Sessione II
Anno Accademico 2013/2014

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

MIRAMBLA MOBILE

Relazione finale in
Mobile Web Design

Relatore
Dott. Mirko Ravaioli

Presentata da
Francesco Maughelli

“Quello che il corpo umano crea
è espressione del DNA quanto e
come il corpo stesso”

- Batou

INDICE

INTRODUZIONE

In un mondo moderno frenetico e sviluppato, dove la rete può raggiungere chiunque ovunque si trovi, la comunicazione fra le persone è divenuta sempre più capillare. Gli smartphone insieme ai tablet ed alle altre tecnologie emergenti hanno rafforzato enormemente la capacità dell'uomo di poter semplificare la propria vita, di sentirsi vicino, di allargare i propri confini e le proprie conoscenze. Questi incredibili mezzi sono oggi la porta che è in grado di dare accesso al mondo intero.

Aristotele sostenne che l'uomo è un *animale sociale*, quest'ultimo tende, infatti, per natura ad aggregarsi con altri individui e a costituirsi in società[1]. Tale affermazione difatti non si discosta dalla realtà e se un tempo la forza per mantenere unita la società era rappresentata dalla comunicazione verbale, con l'avvento della globalizzazione digitale la "parola" della società si sposta sempre di più verso le grandi "memorie esterne" create dall'uomo: piattaforme virtuali come i siti *social* oppure i *social network*. Storicamente l'uomo, per potersi confrontare e dare voce alle proprie opinioni, utilizzava *l'agorà*[2] , il centro rappresentativo del potere del popolo; oggi le persone, sempre più frequentemente, partecipano ad eventi di cui si passano parola on-line tramite chat, messaggi, e-mail, comunicazione non convenzionale o più prettamente social.

L'introduzione degli *smartphone* è stato uno degli eventi più di successo degli ultimi dieci anni. In un periodo relativamente breve di tempo, infatti, la tecnologia *mobile* intelligente è permeata in modo significativo nella società, catturando l'interesse di un ampio spettro d'età, dai bambini nelle scuole agli anziani. Questa rapida diffusione delle moderne tecnologie *mobile* ha trasformato molti aspetti della vita quotidiana; non soltanto il modo di comunicare, ma anche il senso della cultura, della comunità, dell'identità e le relazioni interpersonali (M.N.K. Boulos, 2011).

[1] *Politica*, Aristotele
[2] Agorà (in greco antico ἀγορά, da ἀγείρω = raccogliere, radunare) è il termine con il quale nell'antica Grecia si indicava la piazza principale della polis.

La capacità di miniaturizzazione delle tecnologie ha permesso di utilizzare gli *smartphone* come strumenti per rimanere sempre aggiornati su quello che accade. Tali strumenti consentono, inoltre, di ricevere sempre informazioni su dove e quando ci saranno attività interessanti per coloro che ne fanno utilizzo.

Il presente lavoro, calato nel contesto attuale del mondo mobile sociale, riguarda da vicino l'organizzazione di eventi e si propone di presentare una nuova applicazione in grado di gestire eventi per un portale web. Il lavoro illustrerà il processo creativo completo a partire dall'idea astratta che, tramite il sapiente utilizzo di tecnologie emergenti per la programmazione in ambiente *smartphone*, ha maturato la sua evoluzione in un risultato tangibile e in un prodotto potenzialmente finito e all'avanguardia.

Il presente lavoro in un primo momento andrà ad indagare l'origine e l'evoluzione degli *smartphone* e dei loro sistemi operativi fino ad oggi, dando un ampio spazio alla piattaforma *Android* su cui è stata sviluppata l'applicazione in esame. Successivamente verrà affrontata un'analisi del dominio applicativo sul quale l'applicazione è stata creata: un portale dedicato alla gestione di eventi nel territorio romagnolo. Da tale analisi verranno dedotte le specifiche di progetto necessarie all'implementazione dell'applicazione. Vi sarà poi una parte dedicata allo sviluppo vero e proprio: *tool*, risorse utilizzate per affrontare le difficoltà, implementazione del codice che costituisce le parti che sono state fondamentali nel corso dello sviluppo e *testing*. Seguiranno le conclusioni.

1.0 IL MONDO MOBILE

Con il termine “tecnologie mobile” possiamo intendere tutte quelle famiglie di hardware e software che ci permettono di effettuare uno scambio di dati o di avere libero accesso alla rete internet ed ai suoi servizi in breve tempo ed ovunque sul pianeta. Ad oggi questa definizione si è di molto evoluta rispetto al decennio scorso. In passato con l’hardware, infatti, si indicavano quei dispositivi portatili che servivano solo in ambito lavorativo oppure erano costruiti *ad hoc* per svolgere specifiche funzioni (*task-oriented*). Oggi invece ricadono in questa definizione quei *device* che possono essere utilizzati comodamente in movimento, la cui capacità di calcolo ed elaborazione è aumentata notevolmente: *smartphones*, *tablet* e *wearable*. Occorre però fare una doverosa precisazione circa il momento preciso dell’evoluzione dai cosiddetti telefoni di vecchia generazione chiamati “*feature phones*” verso quelli di nuova generazione detti “*smartphones*”. (Jeon, Kim, Lee & Won, 2011).

1.1 Storia ed evoluzione dei dispositivi mobile

Prima del 1992 non esistevano, di fatto, telefoni cellulari le cui capacità potessero distinguersi di molto da quelle di un comune telefono fisso. In questa data però IBM prototipò il primo *device* della storia ad incorporare le funzionalità di un cellulare a quelle di un PDA (Personal Digital Assistant), dispositivo che fin’ora era soltanto un privilegio diffuso tra gli uomini d’affari. Negli anni successivi al 1994 diversi produttori (*Ericsson*, *Nokia* etc.) iniziarono la ricerca in questo campo nella speranza di trovare un prodotto che potesse soddisfare tali aspettative tecnologiche. Nel 1995 il termine “*smart phone*” fu coniato alla sua prima apparizione per descrivere il “*PowerWriter Communicator*” di AT&T (Savage, 1995). Il modello Nokia 9000 viene, però, considerato il primo *smartphone* “di fatto”; fece il suo debutto nel 1996 ed

introdusse notevoli funzionalità innovative per l'epoca come la gestione di e-mail, calendario, rubrica, calcolatrice, agenda, navigazione web *text-based* e l'abilità di invio/ricezione FAX. In questo periodo anche altri produttori in tutto il mondo si cimentarono in quest'area, portando notevoli migliorie soprattutto nel campo del software, dove nacquero i primi sistemi operativi di tipo mobile (*Palm OS, BlackBerry OS, Windows CE/Pocket PC*). (Sager, 2012).

La vera svolta si ebbe nel 2007, anno in cui *Apple Inc.* introdusse sul mercato il suo primo *iPhone*. L'arrivo del loro telefono rivoluzionò la scena dei dispositivi mobile in quanto, per la prima, volta lo *smartphone* veniva concepito come oggetto per un consumer di tutti i giorni invece che solo per il businessman. Questo modello di nuova generazione ha apportato numerose differenze rispetto al passato; differenze che, a tutti gli effetti, hanno dettato le linee guida del nuovo standard al quale, successivamente, si è adeguata la maggior parte di partner e competitor del settore. Altra principale innovazione è stata il design del dispositivo, composto interamente da un display *touch-screen*, in cui l'esperienza utente poteva essere svolta in maniera semplice sfiorando lo schermo con le proprie mani e senza l'utilizzo di una *stylus*. Al contrario i modelli precedenti erano dotati di tastiere fisiche (le quali sottraevano dimensione agli schermi) e altri metodi di input, diventati da quel momento obsoleti. Questo notevole slancio creativo e il relativo investimento nel progetto diedero i loro frutti quando, nel Settembre del 2007, l'azienda annunciò che le vendite avevano superato il milione di unità a soli 74 giorni dopo l'uscita del primo *iPhone* il 29 Giugno. (GH Forman, 1994).

Nell'Ottobre del 2008, invece, fece il suo debutto HTC[3], che, presentando il suo *Dream G1*, lanciò nel mercato il sistema operativo *Android*, in seguito all'acquisizione e al conseguente sviluppo da parte di *Google* e da *Open Handset Alliance*. Lo scopo preciso di questo progetto fu quello di creare un forte competitor che fosse al passo con la tecnologia di quegli anni, ma che utilizzasse un approccio *open-source* rispetto a quello di altre aziende come *Symbian, BlackBerry OS,* e *iOS*. (Landay, 1993).

[3] *High Tech Computer Corporation*

Il dispositivo del robottino verde divenne molto popolare negli Stati Uniti, dove nell'Aprile 2009 raggiunse il record del milione di vendite. La vera ascesa del prodotto si ebbe, però, con l'uscita della nuova versione di *Android*, montata sul *Motorola Droid* nell'Ottobre dello stesso anno. (Chowney, 2012)

Nel giro di un paio di anni, Apple e Google trasformarono completamente il settore *smartphone*, divenuto ormai una realtà consolidata, usufruibile non più soltanto da enterprise o business users ma da ogni tipo di consumer. Questo costrinse le non poche, ma ormai vecchie aziende leader di settore, ad adattarsi alle innovazioni apportate da questi sistemi, i quali in poco tempo e con grande sorpresa scavalcarono qualsiasi statistica, diventando i più utilizzati in commercio. Un esempio è costituito da *Microsoft* che fece la sua apparizione sul mercato *smartphone* soltanto nel 2010, con un forte ritardo rispetto alla concorrenza, poiché costretta a ricostruire il suo sistema operativo mobile quasi da zero; l'azienda passò, infatti, dal vecchio *Windows Mobile* a *Windows Phone*. Un altro esempio è costituito da *Nokia*, che, nonostante il parziale successo ottenuto grazie a Symbian, abbandonò il sistema in favore della piattaforma di casa Microsoft. Come Microsoft anche *BlackBerry* (formalmente *RIM*), fu costretta a ricostruire un SO da zero. Molti furono poi i produttori di *device* cellulari che adottarono proprio Android come sistema operativo, poiché aderirono alla *Open Handset Alliance* (tra cui *Google, HTC, Sony, Motorola, Samsung Electronics, LG Electronics, T-Mobile, Sprint Corporation* etc.). (O.H.A., 2011)

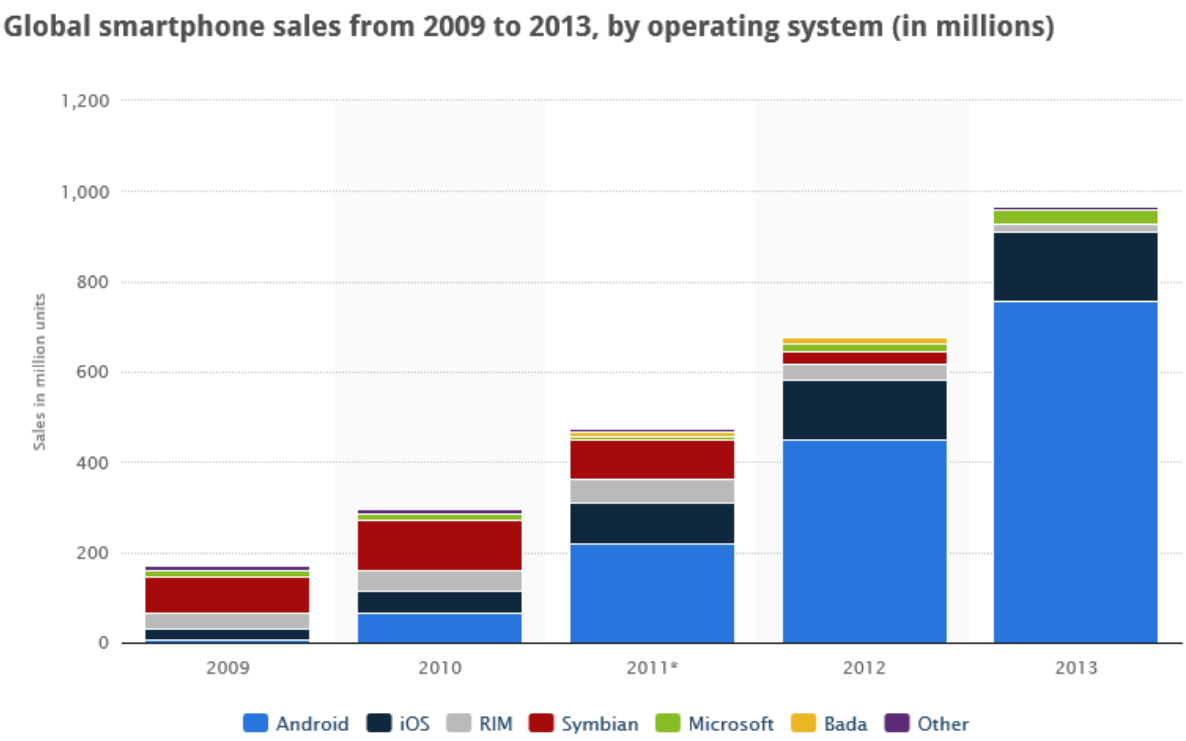

Figura 1 - vendite globali smartphones 2009-2013 (fonte www.statista.com)

1.2 Panoramica attuale

Al giorno d'oggi, dopo gli innumerevoli stravolgimenti dovuti alle innovazioni sia hardware che software degli ultimi anni, lo scenario degli *smartphone* si è di certo stabilizzato. Prendendo in considerazione sia telefoni che tablet, il predominio dei sistemi operativi *mobile* rimane, di fatto, un'agguerrita lotta tra Android ed iOS, combattuta dalle rispettive aziende in termini di quantità di vendite, percentuali di impiego del SO e brevetti. Android è il leader indiscusso delle quantità, detiene circa l'85% (2014Q2) del mercato globale *smartphone* grazie al suo investimento in tutte le fasce di prezzo con numerosi e differenti modelli, contro il 12% (2014Q2) di iOS. Nelle percentuali di share del sistema operativo la disparità è minima tra i due sistemi; a Ottobre, infatti, il robottino verde è risultato in vantaggio su iOS, ma con un distacco così minimo da far sempre scivolare il primato in mani diverse da mese a mese.

Mobile/Tablet Operating System Market Share
Istantanea ad Ottobre, 2014

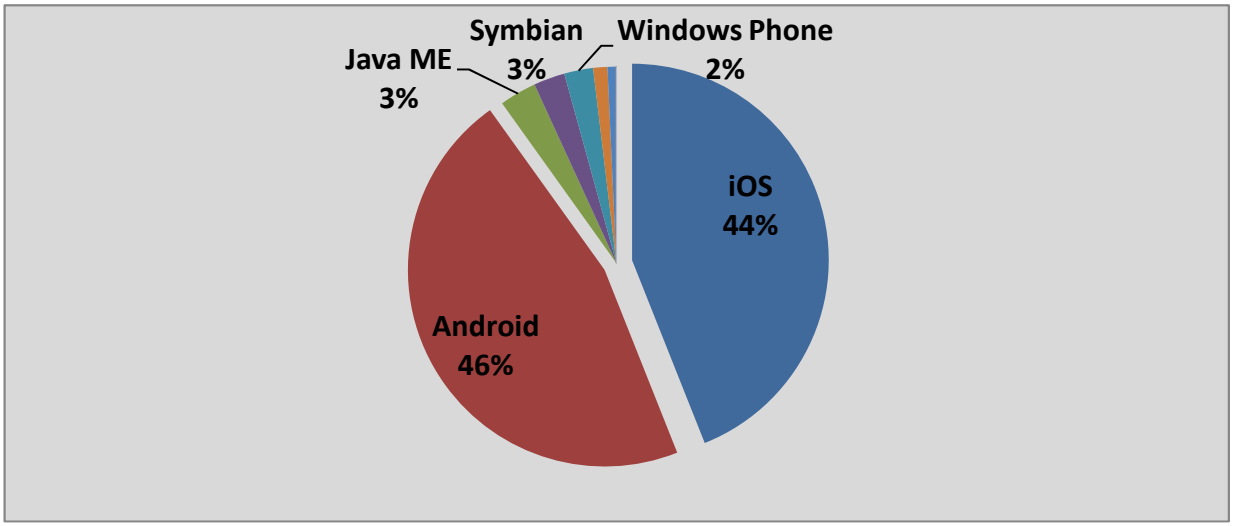

Figura 2 - Mobile OS Market share (fonte www.netmarketshare.com)

Operating System	Total Market Share
iOS	44,23%
Android	46,38%
Java ME	3,05%
Symbian	2,60%
Windows Phone	2,38%
BlackBerry	1,19%
Altri (Kindle, Samsung, Bada, LG, etc.)	0,17%

Occorre approfondire, però, le problematiche ed i limiti che si devono fronteggiare quando parliamo di utilizzo e, soprattutto, di sviluppo su dispositivi *mobile* all'attuale stato dell'arte. Per loro stessa natura, infatti, gli *smartphones* hanno architetture e vincoli differenti da quelli dei più tipici computer desktop o laptop; ciò riflette la precisione e l'attenzione necessarie per la programmazione di tali dispositivi, sempre focalizzata a rendere migliore l'esperienza utente. (Roth, 2005)

Possiamo suddividere le problematiche sopra citate nelle seguenti categorie:

- **Distanze e banda:** gli *smartphone* moderni devono avere un accesso ad internet costante per garantire il buon funzionamento delle loro caratteristiche più avanzate, l'accesso alla rete avviene tuttavia, in maniera più lenta rispetto ad una connessione via cavo e questo rappresenta un limite. Tecnologie radio come il *WiFi* sono utili, ma possiedono uno scarso raggio d'azione e sebbene gli ultimi protocolli di telecomunicazione 3G / 4G permettano alte velocità di download e upload (99 Mbit/s in down tramite *HSDPA* o 3 Gbit/s download tramite *LTE*), queste sono molto costose e soprattutto consumano molta energia, dato vitale per un *hand-held device*.
- **Consumo energetico:** uno *smartphone*, per essere portatile necessita di una propria batteria che possa mantenere in vita il sistema nei suoi cicli di scarica che accompagnano l'utilizzatore durante la sua giornata. Processori potenti, schermi grandi e sensori sono un nemico pericoloso della vita media di una batteria e se non correttamente gestiti, questi elementi possono portare alla rapida estinzione della carica e generalmente, ad un forte *feedback* negativo dell'utente finale.
- **Interazione uomo-macchina:** Gli schermi, sebbene oggi di grandi dimensioni, tendono comunque ad essere piccoli per l'uso di tastiere virtuali che sono solite dimezzarne le dimensioni rendendo non semplice l'approccio iniziale dell'utente al dispositivo. Oltre a ciò, i metodi di input secondari (come il riconoscimento vocale), non sono ancora perfezionati e richiedono sia tempo che "addestramento" per essere utilizzati a dovere.
- **Interferenze:** La distanza dall'antenna radio più vicina, la morfologia del territorio circostante e le condizioni climatiche avverse possono tutte interferire facilmente con il segnale cellulare. La connessione dati mobile soffre ancora di più questo handicap anche all'interno di edifici particolarmente massicci.
- **Sicurezza:** Quando si utilizza il telefono tramite connessione dati mobile, ci si appoggia trasparentemente alle reti pubbliche che devono implementare un uso meticoloso ed attento dei protocolli di sicurezza per permettere ai nostri dati di viaggiare in maniera sicura ed efficiente.

- **Rischi per la salute:** L'utilizzo improprio di uno *smartphone* può portare a conseguenze irreparabili per la propria ad altrui salute; ad esempio le persone che utilizzano un cellulare alla guida sono statisticamente più coinvolte in incidenti spesso fatali. Ci sono inoltre situazioni in cui questi *devices* sono in grado di disturbare o interferire con altri apparecchi elettronici che invece richiedono grande precisione (alcuni dispositivi medici o i sistemi degli aeroplani).

(Landay, 1993) (Pullela) (Repacholi, 2001)

Tenendo in considerazione queste problematiche derivanti perlopiù dalle caratteristiche fisiche dei dispositivi, dobbiamo aggiungere anche quelle legate alle architetture specifiche dei telefoni. Il telefono oggi è diventato un *device* da cui si esige alta responsività e come tale deve saper rispondere agli eventi di input da parte dell'utente. Contrariamente agli ambienti desktop però i processori montati su *smartphone* e *tablet* sono generalmente meno capaci, a causa della minore potenza di calcolo e devono, perciò, concentrarsi sull'applicazione con cui l'utente sta lavorando attivamente sacrificando, se necessario, altri servizi e processi in background. La memoria RAM di un simile dispositivo (e la sua ottimizzazione) rimane di importanza chiave in quanto determina le capacità *multi-tasking* del sistema ed influenza soprattutto l'esperienza utente nel momento dello *switch* tra applicazione ed applicazione andando ad appesantire o snellire il carico di lavoro del sistema. Anche lo spazio di archiviazione non va sottovalutato, se da un lato, infatti, la presenza di memorie di tipo *flash* (o micro SD esterne laddove fattibile) avvantaggi le performance di uno *smartphone*, la dimensione totale ha un forte impatto considerando che sia il sistema che le applicazioni ne occupano una parte non trascurabile. La crescente multimedialità di questi *devices* inoltre crea una mole di contenuti pesanti che cresce solitamente di giorno in giorno nella memoria di archiviazione (basti pensare a video in HD, fotografie, musica ed altri contenuti) ed a questi si sommano i dati (multimediali o anche solo di *caching*) che le applicazioni scaricano o generano in seguito al loro utilizzo quotidiano da parte dell'utente. (Zimmerman, 1999)

Tutti questi aspetti insieme sono la chiave di una buona esperienza utente e se non gestiti in maniera saggia possono portare uno *smartphone* anche di alta categoria a deteriorarsi in poco tempo con un veloce degrado di prestazioni e spreco di risorse inutilizzate. La programmazione per dispositivi mobile dunque deve necessariamente seguire delle disciplinate linee guida nell'utilizzo di queste risorse e nell'uso sapiente di tutto ciò che può fornire uno *smartphone* moderno.

1.3 I protagonisti della scena dei SO

Attualmente il panorama del sistemi operativi mobile si è ridotto drasticamente, soprattutto dopo il debutto e l'evoluzione dei due ormai più grandi colossi in questo ambito: iOS e Android. Tuttavia all'interno del mercato è ancora presente un certo grado di frammentazione software, dovuto a quei sistemi che in questi anni sono riusciti a rinnovarsi in modo efficace, ma che ugualmente non riescono a raggiungere le quote di vendita di questo duopolio. Analizzeremo quindi le caratteristiche principali di questi due protagonisti del mercato, ponendo particolare attenzione ad Android; ambiente su cui è stata sviluppata l'applicazione.

iOS, è il sistema operativo sviluppato da *Apple Inc.*, scritto in C,C++ ed Objective-C per i molti prodotti della sua linea "*iDevices*", viene ottimizzato e distribuito esclusivamente per hardware Apple. Rilasciato nel 2007 per il primo *iPhone* nel corso dello stesso anno il sistema venne adottato da tutti i vari dispositivi *(iPod Touch, iPad, Apple TV*) e dopo un primo aggiornamento che coinvolse il *re-branding* (da iPhone OS a iOS) venne inserito anche un *App Store* all'interno del SO che permetteva l'espansione dell'ecosistema e lo sviluppo di applicazioni native da parte di sviluppatori esterni. iPhone ed iOS hanno detenuto per molto tempo il record di quantità per applicazioni nel proprio store, ed anche oggi che la bilancia è cambiata in favore del suo concorrente, l'App Store resta comunque più remunerativo per l'azienda di quanto non accada per sua controparte. L'interfaccia utente di iOS è basata sul concetto di interazione diretta con il dispositivo tramite l'uso di *gestures touch* che generano risposte a seconda che l'elemento sia toccato, trascinato, sfiorato o

in generale interagito. Il sistema condivide alcune caratteristiche con la sua controparte desktop OS X come i *framework* principali ma il toolkit della UI (*User Interface*) è di diverso tipo. La specificità di questo sistema lo rende non completamente unix-compatibile come anche di natura *closed source*, la sua licenza è infatti di tipo proprietaria (*EULA*). Lo sviluppo su architetture ristrette e ben conosciute tuttavia pone in vantaggio la struttura del sistema di iOS dandogli una grande velocità, fluidità nell'interazione utente ed un comparto grafico molto solido, accattivante e allo stesso tempo semplicissimo da utilizzare. Apple inoltre continua a stupire con il suo design essenziale e la sua cura per i dettagli, cosa che la rende invitante agli occhi dei consumer, i quali ne fanno spesso molto più di un semplice prodotto. (Apple, 2014) (Hillegass & Conway, 2012).

Android è un progetto che nacque inizialmente nel 2003 per concretizzare la visione di uno dei suoi fondatori "*mobile devices più intelligenti, consapevoli sia della locazione dell'utente che delle sue preferenze*"[4]. Le prime intenzioni di questo progetto erano di applicarlo a sistemi per fotocamere digitali, ma quando i suoi creatori capirono che il mercato in questo settore era troppo piccolo, espansero la loro visione e decisero di portare Android su *smartphone*, creando un sistema operativo al pari di nomi come Symbian e Windows Phone. Nell'agosto del 2005 Android venne acquisita da *Google Inc.* e, sebbene il suo futuro non fosse ancora chiaro, la mossa dell'azienda alimentò le voci di un suo possibile ingresso nel mercato dei telefoni cellulari. Durante il periodo fino al 2007 il team di Android sviluppò un kernel Linux su cui si appoggiava una piattaforma studiata per *device*. Google pubblicizzò tale piattaforma ai produttori di dispositivi mobili, promettendo che avrebbe costruito un sistema flessibile in grado di evolversi per incontrare le esigenze di mercato. Ciò portò l'azienda ad avvicinarsi a molti produttori hardware, partners software e carrier telefonici con una solida proposta commerciale e, nel novembre dello stesso anno, venne rivelato l'*Open Handset Alliance*, un consorzio costituito da 47 membri (tra cui la stessa Google ed anche Telecom Italia), oggi divenuti 84.

[4] "smarter mobile devices that are more aware of its owner's location and preferences" – Andy Rubin 2003

L’obiettivo del consorzio era quello di sviluppare un prodotto dagli standard open per *mobile devices* e nell’ottobre del 2008 fece la sua comparsa l’*HTC Dream G1* con Android 1.0.

Da questo momento in poi Android sperimentò una grande varietà di update e notevoli migliorie nelle proprie performance, nell’utilizzo del dispositivo e nell’esperienza utente. Come per iOS, anche Android fu dotata di un suo *Market* per applicazioni sviluppate da programmatori esterni (successivamente divenuto *Google Play Store*). Il sistema di Android è sviluppato in C,C++ e Java ed è di tipo *open-source*, infatti, per questo motivo la sua popolarità è molto elevata. Oltre ai prodotti consumer principali che utilizzano questo sistema (*smartphones*, *tablet*, *Android TV*, *Android Wear* e *Android Auto*), Android viene anche impiegato in alcune consolle da gioco, fotocamere digitali, computer e altri dispositivi elettronici. Il sistema è adottato spesso anche da quelle compagnie che hanno bisogno di soluzioni già implementate a basso costo, ma che possono essere, allo stesso tempo, altamente personalizzabili. (Google, Android, the world's most popular mobile platform, 2014)

L’utilizzo della politica *open-source* ha permesso anche alla comunità di poter lavorare a progetti proprio sul codice sorgente, includendone nuove funzionalità e caratteristiche prima del tutto assenti. La forza di Android è, inoltre, incentrata sulla quantità: il suo successo deriva dall’applicazione del suo sistema a telefoni di tutte le fasce di prezzo di svariati produttori. Nel 2014 si parla, infatti, di oltre un miliardo di telefoni Android venduti, che posizionano questo sistema ad un market share dell’85%.

2.0 MIRAMBLA MOBILE

MiRambla è un portale web creato per pubblicare, gestire, diffondere e condividere eventi social di ogni genere ed è studiato per venire incontro alle esigenze di qualsiasi utente in cerca di eventi. Il nome del portale è un chiaro riferimento alla famosa strada di Barcellona che collega *Plaza de Cataluña* con il Porto Antico, conosciuta in tutto il mondo per la sua vivacità, i suoi spettacoli a cielo aperto e la sua sana frenesia. Mantenendo vivo il collegamento con la Rambla di Barcellona, gli utenti che all'interno del portale pubblicano i propri eventi sono chiamati *Ramblers* (e possono essere locali, aziende oppure band, squadre, staff, individui, etc). Ogni utente registrato al portale può scegliere quali eventi segnarsi in agenda ed allo stesso tempo quali *Ramblers* decidere di seguire, in questo modo è l'utente stesso a crearsi la *propria* rambla. Il nome del portale MiRambla, tradotto come la "*mia rambla*", costituisce una grande novità rispetto agli altri siti che promuovono eventi grazie alla possibilità di seguire i *Ramblers*, e di conseguenza di visualizzare in modo automatico tutti gli eventi che loro stessi pubblicano. Tutti i *Ramblers* sono rintracciabili tramite *nickname* univoco e un proprio profilo caratterizzato da un'immagine, una descrizione e dati per i contatti. Inoltre può essere arricchito con ulteriori immagini e video, in modo da creare una vera e propria pagina di presentazione.

Prima di affrontare la progettazione dell'idea pensata per questo portale verranno introdotte delle nozioni sulla crescente tendenza sociale mobile; quest'ultima ha portato, nel corso dell'ultimo decennio, all'esplosione dell'utilizzo dei siti-social ed al successo dei social-network nella vita di tutti i giorni.

2.1 Il Social nel mondo mobile

Il Web negli ultimi anni ha trasformato la vita della maggior parte delle persone. Coloro che iniziano ad accedere ad internet attraverso l'infrastruttura informatica difficilmente tornano sui propri passi, e al contrario spendono una quantità di tempo on-line sempre maggiore. Come per le linee telefoniche, anche internet non fu creato inizialmente con lo scopo di essere uno strumento per la comunicazione *social*, ma si evolse inaspettatamente per divenire parte essenziale della vita quotidiana. L'interazione sociale attraverso il web, tuttavia, è stata da sempre un argomento di grande successo. Una testimonianza di ciò è costituita dall'ammontare di software *social* esistiti e da quelli in uso tutt'ora, alla cui base vi è l'idea di poter far connettere chi ha già legami nella vita di tutti i giorni. Il Web sociale è sviluppato in 3 fasi, dagli anni '90 fino ad oggi. Il "vecchio" *Web 1.0* era il primo stadio di un complesso sistema di trasporto dati su scala globale in cui l'interazione dell'utente era molto limitata: comunicazioni monodirezionali, pagine con contenuti statici disponibili solo per la consultazione. Il flusso informativo di questi siti era passivo, direzionato esclusivamente dal proprietario (dall'organizzazione o dal privato) delle pagine, la comunicazione end-to-end era difficile e solo pochissimi siti implementavano il sistema di post con possibilità di *reply* sottostante. (Howstuffworks) (Kima, 2010).

Verso la metà degli anni '90 *Amazon* ed altri pionieri fecero grandi passi in avanti nelle tecnologie online, scoprendo come utilizzare i loro database, connettendoli ai propri siti per memorizzare informazioni e contenuti, i quali potevano essere poi rielaborati. Questa ed altre innovazioni permisero la creazione di pagine interrogabili ed applicazioni web che potevano implementare finalmente la comunicazione in entrambi i sensi (user - proprietario). Grazie alla rapida ascesa di popolarità del web, della banda larga e del crescente interesse delle persone, fu chiaro che, per evolversi completamente, era necessario dare la possibilità agli utenti di connettersi con gli altri utenti che navigavano i siti. Anche se tecnicamente era soltanto un piccolo *step* da affrontare, socialmente parlando fu un grande passo in avanti, permettendo l'interazione tra gruppi per la prima volta. E' stato inoltre affermato che proprio questa possibilità di scambio sociale tra i molti utenti è ciò che distingue una *web application*

da una *social web application*. Nel periodo più fertile di questa evoluzione, il creatore del Web *Sir Tim Berners-Lee scrisse*:

"The Web is more a social creation than a technical one. I designed it for a social effect—to help people work together—and not as a technical toy. The ultimate goal of the Web is to support and improve our weblike existence in the world. We clump into families, associations, and companies. We develop trust across the miles and distrust around the corner. What we believe, endorse, agree with, and depend on is representable and, increasingly, represented on the Web"

la cui traduzione è:

"Il Web è più un'innovazione sociale che un'innovazione tecnica. L'ho progettato perchè avesse una ricaduta sociale, perchè aiutasse le persone a collaborare, e non come un giocattolo tecnologico. Il fine ultimo del web è migliorare la nostra esistenza reticolare nel mondo. Di solito noi ci aggreghiamo in famiglie, associazioni ed aziende. Ci fidiamo a distanza e sospettiamo appena voltato l'angolo. Quello che crediamo, proponiamo, accettiamo e da cui dipendiamo è rappresentabile ed è sempre più rappresentato dal Web stesso"

Questa frase può far bene intendere come internet sia mutato proprio per incontrare sempre di più l'esigenza della comunicazione umana. Con l'avvento del *Web 2.0* queste tecnologie hanno aperto le porte a quelli che oggi sono tutti i contenuti social dei vari siti e network che popolano internet (*Facebook, Twitter, Instagram, Foursquare* ma anche siti di eventi come *MiRambla* etc.), evidenziando l’epocale cambiamento che ha portato l'adozione di queste reti nel quotidiano. I contenuti dei siti/pagine/applicazioni che utilizziamo oggi da computer e *smartphone* si sono tanto focalizzati sulla figura dell'utente da non essere più un mero "*service*" da contattare in un server distante, ma sono *commodities* alla portata di tutti e socialmente accettate. Questo salto nel futuro ci permette oggi di vivere a stretto contatto con chi è distante e ci lascia sbalorditi sulla velocità con cui possiamo accedere al mondo circostante con dei semplici tocchi o cercando poche parole chiave sui nostri interessi. Tramite gli *smartphone* possiamo connetterci in tempo brevissimo con tutti coloro di cui abbiamo bisogno e tale *device*

unisce la nostra *social life* in un modo che si personalizzi ai nostri gusti e peculiarità quasi automaticamente. (Andreas M. Kaplan, 2010), (Berners-Lee, 2006) (DiNucci, 1999), (Graham, 2005).

Questo è il social nel mondo mobile di oggi.

2.2 L'Idea ed il Progetto

L'idea di un'applicazione che gestisse eventi deriva da un'analisi condotta circa tre mesi fa nel *Play Store* di Google, dove si poteva constatare che le applicazioni per eventi, sebbene non fossero tante, difficilmente incontravano le aspettative degli utenti. Tralasciando il problema che molte delle applicazioni presenti nello *store* riportavano scarsi contenuti in termini di eventi, i primi risultati ottenuti digitando come keywords di ricerca "*eventi*" o "*events*" risultavano molto macchinosi e non immediatamente semplici da utilizzare. Vi sono poi molteplici esempi di applicazioni sbrigative, create senza un'adeguata progettazione, oppure dotate di eccessivo contenuto. Tali problematiche coinvolgono l'intera interfaccia grafica tanto da rallentare l'esperienza utente al punto che quest'ultimo deve sforzarsi non poco per distinguere le innumerevoli icone che popolano la mappa. L'aspetto critico, tuttavia, rimane la non fluidità durante l'utilizzo della maggior parte delle applicazioni disponibili. Come si è detto, infatti, le prestazioni sono un dato particolarmente importante nelle applicazioni mobile, soprattutto in quelle social dove ci si aspetta un risultato visibile e corretto subito dopo la nostra interazione con il dispositivo. Un buon numero di queste applicazioni, anche di buona riuscita (sia con ottima grafica che utenza), soffrono inequivocabilmente di lacune nella fluidità, causate probabilmente da un errato impiego del codice e dei costrutti di programmazione mobile.

L'idea di avere un'applicazione *mobile social* fluida ha costituito una buona base di partenza per il progetto insieme alla collaborazione con il portale di MiRambla, scoperto grazie al Professore. Insieme si maturò, dunque, l'ipotesi di un progetto per un app che permettesse al sito di espandere il suo utilizzo dedicato anche su

piattaforma *mobile*, con l'intento di creare uno strumento valido, responsivo e semplice da utilizzare su dispositivi Android per la *community* dei futuri *Ramblers*. Inoltre, poiché si basa su un progetto realistico di applicazione, il presente lavoro si discosta da un semplice lavoro accademico, avvicinandosi all'ambito professionale e fornendo l'opportunità di comprendere le possibili esigenze dirette del cliente, al quale spetta la valutazione finale del prodotto.

La linea guida quindi è quella di replicare le funzioni principali del portale MiRambla in un applicazione nativa per Android 4.1 (*Jelly Bean*) e superiore. La scelta di un *API level* così alto rende lo sviluppo, almeno per l'immediato futuro, esclusivo per una buona parte dei *device* Android (circa l'81% dei dispositivi attivi), ma permetterà di utilizzare elementi e costrutti più recenti della piattaforma come anche *smartphone* sempre più nuovi che, tipicamente, presentano anche meno problemi.

Version	Codename	API	Distribution
2.2	Froyo	8	0.6%
2.3.3 - 2.3.7	Gingerbread	10	9.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	8.5%
4.1.x	Jelly Bean	16	22.8%
4.2.x		17	20.8%
4.3		18	7.3%
4.4	KitKat	19	30.2%

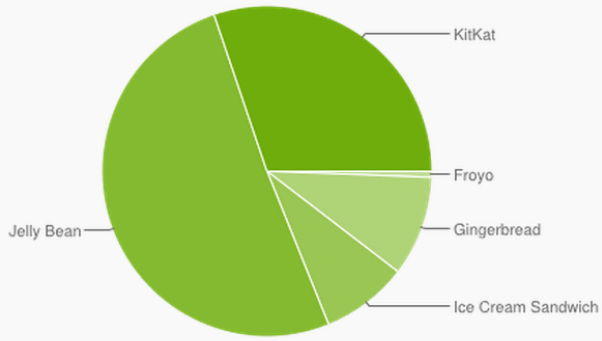

Figura 3 - Diffusione distribuzioni Android (Novembre 2014)

Come ambiente di sviluppo verrà utilizzato *Android Studio*, il nuovo IDE di casa Google che pur essendo ancora in beta, offre un ambiente potente ed adeguato ai bisogni degli sviluppatori. Basato su *IntelliJ IDEA*, l'ambiente porta nuove funzionalità e migliorie rispetto all'utilizzo dell'*ADT* su *Eclipse* ed è destinato a diventare lo strumento ufficiale di sviluppo una volta completato. Android Studio è basato sul *project automation tool Gradle*, che effettua il *build* similmente ad *Apache Ant* e *Apache Maven* ma introduce un *Domain Specific Language* (DSL) invece del tipico XML per dichiarare le configurazioni di progetto. (Pitoura, 1998).

2.3 Analisi

Nel prossimo paragrafo verrà affrontata l'analisi del progetto, iniziando da quella del prodotto online (il portale MiRambla), da cui dedurremo le caratteristiche principali che andranno collocate nell'applicazione. In seguito osserveremo quali sono i requisiti di tali caratteristiche da implementare lato app. Seguirà poi la progettazione vera e propria, dove, tramite uno strumento di *mock-up*, si darà un'idea della bozza implementativa, utile per facilitare il passaggio tra i requisiti iniziali e la programmazione vera e propria.

Il portale **MiRambla** si presenta così:

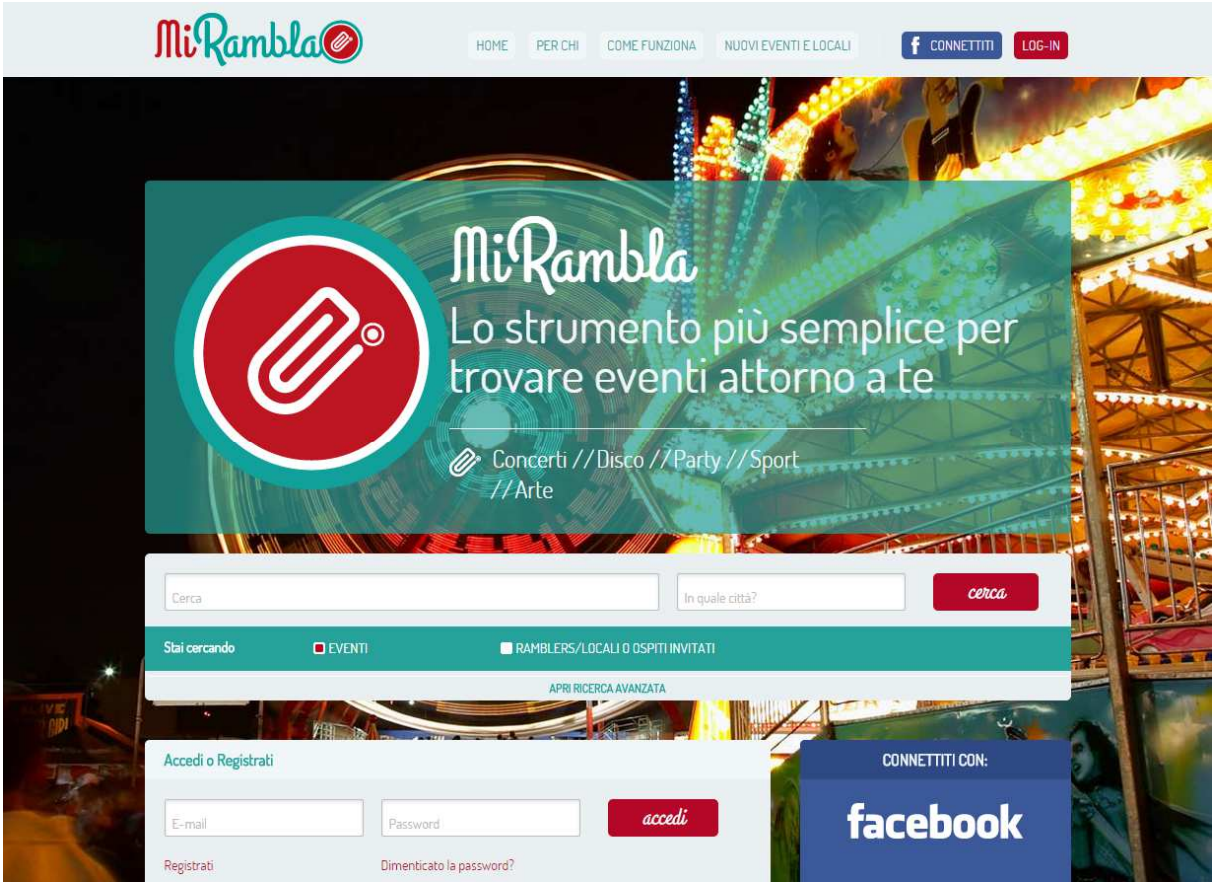

Figura 4 - Home Page MiRambla pt.1

Nella parte superiore dello schermo è presente la barra di navigazione del sito e i bottoni che permettono il log-in tramite dati o *Facebook*. Nella parte centrale prendono posto la barra di ricerca degli eventi (espandibile per ricerche avanzate), il pannello di accesso dati e la registrazione.

Il resto del portale si compone nel seguente modo:

Figura 5 - Home Page MiRambla pt.2

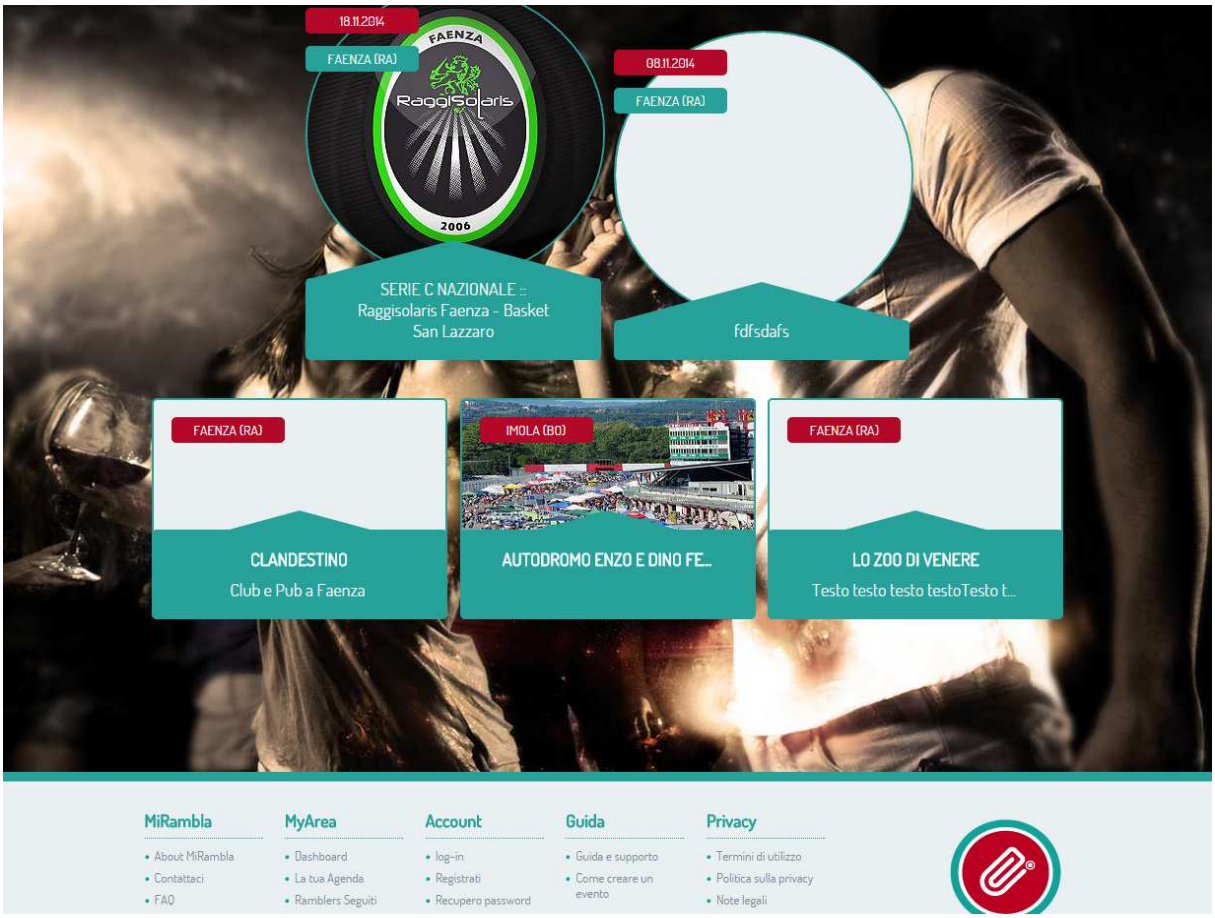

Figura 6 - Home Page MiRambla pt.3

Nel corpo del sito è presente uno *slider* a contenuto variabile con le descrizioni sintetiche del portale insieme a varie informazioni riguardanti l'utilizzo online o *mobile*. Nella parte inferiore, invece, è presente una vetrina degli ultimi eventi aggiunti (basati sulla posizione approssimativa del client), insieme ai vari locali in cui possono avere luogo gli eventi.

L'utilizzo del sito risulta abbastanza semplice ed intuitivo; ad esempio, effettuando una rapida ricerca nella barra possiamo ottenere il seguente risultato:

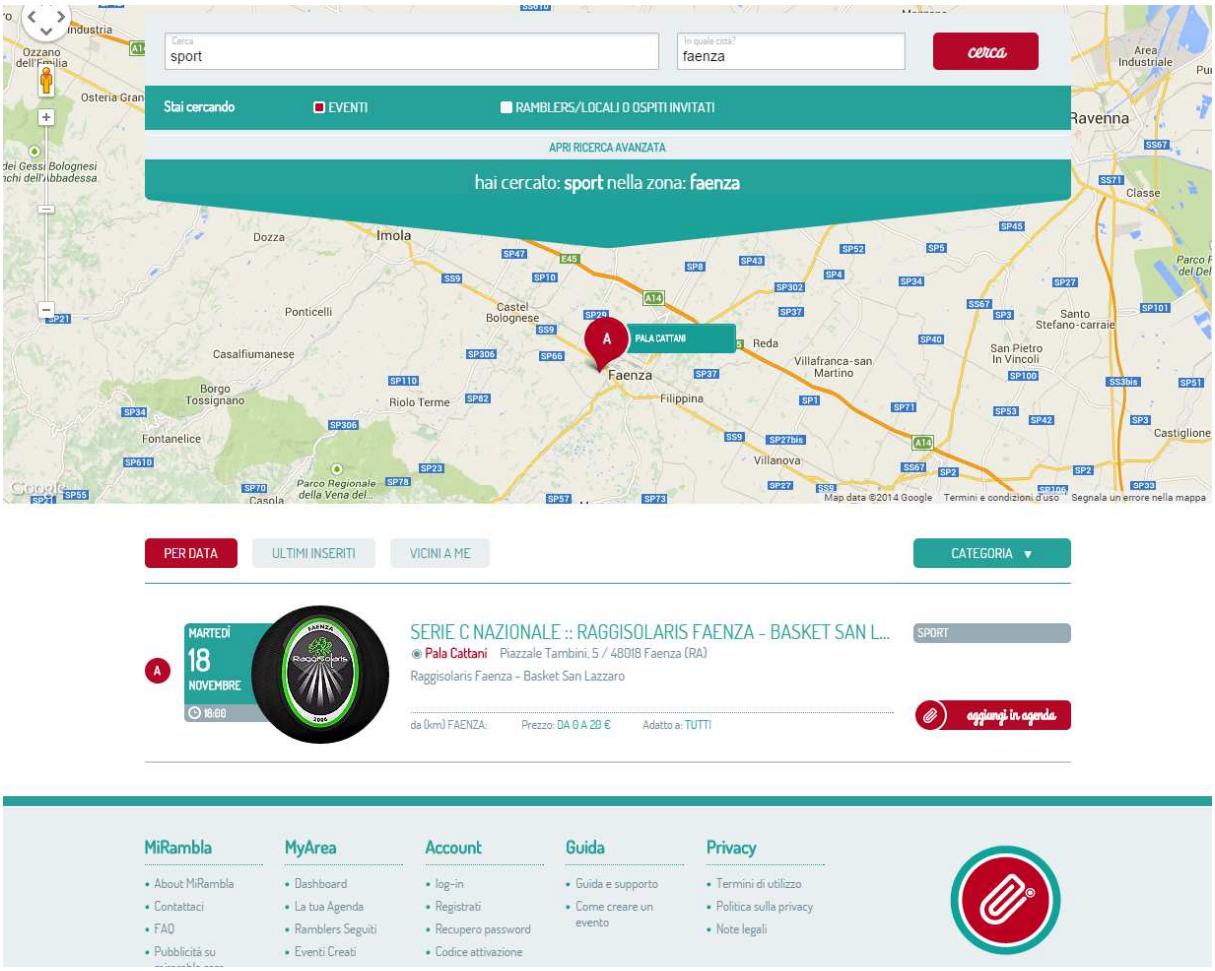

Figura 7 - Risultato di una ricerca

Navigando sul sito ed esplorando le voci di menù, si possono dedurre le sue funzionalità principali. A queste vanno aggiunte quelle ottenute dopo l'operazione di registrazione, che incrementano il livello di dettaglio e soprattutto permettono la creazione di un'agenda personale di eventi. Infine la registrazione consente anche l'utilizzo della parte di integrazione social del portale che verrà esaminata in seguito.

Riassumendo le funzionalità base del sito sono:

- ricerca eventi per keyword e città + altri filtri
- possibilità di registrazione
- possibilità di registrazione con *Facebook*
- ricerca di locali
- visualizzazione di Eventi

- visualizzazione Locali
- visualizzazione *Rambler*

Le funzionalità avanzate invece sono:

- creazione / modifica dell'Agenda eventi
- creazione / modifica profilo *Rambler*
- seguire altri *Ramblers*
- creazione / modifica Evento
- creazione / modifica Locali
- affiliazione a Locali

Tutte queste caratteristiche dovranno entrare a far parte delle possibilità che un applicazione mobile deve fornire, in aggiunta alla possibilità di incentrare le ricerche automaticamente vicino alla nostra posizione attuale (lo *smartphone* permette infatti la geolocalizzazione). E' normale individuare quindi un subset di queste funzionalità che andranno a costituire la versione base dell'applicazione, ovvero quelle di cui non possiamo fare a meno per adempiere alle funzioni di base.

Tenendo conto di quanto detto nella visione d'insieme che abbiamo dato, delle possibilità di linguaggio e di piattaforma, possiamo preporci a grandi linee degli obiettivi che ci aspettiamo di raggiungere nei diversi *step* di sviluppo dell'applicazione. Tali obiettivi devono tenere anche conto della capacità evolutiva del progetto e dell'app in sé, in modo che le successive implementazioni possano essere fatte incrementalmente e non sostituendo il futuro codice. Distingueremo in versione di alto livello il lavoro che il progetto richiede basato sulle funzionalità inserite.

- **1° versione**:
    - lista degli eventi circostanti ordinati sulla distanza
    - ricerca di eventi
    - visualizzazione dettaglio eventi

- **2° versione**:
  - possibilità di log-in
  - visualizzazione dettaglio locali
  - visualizzazione dettaglio *Rambler*
  - ricerca dei locali / *Rambler*
- **3° versione**:
  - possibilità di registrazione / registrazione con *Facebook*
  - gestione profilo
  - gestione agenda eventi
  - creazione / modifica eventi
  - creazione / modifica *Rambler*
- **4° versione**:
  - creazione / modifica / affiliazione Locali
  - implementazione social dinamica
  - gestione *Rambler follower / followed*
  - navigazione libera su mappa

Questa *roadmap* di ***versioning*** è incrementabile sia orizzontalmente che verticalmente e tiene conto anche delle possibilità offerte dal portale. L'inserimento di nuove funzionalità sul portale permetterà di perfezionarle anche sull'app, in quanto la parte server già espone servizi facilmente interrogabili che coprono tutte le esigenze della parte *mobile*. Le prime versioni dell'applicativo sono state mantenute più snelle per bilanciare la difficoltà iniziale rappresentata dalla creazione del progetto da zero e dalla necessità di implementare tutti i layout e gli elementi di base, utilizzabili negli step successivi. Non solo, durante il capitolo riguardante la progettazione assumerà grande rilevanza l'esame del *branding* stesso di MiRambla: identà social, iconografia, colori, font, impatto nell'utilizzo e resa grafica. Si tratta di dettagli che un'app deve possedere per rappresentare un'estensione del portale stesso e non, al contrario, un elemento stand-alone nel quale i dati vengono presi dal web e rielaborati localmente nel *device*.

Le esigenze imposte dai singoli elementi, negli step successivi, richiedono tecnologie e soluzioni differenti per essere implementate. Possiamo iniziare osservando che il contenuto web in transito dal server ai vari client è principalmente di tipo testuale o di immagini. Solo successivamente, si potrebbero aggiungere anche contenuti multimediali più pesanti che in questo contesto non verranno tenuti in considerazione. Per la buona gestione delle immagini, elemento chiave di un'applicazione social, si valuterà l'impiego di una libreria esterna mentre per il contenuto testuale non ci sono particolari considerazioni da fare in questa sede.

Lo scambio dati tra il client ed il server avviene utilizzando lo standard **JSON** (*JavaScript Object Notation*), una tipologia di formato per l'interscambio dati di tipo open, facilmente comprensibile allo sviluppatore. La risposta JSON è costituita da un blocco dati racchiuso tra parentesi graffe **{ … }**, al cui interno prendono posto coppie non ordinate di tipo "*chiave : valore*" che possono essere aggregate in una o più comunicazioni. Tali coppie possono essere anche disposte ordinatamente all'interno di vettori, racchiusi tra parentesi quadre **[…]** e separate dalla virgola. La comodità di questo standard risiede nel fatto che esso è indipendente dal linguaggio sottostante utilizzato ed è quasi universalmente utilizzabile, in quanto rende i dati serializzati. La sua nascita è dovuta alla rapida evoluzione degli strumenti Web 2.0 e deriva dal linguaggio *JavaScript*, conosciuto anche per la comodità delle sue chiamate asincrone al server. I principali tipi di dati rappresentabili in JSON sono:

- *Number*: numeri decimali segnati, possono contenere anche una parte in frazione o usare l'esponenziale in notazione E.
- *String*: una sequenza formata da zero o più caratteri Unicode tra virgolette.
- *Boolean*: inteso come "true" o "false".
- *Array*.
- *Object*: un vettore disordinato ma associativo di dati (chiave/attributo) annidabili tra di loro e racchiusi tramite caratteri speciali (parentesi, virgole o due punti).
- *null*: un valore vuoto, codificato dalla parola stessa "null".

I tipi che JSON mette a disposizione possono sembrare pochi, in realtà questo è uno dei suoi punti di forza. Ad esempio la semplicità del JSON, infatti, restringe la sua struttura rendendolo snello e maneggevole al contrario di XML, dove esistono una moltitudine di tag che possono appesantire anche di molto il corpo del documento.

Le richieste al server di MiRambla vanno inoltrate tramite *GET* del protocollo *HTTP*, trasmettendo nell'URL alcune informazioni aggiuntive basate sul tipo di pagina e servizio al quale vogliamo accedere e/o intervenire. Tutti i servizi vengono quindi chiamati secondo la seguente logica:

"http://dominioMiRambla.com/[lingua]/services/[nome del servizio]"

Il [nome del servizio], o le relative *subdirectory* sono l'insieme delle pagine .php che rispondono in funzione del contesto richiesto. Le risposte seguono una logica JSON parametrizzate a livello di successo, sessione, contesto e data. Per l'utente *loggato* all'interno del sito si avranno alcuni parametri aggiuntivi che lo distingueranno dall'utente in modalità *guest* e gli permetteranno di accedere alle funzionalità avanzate.

Faremo ora delle brevi ipotesi di analisi, *lato Client*, che interesseranno perlopiù la parte dei costrutti utilizzati per implementare l'applicazione. All'attuale stato dell'arte Android ha raggiunto la sua *API level 21* con la recente uscita (12 Novembre 2014) di Android 5.0 *Lollipop*; il progetto qui presentato verrà compilato e testato su questa distribuzione. L'utilizzo di quest'API permetterà, in futuro, di ampliare il comparto grafico dell'applicazione verso il nuovo design chiamato *"material design"*, introdotto da Google stessa al suo I/O di Giugno e a cui *MiRambla Mobile* già si ispira. Il minimo API level specificato, come già accennato, è il 16 (4.1 *JellyBean*), che consente di catturare una buona dose di dispositivi dotati di hardware di ultima generazione. A livello di costrutti, l'applicazione verrà dotata di *Fragment*, che dinamicamente costituiranno l'insieme delle *View*, quest'ultime mostreranno poi il contenuto dell'app. Questi elementi verranno introdotti per spianare la strada verso il futuro sviluppo di un layout dedicato per tablet e dispositivi dagli schermi ampi. L'impiego dei *fragment* consente il riutilizzo grafico di elementi comuni all'interno

dell’applicazione in modo da facilitarne il riuso. Per la navigazione, l’app sarà dotata di un *NavigationDrawer*, elemento che caratterizza ormai tutte le applicazioni con un design moderno. Questo elemento è richiamabile da tutti i punti di navigazione dell’applicativo, rendendone l’esplorazione un’operazione naturale ed istintiva.

2.4 Progettazione

La progettazione di *MiRambla Mobile* è iniziata con l’intento di mantenere, per quanto possibile, il *branding*, l’identità del sito e del marchio. Considerato che la grafica è un aspetto fondamentale per un app social, occorre rispettare le regole di stile viste sulle anteprime del sito presentate nelle pagine precedenti. Le colorazioni principali rosso bordeaux, azzurro, bianco e grigio chiaro saranno predominanti negli elementi grafici finali quali container, bottoni, barre e *TextBox*. Il colore verrà utilizzato principalmente per dare enfasi. L’impiego dell’alto contrasto con palette non esageratamente vivide rende l’applicazione calda al contatto visivo e rientra nei canoni grafici delle linee guida dettate dall’*Android Design*. Occorre dunque stendere un layout per l’activity principale dell’app che possa essere in linea con quanto visto in ***Figura 7*** (risultato della ricerca eventi sul sito), questo per due motivi. Per prima cosa è necessario mantenere la congruenza dello stile con il sito; anche i singoli elementi grafici interni dovranno rievocare il feeling d’utilizzo dello stesso servizio on-site. In secondo luogo la pagina Home dell’app dovrà assomigliare ad una lista di possibili risultati ottenuti mediante ricerca di eventi sulla posizione attuale dell'utente.

Tramite il tap su un qualsiasi punto della cella di un evento si andrà a visualizzarne il dettaglio, caricato tramite *fragment*. Anche il layout del *fragment* dettaglio va pensato considerando il design di MiRambla. Come visto nella vetrina in ***Figura 6,*** gli eventi vengono racchiusi in una cerchio dove alloggia l’immagine dell’evento in questione accompagnato da due *TextBox* colorate che indicano data e locazione. Al di sotto dell’immagine si incastra un piccolo oggetto grafico azzurro contenente il titolo. Questi aspetti saranno replicati il più possibile nel dettaglio dell’evento, insieme ad altri campi che saranno analizzati in breve una volta esaminata la struttura del dettaglio

contenuto nel portale. (Google, Android Design Principles, 2014), (Google, Android Developers Index, 2014).

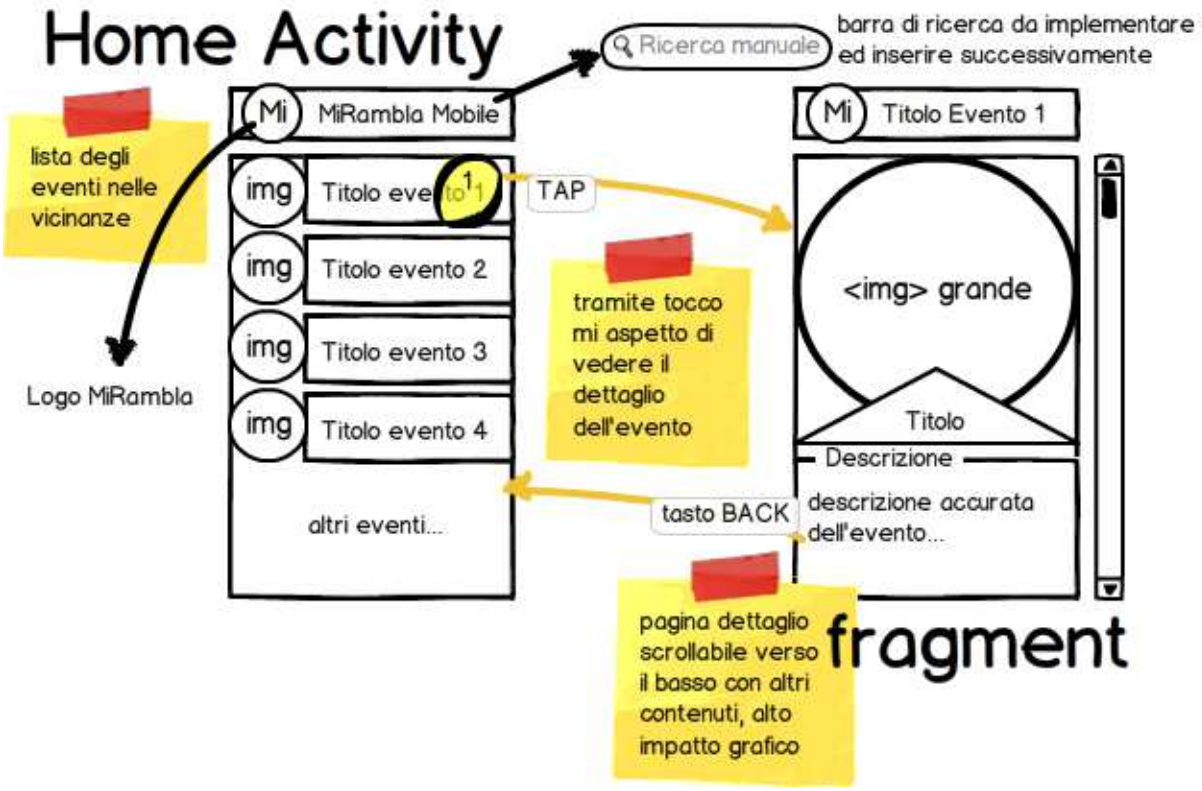

Figura 8 - mock-up dei due layout e della loro navigazione

La Home *activity* di *MiRambla Mobile* implementerà una *ListView* suddivisa in rows caratterizzati a loro volta da un *RelativeLayout* che va a specificare il posizionamento dinamico dei propri sottoelementi (stub di anteprima, titolo, data e luogo). La *home screen* si popolerà autonomamente all'avvio dell'applicazione basandosi sulla quantità di eventi trovati nelle vicinanze, oppure in generale, fino al riempimento della dimensione dello schermo del *device*. Oltre il confine fisico dello schermo la *ListView* ha la comodità di diventare scrollabile automaticamente. Nella parte alta dello schermo prenderà posto una *ActionBar* con il titolo "brandizzato" MiRambla ed alla sinistra di questo il tasto per aprire il *NavigationDrawer*, che *sliderà* dal margine sinistro dello schermo. Naturalmente questo arrangiamento è puramente progettuale, verrà implementato e corretto durante lo sviluppo, ma la sua importanza in questo step è

quella di avere chiara la resa grafica finale *prima* di mettersi al lavoro sul codice XML. Ciò consentirà di non incorrere in indecisioni e dubbi stilistici che renderebbero il procedimento sensibilmente più lento.

Un tipico dettaglio di evento sul portale MiRambla, si presenta nel seguente modo:

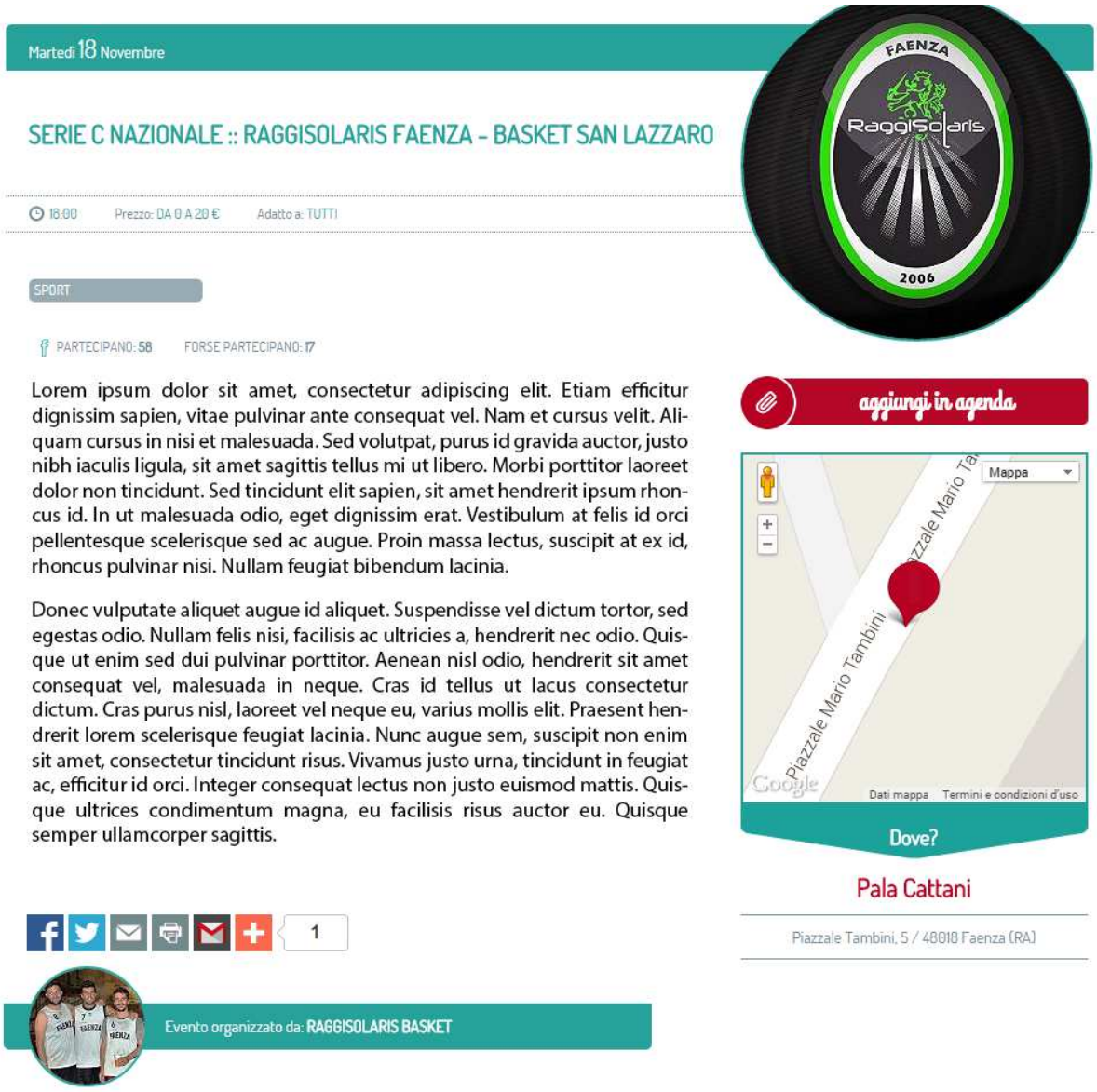

Figura 9 - Dettaglio di un evento

Iniziando dall'alto abbiamo una barra contenente informazioni sulla data, seguono parimenti il titolo sulla sinistra e l'immagine a destra; un piccolo divisore indica alcuni dettagli riguardanti l'ora, il prezzo e la portata dell'evento. Sotto di questi, prende posto un banner che indica la categoria dell'evento e le informazioni sui partecipanti (ottenuti dalla connessione dinamica dell'evento su *Facebook*). Il resto della pagina è abbastanza chiaro: descrizione del testo, mappa del luogo, indirizzo e barra per l'innesto dell'evento sulle varie piattaforme *social*. Le parti più importanti sono il *Rambler* di riferimento (l'organizzatore dell'evento), indicato nella barra in basso e il tasto "Aggiungi in Agenda" che esprime una funzionalità avanzata che richiede il login.

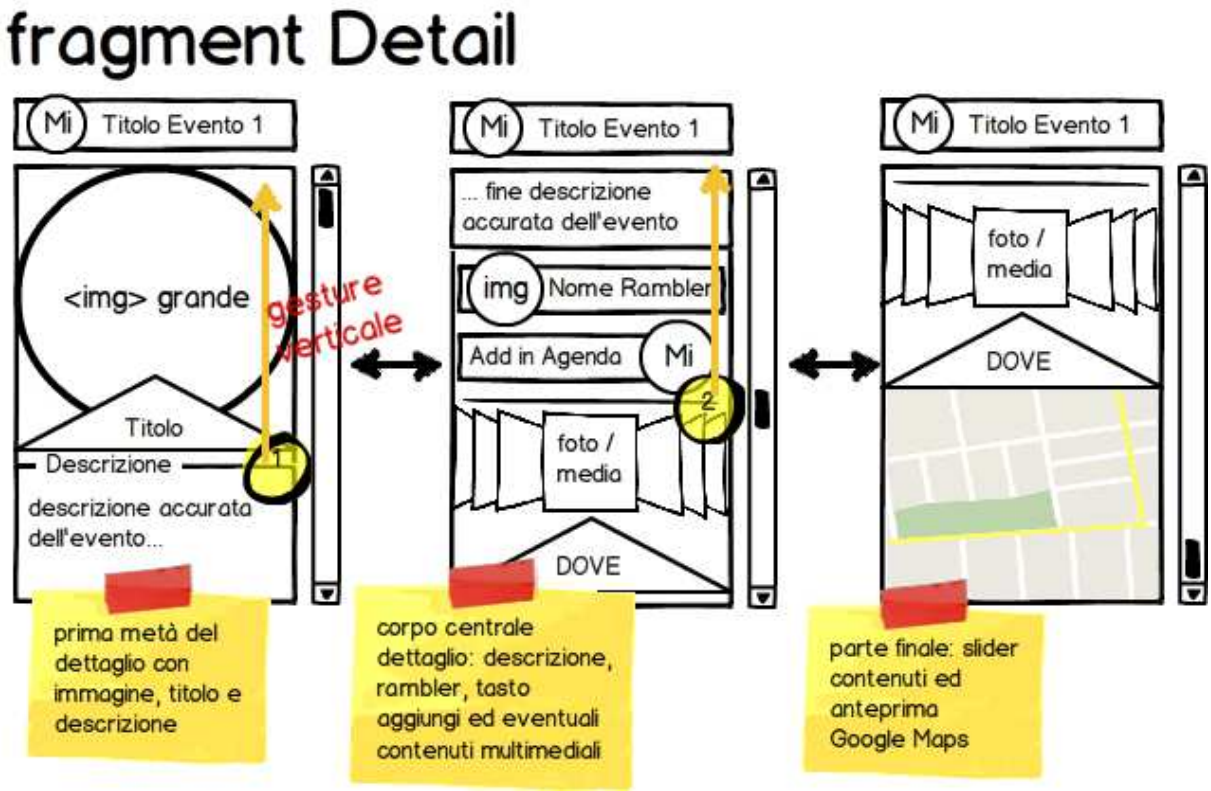

Figura 10 - layout completo fragment dettaglio

Da questo *mock-up* è evidente che il dettaglio dell'evento sarà più complesso da realizzare della *activity* precedente. La parte riguardante i dati non ha un particolare peso ma una volta entrati nel dettaglio serviranno i campi della medesima "istanza evento". L'interazione grafica dei componenti, il loro posizionamento ed il

comportamento individuale, invece, sono elementi particolarmente significativi in questa fase, più che nella precedente, dato che il layout è fortemente *customizzato*. Per implementare questo *fragment* sarà necessaria una *ScrollView* che renderà scorribile dinamicamente tutto il contenuto una volta che viene superata l'eccedenza dello schermo in uso. *TextBox* quali titolo, descrizione e nomi vari sono di facile implementazione; l'anteprima per i contenuti multimediali invece è un aspetto avanzato che sarà rimandato alle versioni successive dello sviluppo. Infine, nell'ultima parte, sarà implementata la *MapView* dell'API di *Google Maps*, strumento molto potente, che richiederà un ulteriore sforzo per l'inserimento dell'anteprima localizzata con il marker della loczione. Una volta entrati nel dettaglio, ovviamente, è sempre possibile tornare indietro verso la *Home Activity* premendo il tasto "back".

L'ultimo elemento necessario per giungere alla nostra prima versione utilizzabile è il *NavigationDrawer*, oggetto che permette la navigazione indipendente verso ogni punto dell'app.

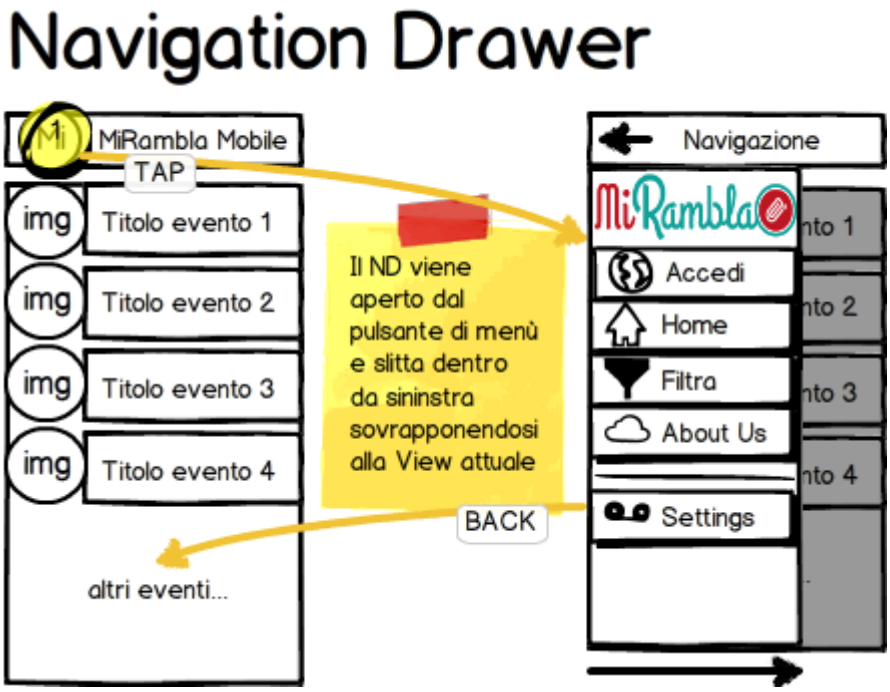

Figura 11 - layout e funzionamento del Navigation Drawer

Tale *drawer* sarà composto da una serie di voci di menu che permettono sia la navigazione alle diverse parti dell'app, sia l'accesso per l'utilizzo delle funzionalità avanzate che richiedono ad esempio, il login. Inoltre posizionare gli elementi in questo modo è anche un'operazione strategica. Anche se nella versione iniziale ci saranno poche voci all'interno del pannello e quest'ultimo risulterà un po' vuoto, con l'inserimento di nuove *activity/fragment* la lista andrà ad aumentare e risulterà immediato richiamarle grazie a quest'oggetto.

La ***versione 1.0*** così progettata di *MiRambla Mobile*, rispecchia la scaletta che abbiamo dato nel paragrafo di analisi ed essa può già funzionare. Visti gli obiettivi prefissati però, vogliamo effettuare anche la progettazione dell'*activity* inerente al log-in, che si renderà necessaria nell'immediato futuro.

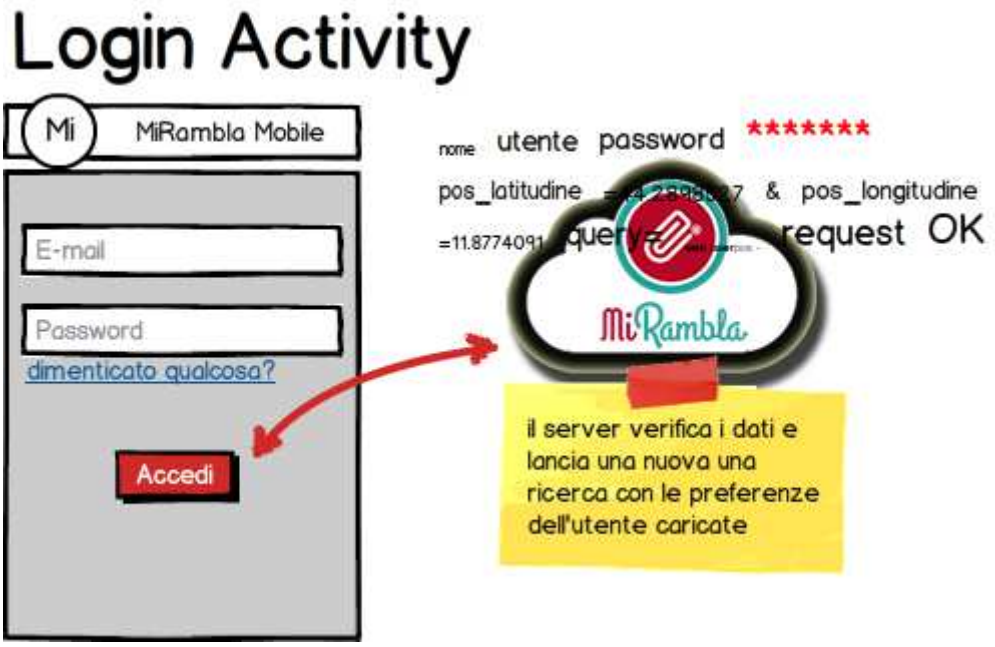

Figura 12 - esempio di comunicazione client-server

Il login per l'area riservata è un elemento cruciale di una app *mobile*, permette infatti il caricamento dei contenuti personalizzati da parte dell'utente e la sua "sincronizzazione" con la propria istanza dell'app sul dispositivo. Questa peculiarità darà anche l'accesso a tutta quella serie di funzionalità avanzate elencate

precedentemente, tra cui anche quelle social come l'integrazione con le altre piattaforme (*Facebook*, *Twitter*, *Google+*, etc.). Lato server il login viene gestito tramite un servizio residente su una pagina apposita .php che risponde alle richieste passate con i parametri di accesso (*user*, *password*, tipo di *device*, latitudine, longitudine, sessione). In caso di login autorizzata il server elaborerà i dati e restituirà una risposta JSON del tipo:

```
{
     "success": true,
     "message": "",
     "sessionid": "sessionid utente",
     "data": {
          "user": {
               "id": "",
               "nickname": "",
               "avatar": "",
               "nome": "",
               "cognome": "",
               "sesso": "",
               "data_nascita": ""
          }
     }
}
```

Il parametro "data" in questo caso è indicativo: potrebbero esserci anche altre informazioni al suo interno in funzione del tipo di utente e dell'evoluzione del portale. Altri servizi del portale MiRambla sono interrogati e rispondono seguendo lo stesso paradigma; sarà opportuno implementare dunque le varie richieste con i giusti campi da inserire. I servizi che *MiRambla Mobile* arriverà ad utilizzare comprendono:

- Login / Logout
- Registrazione / Push service (richiede login)
- Recupero Password (richiede login)
- Conferma attivazione account
- Carica / modifica dati del profilo (richiede login)
- Carica / modifica avatar (richiede login)
- Elimina account (richiede login)
- Richiesta pagine informative (privacy, termini, condizioni etc.)

3.0 SVILUPPO

Il paragrafo relativo allo sviluppo sarà articolato in diversi sottoparagrafi che, a loro volta, sono disposti come le *folder* del progetto di MiRambla Mobile. Ogni *folder*, infatti, al suo interno contiene uno o più file di codice che verranno elencati nei paragrafi seguenti allo stesso modo in cui sono ordinati nella cartella di progetto: *activities*, *adapters*, *fragments*, *models* e *utils*. Inoltre seguirà un paragrafo dedicato ad esplorare le rimanenti risorse di MiRambla Mobile. Ciascun sottoparagrafo descrive in dettaglio un particolare file; in questo modo si vuole inizialmente spiegare il costrutto in questione, dando una delucidazione sul suo funzionamento e scopo all'interno dell'SDK[5] di Android per poi descrivere accuratamente la parte di codice utilizzata. In questo contesto non si specificheranno i dettagli implementativi, che potrebbero risultare verbosi e non necessari.

In generale, la struttura dei file all'interno di queste folder di progetto è suddivisa in *package java* fino al livello di cartella. Nella cartella "*src*" risiedono tutti i sorgenti *.java*, i file di interpretazione del layout in formato XML, le varie parti del layout grafico (i cosiddetti *drawable*, immagini create ad hoc e suddivise per densità dello schermo), i file di configurazione e l'*Android Manifest*. Il Manifest è un documento XML che riassume le informazioni essenziali sull'app e le presenta al sistema; tali informazioni sono necessarie per far si che Android autorizzi l'app ad essere eseguita. Tra le informazioni principali del Manifest abbiamo:

- Nome del package java dell'applicazione: tale nome serve ad identificare univocamente la nostra app sul *Play Store*.
- Descrizione dei componenti nell'app: *activities*, *services*, *broadcast receivers*, *content providers* ed i nomi delle rispettive classi in cui sono implementati.
- Nome del processo che ospiterà i componenti applicativi.

[5] *Software Development Kit*

- Permessi speciali (internet, wi-fi, fotocamera, etc.) che l'app richiede per eseguire il suo lavoro o per contattare altre applicazioni.
- Permessi che le altre app devono avere per interagire con questa.
- Minimo livello di SDK Android necessario per l'applicazione.
- Lista di librerie con cui l'applicazione deve essere compilata.

3.1 Activities

L'*Activity* è un componente applicativo che permette all'utente di avere un oggetto disegnato sullo schermo con il quale interagire per compiere un'azione come: digitare numeri sul tastierino virtuale, scattare una foto, inviare una e-mail o altro ancora. Ad ogni Activity è assegnato uno spazio apposito su cui disegnare il proprio layout, tipicamente il contenuto riempie tutto lo schermo ma vedremo casi specifici dove non sarà sempre così.

Un'app generalmente è costituita da diverse *activities* che a turni alterni occupano lo schermo del *device* con il loro contenuto. L'*Activity* principale di ogni applicazione è quella detta "Main", presentata, cioè, all'utente dopo l'avvio dell'app per la prima volta tramite l'icona nel *drawer*. Le *activities* possono chiamarsi reciprocamente per manifestare l'intenzione di aprire una funzionalità specificata all'interno di un'altra Activity; ogni volta che una di queste ne chiama un'altra le cede il posto. La chiamante viene *stoppata* e la nuova viene inizializzata ed avviata sullo schermo, creando così il ciclo di interazione con l'utente. Ogni volta che l'utente cambia activity le precedenti entrano nel "back stack", una struttura dati di tipo *L.I.F.O.*[6] da cui vengono estratte quando viene premuto il tasto "back" del device. Questa funzionalità serve ad implementare la navigazione all'interno delle applicazioni e fa in modo che l'utente non si perda tra le varie istanze delle *activities*.

Il processo di arresto di un'activity a causa dell'inizio di una nuova viene segnalato e gestito tramite i cosiddetti metodi di *callback* del ciclo di vita. Vi è una certa quantità

[6] *Last In First Out*

di questi metodi a causa degli innumerevoli stati in cui un'activity può trovarsi: appena creata, inizializzata, in arresto, in ripresa, in cancellazione e così via. Ogni metodo di *callback* permette di implementare il codice specifico per diversi tipi di utilizzo. Vedremo un esempio del lavoro delle activities descrivendo quelle di *MiRambla Mobile*.

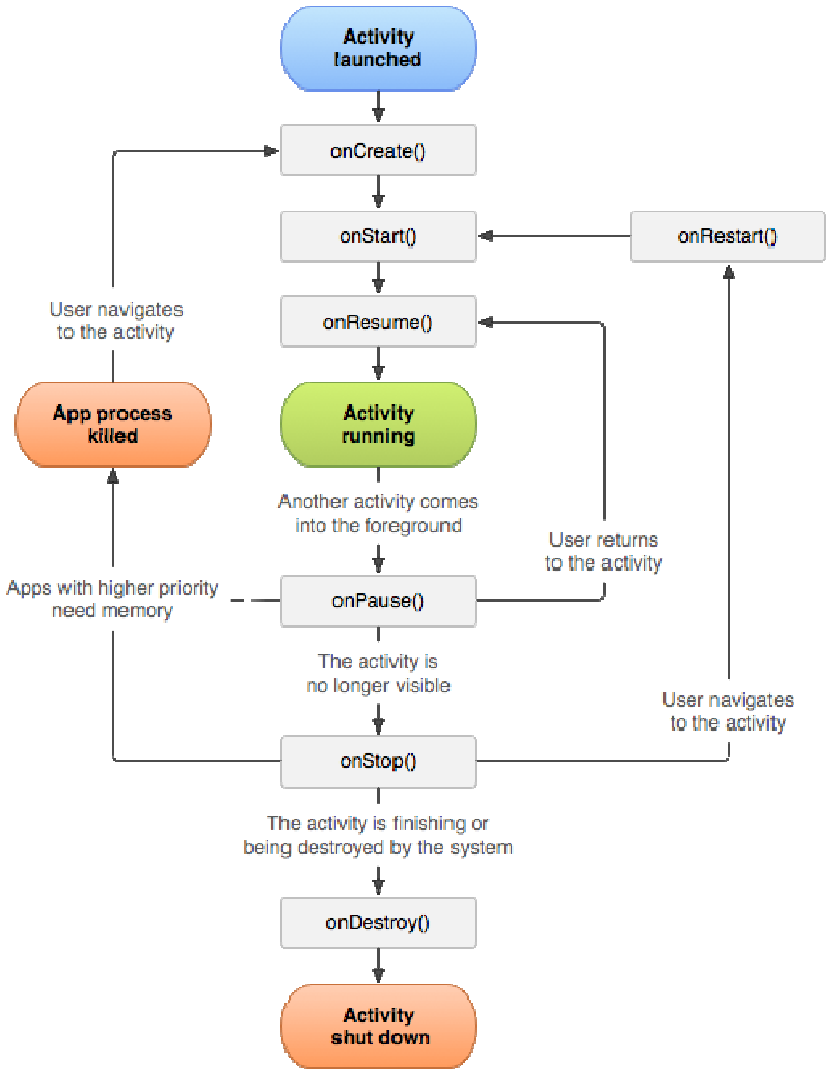

Figura 13 – Ciclo di vita delle Activity e relativi metodi di callback

3.1.1 MainActivity

In *MiRambla Mobil*e la *MainActivity* è la responsabile del funzionamento di buona parte dell'applicazione, viene avviata nel momento del lancio tramite icona e svolge una serie di *task* da eseguire immediatamente. In primo luogo la **MA** (*MainActivity*) è responsabile della creazione della *ActionBar*; dato che il contenuto di questo elemento è personalizzato in MiRambla Mobile la sua istanziazione avviene estendendo la *ActionBarActivity*. L'elemento personalizzato in questione è il *title* della *ActionBar*, che, oltre a variare comportamento in base al pannello/layout visualizzato, è dotato di un font speciale (utilizzato nel brand di MiRambla). L'implementazione del font (caricato nella cartella *assets/fonts/*) avviene tramite una *TextBox* posizionata a sinistra del logo "menù" dell'app. Successivamente, nel metodo di *callback* "onCreate()" la MA si occupa di istanziare il suo *fragment* di layout, a quest'ultimo seguirà la creazione e la gestione del comportamento del *NavigationDrawer*.

Il **ND** (*NavigationDrawer*) ha una forte interazione con *l'ActionBar*, in quanto può essere aperto dal tasto in alto a sinistra posizionato su di essa. Gli eventi di apertura e chiusura naturalmente *triggerano* il cambio del *title* ma per gestire le voci di menu presenti nel ND si è resa necessaria l'implementazione di un *onClickListener* che rimanga in ascolto per gli eventi di *touch* sui blocchi di navigazione. I rimanenti metodi nella *MainActivity* sono di supporto: coprono ad esempio il ripristino dell'app in modo che questa ritorni al suo stato salvato e non a quello iniziale, gestiscono la back *navigation* ed il *replacement* dinamico dei *fragment* in risposta all'interazione con l'utente.

Figura 14 - Layout grafico per la MainActivity ed il NavigationDrawer

3.2 Adapters

La struttura di un layout in XML permette di posizionare con una data disposizione al suo interno un certo numero di elementi di tipo *View*. Tuttavia si possono verificare dei casi in cui non possiamo predeterminare *quanti* elementi dovremo inserire all'interno di tale struttura; in questi casi occorre popolare il layout a *run-time*. Gli *Adapter* sono quei costrutti (sottoclasse di *AdapterView*) da utilizzare per questo scopo, essi, infatti, effettuano il *binding* tra i dati dinamici ed il layout in questione. L'*adapter* si comporta, dunque, come un intermediario tra i dati e il layout *AdapterView*, si fa carico di prendere i dati dalle sorgenti (siano esse array, database o altre) e di convertirne le voci in *View*, aggiunte successivamente all'*AdapterView*.

I layout che necessitano di un *adapter* per essere utilizzati sono principalmente la *ListView* e la *GridView*.

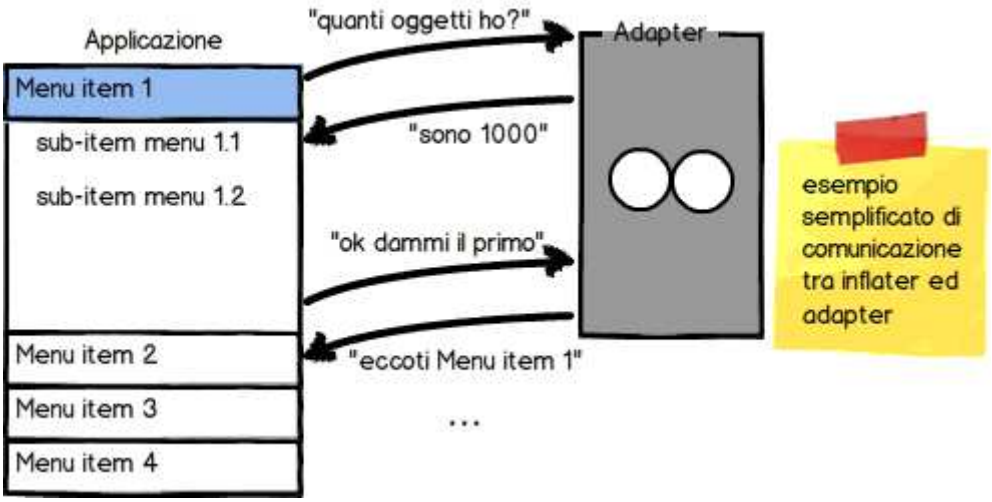

Figura 15 – lavoro svolto dall'Adapter (semplificazione)

3.2.1 MainAdapter

Il *MainAdapter* di *MiRambla Mobile* è il responsabile del caricamento di tutte le celle della *ListView* eventi che vengono poi disposte nella home screen dell'app. Dato che gli eventi possono essere in numero differente a seconda di diversi fattori (distanza, filtro, tipo di ricerca, posizionamento) è chiaro che il loro numero non può essere conosciuto a priori. Il *MainAdapter* lavora altresì a stretto contatto con il model degli Eventi, che immagazzina la struttura dati JSON passata dal portale online.

Come prima cosa il **MA** (*MainAdapter*) istanzia l'*ArrayList* di eventi che utilizzerà per gestire dinamicamente i contenuti nel layout, dopodiché verranno creati i metodi che associano i singoli dati eventi nel model agli elementi fisici nelle *view*. La classe principale del MA è *getView()* in cui vengono prese ed *inflatate*[7] le celle (o *rows*) che compongono la *ListView*; in questo contesto è implementata anche una logica di riuso delle celle. Quando *scrolliamo* la nostra lista verso il basso o l'alto, infatti, gli elementi di *row* che escono dalla vista vengono presi, ripuliti dai dati e ripopolati con quelli che verranno inseriti ordinatamente. Questo tipo di riciclo è alla base delle performance

[7] "inflate" è il nome del metodo Android che "riempie" la gerarchia degli elementi visivi specificati in XML

per una struttura di questo tipo; molte applicazioni sul Play Store soffrono spesso di *lag*[8] derivante dallo scroll dei dati perché l'applicazione impegna le proprie risorse nel ricreare gli elementi della *view* da zero. Un'altra ottimizzazione apportata è quella che mantiene, all'interno di un *Holder*[9], il riferimento ai vecchi dati ripuliti, i quali, invece di essere eliminati, sono semplicemente messi da parte e, successivamente, riconessi ad una cella quando deve riapparire a vista sullo schermo. Il resto della classe valorizza tutti gli elementi grafici di cui è composto il *row* e restituisce appunto la *View* così creata.

3.2.2 NavigationDrawerAdapter

Il *NavigationDrawerAdapter*, il cui comportamento è simile al precedente *adapter*, si occupa di popolare tutto il contenuto del menu inserito all'interno del *NavigationDrawer*. Anche la struttura del file è molto simile, il **NDA** (*NavigationDrawerAdapter*) sfrutta, infatti, un *ArrayList* di *items* generici creati specificatamente per manipolare le voci del menu. Tramite alcuni campi statici è possibile identificare il tipo di tasto premuto o la voce di menu selezionata, passando quindi alla sua implementazione.

3.3 Fragments

Un *Fragment* in Android rappresenta un comportamento di una porzione dell'interfaccia utente all'interno di un'*Activity*. I *fragment* possono essere utilizzati e combinati assieme per creare un'interfaccia utente multi-pannello i cui comportamenti sono indipendenti tra di loro e possono essere riutilizzati in diversi punti dell'app senza la necessità di doverli riscrivere di nuovo. Dato che un *fragment* è un singolo modulo grafico di un layout esso ha un suo ciclo di vita particolare (proprio come una *Activity*), risponde a metodi di *callback* differenti e riceve input da parte dell'utente. All'interno dell'*activity* un *fragment* può essere creato ed eliminato "al volo" durante

[8] Inglesismo che definisce il ritardo in informatica, sia esso derivante dalla rete o da altri fattori tra i quali anche le scarse prestazioni.
[9] L'Holder è implementato come una classe statica interna

l’esecuzione come se fosse una *sotto-activity* indipendente. Dato lo stretto rapporto tra *fragment* ed *activity* è chiaro che per poter funzionare questa struttura dovrà essere sempre allacciata ad una sua *activity host* ed anche il ciclo di vita, sebbene diverso, dipenderà comunque da quello del suo *host* (ad esempio, se l’*activity* viene messa in pausa, anche tutti i suoi *fragment* verranno messi in pausa).

L’implementazione dei *fragment* è trasparente all’utente dato che questi vengono inseriti anche all’interno del *back stack* per una corretta navigazione dell’app. Lo scopo principale dell’introduzione dei *fragment* era quello di supportare *UI* (*user interfaces*) più dinamiche e flessibili, adattabili soprattutto anche alle esigenze di *tablet* o dispositivi con schermi di più grande dimensione. Tramite *fragment*, infatti, è possibile combinare diversi pannelli in schermi più grandi oppure mantenerli intercambiabili in uno schermo più piccolo, come mostrato in **Figura 16**. Questi vantaggi si possono ottenere con una minima personalizzazione di codice dopo che i *fragment* sono stati già programmati una prima volta; ognuno di questi moduli va progettato come un componente stand-alone che però può essere riutilizzato in diversi punti dell’app. Ogni *fragment*, infatti, è dotato di un suo file di layout XML che ne definisce la struttura a fronte del suo comportamento, scritto nel file di codice.

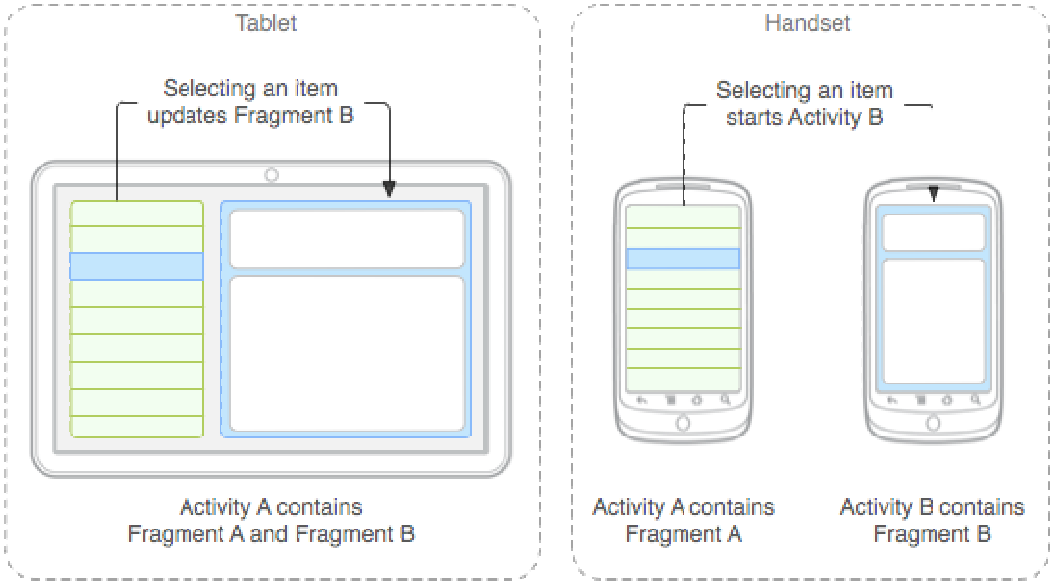

Figura 16 - Esempio d'uso dei Fragment

3.3.1 MainFragment

Il *MainFragment* è una classe che deriva da quella più generica di *Fragment*, entrambi fanno parte della libreria *android.support.v4.app.Fragment*, che serve a renderli implementabili anche in versioni precedenti di Android. Nel **MF** (*MainFragment*) vengono definiti all'inizio i metodi di *onDetach()* ed *onAttach(),* chiamati nel momento in cui il *fragment* stesso viene assegnato alla *host activity* corrispondente; tramite questi metodi viene gestita l'assegnazione del riferimento ai *fragment* nel *FragmentChanger*. Successivamente viene implementato il metodo principale della classe ovvero *onCreateView(),* chiamato in fase di caricamento del layout visivo. Tale componente si occupa di *inflatare* il container dove risiedono gli elementi XML principali e crea un'istanza dell'*adapter* dati visto in precedenza. Durante la procedura vengono *bindati* i dati recuperati precedentemente dalle varie istanze della *ListView* mediante la chiamata *list.setAdapter(nomeAdapter)*. Infine, dato che vogliamo ottenere una risposta dalla pressione sull'area della cella in corrispondenza di un evento, viene creato un *listener* di tipo *OnItemClick* che ci permette non fare distinzioni tra la zona premuta dall'utente all'interno del row dell'evento. Questo evento di tap, comunicherà al *FragmentChanger* che occorre rimpiazzare il *fragment* corrente con quello di dettaglio, passando come parametro l'ID corrispondente dell'evento interessato. L'aspetto finale del MF è quello riportato in ***Figura 14.***

3.3.2 DetailFragment

Il *DetailFragment* rimane strutturato alla stessa maniera del *main fragment* ma, ovviamente, ha un layout strutturato in maniera molto più complessa. Dato che la sua funzione principale è quella di mettere in vetrina un evento per l'utente la sua grafica ha un impatto diverso rispetto alla lista visualizzata nella home screen dell'applicazione. Per conoscere quale evento va mostrato all'interno del **DF** (*DetailFragment*) viene passato un KEY_ID corrispondente all'evento interessato e durante la procedura di *onCreateView()* il *fragment* scarica dalla rete le informazioni

dettagliate e le dispone dentro i blocchi grafici. Come per il *main*, viene effettuato un *inflate* di tutti gli elementi presenti poi visualizzati su schermo.

In seguito sono stati predisposti i *listener* che implementano l'eventuale passaggio tra la visualizzazione del dettaglio-evento a quella di un locale o di un *Rambler*.

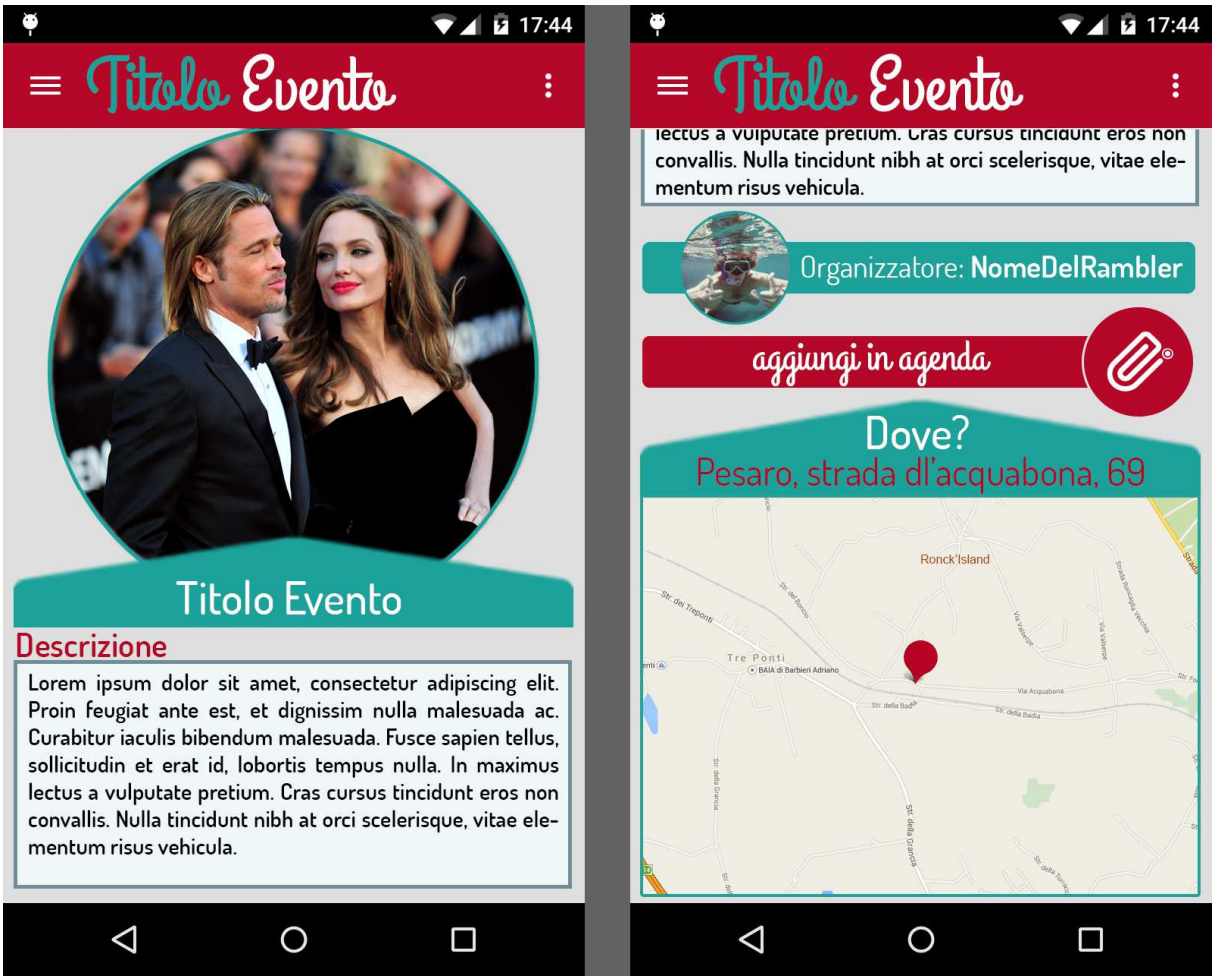

Figura 17 - dettaglio grafico del DetailFragment

3.4 Models

Il *model* generalmente può avere più definizioni che dipendono molto dal contesto applicativo, informalmente possiamo dire che è un costrutto che si riferisce a classi definite sui dati che incapsulano oggetti applicativi di alto livello. All'interno del progetto di *MiRambla Mobile*, i *model* servono a codificare le diverse tipologie di oggetti del dominio applicativo in delle classi java. Concetti astratti come "*Evento*" o "*Rambler*" necessitano di diversi campi speciali per essere modellati, campi che tuttavia saranno comuni alle varie istanze di queste due classi. Il *model* ci permette di dichiarare i dati per fare si che essi siano riutilizzati tra i vari eventi o *Ramblers*. I campi in questione sono direttamente collegati con le risorse web che vengono scaricate dal portale di MiRambla e sono una diretta implementazione dei dati *parsati* nelle risposte in formato JSON. I due *models* interagiscono con diversi componenti del progetto (il model *Event* lavora con il *MainAdapter* ad esempio) ed in fase di caricamento popolano le istanze delle *view* o dei layout che si costruiscono con i dati. Il model Events contiene al suo interno tutti i campi relativi agli eventi (come titolo, durata, luogo, organizzatore, link all'immagine etc.) che vengono scaricati in blocco all'avvio dell'app ed aggiornati in fase di *refresh* della home page. Il model del *Rambler* invece è più elastico e contiene informazioni sugli utenti concepiti come gli organizzatori (o partecipanti) agli eventi. Per estensione questo model sarà utilizzato anche per modellare l'istanza di un utente registrato, dato che anch'esso è di fatto un *Rambler*.

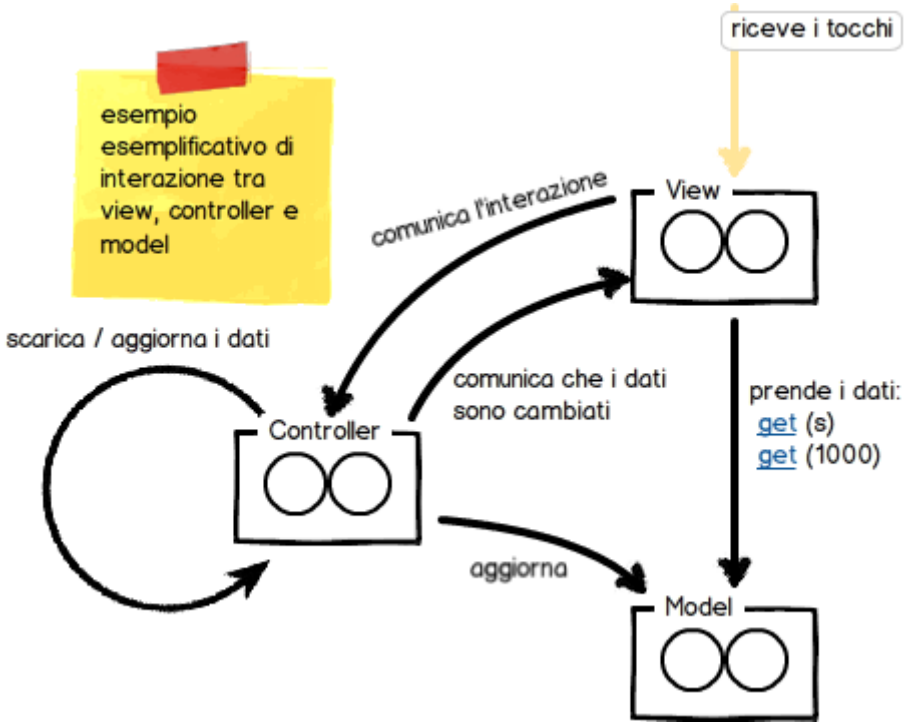

Figura 18 - interazione del Model

3.5 Utils

Le *utils* o utilità rappresentano tutti quei costrutti di supporto utilizzati all'interno del codice per fornire funzionalità aggiuntive all'applicazione. L'unico file .java presente all'interno di questa cartella è "CircleTransform.java" che descrive un'implementazione specifica di una funzione offerta da una libreria esterna, usata per la gestione delle immagini dell'applicativo. La libreria in questione è *Picasso*. Creata da *Square*, comunità *OpenSource* di *Github*, Picasso è una formidabile libreria che permette di implementare un'ottima gestione di *download/caching* per immagini su applicazioni Android e riesce a farlo in poche linee di codice. Tra gli aspetti critici della gestione immagini su Android, Picasso copre anche:

- *Handling* delle *ImageView* utilizzate nel riciclo, scaricamento ed eliminazione di un *adapter*.
- Trasformazioni anche complesse delle immagini utilizzando poca memoria.
- *Caching* automatico delle immagini in memoria o in supporti secondari.

All’interno di MiRambla Mobile, Picasso viene utilizzato per tutti i caricamenti di immagini relative agli eventi o *Rambler*, sia per le anteprime delle immagini evento (ad esempio nella *Home Screen*), sia per i dettagli più specifici (come nella *DetailActivity*).

La diversa dimensione dell’anteprima è, infatti, gestita tramite una trasformazione nella CircleTransform.java presente in questa cartella. Picasso in questa classe inizia prendendo un file sorgente di cui calcola dimensioni assolute e crea una *bitmap* con tali dimensioni per poi inizializzarla dentro ad un *canvas*. Con il metodo *paint* e sfruttando un *BitmapShader* disegna tale *canvas* (utilizzando *tilemode.clamp*) ed aggiunge per ultimo un anti-alias. Infine il processo disegna la figura geometrica interessata tramite coordinate (un cerchio in questo caso) e piazza l’immagine ottenuta all’interno del nuovo *canvas* appena creato.

Questo metodo restituisce una figura personalizzata con un proprio id e dimensioni generiche applicate per effettuare la trasformazione. Tramite l’uso di Picasso le prestazioni delle immagini sono notevolmente migliorate, come anche la stesura del codice.

3.6 Altre risorse

Abbiamo visto la completa panoramica dello sviluppo dietro quasi tutti gli elementi che costituiscono l'app di MiRambla Mobile. Per la versione a cui giungerà l'applicazione descritta al momento della scrittura di questo documento, mancano soltanto alcune pagine di minore interesse. Discuteremo dunque in questo paragrafo tutte quelle *feature out-of-the-box* che saranno incluse nell'app, che abbiamo (o non abbiamo) introdotto nell'analisi/progettazione ma che sono state saltate durante la stesura del resto del capitolo di sviluppo.

Il motivo di tale omissione deriva dal fatto che si tratta di caratteristiche laterali all'applicativo. Esse completano comunque l'immagine generale del prodotto ma utilizzano codice largamente implementato altrove (in altri *files*) o sfruttano layout,

strutture dati, costrutti già programmati ed appositamente riutilizzati (ne sono un esempio i *fragment* stessi). Le voci interessate sono dunque tutte quelle che si possono scorgere dall'anteprima del *NavigationDrawer* aperto (***Figura 14***).

La voce "Accedi" permetterà di arrivare nel *loginFragment*, costituito da due campi in cui inserire i propri dati per poter accedere all'applicazione; contestualmente ai campi vi sarà anche la possibilità di avviare il recupero dei propri dati in caso di smarrimento.

La voce "Filtra", in fase di deploy verrà probabilmente cambiata. Il concetto che esprime è quello di raffinare una ricerca già visualizzata, dunque inizialmente potrebbe portare soltanto a cambiare la distanza/posizione correlata agli eventi in zona. In seguito, invece, dovrà diventare una ricerca avanzata simile a quella nel portale online, permettendo di effettuare operazioni più complesse. La ricerca base vera e propria, infatti, verrà implementata tramite l'aggiunta di una *action search icon* sulla barra principale dell'app in modo che sia pienamente intuitiva.

La voce "About Us" è un segnaposto e porterà all'insieme di tutte quelle pagine informative posizionate nella parte inferiore del portale di MiRambla, in aggiunta ad altre dedicate all'app mobile (tra cui About, Contatti, FAQ, Termini d'utilizzo, Politica sulla privacy, Note legali, riferimenti a librerie esterne, etc.). Le pagine in questione verranno richieste al sito e verranno visualizzate come testo semplice all'interno di un layout di base oppure saranno *renderizzate* all'interno di una *WebView* a seconda della necessità.

La voce "Impostazioni" porterà, abbastanza chiaramente, ad un piccolo menù contenente i settaggi dedicati per l'applicazione mobile. La versione di base di MiRambla Mobile non necessita di grandi personalizzazioni in realtà, tuttavia sarà inserita la struttura di un menù con poche voci in modo da poter essere agilmente ampliate successivamente. Settaggi come la frequenza di aggiornamento degli eventi e le impostazioni per la geolocalizzazione sono solitamente le più comuni per questo tipo di app.

Il *NavigationDrawer* è un elemento molto comodo, come già accennato, e viene in aiuto soprattutto per la parte evolutiva di MiRambla Mobile. Le voci che fanno parte

di questo menù possono essere aggiunte e rimosse con facilità, permettendo modifiche molto specifiche che rispecchiano la massima compatibilità grafica e funzionale dell'applicazione mobile con quella del portale stesso.

4.0 CONCLUSIONI

In conclusione il lavoro svolto ha preso in esame differenti aspetti della programmazione *Mobile*. Iniziando da una descrizione dell'impiego di queste tecnologie nel mondo moderno e dal modo in cui esse sono diventate virali all'interno della nostra società, si è giunti alla presentazione di un'idea dalla quale è stato possibile concepire un progetto reale. La realizzazione di un idea non è mai un processo semplice da portare a termine, soprattutto quando ci si prefissa di arrivare ad un risultato tangibile e concreto.

Data la vastità dell'argomento *Social-Mobile,* è stata presa in considerazione l'ipotesi di un suo impiego pratico calato nella realtà online di MiRambla, piattaforma dalle grandi potenzialità sociali. Sono stati presi in considerazione, inoltre, un certo numero di aspetti chiave selezionati attraverso l'indagine di mercato, che hanno accompagnato lo sviluppo di tutto il lavoro e che mirano a contraddistinguerlo dalla concorrenza: branding del marchio, originalità, semplicità d'uso, performance e concretezza. Dalla descrizione iniziale, all'analisi, alla progettazione e sviluppo, tutto il lavoro ha cercato di soddisfare al massimo le aspettative create fino alla realizzazione finale.

MiRambla Mobile è un'applicazione agli albori ed è, dunque, descrivibile come semplice ed immediata anche se limitata per quegli aspetti che costituiscono i limiti propri di un progetto di questa dimensione. Al contrario di molti lavori che non hanno l'obbligo di interagire con le realtà di mercato, il progetto di MiRambla Mobile ha seguito dei canoni più rigorosi, dettati dall’esigenza di assecondare le aspettative del cliente. Questo ha dato all'applicazione uno slancio in più, rappresentato dalla volontà di seguire gli aspetti chiave sopra citati, capaci di trasformare il lavoro di tesi in un processo professionale. L'app così sviluppata porta uno strumento innovativo nelle mani di un potenziale utente del portale di MiRambla ed incontra le sue aspettative in quanto mezzo di comunicazione privilegiato all'interno della rete sociale dell'individuo. Tale mezzo potenzia le capacità della persona connettendola non solo agli eventi intorno a lei ma sviluppando tramite gli altri *Ramblers* quell'interazione

social che oggi è tanto richiesta. Il prodotto, nelle mani dell'utente, si presenta accattivante, aggiornato e soprattutto dalle ottime prestazioni.

L'applicazione ha tante prospettive nel suo futuro prossimo, sebbene sia già pronta per l'impiego nel *Play Store*. MiRambla Mobile dovrà maturare con l'evolversi del portale online ed il primo obiettivo di un suo miglioramento dovrebbe essere l'implementazione delle versioni successive, già pianificate nel presente lavoro in fase di analisi. Le basi per questi aggiornamenti di versione sono quindi già gettate e possono essere suddivise in ulteriori step evolutivi.

Questo lavoro ha permesso di ampliare sia le conoscenze tecnologiche in merito al panorama dei *device mobile* presenti oggi sulla scena, sia di incontrare dal vivo la piattaforma Android. Il sistema operativo del robottino verde offre grandi potenzialità ed aiuta, ogni giorno, i suoi innumerevoli utenti nell'affrontare i propri impegni quotidiani; *MiRambla Mobile* vuole offrire le stesse possibilità nei limiti di un app social degna del proprio nome.

Il progetto intero è stato reso possibile grazie alla gentile concessione dell'azienda "Zoo di Venere" situata a Faenza, sviluppatrice dell'intero progetto: portale web, grafica e marchio di MiRambla. Un particolare ringraziamento va al suo titolare ed ideatore Andrea Castellari.

INDICE DELLE IMMAGINI

BIBLIOGRAFIA

Andreas M. Kaplan, M. H. (2010). *Users of the world, unite! The challenges and opportunities of Social Media.* Elsevier.

Apple. (2014). *What is iOS*. Retrieved from iOS, official site: http://www.apple.com/ios/what-is/

Berners-Lee, T. (2006, 08 22). DeveloperWorks Interviews: Tim Berners-Lee. (S. Laningham, Interviewer)

Chowney, V. (2012). Android takes 52.5% of smartphone market. *Econsultancy*.

Defined, W. 1. (n.d.). *Is there a Web 1.0?* Retrieved from Howstuffworks: http://computer.howstuffworks.com/web-101.htm

DiNucci, D. (1999). Fragmented Future. *Print*.

GH Forman, J. Z. (1994). *Computer*.

Google. (2014). *Android Design Principles*. Retrieved from Android Design: http://developer.android.com/design/index.html

Google. (2014). *Android Developers Index*. Retrieved from Android Developers: http://developer.android.com/develop/index.html

Google. (2014). *Android, the world's most popular mobile platform*. Retrieved from Android Developers: http://developer.android.com/about/index.html

Graham, P. (2005). Web 2.0. *"I first heard the phrase 'Web 2.0' in the name of the Web 2.0 conference in 2004"*.

H. Lee, S., & Kim, Y.-Y. A. (2010). *"Googling Social Interactions: Web Search Engine Based Social Network Construction.* PLoS ONE.

Hillegass, A., & Conway, J. (2012). *iOS Programming: The Big Nerd Ranch Guide (3rd ed.).* Pearson.

Kima, J. L. (2010). *On social Web sites.* Elsevier.

Landay, J. K. (1993). *User interface issues in mobile computing.* Workstation Operating Systems.

M.N.K. Boulos, S. W. (2011). *How smartphones are changing the face of mobile and participatory healthcare.* Drake Circus, Plymouth: © 2011 Boulos et al; licensee BioMed Central Ltd.

Members, A. (2011). *Open Handset Alliance*. Retrieved from Sito web Open Handset Alliance: http://www.openhandsetalliance.com/oha_members.html

Pitoura, E. (1998). *Data Management for Mobile Computing.* University of Ioannina, Greece: Computer Science Department.

Pullela, S. (n.d.). *Security Issues in Mobile Computing.* University of Texas at Arlington: Department of Computer Science.

Repacholi, M. (2001). *Health risks from the use of mobile phones.* Elsevier.

Roth, J. (2005). *Mobile Computing.* Germany.

Sager, I. (2012). Before IPhone and Android Came Simon, the First Smartphone. *Bloomberg Businessweek.*

Savage, P. (1995). *Business Telephone.* Middletown, N.J.

USA, I. C. (2014). *International Data Corporation*. Retrieved from Sito Web IDC: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Zimmerman, J. B. (1999). *Mobile Computing: Characteristics, Business Benefits, and Mobile Framework.*

RINGRAZIAMENTI

Scrivere questi ringraziamenti non è facile per me. Il mio percorso in questo CdL è stato lungo, non privo di ostacoli e le persone che mi hanno accompagnato lungo la strada sono state tante. Cercherò di andare per ordine e di non scordarmi nessuno. Il primo ringraziamento va al mio relatore il *Professore Mirko Ravaioli*, che ha reso possibile tutto il mio progetto ed ha portato la mia idea iniziale verso qualcosa di più di una semplice tesi, realizzando al meglio le mie aspettative. Un grazie immenso va alla mia Franci, che con tanto amore e voglia (anche costretta a volte!!) ha tradotto tutto il mio lavoro in italiano corretto e i miei molti refusi grammaticali (nonché logici!). Grazie mille a Massimo, sviluppatore ed insegnante bravissimo, per i suoi tanti consigli e numerose dritte su Android! Grazie davvero tanto anche a Michele, alla sua passione e voglia che ha impiegato senza sosta nel preparare assieme a me l'esame di CPS. Grazie alla Leti e a Giovi, senza i quali non sarei riuscito a studiare nemmeno lontanamente almeno due esami (se non tre!). Grazie a Fedi, immancabile compagno di studi ed inossidabile amico, sempre lì a ricordarti di tenere la testa bassa sui libri quando è ora per poi fare pausa con due pieghe in pano. Grazie ai miei compagni d'avventura marchigiani: Diego, Nicola, Matteo e Alessio; i momenti passati con voi sullo studio, sul treno ed ovunque siamo stati insieme sono stati esilaranti a dir poco e non li scorderò mai! Grazie anche ai compagni d'avventura romagnoli e san marinesi: Eros, Luca, Steve, Mike per tutta la strada fatta insieme! Grazie a quei geni di Giulio e Spada, i quali hanno dato, lungo il percorso, un numero di aiuti che tende esponenzialmente a più infinito! Grazie alla Vale e alla Lu, le mie S.P.R.I.Te.-rs preferite con cui abbiamo preparato gli ultimi odiatissimi esami! Grazie a Viviana, per il suo aiuto, i consigli e la sua grande simpatia! Grazie a Valy e la Baby, che da quando erano qui a dove sono ora (6000 km!) mi hanno sempre spronato a studiare e ad andare avanti! Grazie anche a tutti quanti i miei amici: Diego, Giuly, Fabio, Vero, Penny, Raffy, Enry, Ali, Marco, Giorgio, Eli, Luca, Andri, Mary, Mike, Fabri, che mi hanno accompagnato nei momenti difficili per tutto il percorso. Grazie a Giada, grande amica, per avermi trasmesso la sua voglia di fare e studiare. Grazie a tutta la famiglia, mamma, papà e sorella, che hanno creduto in me fino a questo grande giorno.