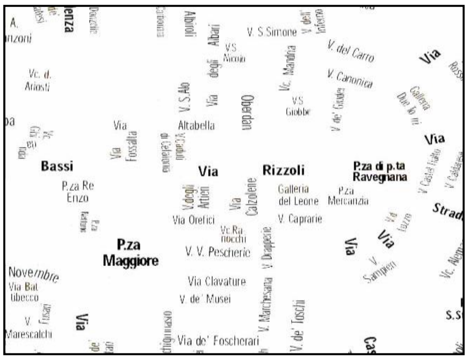

Figura 3.3: Livello nomi delle strade

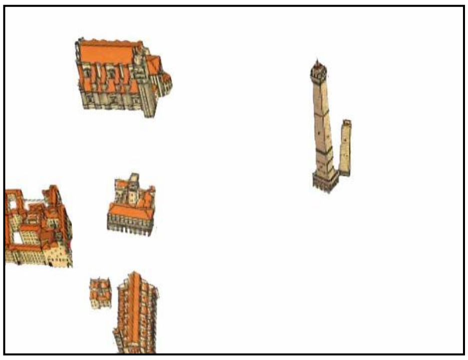

Figura 3.4: Livello edifici

Possiamo pensare alla mappa rappresentata nella figura 3.1 non come un'unica entità, ma come risultato della sovrapposizione di più livelli (figure 3.2, 3.3, 3.4) . Questa stratificazione rende l'oggetto risultante aggiornabile, estendibile e utilizzabile per risolvere diverse problematiche.

Immaginiamo per esempio venga demolito un edificio. Con questa struttura basterà andare a eliminare lo stesso dal livello degli edifici, senza andare a modificare l'intera mappa, inoltre, se si è interessati solo ad un certo set d'informazioni si può evitare di riportare quelle superflue avendone così un guadagno in termini di leggibilità della mappa stessa.

Utilizzando questa tecnica di base, vengono creati sistemi cartografici che riescono a risolvere problemi complessi. Per il nostro progetto non abbiamo la necessità di avere a disposizioni sistemi con questo livello di dettaglio che tra l'altro necessiterebbero di spese non indifferenti, potremmo invece utilizzare un sistema meno complesso con un numero limitato di livelli che ci permetta di segnalare dei punti o regioni su di una mappa.

Un sistema come Google maps soddisfa a pieno le nostre principali esigenze, esso inoltre, offre la possibilità di essere incorporato e utilizzato nelle proprie applicazioni web in maniera del tutto gratuita tramite delle semplici api. Inoltre, tale piattaforma e abbastanza diffusa e conosciuta,

tanto che la sua interfaccia è utilizzata già dalla quasi totalità dell'utenza di applicazioni come la nostra.

Tutte le informazioni su come includere le mappe di Google all'interno della propria web application si possono trovare al sito http://code.google.com/intl/it-IT/apis/maps/index.html.

3.3 La geolocalizzazione

La geolocalizzazione è l'identificazione della posizione geografica nel mondo reale di un dato oggetto, come ad esempio un telefono cellulare o un computer connesso a Internet.
Esistono vari modi per conoscere la posizione di un oggetto sul globo terrestre tra i principali ci sono:

- GPS;
- localizzazione Tramite le reti WiFi o WLAN circostanti;
- localizzazione tramite le celle della rete telefonica.

3.3.1 Il GPS

Il GPS è un sistema di posizionamento terrestre particolarmente preciso creato dal Ministero della Difesa Americano per fini militari ed in seguito utilizzato anche per scopi civili.

Il suo funzionamento è legato a 27 satelliti orbitanti di cui 24 effettivamente operativi e tre di riserva. Le orbite sono circolari su 6 piani orbitali paralleli inclinati di 55° rispetto al piano equatoriale.
Ogni satellite si trova a circa 20 Km dalla terra e compie due rotazioni del pianeta al giorno (il periodo di rivoluzione è di 11 ore e 58 minuti). Le orbite dei satelliti sono state studiate in modo che in qualsiasi momento, ogni punto della terra venga visto da almeno 4 di essi contemporaneamente.

Oltre ai satelliti, ci sono anche 4 stazioni di controllo a terra che si occupano costantemente di verificare il loro stato, di correggere i loro orologi atomici e la loro posizione orbitale. Senza queste stazioni terrestri il sistema non sarebbe in grado di funzionare.

Il resto del lavoro viene fatto dal GPS receiver che esegue le seguenti operazioni:

- localizza 4 o più satelliti;
- calcola la distanza da ognuno dei satelliti;
- usa i dati ricevuti per calcolare la propria posizione mediante il processo di trilaterazione;

La trilaterazione è il metodo usato per il calcolo effettivo della posizione. Vediamo come funziona nell'esempio seguente basato su un spazio bidimensionale per facilitarne la comprensione. Quello effettivo ovviamente lavora sullo spazio tridimensionale ed usa lo stesso concetto. Supponiamo di esserci persi e di voler capire qual è la nostra posizione. Chiediamo aiuto ad un passante che ci dice "Ti trovi esattamente a 215 Km da Napoli". Possiamo rappresentare questa informazione come in figura.

Figura 3.5: Circonferenza rappresentante i possibili punti sui quali posso trovarmi

Se Napoli è al centro, significa che possiamo essere su un qualsiasi punto della circonferenza visto che ogni punto si trova proprio a 215 Km.

Supponiamo di incontrare un altro passante che ci fornisce un altra indicazione: "Ti trovi esattamente a 271 Km da Firenze". Rappresentando graficamente anche questa informazione avremo la seguente situazione:

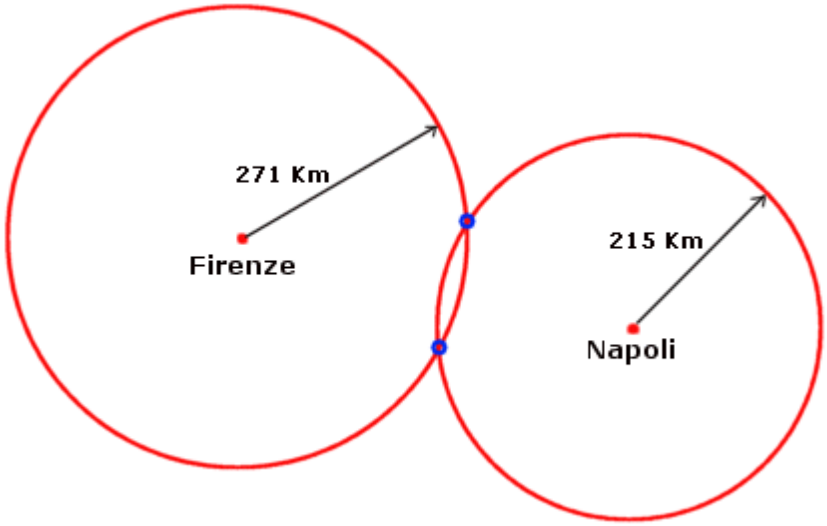

Figura 3.6: Gli unici due punti possibili sui quali posso trovarmi, risultanti dall'intersezione delle due circonferenze

Considerando le due informazioni, possiamo essere sicuri di essere in un punto che dista 271 Km da Firenze e 215 da Napoli. Come si vede nella figura, solo 2 punti (quelli cerchiati in blu) rispondono a queste caratteristiche.

Per capire in quale dei due punti effettivamente mi trovo ho bisogno quindi di una terza informazione. Incontro un terzo passante che mi dice "Ti trovi esattamente a 80 Km da Roma" e posso a questo punto capire senza il minimo dubbio dove mi trovo, ossia nell'unico punto al mondo che dista 271 Km da Firenze, 215 da Napoli e 80 da Roma (figura 3.7).

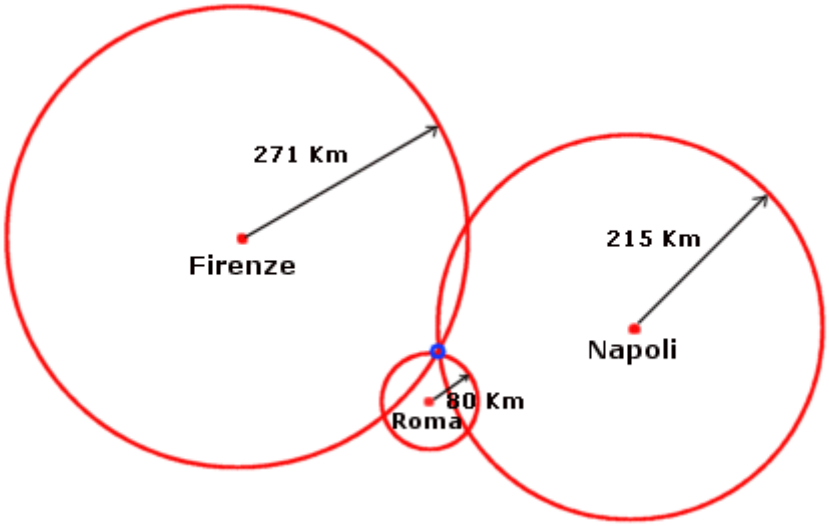

Figura 3.7: Il punto esatto in cui mi trovo, ottenuto dall'intersezione delle tre informazioni

Nella realtà, come abbiamo già detto in precedenza, il calcolo viene fatto nello spazio tridimensionale e usando 4 misurazioni per cui invece dei cerchi dobbiamo immaginare delle sfere che si intersecano tra loro fino ad identificare un unico punto.

3.3.2 Localizzazione cell-based

La posizione di un terminale GSM può essere stabilita ricorrendo alle informazioni che sono già disponibili nella rete GSM. Si parla in questo caso di localizzazione cell-based, cioè basata sulla cella che "ospita" istante per istante un certo dispositivo mobile. La precisione con cui la posizione può essere rilevata, dipende dalla pianificazione delle celle nella rete e dallo stato di accensione o meno del dispositivo stesso.
Sono possibili quattro tipi diversi di localizzazione:

Cella omnidirezionale

Se la cella é di tipo omnidirezionale, é possibile conoscere la posizione di un terminale con precisione massima pari al raggio della cella stessa. Il dispositivo é quindi all'interno della zona di copertura della cella. Si ricorda che una cella omnidirezionale é creata da antenne che irradiano in tutte le direzioni indistintamente.

Cella omnidirezionale e timing advance

Per aumentare la precisione di localizzazione si può ricorrere al timing advance. La rete, stimolando il dispositivo mobile ad una connessione, stima la distanza del dispositivo da essa, calcolando i tempi di ritardo nella trasmissione. La zona di localizzazione si riduce da un cerchio a una corona circolare. Il dispositivo é quindi all'interno della corona circolare.

Cella settore

Se il sito irradia più settori, é possibile localizzare il terminale con la precisione massima del settore circolare coperto dalla cella medesima. Il dispositivo mobile é quindi all'interno del settore circolare.

Cella settore e timing advance

Per aumentare la precisione di localizzazione si può ricorrere, anche in questo caso, al timing advance. La zona di localizzazione si riduce da un settore circolare, ad un settore di corona circolare. Il dispositivo é quindi all'interno del settore di corona circolare.

Triangolazione

Per aumentare la precisione della misura é possibile fare ricorso a più celle e quindi, stimare la posizione del mobile triangolando i dati relativi alle distanze da ciascuna BTS. L'operazione risulta più complessa ma consente di raggiungere risultati molto più precisi.

3.4 Il QR-code

3.4.1 Un po' di storia

Il QR-code non è altro che l'evoluzione del codice a barre.
Il codice a barre nasce negli anni '50 grazie a un progetto di ricerca intrapreso da due studenti della facoltà di Ingegneria dell'Università di Drexel, in Pennsylvania. L'idea fu sviluppata a partire dall'esigenza del presidente di un'azienda del settore alimentare, di automatizzare le operazioni di cassa.

Così, gli studenti Joseph Woodlan e Bernard Silver pensarono inizialmente di estendere in verticale il Codice Morse, visualizzandolo come una serie di barre strette e barre larghe. Partendo da questa intuizione, arrivarono alla realizzazione del primo codice a barre dalla forma ovale, simile ad un bersaglio.

L'idea di base era quella che sarà poi usata per i codici a barre a una sola dimensione che conosciamo oggi. Si partiva da un particolare alfabeto che assegnava ad ogni parola una stringa di bit, le stringhe venivano poi rappresentate assegnando un simbolo a ciascun bit (1 = linea

bianca 0 = linea nera). Codificando poi, ogni parola dell'alfabeto con lo stesso numero di bit, il decodificatore poteva facilmente stabilirne inizio e fine, conoscendo a priori la lunghezza di ogni parola. L'eccesivo rumore prodotto dal dispositivo di lettura e le imperfezioni dovute alla stampa dei codici ovali impedirono il diffondersi di questa nuova tecnologia.

Grazie alla nascita della tecnologia laser e allo sviluppo del circuito integrato, nel 1973, presso IBM, Woodlan sviluppò gli Universal Product Code (UPC), meglio conosciuti come Bar-code o codici a barre lineari tecnologia ancora oggi largamente utilizzata nell'industria, soprattutto nella grande distribuzione commerciale.

I codici UPC possono codificare 12 cifre decimali all'interno di sequenze che rispettano il seguente schema:

SLLLLLLMRRRRRRE

dove S rappresenta il simbolo di inizio sequenza ed E il simbolo di fine, e vengono rappresentati con la terna di bit 101, M è il simbolo che indica la metà della sequenza ed è rappresentato dalla sequenza 01010, L e R sono i dati rispettivamente a sinistra e a destra di M e ognuno è rappresentato con una stringa da 7 bit, per un totale di 95 bit.

Il processo di decodifica di un codice a barre è descritto nella figura sottostante:

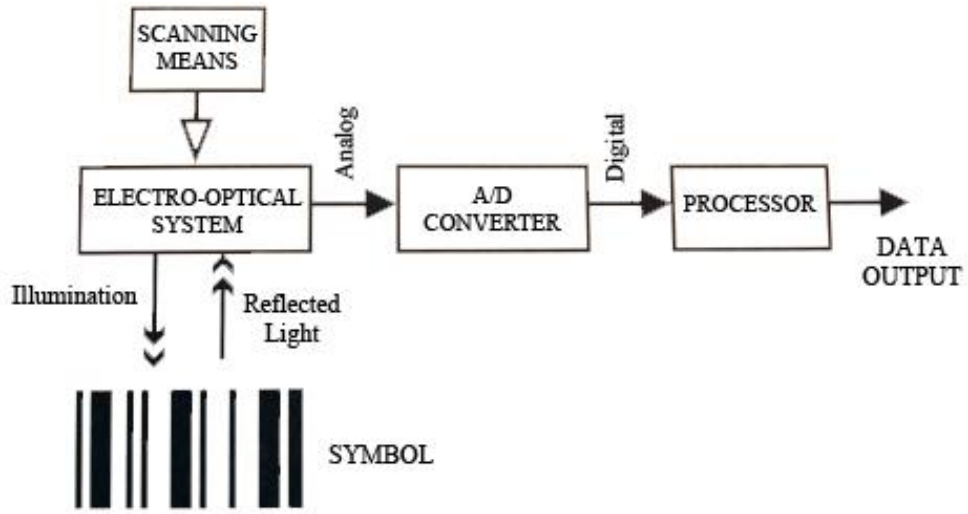

Figura 3.8: Fasi della decodifica di un codice a barre

La decodifica si basa sulla lettura della riflettenza, ossia il rapporto tra l'intensità dell'onda luminosa incidente sul codice e quella dell'onda riflessa da esso. Istante per istante, la luce emessa (generalmente un fascio laser) viene riflessa dal simbolo e direzionata verso un fotodiodo, che genera una piccolissima corrente proporzionale all'intensità dell'onda ricevuta in quell'istante.

Come si può vedere in figura, successivamente il segnale elettrico viene inviato in input a un amplificatore che lo "amplifica" e lo converte in un segnale in tensione. Il segnale analogico risultante passa in un convertitore e viene trasformato in digitale il quale verrà poi passato in input ad un microprocessore che lo codificherà utilizzando ad esempio un algoritmo di thresholding ritornando infine il risultato.

Date le potenzialità di questa tecnologia, il mercato ha iniziato a richiedere codici in grado di memorizzare una maggior quantità di informazioni, per aumentarne l'efficienza ed espanderne le possibili applicazioni. Tuttavia, adattando i codici a queste nuove richieste, sono sorti numerosi problemi, come ad esempio l'ingrandimento delle dimensioni del codice che provoca un aumento dei costi di stampa e complica il processo di lettura e decodifica.

Per far fronte a queste nuove richieste, la ricerca si è spostata sullo sviluppo di codici bidimensionali in grado cioè di contenere maggiore informazione e di ridurre al minimo costi di stampa e dimensioni.
Tra questi, spiccano per notorietà e applicabilità i Qr-code dove QR è l'abbreviazione dell'inglese *quick response* (risposta rapida), in virtù del fatto che il codice fu sviluppato per permettere una rapida decodifica del suo contenuto

I QR-code sono stati creati nel 1994 dalla Denso, un'azienda giapponese leader nella progettazione di apparati per l'automazione industriale. Inizialmente l'industria nipponica utilizzava questi codici per tracciare dettagliatamente i componenti dei propri prodotti, in seguito però, con il diffondersi in Giappone di telefoni cellulari dotati di fotocamera, il Qr-code cominciò ad entrare anche nel mercato consumer, in particolare,

apparve su alcuni cartelloni pubblicitari di una delle principali compagnie telefoniche giapponesi: la NTT Docomo. Quest'ultima permise, così ai propri utenti di ricevere maggiori informazioni sul prodotto pubblicizzato, semplicemente inquadrando il codice con la telecamera del proprio smartphone.

Con il passare del tempo, usare lo scatto di una foto per veicolare un utente ad un determinato url, è divenuto uno dei maggiori impieghi di questa tecnologia. L'utente fotografa il codice con la telecamera del proprio smartphone, il quale decodifica il codice tramite apposito software e apre automaticamente il browser all'indirizzo codificato. In questo modo si evita il fastidioso compito di inserire dati da tastiera o da touch-screen, e si garantisce un link rapido a informazioni aggiuntive e multimediali, il tutto realizzato senza la necessità di un dispositivo creato appositamente, ma tramite un semplice smartphone.

Figura 3.9: Esempio di Qr-code: codifica l'indirizzo web www.unibo.it

3.4.2 Caratteristiche e struttura

Il Qr–code è un codice bidimensionale: al contrario del classico codice a barre , l'informazione è distribuita sia in verticale che in orizzontale. Esistono attualmente 40 versioni di Qr–code, ognuna con capacità e dimensioni diverse: si passa dalla versione 1 composta da 21x21 moduli, alla versione 40 composta da 177x177 moduli (vedi 3.10)

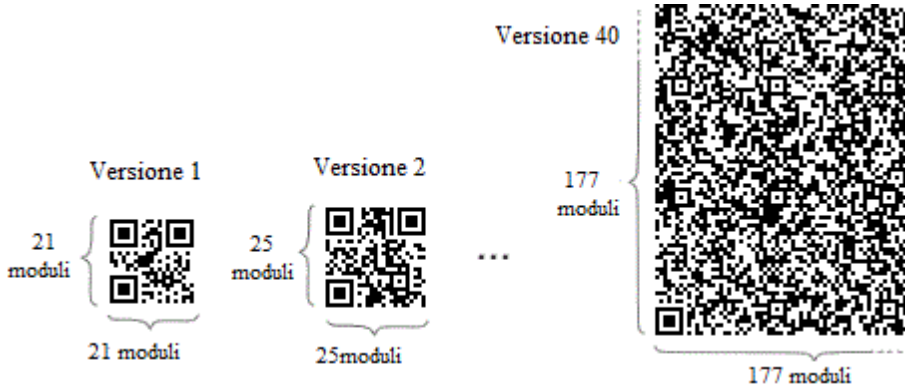

Figura 3.11: Alcune versioni di QR-code

La dimensione cresce di 4 moduli per lato in ciascuna versione. Nella sua dimensione massima un codice può contenere fino a 7089 caratteri numerici e 4296 caratteri alfanumerici, quantità di gran lunga superiori alle 12 cifre decimali codificabili con un codice UPC. Questo inoltre permette di ridurre di un decimo l'area di stampa rispetto ai vecchi standard.

La capacità di correzione d'errore di un Qr–code è molto elevata ed è classificabile in 4 livelli a seconda di quanto vogliamo proteggere il codice:

- Livello L: il 7% delle parole affette da errore può essere ripristinato;
- Livello M: il 15% può essere ripristinato;
- Livello Q: il 25% può essere ripristinato;
- Livello H: il 30% può essere ripristinato.

Queste prestazioni sono raggiungibili grazie all'utilizzo, da parte dell'apparato decodificatore, di codici Reed-Solomon per la rilevazione e la correzione dell'errore.

L'idea chiave che sta dietro al codice Reed-Solomon è quella di vedere i dati da proteggere come un polinomio. L'algebra lineare dice che un numero k di punti distinti determinano univocamente un polinomio di grado al massimo k-1. Il polinomio è quindi "codificato" tramite il calcolo di diversi suoi punti e i valori di questi punti sono ciò che viene effettivamente trasmesso. Durante la trasmissione, alcuni di questi punti possono essere corrotti. Per questo motivo sono trasmessi k+n punti, con n che sono i dati aggiunti per la protezione dell'informazione.

Il ricevitore analizza i punti ricevuti e determina il polinomio di grado k-1 che meglio approssima la sequenza di punti ricevuti. Finché "non troppi" punti sono ricevuti corrotti, il ricevitore può ricostruire quello che era il polinomio originario e quindi decodificare i dati. Nel caso di un numero eccessivo di errori comunque il decodificatore ricostruisce un polinomio "simile" a quello di partenza. Quindi l'algoritmo mostra una dipendenza debole dal numero di errori, non vi è perciò un numero di errori oltre il quale la ricostruzione fallisce, ma l'algoritmo ricostruisce sempre un polinomio verosimile, la verosimiglianza del polinomio ricostruito dipende dal numero di errori ricevuti. Alla luce di quanto detto perciò si deduce che si può aumentare o diminuire il livello di protezione del codice inserendo ridondanza nei dati nella misura data dal parametro n.

Utilizzando questo metodo per la correzione, un Qr–code può essere letto e decodificato anche se il simbolo è stato parzialmente sporcato o danneggiato come negli esempi in figura 3.11.

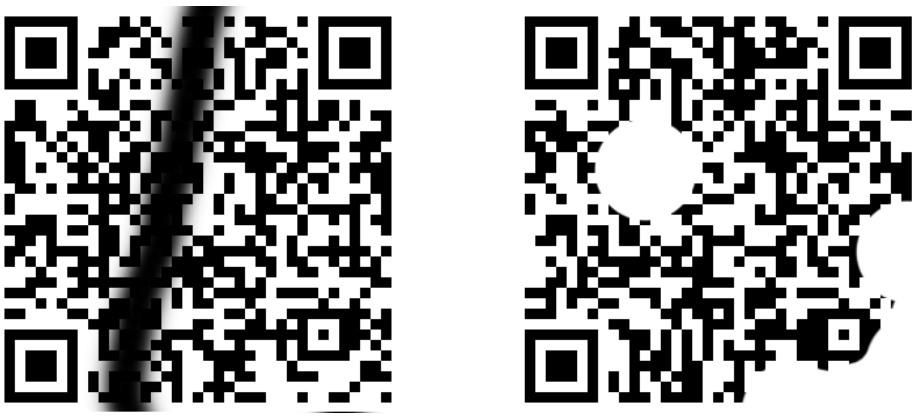

Figura 3.11: Esempi di QR-code danneggiati

Grazie alla sua grafica, un Qr–code può essere letto in qualsiasi direzione esso si trovi, rispetto al decodificatore.

Utilizziamo la figura 3.12 per descriverne la struttura.

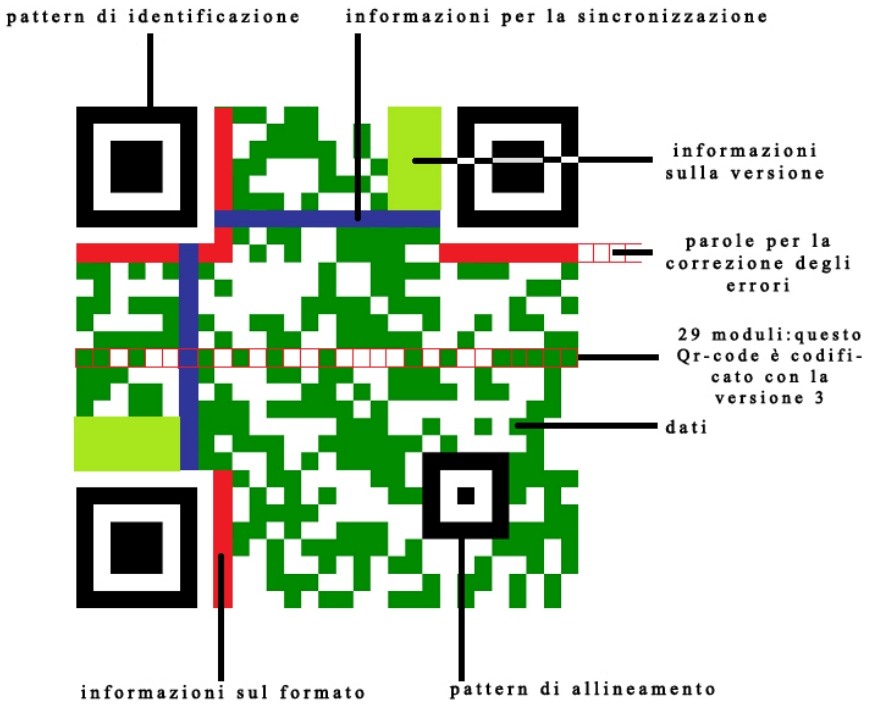

Figura 3.12: Struttura di un Qr-code

Principalmente la struttura grafica di un Qr-code è composta da una parte contenente l'informazione vera e propria (la parte in verde scuro nella figura 3.12) e di tutta una serie di elementi che possiamo vedere come informazione accessoria.

In particolare un ruolo importante per il corretto utilizzo del Qr-code, è svolto dai tre pattern d' identificazione, due posti nei vertici alti della struttura e uno nel vertice in basso a sinistra, che visti come vertici di

un immaginario triangolo isoscele servono a individuare il Qr-code all'interno di un'immagine e a capirne l'orientamento.

All'interno del simbolo sono presenti anche patterns di allineamento, necessari al software di decodifica per la correzione dell'eventuale distorsione geometrica dell'immagine acquisita. I patterns di allineamento variano in numero a seconda delle dimensioni del Qr-code: più grande è il simbolo, maggiore è la probabilità che l'immagine acquisita risulti più o meno distorta.

Sia i patterns di identificazione che quelli di allineamento possono essere letti in orizzontale o in verticale, in quanto individuati dalla stessa sequenza di colori come si può vedere dalla figura sottostante.

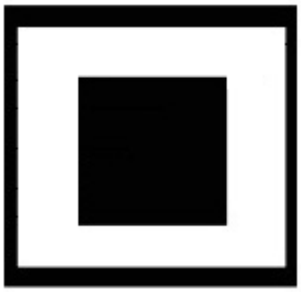

Figura 3.13: Grafica di un Pattern di identificazione o di allineamento

Infine altre porzioni della struttura contengono informazioni relative al formato alla versione del codice, nonché le parole necessarie alla correzione degli errori.

La struttura e la rappresentazione grafica di un Qr–code sono state definite dallo standard ISO/EAC 18004:2000.

3.4.2 La decodifica e le sue problematiche

Un lettore per codici a barre può essere usato solamente per il riconoscimento e la decodifica di un codice, e un lettore per codici bidimensionali è molto costoso.

Oggi i nuovi smartphone possono integrare numerose applicazioni, tra cui la possibilità di sfruttare la telecamera integrata per leggere i codici, sia lineari che bidimensionali. Una volta fotografata la scena, il software di decodifica analizza l'immagine risultante cercando prima di tutto di individuare i patterns di identificazione del QR-code, per poter studiare poi solamente il codice, scartando il resto dell'immagine.

Necessità fondamentale per un'ottima decodifica infatti è il riconoscimento del Qr-code nella scena acquisita. Sia il riconoscimento che la successiva analisi e decodifica devono avvenire in real-time per ridurre i tempi di attesa da parte dell'utente. Individuare i patterns di riconoscimento può essere, in certe condizioni, veramente difficoltoso. Ad esempio se è presente del rumore totalmente concentrato su uno dei tre pattern, tale da renderlo completamente irriconoscibile, il decodificatore non può stabilire l'orientamento del simbolo e portare a termine la decodifica. Altre condizioni svantaggiose possono essere rappresentate dai seguenti casi:

- zone in forte sovraesposizione o sottoesposizione;
- diffusione non omogenea della luce nella scena;
- basso contrasto;
- rumore elevato;
- presenza nella scena di elementi molto simili ai patterns d' identificazione.

Le applicazioni per smartphone dedicate alla lettura dei Qr-code utilizzano vari algoritmi, diversi tra loro, per il riconoscimento in real-time di questi codici in questo paragrafo ne analizzeremo uno tra i più recenti ed efficienti quello di Liu-Yang.

Questo algoritmo di riconoscimento e decodifica prevede in input un'immagine RGB catturata con la fotocamera dello smartphone e tramite diversi passaggi (schematizzati nella figura 3.14) restituisce in output il risultato della decodifica.

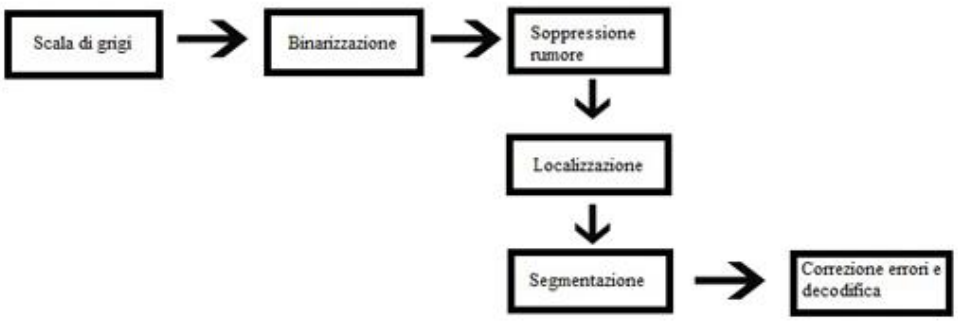

Figura 3.14: Fasi dell'algoritmo di Liu-Yang

Nello specifico l'immagine viene inizialmente convertita in scala di grigi, dato che il Qr–code è per definizione in bianco e nero e lavorare su immagini più leggere rende il processo più veloce.

Successivamente si passa alla binarizzazione la quale utilizza un thresholding adattivo a multilivello che integra soglie locali e soglia globale. Qui risiede il punto di forza di questo algoritmo. Utilizzando la sola soglia globale, si può ottenere facilmente un risultato scadente se le condizioni di illuminazione non sono sufficientemente omogenee. Sarebbe più efficiente utilizzare una soglia locale, anche se non è sempre immediato ottimizzare le dimensioni della finestra da utilizzare e ridurre i tempi di calcolo.

Unendo le due tecniche si ottengono risultati soddisfacenti. Partendo dall'istogramma dell'immagine in scala di grigi, opportunamente filtrato per ridurre l'effetto del rumore, se ne studiano le caratteristiche dei punti di massimo. Se l'istogramma filtrato ha una distribuzione bimodale, il valor medio della depressione massima sarà usato come soglia globale; se l'istogramma ha un solo picco, viene adottato un metodo iterativo per il calcolo della soglia, mediando tra il vecchio valore e il centro dell'area più scura o più luminosa a seconda di dove è situato il picco; se la distribuzione è multimodale si usa un algoritmo di soglia locale. Questa tecnica permette un'ottima binarizzazione e può essere implementata in real-time.

Lo step successivo individua i pattern di identificazione per localizzare il Qr–code . Se un pattern è danneggiato si utilizzano le linee relative alla sincronizzazione, che forniscono un' informazione ausiliaria alla localizzazione del simbolo. Questo metodo non prevede la rotazione dell'immagine, evitando così un ulteriore processo che prolungherebbe il tempo di calcolo.

Una volta localizzati i pattern di identificazione, si conoscono le dimensioni del codice e il suo angolo di rotazione rispetto all'asse orizzontale (figura 3.15).

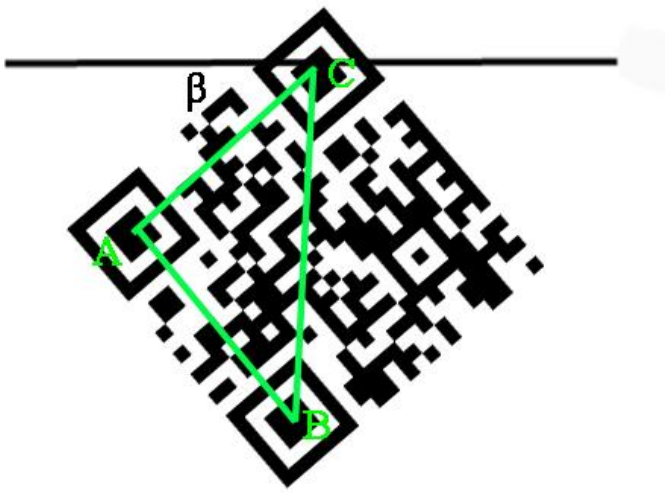

Figura 3.15: Calcolo delle dimensioni di un QRCode e del suo angolo di rotazione rispetto all'asse orizzontale

Una volta ottenute quest'ultime informazioni, viene generata una griglia delle stesse dimensioni del Qr-codice e viene sovrapposta ad esso ottenendone la segmentazione.

Ora i pixel corrispondenti ai quadratini bianchi e neri sono pronti per essere decodificati.

CAPITOLO 4

LE RISORSE DISPONIBILI

4.1 Panoramica

Nel capitolo precedente abbiamo analizzato quelli che saranno gli strumenti principali che utilizzerà la nostra applicazione. Ora cercheremo di capire come gestire nel concreto ogni singola componente riprendendo così nuovamente contatto con il nostro progetto.

Ricapitolando, si è partiti dal voler realizzare un'applicazione turistica, si è deciso di distribuirla sul maggior numero di piattaforme mobili possibili, si è cercato una soluzione che non preveda una nuova riscrittura del codice per ogni versione ma che allo stesso tempo riuscisse a sfruttare le risorse hardware di un dispositivo come il Gps e la fotocamera.

Partendo da queste esigenze, si è arrivati ad una soluzione ibrida la quale prevede la comunicazione tra due componenti: una web-application (della quale parleremo nel prossimo capitolo) che non è in grado di accedere all'hardware di un dispositivo e un modulo (uno per ogni piattaforma) che integri tale deficit.

Arrivati a questo punto in che modo e con che risorse intendiamo realizzare tutto ciò? In questo capitolo approfondiremo le risorse a nostra disposizione per la gestione della fotocamera e della geolocalizzazione, non parleremo invece della web application che verrà trattata nel capitolo successivo. Come risulta dalle analisi e dalle scelte fatte nel capitolo 2, la realizzazione dei moduli per fotocamera e geolocalizzazione, è l'unica parte del nostro progetto che prevede la scrittura di un app, in particolar modo una per ogni sistema operativo (vedremo più avanti che potremmo ridurre ulteriormente questa parte limitandoci a riscrivere solo la gestione della fotocamera delegando la gestione della geoloacalizzazione alla web application).

Per quanto riguarda questa tesi si è scelto di fornire esempi sulla realizzazione di questi moduli solo per due dei principali sistemi operativi sul mercato: Android di Google[1] e iOS di Apple. Questa scelta non limiterà le aspettative iniziali, si vedrà infatti come con una scelta accurata delle risorse renderà il processo di realizzazione di questi moduli simile per ogni sistema operativo mobile moderno.

4.2 Sdk e ambienti di sviluppo

In questo paragrafo faremo una breve descrizione degli ambienti di sviluppo e degli sdk per le due piattaforme scelte.

4.2.1 Android

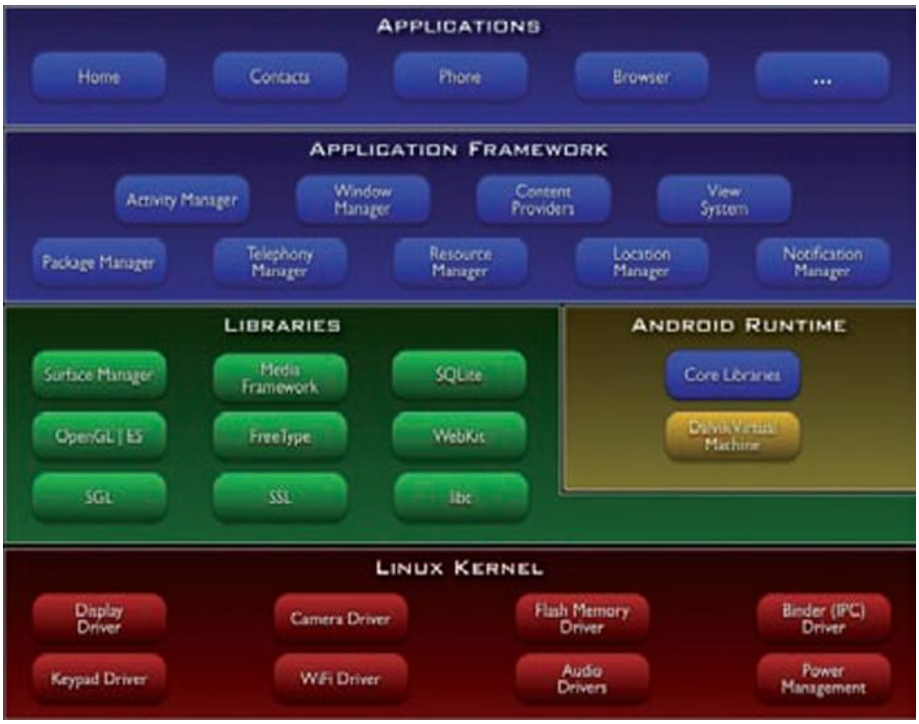

Figura 4.1: L'architettura di Android

Android, essendo un sistema operativo di moderna fattura, è abbastanza complesso. Anche se il suo target sono i dispositivi mobili,

[1] Per essere precisi, dietro Android non c'è soltanto Google. Il colosso di Mountain View ha fatto la prima mossa e di sicuro è l'attore di maggior peso, tuttavia l'evoluzione di Android è curata da un consorzio denominato Open Handset Alliance il cui sito ufficiale è: http://www.openhandsetalliance.com/

l'architettura di Android ha poco da invidiare a quelle dei comuni sistemi per desktop o laptop. Tale architettura è presentata schematicamente in Fig.3.1dalla quale si evince come Google ha attinto a piene mani dal mondo Open Source.

Il cuore di ogni sistema Android, tanto per cominciare, è un kernel Linux. Direttamente nel kernel sono inseriti i driver per il controllo dell'hardware del dispositivo: driver per la tastiera, lo schermo, il touchpad, il Wi-Fi, il Bluetooth, il controllo dell'audio e così via.

Sopra il kernel poggiano le librerie fondamentali, anche queste tutte mutuate dal mondo Open Source. Da citare sono senz'altro OpenGL, per la grafica, SQLite, per la gestione dei dati, e WebKit, per la visualizzazione delle pagine Web (queste librerie saranno di fondamentale importanza per il nostro progetto). L'architettura prevede poi una macchina virtuale e una libreria fondamentale che, insieme, costituiscono la piattaforma di sviluppo per le applicazioni Android. Questa macchina virtuale si chiama Dalvik[1], e sostanzialmente è una Java Virtual Machine.

Nel penultimo strato dell'architettura è possibile rintracciare i gestori e le applicazioni di base del sistema. Ci sono gestori per le risorse, per le applicazioni installate, per le telefonate, il file system e altro ancora: tutti componenti di cui difficilmente si può fare a meno. Infine, sullo strato più alto dell'architettura, poggiano gli applicativi destinati all'utente finale. Molti, naturalmente, sono già inclusi con l'installazione di base: il browser ed il player multimediale sono dei facili esempi. A questo livello si inserirà anche la nostra applicazione.

Per sviluppare applicazioni in grado di girare su sistemi Android, è necessario installare sul proprio PC un apposito kit di sviluppo (SDK), che

[1] Citazione tratta da Wikipedia: Dalvik è una macchina virtuale, progettata da Dan Bornstein, dipendente Google, ed è uno dei componenti di Android. È ottimizzata per sfruttare la poca memoria presente nei dispositivi mobili, consente di far girare diverse istanze della macchina virtuale contemporaneamente e nasconde al sistema operativo sottostante la gestione della memoria e dei thread. Dalvik è spesso associato alla Macchina virtuale Java, anche se il bytecode con cui lavora non è Java.

è disponibile gratuitamente per i principali sistemi operativi all'indirizzo: http://developer.android.com/sdk/. All'interno di questo kit di sviluppo è presente anche un Android Virtual Device (AVD), ovvero un emulatore di dispositivi Android, con il quale si può testare un'applicazione senza il bisogno di acquistare necessariamente un dispositivo reale. Nel nostro caso questo emulatore ci è servito solo nella fase iniziale, infatti non essendo direttamente in grado di simulare la fotocamera del dispositivo emulato (ad esempio attraverso una webcam), rendeva impossibile testare la scansione di codici QR-code.

Per programmare applicazioni in qualsiasi linguaggio, uno degli strumenti fondamentali è l'ambiente di sviluppo (ide). Per quanto concerne lo sviluppo in Android l'ide ufficiale è Eclipse (corredato di un plugin aggiuntivo), anch'esso reperibile per tutto le piattaforme gratuitamente all'indirizzo: http://www.eclipse.org/downloads/.

Infine, Android per i suoi sviluppatori mette a disposizione all'indirizzo: http://developer.android.com tutte le informazioni che possono servire ad uno sviluppatore come: la documentazioni delle classi, guide, tutorial, ed esempi pratici, ecc.

A questo punto una volta creata la nostra applicazione non ci resta che registrarci come sviluppatori nell'Android market, pagando 25$, firmarla in maniera adeguata e pubblicarla sul market.

4.2.2 iOs

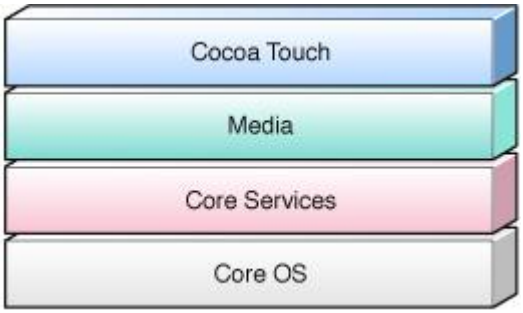

Figura 4.2: L'architettura di iOS

iOS è il sistema operativo sviluppato da Apple per iPhone, iPod touch e iPad. Come Mac OS X è una derivazione di UNIX e usa un microkernel XNU Mach basato su Darwin OS[1].

Come si può vedere nella figura 4.2, l'architettura di iOS è composta da quattro livelli uno sopra l'altro sui quali ovviamente poggia il livello application. Esaminiamoli dal basso verso l'alto.

Il Core OS è il nucleo del sistema operativo, la parte che lavora a più stretto contatto con l'hardware e gestisce le operazioni vitali del nostro device e non solo. Qui sono implementati i metodi per la gestione dei certificati, della sicurezza, quelli per il file system, ecc.
Una delle funzioni più importanti è la gestione della batteria e della potenza erogata: ad esempio quando il device disabilita la wireless, questo strato andrà a spegnere la scheda wifi del dispositivo. Inoltre, gestisce anche:

- threading (POSIX threads);
- networking (BSD sockets);
- standard I/O;
- bonjour e DNS services;
- informazioni sulla lingua Locale;

[1] Citazione da wikipedia: Darwin è un sistema operativo libero che utilizza il kernel XNU. Insieme all'interfaccia grafica proprietaria Aqua, forma il sistema operativo Mac OS X.

- allocazione di memoria;
- calcoli matematici.

Lo strato Core Services implementa le funzioni di gestione delle connessioni di rete, la gestione dei database SQLite, la lista contatti e le preferenze di sistema salvate dall'utente. A questo livello si trova l'implementazione delle features di base più importanti come ad esempio la localizzazione, gli eventi, la parte telefonica, la gestione a basso livello dei Media, ecc. Inoltre include anche una serie di librerie in C di base per la manipolazione degli oggetti come ad esempio le stringhe, le date, gli urls, i threads, ecc.

Questo strato gestisce tutto ciò che concerne la multimedialità, quindi la gestione di audio, video, immagini come jpg, png, tiff, la lettura e la manipolazione di PDF, l'accesso e la modifica della libreria fotografica dell'utente, ecc. Inoltre la funzione principale è l'implementazione delle librerie Open AL per la gestione dell'audio e delle librerie OpenGL ES per la gestione della grafica 2D e 3D. Dalla versione 4.x di iOS qui si trova anche la gestione di AirPlay, la features che permette la riproduzione su vari dispositivi Apple, collegati in rete, di flussi multimediali presenti solo sul nostro device.

E' lo strato che lavora più ad alto livello. Già dal nome si può intuire che si occupa della gestione e del riconoscimento dei movimenti sullo schermo, del touch e del multi-touch dell'utente appunto, ed è in grado di interpretare, in maniera corretta, le gesture (zoom-in, zoom-out, pinch, rotazione, ecc..) compiute da quest'ultimo. Oltre a gestire il touch dell'utente, questo strato si occupa di gestire funzionalità come l'accelerometro ed il giroscopio riuscendo dunque a capire in che modo è orientato il dispositivo rispetto ad un asse orizzontale. In più si occupa della gestione della gerarchia delle view, del multitasking, della fotocamera del dispositivo, della stampa, delle alert, delle notifiche push, del file-sharing e del Peer-to-Peer A questo livello poi sono presenti i framework aggiuntivi più utilizzati ad un più alto livello, come il

framework per la gestione della rubrica, degli eventi, dell'iAd, delle connessioni con il Game Center. Notizie più dettagliate e precise sulla struttura di iOS si possono trovare all'indirizzo:

http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/IPhoneOSOverview/IPhoneOSOverview.html#//apple_ref/doc/uid/TP40007898-CH4-SW1.

Come per Android, sviluppare applicazioni per iOS richiede il download di un sdk. In questo caso però bisogna prima accedere al sito di Apple all'indirizzo http://developer.apple.com/devcenter/ios/index.action registrarsi ed ottenere gratuitamente un Apple Id, con il quale si potrà accedere alla sezione downolad e scaricare l'sdk, insieme ad Xcode che è l'ide ufficiale per programmare per iOs.

Il linguaggio di programmazione nel quale vengono scritte le applicazioni per iOS è Objective-c, che come lo stesso nome suggerisce, è un'estensione a oggetti del linguaggio C. Esso mantiene la completa compatibilità col C. Le due principali estensioni dei file Objective-c sono *.h* e *.m* con significato rispettivamente equivalente al *.h* e *.c* nel linguaggio C. Un altro tipo di file è quello con estensione *.mm* che non è nient'altro che una versione del *.m* con la possibilità di scrivere codice in C++.

Il pacchetto contenente Xcode, comprende anche Interface Builder per creare l'interfaccia grafica e tutta una serie di utility tra le quali anche un emulatore che come nel caso precedente non riesce però a emulare la fotocamera. In questo caso inoltre, l'acquisto della licenza da sviluppatore è necessaria anche per il test dell'applicazione su un dispositivo reale. La licenza costa 99$ e senza di essa non si può ne testare su device ne ovviamente pubblicare sull'Apple store.

4.3 La geolocalizzazione

Dopo aver fatto una panoramica sui due sdk che andremo ad utilizzare, veniamo ora alla gestione della geolocalizzazione. I principali modi per conoscere la posizione di un determinato device sono state presentate nel paragrafo 2.3, cerchiamo adesso di capire che tipo di risorse esistono per gestirle.

Entrambi i sistemi: Android e iOS, hanno un loro supporto nativo per la geolocalizzazione. Inizialmente si pensava di utilizzare questa strada per fornire alla nostra applicazione la possibilità di essere localizzata, ma dopo un'attenta ricerca si è trovato un modo per conoscere la posizione di un device direttamente da web application. Una tale soluzione è ovviamente la migliore, in questo modo infatti si evita di scrivere una nuova funzione per ogni piattaforma. Il nostro Graal in questo caso è rappresentato dall'oggetto *navigator*, utilizzabile all'interno di un qualsiasi codice javascript, il quale secondo le specifiche w3c del 10 Febbraio 2010[1], con l'ausilio di due metodi *getCurrentPosition()* e *watchPosition()* ci permette di ottenere informazioni sulla posizione geografica di un device, rispettivamente una sola volta nel primo caso, ad ogni cambio di posizione nel secondo.

La grande utilità di questi due metodi non sta solo nel fatto che ritornino la posizione, ma anche nella loro capacità di astrarsi dal metodo di acquisizione attivo su di un device. In pratica, all'oggetto navigator che sostanzialmente rappresenta il web browser, non interessa se il device calcoli la posizione utilizzando il metodo delle reti wifi, cell-based o GPS, ma semplicemente utilizza la posizione calcolata con il sistema più preciso presente e attivo nel sistema. Nell'ordine abbiamo Gps, Wifi, cell-based.

[1] Geolocation API Specification link: http://dev.w3.org/geo/api/spec-source.html

4.4 ZXing

Diversamente da quanto successo con la geolocalizzazione, per quanto riguarda l'acquisizione del QR-code non possiamo cavarcela con una versione unica da incorporare nella web application, ma bisogna creare un modulo per ogni piattaforma.

Anche in questo caso la situazione che inizialmente sembrava scomoda, si è rilevata indolore e non solo per ciò che riguarda le due piattaforme prese in considerazione.

Cercando per il web ci si è imbattuti in ZXing (si legge "zebra crossing") un progetto open-source che fornisce una libreria scritta in java ma con porte anche per altri linguaggi, per la scansione di codici a barre 1D/2D attraverso l'uso di una fotocamera. In particolare supporta i seguenti formati di codici a barre:

- UPC-A and UPC-E;
- EAN-8 and EAN-13;
- Code 39;
- Code 93;
- Code 128;
- QR Code;
- ITF;
- Codabar;
- RSS-14 (tutte le varianti);
- Data Matrix;
- PDF 417 ('alpha' quality);
- Aztec ('alpha' quality).

Praticamente questa libreria fa addirittura più del dovuto, fornendo anche, la possibilità di integrarla in applicazioni per i principali sistemi operativi mobili come iOS, Android, Rim, Symbian e da gennaio 2011 è disponibile in versione beta anche il supporto per Windows Phone 7 all'indirizzo: http://silverlightzxing.codeplex.com/, mentre il progetto

principale con tutta la documentazione ufficiale è disponibile all'indirizzo: http://code.google.com/p/zxing/.

CAPITOLO 5

LA WEB APPLICATION

5.1 Panoramica

Dopo aver esaminato quelle che possiamo definire le componenti ausiliari del nostro progetto, passiamo ora a descriverne il fulcro: la web application. In questo capitolo faremo una descrizione della web application usata, e della sua architettura

5.2 QRPlaces

QRPlaces è una piattaforma web/mobile pensata e sviluppata da NSi (Nier Soluzioni Informatiche) a supporto dell'esplorazione turistica del territorio fruibile da PC e da dispositivi mobili di nuova generazione connessi alla rete Internet (Wi-Fi/3G/HSPA). A fianco delle funzioni informative QRPlaces offre funzioni di comunity tra gli utenti della piattaforma.

La piattaforma QRPlaces da la possibilità di censire, documentare e mettere a disposizione del visitatore informazioni su diversi circuiti informativi tra loro relazionati.

Ogni circuito consiste in un contenitore di "place". Un place rappresenta ciò che nel capitolo 2 abbiamo definito come "oggetto", contiene cioè tutte le informazioni che servono a descrivere un bene culturale/ambientale, un evento, o un'informazione di tipo business.

Ogni place è associato ad un canale. Attualmente i canali presenti sono: cultura, eventi, mangiare e bere, acquisti e hotel. I canali caratterizzanti e trainanti l'iniziativa sono quelli relativi ai beni culturali/ambientali e agli eventi, gli altri invece sono messi a disposizione in un'ottica di completamento di risposta agli interessi e ai bisogni reali del visitatore.

In particolare i principali dati utilizzati per descrivere un places sono: nome, descrizione (in più lingue), indirizzo, coordinate della sua posizione, contatti, circuito e canale di appartenenza.

Figura 5.1: Schermata iniziale con la lista dei canali attualmente disponibili su QRPlaces

Entrando più nel dettaglio, QRPlaces, utilizzando la geolocalizzazione (come descritta nel paragrafo 4.3) fornisce all'utente un elenco di place appartenenti al circuito e al canale scelto, filtrato e ordinato in base alla distanza dalla sua posizione (figura 5.2).

Figura 5.2 Elenco dei places culturali filtrati e ordinati secondo la posizione dell'utente

Un singolo place viene presentato all'utente corredato di un set d'informazioni quali: immagini e descrizioni in formato sia audio (utilizzando un sintetizzatore vocale) che testuale (figura 5.3).

Figura 5.3: Interfaccia di presentazione di un singolo place

L'accesso al place così come si vede nella figura 5.3 può avvenire principalmente in due modi: da un elenco come quello in figura 5.2, oppure direttamente da un url dedicato. Quest'ultimo accesso è stato pensato principalmente per permetterne l'accesso tramite la scansione di un QR-code con l'ausilio di un applicazione esterna e sarà proprio questo il punto d'incontro tra la nostra applicazione e la web application ma questo verrà affrontato nel prossimo capitolo.

Anche se non considereremo questa funzionalità per il lato mobile, la web application, prevede anche un' interfaccia pensata unicamente per browser desktop che permette agli utenti che nel capitolo 1 abbiamo chiamato "producer", di creare e immettere nel sistema uno o più place (figura 5.4).

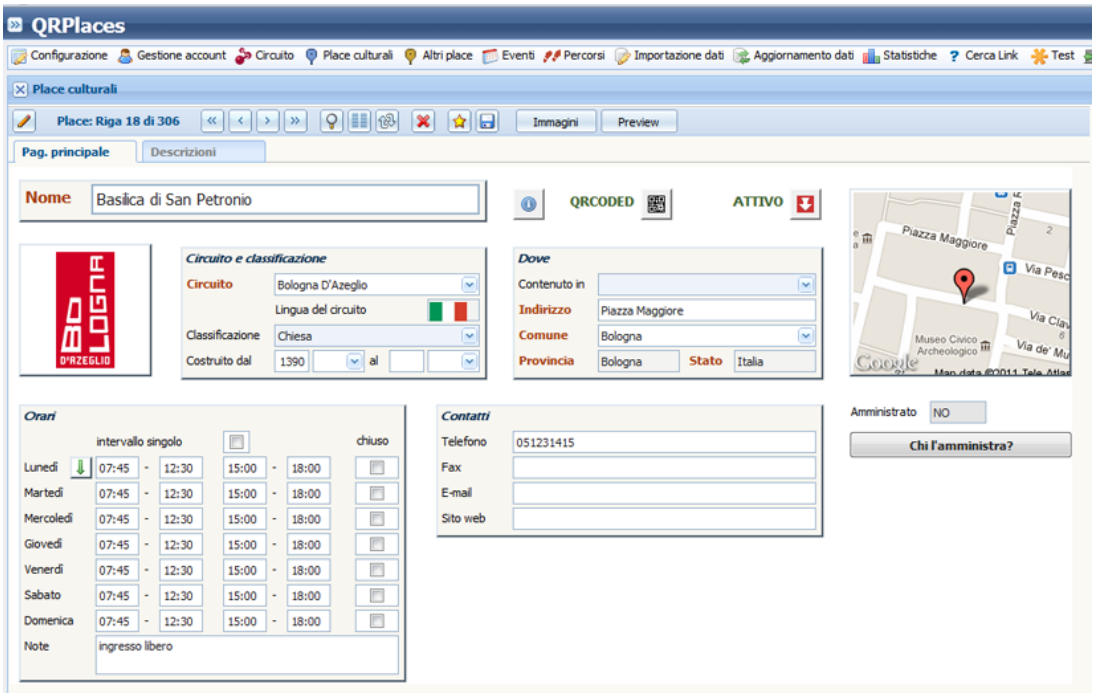

Figura 5.4: Interfaccia per l'inserimento di un Places

5.3 L'architettura

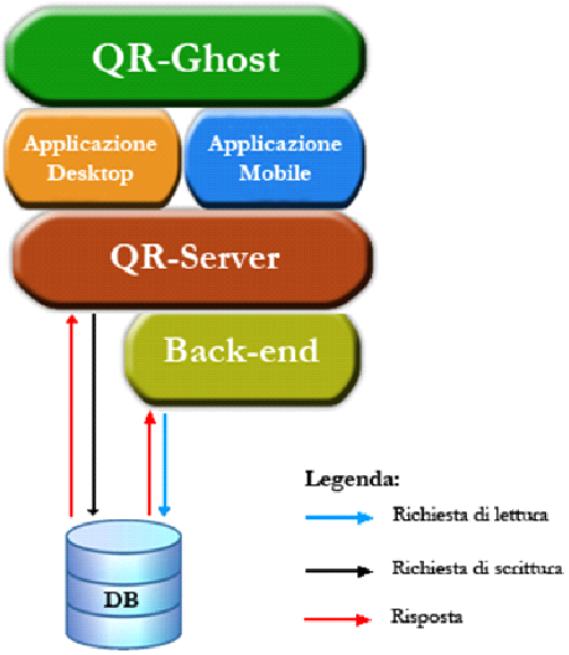

Figura 5.5: Schema sull'architettura di QRPlaces

La figura 5.5, rappresenta graficamente l'architettura di QRPlaces. La piattaforma se si esclude la base di dati, è composta da cinque parti. Cominciamo dall'alto verso il basso.

Il livello più alto, è rappresentato da QR-Ghost è la parte che riceve la richiesta dal client, riconosce il tipo di browser e dopo aver memorizzato alcune informazioni di tipo statistico sugli accessi, richiama il modulo mobile o quello desktop a seconda del tipo di client rilevato.

Al livello inferiore ci sono le due applicazioni con le quali realmente l'utente interagisce. Entrambe per effettuare qualunque tipo di accesso alla base di dati hanno bisogno del livello sottostante ovvero QR-Server.

Per le richieste di lettura dei dati, il QR-Server s'interpone tra le applicazioni e il database mentre, le richieste di scrittura le inoltra allo strato back-end, che è l'unico a poter scrivere sul database.

Tutte le componenti, comunicano tra loro con richieste Ajax, e con risposte in json.

CAPITOLO 6

L'APPLICAZIONE FINALE

Dopo aver studiato nei due capitoli precedenti le risorse a nostra disposizione, il passo finale sarà quello di integrare il tutto, realizzando la nostra applicazione finale.

6.1 Linea guida per l'integrazione

Per questa tesi, come detto in precedenza, sono state realizzate due app: una per iPhone, e una per Android e rese disponibili sui market di Apple e Android. Ma lo scopo finale di questo progetto non e la realizzazione di queste due applicazioni ma, quello di trovare una procedura il più possibile generalizzabile per creare più versioni (una per ogni sistema operativo mobile), evitando ogni volta di riscrivere tutta l'applicazione. Una volta individuato, tale processo di produzione, è stato quindi testato sulle due piattaforme citate e ha prodotto dei tempi di realizzazione soddisfacenti.

6.2 Il processo di produzione

Cerchiamo a questo punto di sintetizzare il processo di produzione ottenuto.

Il primo passo affrontato è stato quello di trovare un modo di "incorporare" una web application all'interno di un app. A tal proposito, si è notato che i linguaggi per le moderne piattaforme mobile, permettono di incorporare all'interno delle proprie app un browser nativo. Nel caso di Android e iOS questo procedimento viene effettuato con una vista: rispettivamente, una WebView[1] in Android e una UIWebView[2] in iOS.

[1] WebView Class: http://developer.android.com/reference/android/webkit/WebView.html

[2] UIWebView Class http://developer.apple.com/library/ios/#documentation/uikit/reference/UIWebView_Class/Reference/Reference.html

Inoltre questi oggetti possono essere facilmente manipolati e controllati, ad esempio richiamando su di essi un metodo per caricare una pagina web con un certo url.

Abbiamo già visto nel paragrafo 4.3 che il problema della geolocalizzazione era stato risolto direttamente all'interno della web application perciò a questo punto rimane da illustrare solo il processo di integrazione con la lettura del QR-code.

La soluzione proposta da Zxing si è rivelata particolarmente adatta per essere utilizzata all'interno della nostra struttura. Infatti, importando infatti la libreria all'interno di un progetto abbiamo la possibilità di gestire l'acquisizione del codice a barre e la sua decodifica, potendo poi passare il risultato (che ricordiamo nel nostro caso essere un url) al metodo dell'oggetto browser responsabile del caricamento della pagina. La figura 6.1 rappresenta quanto detto.

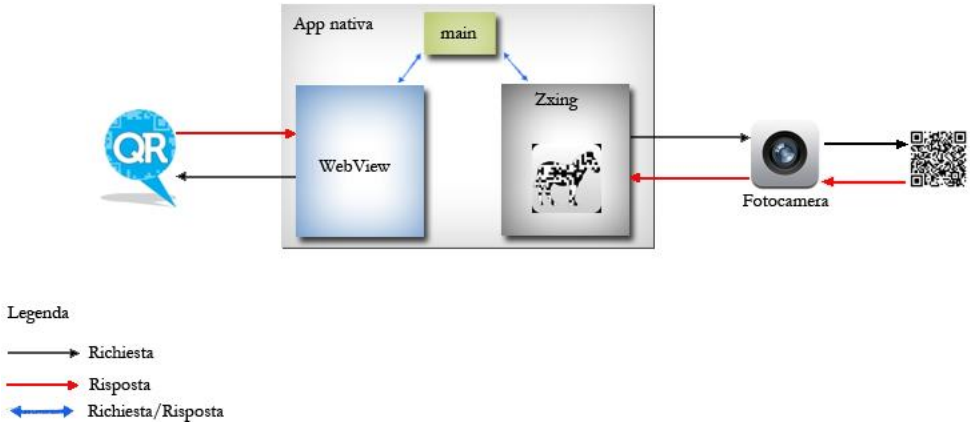

Figura 6.1: Architettura di un'app nativa

Se esaminiamo la figura vediamo che le due componenti vengono gestite e tenute insieme da un elemento "main". Per comprendere meglio come l'app gestisca le componenti interne ed esterne esaminiamo due dei possibili casi d'uso: accedere alle informazioni di un place utilizzando le informazioni derivanti dal gps, e accederci in maniera diretta tramite Qr-code.

Ricordiamo che in questo capitolo non affrontiamo il lato "producer", in quanto non è previsto per la versione mobile.

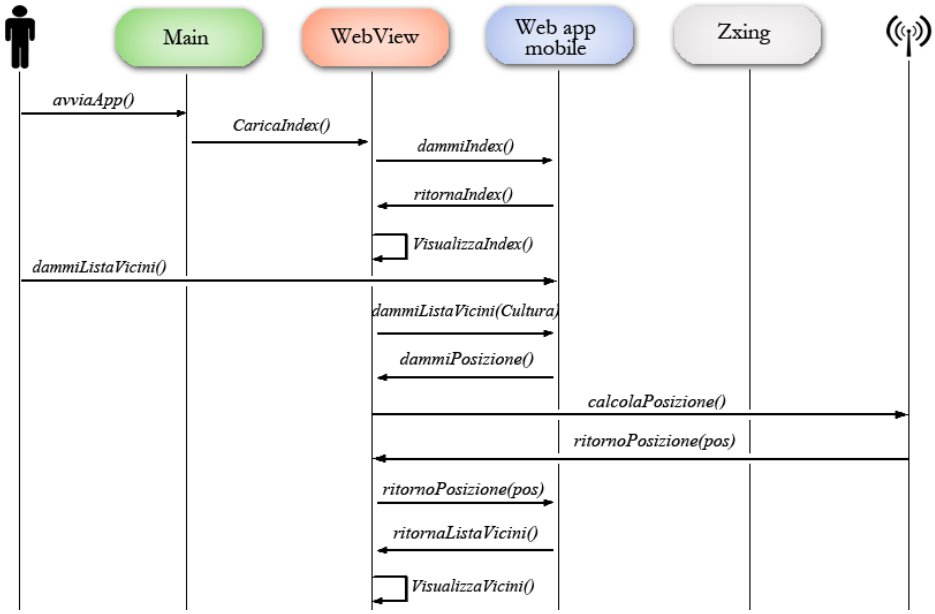

Figura 6.2: Esempio di utilizzo della geolocalizzazione da parte dell'app

Nella figura 6.2 è schematizzato il caso in cui un'utente vuole conoscere la lista dei places appartenenti al canale cultura vicini a lui. Quando l'utente avvia l'app, il main chiede alla WebView di caricare la pagina iniziale di QRPlaces. La WebView richiede la pagina alla web application che gli ritorna l'index. Una volta visualizzata la pagina iniziale, l'utente chiede di conoscere i places di tipo culturale vicino alla sua posizione.

La WebView richiede l'elenco dei vicini alla web application, la quale chiede a sua volta di definire la posizione alla webView, che per calcolarla utilizza il gps o qualche altro metodo di geolocalizzazione per ottenere la posizione. Una volta ottenuta la posizione l'inoltro alla web application la quale adesso può costruire l'elenco dei vicini e può inoltrare la pagina alla WebView.

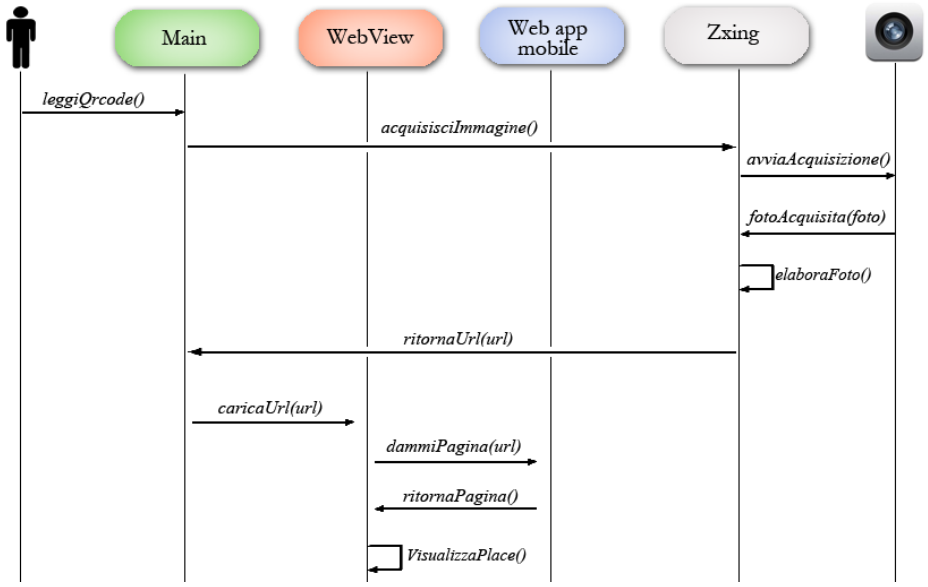

Figura 6.3: Esempio di utilizzo del lettore di QR-code

Nella figura 6.3, invece, è rappresentato il caso d'uso per cui è stata pensata tutta questa architettura, quello cioè di accesso ad un places direttamente da QR-code.

L'utente ad un certo punto decide di scansionare un QR-code, e fa una richiesta al main, a questo punto, il main invia una richiesta a Zxing, il quale, attiva la fotocamera, che avvia la scansione. A questo punto la procedura sarebbe più complessa, ma per semplificarne la comprensione, supponiamo che al primo tentativo la fotocamera ritorni l'immagine con il QR-code all'interno.

A questo punto, Zxing elabora l'immagine, la decodifica e invia l'url alla WebView affinché visualizzi il place corrispondente. Adesso la WebView richiede la pagina alla web Application e una volta ricevuta, la pagina viene visualizzata.

CONCLUSIONI E SVILUPPI FUTURI

Questa tesi, è nata per cercare di fornire alla piattaforma QRPlaces uno strumento aggiuntivo come quello del supporto al QR-code e si è trasformata poi in una ricerca di linee guida e di strumenti utili per un processo produttivo adeguato sia in termini qualitativi che in termini di efficienza produttiva che permettesse la distribuzione del prodotto finito sul maggior numero di dispositivi mobili.

Inoltre, per la scelta delle risorse da utilizzare, si è cercato di puntare sull'utilizzo di strumenti fruibili gratuitamente e liberamente sul web, riducendo i costi di una soluzione di questo tipo ai soli dazi previsti dai diversi canali di distribuzione. Ulteriori benefici derivanti dalla soluzione scelta, si hanno anche in termini di estendibilità. Pensiamo infatti, se un domani alla moda dei QR-code si aggiunga qualche altro bizzarro metodo per accedere alle informazioni. Nulla ci vieterà di creare un modulo aggiuntivo che interagirà anch'esso con la nostra web application, il tutto senza dover toccare una riga di codice dell'applicazione centrale. È ovvio che una soluzione di questo tipo presenta anche dei limiti, ad esempio se al posto di andare a modificare il modo di accedere ai dati, si modifica il modo di presentarli, tutti i discorsi relativi a non dover faticare vacillano (in questo caso entra in gioco la bontà della web application). Inoltre la maggior parte del lavoro in una soluzione di questo tipo viene svolto dalla web application, con ovvi rallentamenti rispetto ad un'applicazione non ibrida.

La piattaforma QRPlaces è un progetto ancora giovane, ma con grosse potenzialità e soprattutto versatile. Tale applicazione, già così com'è potrebbe essere utilizzata in diversi ambiti, ad esempio all'interno di mostre o fiere come strumento di navigazione e informazione, oppure in progetti per la comunità sia per scopi ludici che educativi. Pensiamo alla possibilità di creare un circuito apposito per i bambini, che permetta a un bambino di visitare una città con a disposizione contenuti e descrizioni creati da un suo pari. Se poi la tali informazioni venissero create all'interno

di realtà come le scuole, mediante appositi progetti didattici paralleli a quelli standard, si potrebbe mettere in moto un meccanismo dal quale tutti sia “baby consumer“ che “baby producer” ne trarrebbero giovamento. La piattaforma QRPlaces è un progetto in continua evoluzione, nuove funzionalità sono in programma come l’aggiunta di percorsi tematici lungo i quali si può essere guidati grazie all’ausilio del gps, è ancora molto giovane e ancora molto c’è da migliorare, ma di sicuro è un progetto con buone potenzialità.

SITOGRAFIA

http://ilsistemagsm.interfree.it/Approfondimenti/Localizzare.htm
Articolo sulla localizzazione cell-based.

http://www.phonegap.com
Sito ufficiale di Phonegap.

http://rhomobile.com
Sito ufficiale di Rhomobile.

http://www.comscore.com/Products_Services/Product_Index/MobiLens
Sito ufficiale di comScore, una società di ricerca marketing via internet.

http://www.losnaweb.com/
Sito ufficiale di Losna, una web agency che opera in vari ambiti tra cui la realizzazione di applicazioni informatiche a supporto dei beni culturali.

http://www.wigoapp.com/
Sito ufficiale di WIGO.

http://www.whaiwhai.com/
Sito ufficiale whaiwhai.

http://www.comefunziona.net
Sito di cultura generale.

http://www.openhandsetalliance.com/
Sito del consorsio fondatore di Android.

http://dev.w3.org/
Sito ufficiale w3c.

http://developer.apple.com/
Sito ufficiale sviluppatori Apple.

http://developer.android.com
Sito ufficiale sviluppatori Android.

http://code.google.com/p/zxing/
Sito ufficiale progetto Zxing.