Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Progettazione ed Implementazione di un sistema di localizzazione di oggetti tramite Beacon

Relatore:
Chiar.mo Prof.
Marco di Felice

Presentata da:
Faienza Antonio

Sessione III
Anno Accademico 2015/2016

A mia MADRE *e a mio* PADRE,
che mi hanno trasmesso la cultura del lavoro,
insegnandomi a non arrendermi mai
e a credere sempre in me stesso ...

Introduzione

Sono passati ormai tanti anni dalla comparsa dei primi telefoni cellulari; dispositivi che con il tempo si sono evoluti sotto tutti i punti di vista, estetici e funzionali; e con essi la nostra percezione della telefonia mobile. Prima infatti possedere cellulari dalle dimensioni ridotte, significava essere alla moda, ma con il loro graduale aumento, ciò è cambiato e con esse anche i servizi che sono riusciti a offrirci. Oggi infatti parliamo di *smartphone* e a questa parola, ormai entrata nel nostro vocabolario, si sono aggiunte termini come *tablet* e *phablet*.

Ciò che ne ha determinato il successo è stato l'impatto che hanno avuto nella nostra quotidianità, riuscendo a penetrare i più svariati settori: dall'intrattenimento, fino al turismo, all' economia,all' agricoltura e a tanti altri.

Si stima che negli USA, l'81% di persone con un età compresa tra i 25 e i 34 anni e il 50% di persone con un età superiore ai 55 anni posseggono uno smartphone, mentre il 70% dei teenager tra i 13 e i 17 anni ne usano uno. [1]

Di conseguenza è facile intuire che il numero di smartphone è molto elevato e con il passare del tempo continuerà a crescere. Secondo i dati Forbes, Apple aumenterà le sue unità di vendita tra il 2015 e il 2019, da 237 milioni a 274; nello stesso periodo, il numero di smartphone che montano Android crescerà fino a 1.53 miliardi, rispetto gli attuali 1.33 attuali. [4].

Se a ciò si aggiunge che ben l'80% usano i propri devices durante lo shopping, che l'53% delle ricerche di acquisto sono il risultato di ricerche fatte da smartphone e che per finire l'86% della gente usa i propri telefoni come strumento di informazione, [1] ne consegue che c'è un terreno fertile ancora

non ampiamente sfruttato, legato al possesso di questi device di ultima generazione, che mira a proporre agli utenti ciò che più li lega ai propri interessi. Come? attraverso la *pubblicità*. Questa nuova tipologia di vendita è chiamata *"Mobile Advertising"*. Volendone dare una prima breve introduzione, può essere definito come quello strumento che dà la possibilità alle società commerciali di essere presenti al fianco dei clienti con messaggi pubblicitari ad hoc [2].

In uno studio viene definito come l'insieme delle attività commerciali che un utente realizza con il suo smartphone [1]. Ci si può chiedere: come vengono mandati questi messaggi pubblicitari ? Nei primi anni dalla sua comparsa, gli SMS e gli MMS erano lo strumento sicuramente più utilizzato [3], ma come si è già anticipato la tecnologia è in perenne cambiamento, infatti negli ultimi anni sta prendendo sempre più piede l'utilizzo di un nuovo dispositivo che prende in nome di *Beacon*. In generale un Beacon è un congegno Bluetooth in grado di mandare segnali, segnali che saranno poi captati da uno smartphone per poter produrre un contenuto. Una domanda potrebbe essere: ma quando si parla di Beacon ci si sta riferendo esclusivamente al mondo del Mobile Advertising? Ovviamente no. In generale può essere accostato all'ormai famoso mondo dell'*Internet of Things (IoT)*. Il concetto che ne è alla base è quello di di dar vista alle cose, facendo sì tutto il mondo possa essere connesso. Per cui gli oggetti, in questo modo, acquistano intelligenza e possono comunicare tra loro fornendo, dati informazioni senza la necessaria supervisione di un essere umano. Le potenzialità di questo mondo sono enormi. Cosa succederebbe se un *pacemaker* fosse in grado di segnalare eventuali malfunzionamenti consentendo un immediato soccorso ad un paziente? Si potrebbero fare innumerevoli altri esempi: una sveglia suona prima in caso di traffico, una lampada si accende per segnalare dove si trova esattamente un oggetto, scarpe da ginnastica che segnalano tempi di corsa, velocità, distanze e quant'altro consentendo a persone che si trovano in parti diversi della terra di poter competere tra loro, come se si stesse disputando una partita *online* davanti ad un computer. Il tutto perché ogni cosa è con-

nessa alla rete. I Beacon rappresentano un modo per poter dare vita agli oggetti, oggetti che non sono dotati di circuiti interni. Si pensi ad un mazzo di chiavi: è possibile monitorarne la posizione? In teoria no, ma se lo si associasse ad un Beacon tutto diventerebbe possibile. Lo scopo dell'elaborato sarà quello di addentarsi alla scoperta del *Mobile Advertising*, nella prima parte, dandone una definizione più specifica definendone i vantaggi e quali aspetti della psicologia intervengono a favore del suo successo. Nella seconda parte si cercherà inizialmente di correlare questa nuova tendenza del Mobile Advertising all' utilizzo dei Beacon per poi spostare il centro dell'attenzione su questo dispositivo, cercando di far' luce su aspetti non ancora molto conosciuti al grande pubblico, illustrando quella che è la tecnica utilizzata, i protocolli che sfrutta, riportando esperimenti inerenti le sue performance e indicando casi reali di utilizzo. Infine per sottolineare il grande potenziale che sono in grado di esprimere si descriverà nel dettaglio l'implementazione di un progetto, cercando di spiegare il codice nelle sue fasi saliente, riguardante la localizzazione di oggetti associati ai Beacon. L'obiettivo di fondo è quello di rendere possibile la realtà descritta poc'anzi: cercare di dare vita a degli oggetti. L'applicazione, sviluppata in Android, consente agli utilizzatori che decidono di munirsi di Beacon di evitare potenziali smarrimenti. Nel caso in cui quest'ultima situazione dovesse verificarsi, dopo aver settato il dispositivo come perso, verrà attivato il monitoraggio consentendo una serie di operazioni: la ricezione di una notifica nel momento in cui si entra nuovamente nel suo range di competenza; la possibilità di consentire ad altri utenti di aggiornare automaticamente la posizione consultabile da un'apposita schermata oltre che la facoltà di poter avvisare il possessore di un eventuale Beacon ritrovato e infine fornire uno strumento di localizzazione indoor che consenta di individuarne il punto in cui è situato.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Overview sul Mobile Advertising

Molti siti web sono stati ottimizzati e adattati affinchè possano essere consultati comodamente attraverso l'utilizzo degli smartphone. A tal proposito Forbes dichiara che nel maggio 2015 Google ha annunciato che il 50.3% degli acquisti vengono effettuati da smartphone e tablet piuttosto che da personal computer [8]. Non c'è da meravigliarsi quindi che in paesi come Stati Uniti e Cina, la maggior parte delle query provengano da questi dispositivi piuttosto che da personal computer e che di conseguenza molte delle asserzioni pubblicitarie non siano più attraverso link ipertestuali, la cui apertura è contrastata da strumenti come Adblock Plus [1] [8] ma come Google sta proponendo, attraverso un nuovo tipo di *Advertising* che sfrutta il *Mobile*,il quale punta alla propagazione di immagini e alla loro gestione mediante swipe [9][10].
Stiamo parlando quindi di questa nuova tendenza, che sta prendendo piede ed espandendosi negli ultimi anni chiamata *Mobile Advertising*. Ma cosa si vuole esprimere a pieno con questa nuova forma di pubblicità?
Con Mobile Advertising si intende non solo l'invio di messaggi ad hoc, unici e personalizzati all'utente, ma anche come strumento atto a suscitare la sua curiosità stimolando l'interazione con il contenuto che gli viene offerto, in

[1] Adblock Plus: rappresenta un estensione gratuita per i maggiori browser in circolazione e Android che permette eliminare banner publicitari, pop-upd, e malware scaricabile da https://adblockplus.org/

modo che possano essere influenzate le sue attitudini, intenzioni e comportamenti.
La sua potenza riguarda la possibilità di poter far riferimento a un vasto pubblico, dare una forte interattività, e predisporre delle pubblicità che siano ad hoc per ogni tipologia di gruppo. La forza di questi messaggi sono superiori ai canali televisivi, ai messaggi delle radio e può essere considerata una nuova vera piattaforma. La crisi del 2008 ha reso possibile la crescita di questo sistema come riferimento per l'advertising. Inoltre da studi effettuati emerge che incide notevolmente sul comportamento degli utenti innovatori, esercitando quasi una funzione manipolatrice, riuscendo a incidere fortemente su quelle che sono le intenzioni dell'utente [11]. L'obiettivo di fondo è quello di addentrarci alla scoperta di questo nuovo mondo, cercando di dare una chiara delucidazione sui concetti che ne sono alla base.
Andando con ordine, l'American Marketing Association definisce l' **Advertising** come " *il posizionamento di qualsiasi messaggio e/o annuncio proposto da varie imprese commerciali, organizzazioni no profit, agenzie governative e individui che cercano di informare e / o di persuadere i membri di un particolare target di mercato o il pubblico circa i loro prodotti, servizi, organizzazioni, o idee* "[7].
Avendo ora data una definizione esplicita, del concetto di pubblicità in generale, ci si accinge a descrivere e chiarire quali sono le caratteristiche che deve possedere tale pubblicità se proposta sugli smartphone [5] e perché la rendono un fenomeno così vincente, ma che si discosta da un altro tipo di pubblicità assai famosa e conosciuta ma per ragioni diametralmente opposte: lo *spam*.

1.1 Struttura del messaggio

Per i consumatori ciò che conta maggiormente riguarda essenzialmente ciò che viene annunciato nel messaggio, quindi si può dire che l'informatività viene vista come una delle fonti maggiori di valore per il messaggio propo-

sto. Infatti, se interessante costituisce una grande forma di stima per questo sistema pubblicitario.
Il suo contenuto rappresenta lo scambio che c'è tra il consumatore e la pubblicità, ma molto spesso ci si sofferma su quale lunghezza sia più consona alle esigenze del consumatore. Ovviamente non esiste una regola fissa che sancisce quale ipotesi sia migliore dell'altra, ma sulla tipologia di pubblicità si possono fare delle ipotesi [3]. Se il contenuto del messaggio è a scopo *informativo*, magari una maggiore prolissità sarebbe gradita; al contrario un messaggio di *intrattenimento* eccessivamente lungo potrebbe risultare tediante. In questo caso l'obiettivo principale è lasciare stupito l'utente per cui deve essere sviluppato nei minimi dettagli, come verrà dibattuto più avanti, cercando di fare in modo che sia esaustivo ma al tempo stesso conciso. L'utilizzo di immagini significative può essere sicuramente lo stratagemma più rilevante per poter raggiungere tale intento. L'umorismo e il l'ironia sono gli ingredienti fondamentali per poter risaltare il contenuto magari con l'ausilio di alcuni strumenti come può ad esempio il Viral Marketing, la cui propagazione è molto importante per gli advertisements, soprattutto tra gli adolescenti.
Una pubblicità deve essere credibile affinché possa essere percepita la veridicità di quel che si pubblicizza. Come è ipotizzabile, il marchio ha un peso rilevante. Infatti i messaggi reperibili in rete raggiungono maggior successo se realizzati da un marchio autorevole e quindi riallacciandosi a quello detto poc'anzi un messaggio breve in questo caso potrebbe avere il suo effetto perché "tutelato" dal marchio. La credibilità è il vero beneficiario del m-advertising; in aggiunta, l'utilizzo di nuove tecnologie di business non può che incrementare l'impatto che la pubblicità può esprimere.
La credibilità dipende da tre fattori principali riguardanti il messaggio pubblicizzato:

- la discrepanza percepita;
- la credibilità del pubblicitario;
- la sua credibilità in generale;

Sulla base di queste tre considerazioni, il TAM *(Tecnology acceptance model)* il quale rappresenta un modello che spiega in base a cosa gli utenti utilizzano e accettano una nuova tecnologia, asserisce che tale tecnologia, la quale dà la percezione agli utenti di facilità di utilizzo e di utilità può migliorare i risultati e l'adozione della tecnologia stessa in uso. Principalmente il TAM si basa su alcuni criteri cardini di base, quali:

- Il sistema di uso comportamentale che mettono in pratica gli utilizzatori della nuova tecnologia;
- Quanta intenzione c'è da parte degli utilizzatori di voler adottare una nuova tecnologia;
- L'impressione che la tecnologia riscuote, che sia positiva o negativa;
- La possibilità di utilizzo della nuova tecnologia e soprattutto il possibile miglioramento che gli utenti utilizzatori possono apportarvi;
- La percezione di facilità di uso e gli sforzi che si compiono per adoperarla [11].

Che relazione esiste tra *utilità e facilità di uso* ? La percezione di utilità e di facilità di uso ha un effetto positivo sulle intenzioni di ricezione del messaggio pubblicitario se coadiutavate da strumenti come interattività e locazione.
A ciò si aggiunge che la *facilità di uso* ha essa stessa l'impatto di utilità sul mobile advertising.

1.2 Performance & Interattività

Nell'invio di un messaggio, tre sono i fattori che incidono maggiormente sull'impatto che questo avrà sul pubblico: *informatività, intrattenimento, irritazione*. La qualità dell' informazione agisce sulla qualità che il consumatore percepisce correlata ai suo interessi; l'intrattenimento d'altro canto mira a sollecitare lo stato d'animo e l'umore degli utenti a cui è indirizzato.

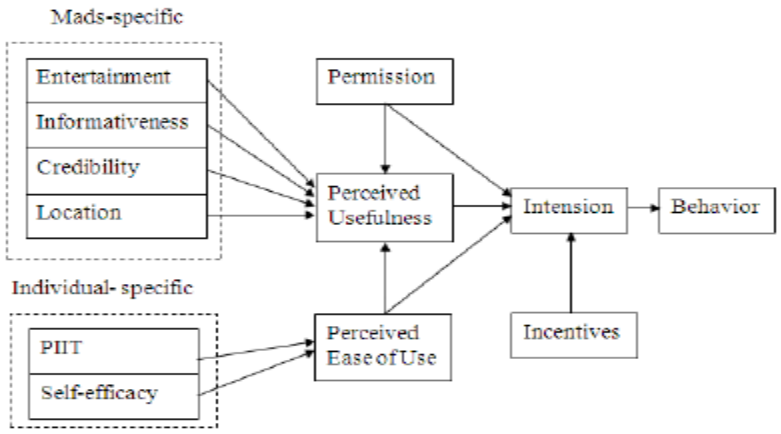


Figura 1.1: Modello concettuale rappresentante l'adozione del mobile advertising

Magari l'associazione ad un gioco può incentivarne il divertimento. L'irritazione si verifica nel momento il cui un messaggio non è ne informativo ne di intrattenimento, provocando noia, risentimento, stizza, oltre un presumibile rifiuto. Per le stesse ragioni appena asserite è importante aggiungere che un messaggio deve evitare che sia offensivo, noioso, invedente.
Tornando sui primi due criteri precedentemente esposti, ossia l'informatività e l'intrattenimento, è stato dimostrato che esiste una correlazione chiamata *sinergia d'effetto (synergy effect)* che analizza come entrambi si correlano tra loro. Mentre prima si pensava che i due fattori fossero indipendenti l'uno l'altro, invece ora non più. Per spiegarla partiamo dal presupposto che una motivazione estrinseca può diminuirne una intrinseca e viceversa. Per cui se associassimo l'informatività come valore estrinseco perché utilitario, mentre l'intrattenimento come valore intrinseco perché piacevole, in quanto appartiene alla natura umana, è facile immaginare come l'aumentare o il diminuire dell'uno incide sull'altro in modo direttamente proporzionale [3].
Ogni volta, bisogna chiedersi cosa si vuole trasmettere attraverso una pubblicità, focalizzandosi su criteri precisi, in modo tale che questi possano essere usati come punti di forza o di richiamo rispetto ad altre concorrenti.

Ci sono svariati modi per proporre al pubblico un'advertisment, per esempio tramite SMS, video, notifiche e altro. Ad ognuno di questi modi ne consegue un'azione che mira a innescare un'acquisizione di interesse prima e successivamente un'interazione con esso tale per cui l'utente sarà stimolato a conoscere cosa c'è dietro quello sconto, quell'offerta, quell'avviso che gli viene proposto.
L'interattività rappresenta quindi una sfida da portare avanti, un'idea da mettere in pratica, la visualizzazione di un qualcosa di interessante. Dare una buona interattività può accattivare maggiormente l'attrazione da parte del consumatore, perché esprime quasi una forma di comunicazione tra venditori e compratori.
Se conforme agli interessi, si interagirà con il messaggio, e si cercherà di andare oltre ad esso cercando conferme oltre quel primo riscontro.
E'importante sottolineare come una netta discrepanza è presente qualora uno stesso messaggio sia proposto ad un pubblico femminile o ad uno maschile. Generalmente da studi effettuati, le donne cercando di catturare tutte le informazioni prima di definirne l'importanza, mentre gli uomini cercano da subito le informazioni salienti. Soffermandoci sugli adolescenti invece, generalmente sono sempre in linea con l'avanzamento tecnologico, per cui le grandi compagnie tentano di sfruttare questo, facendo leva su tendenze diffuse tra di essi. Non si sta facendo riferimento ai social network in generale, ma ciò che vi è alla loro base: la condivisione. Quest'ultima si riferisce al mondo del Viral Marketing. Come definisce l'AMA [7], con queste due parole viene descritto un fenomeno che facilita e incoraggia le persone nel trasmettersi un messaggio. E'definito *virale* perché il numero di persone esposte al messaggio imita il processo di passaggio di un virus. Con questa tecnica si cerca di dare la possibilità di condividere il messaggio con altri, oppure rendere un link accessibile a tutti [3]. I contenuti condivisi possono essere di vario tipo e affinché possano essere appetibile agli utenti, è necessario fare in modo che siano personalizzabili.

1.3 Personalizzazione del contenuto e del layout

La personalizzazione è un qualcosa che come il nome stesso esprime, riguarda il punto di vista estetico, in particolare, parlando di mobile advertising ci si riferisce all'interfaccia grafica dell'app che ospiterà il messaggio pubblicitario. In genere si considerano la lunghezza, il colore, una musica o un suono. [3] Ovviamente un'interfaccia *user-friendly*, può far sì che l'app possa essere più appetibile e fruibile. Ogni dispositivo ha una dimensione diversa e dovrà per forza di cose adattarsi ai diversi display e ad un set di colori che possono essere nettamente diversi rispetto ad altri, per cui la ricezione del messaggio può essere problematica qualora il messaggio non sia particolarmente performante per certi dispositivi, che magari costruiscono, sulla brillantezza dei colori il loro punto di forza di una o più delle loro pubblicità.
Ciononostante, quando si parla di personalizzazione ci si riferisce in questo contesto anche, forse soprattutto, al contenuto che la pubblicità vuole trasmettere, cercando di creare un identikit che meglio identifichi una tipologia di utente, rispetto quelle ormai obsolete che mirano invece alla massa.
Ciò può essere realizzato tenendo in considerazione diversi fattori come per esempio, interessi, cultura, stile di vita, bisogni oltre a fattori come età, sesso, reddito, hobby e posizione geografica [11][12].
Normalmente gli "advertiser" e gli utenti hanno una relazione chiusa tra di loro. Questi ultimi usano la navigazione da mobile per usufruire dei servizi di intrattenimento quali gossip, news sportive, shopping e altro. Con l'aggiunta di servizi wireless e tecnologia GPS è possibile mirare a un target più circoscritto di utenti in modo tale da potergli proporre delle offerte ad hoc per i loro gusti, riuscendo a creare l'identikit di un "utente modello". [11] Per cui il primo passo dovrà prevedere l'identificazione degli appena citati *target* in modo da intersecare le informazioni demografiche, ottenute con servizi wireless, con gli interessi della popolazione locale, per avere un vantaggio competitivo rispetto ad altri concorrenti. Se ad esempio un consumatore è

interessato al campo dell'informazione, sarà maggiormente incuriosito da informazioni su prodotti, servizi e compagnie; al contrario se è maggiormente interessato all'intrattenimento, bisognerebbe focalizzarsi su messaggi riguardanti piaceri estetici, esperienze sociali, il tutto per evitare pubblicità che non sono in linea con le "esigenze" del consumatore [11].

1.4 Cosa c'è dietro il Mobile Advertising ?

Dietro il mondo del m-advertising c'è tutto lo sviluppo tecnologico che riguarda i nuovi device, dove la comunicazione si è evoluta profondamente riuscendo adesso a riguardare non solo quella vocale ma anche quella di dati attraverso l'utilizzo di reti GSM prima e adesso con reti wireless. Nel caso che si sta trattando, ciò che conta maggiormente è sicuramente la velocità di connessione e l'utilizzo in maniera accurata dei sistemi di locazione.
Dal punto di vista economico, c'è la necessita di fare una pubblicità che sia in linea con la catena del valore degli attori in gioco. I partecipanti sono:

- L'inserzionista
- le società di pubblicità
- i proprietari dei media
- le tradizionali agenzie di pubblicità
- gli operatori di rete
- i fornitori di tecnologia
- i clienti

Il primo è sicuramente il ruolo più importante che entra in gioco, in quanto vengono valutate le performance, i canali usati, l'invenzione della pubblicità; i proprietari dei media invece sono i possessori dei database i quali contengono i potenziali clienti a cui inviare il messaggio pubblicitario; L'altro perno

importante riguarda la tecnologia usata per l'advertising, che si integra nel sistema delle telecomunicazioni, inoltre offre un importante strumento per valutare se il messaggio inviato ha avuto il suo effetto o meno cercando di dare un benefit, come possono esserlo gli incentivi: "ricompense finananziarie" per coloro che accettano di ricevere messaggi pubblicitari, senza dei quali il Mobile Advertising non avrebbe futuro.

1.5 Privacy & Sicurezza

Nel momento in cui un utente è soggetto alla ricezione di messaggi pubblicitari, avendo premesso che devono essere attrattivi per un generico soggetto, entra in gioco un'altra componente, a cui prestare molta attenzione. Stiamo parlando della *privacy*. Con questo termine si intendono le informazioni che non sono conosciute dagli altri utenti. Questi ultimi sono molto influenzabili sulla ricezione di un messaggio da parte di qualcuno che non conoscono. Discorso cambia se tali messaggi vengono inviati da compagnie commerciali. Dal punto di vista dei consumatori infatti l'invasione della privacy identifica uno dei principali ostacoli alla crescita del Marketing di prossimità. Considerando infatti che il m-advertising porta vantaggi nella vita di un consumatore, allo stesso tempo viene creato un profilo dello stesso lasciando porte aperte alla possibilità di intrusione all'interno della sua sfera privata
Da sottolineare inoltre è che ciò di cui si sta discutendo è nettamente diverso dal concetto di spam il quale rappresenta uno dei grandi rallentamenti di quel che il Marketing di prossimità vuole esprimere. Per spam infatti si intende l'invio di messaggi a un destinatario, solitamente tramite mail, di qualsivoglia tipo di contenuto, senza che il questi l'abbia esplicitamente richiesto con una conseguente occupazione di banda considerando che tali messaggi vengono inviati automaticamente a una moltitudine di consumatori [6].
Uno smartphone, come è ovvio che sia, non riesce a distinguere tra quella che è un messaggio di advertising autorizzato da un messaggio spam. In alcuni paesi, questi ultimi sono illegali. Come riconoscere la differenza? Per esempio

attraverso la frequenza con cui vengono mandati, il controllo, la confidenza indesiderata. Il Mobile Marketing Association ha stipulato sei principi da considerare per lo sviluppo di quest'ambito: *scelta, controllo, vincoli, considerazione, confidenza*. Tramite questi, si possono cercare di inculcare fiducia nei consumatori in modo da scongiurare la loro paura di possibili attacchi spam.

Il mobile advertising d'altronde rappresenta qualcosa di molto diverso: contenuti multimediali, interattività, strumento di comunicazione personale con gli utenti e altro. La grande sfida consiste quindi nel cercare di discernere il concetto di spam da esso.

Tornando sulla privacy in generala, ci sono delle normative che si occupano di regolarizzare il mobile advertising, questo perché come ripetuto più volte la creazione di un profilo apre le porte all'irruzione nella sfera privata di un utente. Magari messaggi che possono sembrare "poco intrusivi" per alcuni, in realtà lo sono per altri. Se a ciò si aggiunge che il campo della telefonia è quello che presenta una maggiore percezione su un eventuale uso o abuso dei dati personali, allora risulta rilevante conoscere quali leggi regolamentano la privacy. La direttiva Europea (95/46/EC) e la (2002/58/EC) proteggono le informazioni relative la privacy e in particolare quelle inerenti al campo del mobile, perché il consumatore deve sapere ed essere libero nel voler ricevere o meno questi avvisi pubblicitari. Per assicurare la trasparenza l'utente deve fornire nome e indirizzo e tutte le informazioni necessarie per poter usufruire di un trattamento equo (Direttiva 95/46/EC); deve essere sempre a conoscenza di eventuali modifiche, cancellazioni o accesso ai dati (Direttiva 95/46/EC, recital 41) e infine deve sapere cosa c'è dietro il trattamento dei sui dati (Direttiva 95/46/EC, art. 12) [12].

L'Unione Europea, inoltre ha emanato una nuova direttiva nel 2002 per regolamentare la trattazione di dati personali e la privacy ad esso connesse (Direttiva/58/EC). Ciò prevede il dover richiedere l'autorizzazione per l'invio di messaggi a scopo commerciali. Ciò è chiamato *opt-in Mobile Advertising*. In pratica , il pubblicitario può inviare messaggi promozionali, ma l'utente

deve dare esplicitamente il suo consenso, prima che il fornitore dei servizi li invii, oltre al fatto che in qualsiasi momento il consumatore può evitare la loro ricezione [12]. Al contrario si parla di *opt-out* nel caso il consumatore si oppone da principio alla ricezione dei messaggi. Ovviamente per certi aspetti, c'è sempre un limite che delle volte si supera o si deve superare; sebbene tutto questo aspetto dell'm-advertising possa essere visto come un ostacolo, ci sono eccezioni che riguardano quegli utenti che decidono di "permette" una maggiore invasione della privacy, magari a fronte di maggiori benefici.

Capitolo 2

Bluetooth Low Energy: Cos'è? Quali miglioramenti ha portato?

Nel prossimo capitolo si discuterà dei Beacon, device che utilizzano la tecnologia Bluetooth 4.0 o comunemente chiamata Low Energy (BLE). Per cui prima di addentrarsi alla scoperta di questi dispositivi e il loro utilizzo è importante fornire una visione dall'alto di questa tecnologia e sottolineare le differenze tra l'attuale versione e quella antecedente. Lo sviluppo dei sensori wireless ha spianato la strada a innumerevoli sfide da superare: tra cui l'impatto che hanno sul consumo della batteria; o anche la limitatezza che hanno in potenza di calcolo e memoria. Ciononostante sono stati messi a disposizione servizi prima impensabili. I Bluetooth vengono considerati i pionieri della standardizzazione di tecnologia wireless [15]. Pensati per offrire ai consumatori un metodo economico e sicuro di scambio dati tra diversi dispositivi attraverso una frequenza radio a corto raggio.
Sfortunatamente il primo protocollo, pubblicato nel 1999 rappresentava un grosso ostacolo a ciò che si auspicava di poter raggiungere, per una serie di motivi quali:

- Alto consumo energetico;
- Lenta connessione e in generale comportamento subottimale;

- Grande allocazione di memoria richiesta;
- Pacchetti da inviare eccessivamente grandi con conseguenti spese ingenti per la comunicazione.

Nonostante questi problemi la tecnologia è andata avanti migliorando sempre più fino ad arrivare all'ultima versione la 4.0 (esiste anche la 4.1 che risolve un problema di sovrapposizione di banda) la quale porta delle ingenti migliorie. Globalmente i bluetooth 4.0 si contraddistinguono per:

- Bassa larghezza di banda;
- Bassa latenza (intervallo di tempo tra input che arriva al sistema e il momento in cui è disponile l'output);
- Efficiente comunicazione.

Mettendo a confronto i due protocolli, è possibile evidenziare quelle che sono le principali differenze. La prima sicuramente riguarda il numero di canali usati che per quanto riguarda i "classici" bluetooth o per meglio dire i *Bluetooth BR*, il numero di canali che vengono sfruttati sono 111 con una larghezza di 1 MHz; al contrario i Bluetooth 4.0 o Bluetooth Smart, i quali hanno bisogno di sensori appositi per poter funzionare, permettono un uso su un massimo di 40 canali, larghi 2MHz. Entrambi i protocolli predispongono un numero di canali appositi per la scansione che per i BR è di 32, contro i 3 dei BLE. In quest'ultimo caso, usandone solo tre c'è una maggiore rapidità di connessione e non vi è interferenza [14]. In generale utilizzano lo stesso quantitativo di banda che è nell'ordine dei 2.4 GHz ISM, però i BLE possiedono una potenza di trasmissione che varia tra 0.01 mW (-20 dBm) e 10 mW (+10 dBm). Inoltre combinano due schemi di accesso uno basato sull'accesso multiplo chiamato Frequency Division Multiple Access (FDMA) che prevede diversi canali di frequenza per ogni flusso di dati; quindi una comunicazione ad ogni canale; mentre l'altro denominato TDMA (Time division multiple access), mira a fornire su uno stesso canale dei "time-slot" per ogni flusso di

dato [13]. Uno dei più grandi miglioramenti che ha portato il BLE è la semplicità del protocollo implementato, il che porta ad incentrare il tutto su un singolo microcontroller. Di conseguenza molti produttori lo adottano. A tal proposito, il diagramma di stato dei BLE, grafico che definisce i cambiamenti di stato, è molto semplice, da come si può osservare:

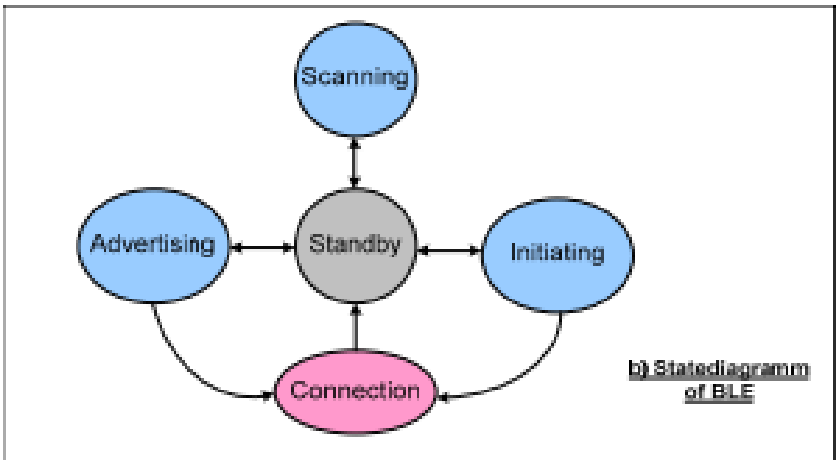


Figura 2.1: Descrizione del diagramma di stato bluetooth 4.0

Nello stato di Advertising i dati possono essere trasmessi via Broadcast ai device preposti alla connessione. Tra questi ultimi, quelli che si trovano nella fase di "Scanning State" e "Initialiting State" riceveranno pacchetti dai microcontroller pronti ad inviare. In quel momento, risponderanno al device trasmettitori e stabilita successivamente una nuova connessione. Quando è in fase di connessione si dice che i device si trovano in "Connection State". Questa fase può essere di due tipi:

Master Role: quando si passa da "Initiating State" a "Connection State";

Slave Role: quando ci si connette passando da "Advertising State" a "Connection State".

E'importante sottolineare come quando si è in Master Role vengono definiti i tempi di esecuzione. Ciò da cui si differisce particolarmente dai Bluetooth

BR riguarda la molteplicità di stati intermedi rispetto quelli appena descritti oltre al fatto che il numero massimo di dispositivi connettibili è otto, provocando una maggiore occupazione di banda, cosa che invece non avviene con i BLE i quali permettono un numero di connessioni simultanee nell'ordine di 2^{32}.

Nel momento in cui viene stabilita la connessione tra un microcontroller e un device che supporta la tecnologia BLE possono essere propagati dati su uno specifico canale. Questa azione è chiamata "connections event". Il tempo che intercorre tra due di esse: il "Connection Interval" varia da un minimo di 7.5 ms a un massimo di 4s. Quando ci si trova nella fase di Connection State è possibile mandare notifiche al dispositivo ricevente senza la necessità di stabilire una connessione, per cui tutto avviene in broadcast, cosa che non è possibile che avvenga, come si è asserito poc'anzi, quando c'è uno scambio di dati tra due dispositivi. In questo frangente un dato che bisogna sempre tenere sotto controllo è il *throughput*. Tale dato indica l'effettivo utilizzo del canale, da non confondere con la *capacità*, che al contrario indica la frequenza trasmissiva massima alla quale i dati possono viaggiare. Secondo i dati di Elke Mackensen, Matthias Lai e Thomas M. Wendt [15], si evince come il valore massimo di throughput sia 1.3 kByte.

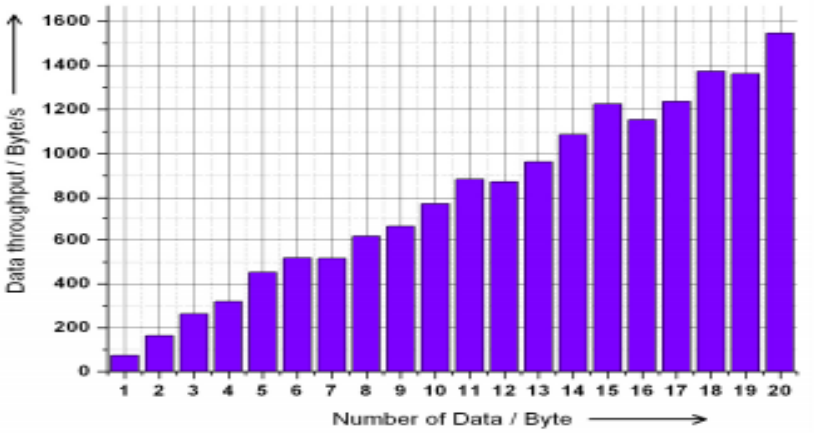


Figura 2.2: Misura del *Data-Throughput*

Generalmente un pacchetto inviato, nel caso specifico di un Beacon che si vedrà nel prossimo capitolo, consta di 4 parti. La prima parte e quindi il primo byte è detto *preamble* ed è un byte introduttivo. Segue l'indirizzo di accesso a cui si accede al pacchetto che è sempre uguale per tutti *(0x8E89BED6)*; il PDU che contiene al suo interno il carico utile di informazioni [26] oltre che l'indirizzo fisico (MAC) del dispositivo e infine il CRC per il rilevamento di errori [35]. Il PDU ha 16 byte per lo *header* e il restante numero di Byte è

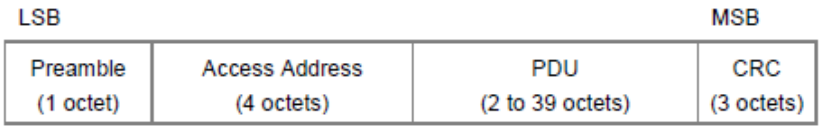


Figura 2.3: Formato data-packet

variabile e dipende dal carico utile che viene trasmesso. [35]
Un altro particolare in cui differiscono le due versione bluetooth prese in esame, si riferisce al numero massimo di pacchetti trasmissibili. Per i BLE sono 47 byte. Ciò rappresenta un punto a sfavore rispetto i BR, infatti per questa precedente versione i pacchetti trasmettibili ne erano 359 byte. Questa riduzione è stata giustificata dal ritardo di trasmissione con cui venivano ricevuti i pacchetti. Ulteriori differenze oltre quelle già menzionate sono riassunte nella seguente tabella:

Specifiche Tecniche	Bluetooth BR	Bletooth Low Energy
Frequenza	Tra 2400 e 2483.5 MHz	Tra 2400 e 2483.5 MHz
Data Channel	79	37
Advertising Channel	32	3
Criptaggio	$64/128bit$	AES 128 bit
Range	100 m	> 100m
Throughput	$0.7 - 2.1$ Mbit/s	< 0.3 Mbit/s
Latenza di Connessione	≈ 100 ms	6 ms
Tempo minimo di invio dati	100 ms	3 ms
Potenza di consumo	1 W	Tra 0.01 a 0.5 W
Picco di corrente	$22 - 40$ mA	$10 - 30$ mA

Tabella 2.1: Differenze tra Bluetooth BR & Bluetooth 4.0

Avendo fatto queste premesse, ci si apetta un consumo nettamente migliore dei BLE. In realtà da come si apprende [14], il risparmio è meno del 10%, se comparato con i classici Bluetooth. In aggiunta un dato che fa ancora più riflettere riguarda il consumo di energia di un app che utilizza la tecnologia beacon. Infatti tale consumo è lo stesso sia che si usi i BLE, sia che si usi i BR. Questo problema si ha perché le API di Android non possono utilizzare esclusivamente i bluetooth 4.0. Di conseguenza nel momento in cui viene ad essere utilizzata l'interfaccia 4.0, simultaneamente viene ad essere attivata anche quella classica. In conclusione si può asserire che l'ultima versione del protocollo ha portato si miglioramenti, ma che la strada per avere un servizio ottimale ed efficiente e ancora lunga.

Capitolo 3

Tecnologia Beacon

Il precedente capitolo rappresenta il collante che unisce informatività, su una tecnologia importante e diffusa da dover obbligatoriamente menzionare a un potenziale tecnologico, che sta prendendo piede quello dei Beacon che incentra il suo successo sull'adozione dei BLE sfruttando i vantaggi (e ovviamente anche gli svantaggi) ad essa legati.
Nel mondo attuale si sente spesso parlare di *Internet of Things (IoT)*. Con questo termine si descrive un nuovo modo di connettersi con il mondo reale. Alla base c'è il voler esprime l'idea di interconnessione tra gli esseri umani e gli oggetti, nei più svariati campi: sanitari, economici, logistici e manifatturieri [16]. I Beacon possono considerarsi come un modo per poter "dar' vita" agli oggetti e il loro uso è assai vasto. Oggi forse ancora non si conoscono appieno i possibili usi che sono in grado di offrire. Più avanti saranno menzionati infatti solo quelli più consueti che se ne fanno oggi e per tale motivo si descriverà nel dettaglio l'implementazione di un progetto Android che mira a tener traccia e localizzare oggetti. Volendone dare una definizione universale, i Beacon sono dispositivi dotati di batteria, (eventualmente possono far riferimento a batterie esterne per alcune versioni), che utilizzano tecnologia bluetooth, precisamente la 4.0 e che grazie a ciò riescono a trasmettere segnali attraverso un microcontroller agli smartphone o a qualunque altro device preposto alla ricezione. Apple lo definisce come "un' eccitante nuova tecno-

logia" che consente alle app di conoscere la posizione in cui sono collocati [22]. Ce ne sono di svariati tipi, alcuni nettamente diversi tra loro per forma, dimensione e funzionalità che sono in grado di offrire, se si pensa per esempio alla possibilità di rilevamento della temperatura. In questo capitolo e nei successivi, si farà riferimento a un *Jaalee con chip N51822 e batteria CR2032* con il quale sono stati effettuati alcuni esperimenti per poterne testare il funzionamento, e con cui è stato realizzato il progetto poc'anzi introdotto. Nel dettaglio si presenta nel seguente modo: Da come si può notare, le estremità

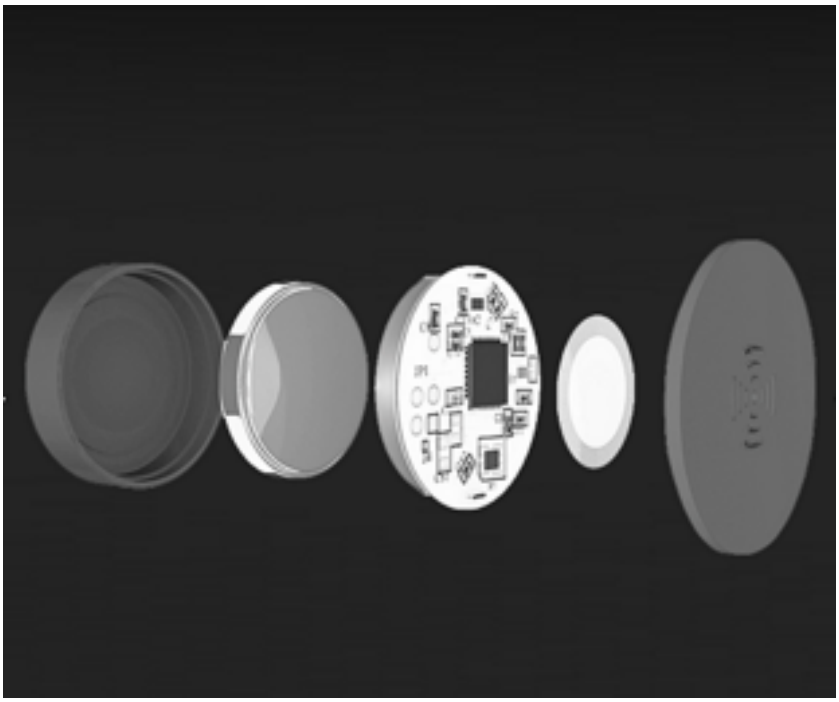


Figura 3.1: Esempio di un beacon nel dettaglio

di un beacon servono a dare un aspetto estetico più gradevole e a proteggerne il circuito. Costituito quest'ultimo da un chip Bluetooth e da una batteria rimovibile.

Esistono vari modi per interagire con esso: *connettendosi* e successivamente recuperando le informazioni utili; *monitorando* la sua regione di competenza, entro la quale può essere fornita la stima della distanza; *informare* gli utenti

della loro presenza con l'ausilio di notifiche; *scansionando* i fari disponibili per poter cercare quello di interesse. Uno stesso dispositivo può riuscire a ricoprire una superficie che va dai 30 metri, denominati in questo caso a corto raggio, o in alternativa fino ai 75 metri (a lungo raggio) [22]. Se per esempio ci si trova all'interno della regione lo scanner rileverà i dispositivi disponibili e calcolerà la distanza tra il beacon e lo smartphone, la quale può essere raggruppata in 4 macro gruppi (nel caso di utilizzo di un Beacon a corto raggio):

- Sconosciuta: distanza maggiore di 30 metri;
- Lontano: entro i 30 metri;
- Vicino: entro i 2 metri;
- Immediato: entro i 50 centimetri;

Da sottolineare come il valore della distanza sia calcolato tramite l'RSSI: potenza del segnale del dispositivo in scansione [24]. Ovviamente l'accuratezza aumenta nel momento in cui aumenta la potenza del segnale.
Generalmente viene venduto con un adesivo per essere incollato su una qualsiasi superficie, la quale di solito è un muro. Sebbene questo particolare possa essere considerato irrilevante, in realtà capire il motivo di questa scelta giustifica il suo principale uso di cui si è dibattuto nel primo capito: il Mobile Advertising. Di norma viene adoperato nei musei, negli aeroporti, e negli store per offrire un servizio qualitativamente migliore ai consumatori permettendo per esempio di evitare code o come strumento per invogliare all'acquisto di qualche prodotto senza poi contare che Apple, che dal 2013 è stata la prima azienda ad introdurlo, lo usa come mezzo di pagamento. Con il tempo anche altri produttori tra i quali ovviamente Android, stanno pian piano iniziando ad utilizzarlo, in considerazione del fatto che la tecnologia NFC diretta concorrente di quella beacon non si è diffusa come si sperava, al punto tale che l'azienda di Mountain View ha ideato un suo protocollo di trasmissione per l'invio e la ricezione dei dati chiamato Eddystone che sarà approfondito più avanti.

3.1 Integrazione dei Beacon con gli smartphone

Avendo visto cosa sono i Beacon, ora si necessita la spiegazione di quelli che sono i formati principali di trasmissione dati. Come suggerisce Google [25] i più importanti sono:

- iBeacon
- Eddystone
- AltBeacon

Cosa contraddistingue ogni formato, dipende dal contenuto del PDU che è alla base e che verrà successivamente ricevuto dagli smartphone.

3.1.1 iBeacon

Soffermandosi sul quello iBeacon, generalmente i bit che contengono le informazioni utili prevedono:

prefisso: serie di byte che identifica la compagnia fornitrice;

UUID: sequenza di byte i quali permettono di distinguere vari tipi di Beacon;

Major: byte che raggruppano un set relativo di Beacon;

Minor: identificativo univoco di uno specifico beacon

Tx Power: indice identificativo di vicinanza ad un Beacon. Esprime quanto si è vicino ad esso.

La sequenza di bit relativi l'ID della compagnia non sono modificabili. Ogni pacchetto ha una lunghezza definita che secondo alcuni studi [26] risulta essere di 30 byte (9 per il prefisso, 16 per l'UUID, 4 totali tra Major e Minor e 1 per il Tx Power), mentre per altri [23] il numero dei byte che sancisono le informazioni utili risulta essere 25 (2 byte per l'ID, 1 per il tipo

di dato, 1 per la lunghezza dei dati, UUID, Major, Minor e Tx Power come nel caso precedente). Graficamente, nel caso in cui si prenda come modello esemplificativo quello da 25 byte, un semplice esempio di rappresentazione di un pacchetto si presta nel seguente modo:

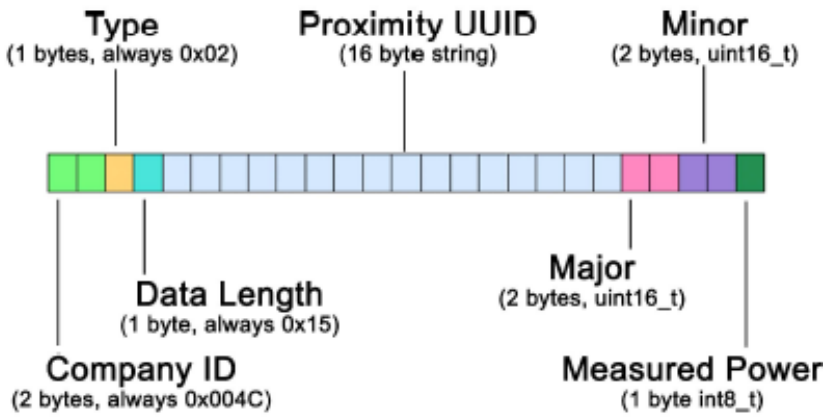


Figura 3.2: iBeacon Data Format

3.1.2 AltBeacon

Un formato alternativo di utilizzo, il quale è stato usato anche per l'implementazione del progetto, come si vedrà più avanti è quello AltBeacon [27]. Nato perché prima della sua creazione come si legge dal sito ufficiale non esisteva ancora una specifica aperta e interoperabile. Il suo obiettivo è quello di definire un formato dati che sfrutta la tecnologia Bluetooth Low Energy con l'intento di creare un mercato competitivo e aperto, sfruttando l'open source ed esentandolo da tasse, senza cercare di favorire un fornitore piuttosto che un altro, e per questo motivo, diverso degli standard tecnici in uso. Come per il formato iBeacon, e come tutti gli altri, si pone come obiettivo l'acquisizione delle informazioni dei beacon al fine di riuscire ad interagire con esso e dare la possibilità agli utenti di calcolarne la distanza. Lo sviluppo di questa specifica è stata dettata dai seguenti obiettivi [28]:

- Fornire un messaggio di advertising conciso per lo scambio di informazioni tra i pubblicitari e i consumatori;
- Mantenere la conformità con le specifiche bluetooth 4.0;
- Incoraggiare l'adozione da parte di tutti gli interessati per evitare restrizioni;
- Renderlo *open-source* dando la possibilità agli sviluppatori di aggiungere caratteristiche;

Gli annunci pubblicitari o qualunque altra informazione che si vuole rendere disponibile è incapsulata nella struttura PDU. Il *data packet* è costituito da 1 byte come indice di lunghezza, 1 byte per il tipo, 2 byte per la compagnia produttrice, 20 byte per l'UUID, 1 byte per rappresentare il valore RSSI e 1 byte che viene lasciato libero per poter aggiungere eventuali feature *(MFG RESERVED)*.

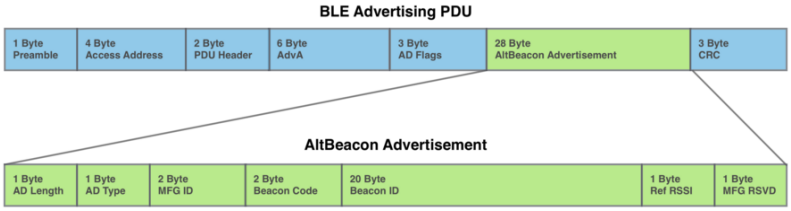


Figura 3.3: AltBeacon data format

3.1.3 Eddystone

L'ultimo formato ancora di cui non si è parlato è il quello Eddystone creato da Google. Quest'ultimo permette l'interoperabilità oltre che con Android anche con iOS. A differenza dei precedenti formati quello Google include un servizio cloud chiamato Proximity Beacon API [29] che conferisce la possibilità di gestione di dati associati ai beacon utilizzando un'interfaccia REST. I dati possono essere associati ad un beacon sotto forma di allegati. Questi

ultimi sono dati *blob* rimandati alle app Android o iOS attraverso le API Nearby [30] (API che consentono il passaggio di dati attraverso una connessione Internet in cui sono connessi i dispositivi Android e iOS combinando tecnologia Bluetooth 4.0 e Wi-Fi). Con il Proximity Beacon API è possibile tenere sotto controllo lo stato di salute del Beacon, verificando il suo livello di batteria, visualizzando la latitudine e la longitudine e altre operazioni.
Il formato Eddystone prevede un payload di dati trasmissibili che si differenzia dai precedenti. A tal proposito le specifiche [31] asseriscono quelli che sono stati gli obiettivi che hanno guidato gli sviluppatori:

- Interoperabilità delle API Google con dispositivi Android e iOS;
- Semplice implementazione su una vasta gamma di dispositivi esistenti BLE;
- Architettura flessibile per conferire elasticità ai pacchetti da inviare
- Piena compatibilità con le specifiche Bluetooth

Ogni frame Eddystone contiene un PDU conforme a quelle che sono le specifiche Bluetooth che prevede la dichiarazione di 16 byte per l'UUID, più 1 byte per il tipo. Per quanto riguarda quest'ultimo punto i tipi disponibili sono:

UID: composto da 16 byte, 10-byte per il namespace e 6-byte per l'istanza. Il primo è destinato a garantire un ID univoco attraverso multiple piattaforme Eddystone. E'utilizzato per filtrare i beacon sottoposti a scansione. [32]

URL: possibilità di compressione di qualsivoglia URL, mandato attraverso il payload a cui l'utente potrà successivamente accedervi. Una volta decodificato, un generico user con un qualsivoglia client potrà accedervi tramite un browser a sua scelta; [33]

TLM: formato dati usato per avere sotto osservazione la telemetria del beacon. Questo tipo di osservazione prevede la possibilità di monitoraggio

dello stato di salute del beacon. Non è munito di un ID proprio ma viene accoppiato ad un frame che ne possiede uno (UID o URL)[34];

Questi valori appena descritti a cosa servono nel concreto? Si immagini una catena di centri commerciali, che utilizzi un numero arbitrario di beacon per ampliare il potenziale di acquisto da parte dei consumatori. La seguente tabella [22] mostra a cosa possano servire valori appena descritti:

Store Location		**San Francisco**	**Paris**	**London**
UUID		D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C		
Major		1	2	3
Minor	**Clothing**	10	10	10
	Housewares	20	20	20
	Automotive	30	30	30

Tabella 3.1: Esempio di un possibile utilizzo di paramentri di un beacon

L'UUID è condiviso da tutti gli store di una stessa catena di centri commerciali. Questo permette ai device in uso di identificare univocamente ognuno di essi in una singola regione. Seguendo l'esempio, per ognuno di essi: San Francisco, Parigi e Londra è assegnato un valore univoco (il Major) che lo distingue dall'altro. Entro ogni singolo store, sono disposti dei beacon con UUID e Major assegnati nel modo in cui è stato descritto, ma il valore Minor è diverso per ogni reparto. In questo modo uno smartphone leggendo questo raggruppamento riesce a sapere dove si trova e permettere l'invio di notifiche relative nel reparto di competenza.

3.2 Tecnologia & Specifiche tecniche

3.2.1 Batteria

La maggior parte dei Beacon come detto fa uso di batterie che conferisce loro autonomia per un periodo stimato di circa due anni. Tale stima ciòno-

nostante è da considerarsi relativa e si basa su una serie di fattori settabili, uno fra tutti la potenza del segnale *Tx Power* ossia il valore della potenza del segnale in uno spazio di un metro. Se in una situazione normale viene propagato 1 segnale ogni 1000 ms, allora la batteria potrebbe durare quanto indicato; se al contrario venisse impostato il dispositivo in modo da diramare un segnale ogni 100 ms, significherebbe che verrebbero propagati 10 segnali al secondo con un conseguente consumo di batteria 10 volte maggiore, il che potrebbe causare il consumo della stessa solo dopo pochi mesi [23]. La potenza del segnale non è l'unico fattore che incide sul consumo dei beacon, infatti vi è differenza tra scansioni effettuate in *background* o in *foreground* dove i consumi differiscono di $\approx 30\%$ [14].

3.2.2 Prestazioni inerenti le performace di trasmissione

Altri due parametri chiave dei beacon sono la *frequenza di trasmissione* o *intervallo di scansione* indicato con (T_s) e il *duty cycle* indicato con (f_D). Da premettere che normalmente è possibile impostare in un'applicazione che utilizza i beacon il suo periodo di attività permettendo quindi la determinazione dei momenti in cui il beacon può mandare segnali agli smarphone adibiti alla ricezione. Il primo parametro (T_s) indica proprio questo aspetto. Il secondo (f_D) esprime il rapporto tra il periodo di scansione e la frequenza di trasmissione o ancora quanto tempo la scansione risulta effettivamente essere attiva.
Se settati nel modo corretto questi due parametri conferiscono una maggiore autonomia al dispositivo o per meglio dire fanno sì che il consumo giornaliero sia al massimo del 30%. Se contrariamente vi fosse necessità di scansioni continue, sarebbe necessario settare il l'intervallo di scansione intorno ai 5 secondi per evitare che vi sia un consumo eccessivo [14].

3.2.3 Controindicazioni

Nell'utilizzo dei beacon ci sono anche alcune questioni da tenere in considerazione. Una ad esempio riguarda il formato con cui vengono inviati i dati. Infatti per i dispositivi Apple questo formato non è settabile. L'unico modo per poterlo cambiare è implementare un proprio formato dati Bluetooth che per esempio permette di aggiungere e registrare alcune funzioni come può essere il sensore di temperatura [1].
Un altro problema da considerare attentamente riguarda come le onde si propagano e soprattutto quali sono i materiali che arginano la potenza del segnale. Normalmente i raggi vengono trasmessi con un segnale di 2.4 GHz, ma il corpo umano o i muri tendono ad attenuare la forza di trasmissione. Apple ha realizzato uno studio [23] dove viene spiegato quali sono i materiali che assorbono o rifrattano meglio il segnale.

Tipo di materiale	**Interferenza potenziale**
Legna	Bassa
Materiali sintetici	Bassa
Vetro	Bassa
Acqua	Media
Mattoni	Media
Marmo	Media
Malta	Alta
Calcestruzzo	Alta
Vetri antiproiettili	Alta
Metallo	Molto alta

Tabella 3.2: In ordine di interferenza sono elencati i materiali che riescono ad assorbire le onde propagate dai beacon

Il risultato di questo esperimento espone che il materiale che attenua maggiormente il segnale è il metallo. Se considerassimo l'acqua come esempio, ad

[1] I beacon Gimbal permettono di monitorare la temperatura. http://www.gimbal.com/

una distanza di 0.10 metri tra un beacon e uno smartphone e anteponessimo tra loro ad una distanza di 0.04 m uno strato d'acqua, la distanza calcolata che si ricorda dipendere dall'RSSI, è di 0.49 m. Se invece si inserisse una mano tra i due dispositivi, la distanza risulterebbe essere di 1.50 m.

3.3 Stato dell'arte dei Beacon

La domanda che molti si pongono, forse, è sapere chi utilizza questo dispositivo, se è diffuso e quali sono le vendite che ne derivano. Per quel che riguarda le vendite un report della rivista *Business Insider* ha stimato che nel 2015 il valore che è stato in grado di generare si aggiri sui 4 miliardi di dollari, influenzando quindi 1% del totale degli acquisti e che nel 2016, questo dato crescerà circa 10 volte di più, portando le vendite e 44 miliardi. In questo frangente il peso maggiore lo assumono i coupon proposti tramite messaggi ai consumatori.Ognuno di essi può essere personalizzato, in linea con il principi del Mobile Advertising, per creare un esperienza interattiva che racchiuda *intrattenimento* ed *informatività*, come è stato asserito nel primo capitolo [17]. Le modalità di utilizzo sono svariate. Principalmente si identificano tre luoghi dove si può venire a contatto con il mondo Beacon.

3.3.1 Musei

Nei musei può essere usato per creare un'esperienza interattiva. Nel momento in cui si entra viene mandata una notifica di benvenuto agli smartphone dei visitatori che hanno preinstallato l'app suggerita e attivato i Bluetooth. Una volta all'interno, possono essere mostrate le varie esposizioni disponibili, sceglierne una ed essere guidata fino ad essa seguendo il percorso più congeniale e più rapido. In aggiunta per offrire un servizio qualitativamente migliore può essere implementa la possibilità di ricezione video, immagini o giochi inerenti il contesto in cui ci si trova. Sui bambini potrebbe far leva l'aspetto ludico e quindi ad esempio essere guidati fino all'opera con una caccia al tesoro. Allo stesso tempo i genitori potrebbero monitorare la posizione

dei loro figli i quali hanno conservato con loro un dispositivo e individuarli con il sistema di locazione di cui ne è dotata l'applicazione che ne fa uso. Le finalità sono quelle di invogliare e stimolare le persone, a rendere più accattivante e allo stesso tempo più sicura un'esperienza come la visita ad un museo soprattutto per i più piccoli che sono i più restii ad andarci.

3.3.2 Centri commerciali

Nel mondo dello shopping si è sempre alla ricerca di soluzioni, politiche di marketing, strumenti che riescano a incrementare i ricavi di vendita. I beacon rappresentano una risposta a queste esigenze. Si immagini una situazione ideale che consenta un modo tutto nuovo di fare acquisti: si comincia all'entrata, nel momento in cui un utente entra nel centro per poter parcheggiare. La barriera che consente il transito all'interno del centro viene azionata direttamente dall'app. Una volta entrato l'utente sarà guidato fino al primo parcheggio disponibile (sede in cui è posto uno dei tanti beacon). Attraverso il sistema GPS che il telefono utilizza viene calcolata la distanza rispetto l'ingresso del centro commerciale e quindi stabilito un percorso che lo guidi. Li riceverà una notifica la quale informa sulle possibili offerte, news, e tant'altro. Una volta all'interno di uno dei tanti store del centro commerciale, nel caso in cui si decida di acquistare un oggetto di interesse, sarà possibile pagare tramite l'app (sistema attualmente disponibile solo su iOS). A quel punto una volta usciti dal range di competenza dei beacon, l'applicazione ricorderà dove si era parcheggiato e ricondotti fin li. Anche in quel caso l'utente avrà la possibilità di pagare tramite l'applicazione stessa.

3.3.3 Aeroporti

Negli aeroporti invece possono essere considerati strumenti di tracciamento dei bagagli. Una visione esemplificativa potrebbe essere quella che prevede dei beacon inseriti all'interno di ogni valigia di una famiglia in procinto di partire.

La prima operazione da fare è settare l'UUID, il Major e il Minor. L'UUID è univoco, per ogni tipo di famiglia, indicativo del contesto. Ciò che cambia riguarda il Major, diverso per ogni famiglia cosi come i valori Minor segno di distinzione per ogni componente. La situazione descritta sarà quindi la medesima:

Famiglia		**Famiglia 1**	**Famiglia 2**	**Famiglia 3**
UUID		8803EC43-D405-14E3-93A7-3B514933A4B1		
Major		10	20	30
Minor	**Padre**	1	1	1
	Madre	2	2	2
	Figlio	3	3	3

Tabella 3.3: Esempio di un possibile utilizzo di paramentri di un beacon all'interno di un aereoporto da parte di una famiglia

Ricordando che ci sono diversi modi di utilizzo dei beacon tra i quali uno come sistema di monitoraggio in background e l'altro come strumento di scansione, nel momento in cui si entra nella regione di interesse viene mandata una notifica ai telefoni che individuano in background il "faro". In alternativa potrebbe essere realizzata una scansione per cercare un particolare dispositivo. In ogni caso, viene letta la distanza che divide un componente della famiglia da un proprio bagaglio. Se l'implementazione lo consente, creata una mappa che sfrutti la forza del segnale RSSI per la loro localizzazione. Con questo sistema si è in possesso di uno strumento che sia in grado di evitare eventuali smarrimenti o in alternativa il suo uso può essere un espediente per ridurre sensibilmente il tempo di attesa nelle code all'interno degli aeroporti.

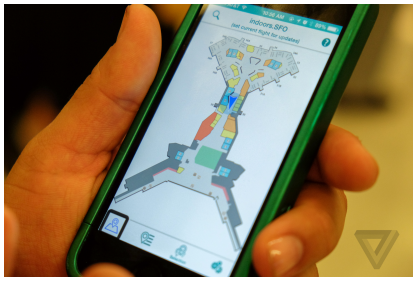
3.4 Casi pratici: realtà di utilizzo dei Beacon

In Italia l'utilizzo dei Beacon è ancora inesplorato, o per lo meno sono ancora poche le compagnie che ne fanno uso. Questo perché la sua diffusione

si è avuta soprattutto negli Stati Uniti, sua patria. Ad oggi però le aziende italiane che sfruttano la tecnologia apportata da questo strumento stanno crescendo [2]. Negli USA sono invece molte le aziende che impiegano effettivamente già da ora ciò che tali dispositivi sono in grado di offrire.
Macy's, uno dei più grandi distributori al dettaglio, fondata nel 1858, ha acquistato più di 4000 dispositivi da installare nei sui store come politica di marketing. Il loro utilizzo è intento ad offrire offerte migliori a livello di reparto, sconti, consigli e ricompense [18]. Sulla stessa scia, Lord & Taylor, antico negozio di lusso americano dichiarò nel 2014 tramite il suo vicepresidente l'intento di installare un numero di dispositivi, limitato per ogni store, (fino a 3 o 4 per non assalire il consumatore) al fine di poter dare consigli, oppure proporre offerte [19]. Per raggiungere tale fine sono stati integrati applicazioni come SnipSnap, un app creata per proporre offerte di acquisto. Ancora, la Major League di Baseball Americana ha concesso alle squadre di baseball di installare beacon all'interno dei propri stadi, equipaggiando gli spettatori di mappe, video clip o per tenere aggiornati i posti a sedere [20]; Uno dei pochi esempi italiani, ma non unico, che usa tecnologia beacon è rappresentato dal Perugia Calcio integrando due servizi con un duplice scopo: mandare notifiche ai frequentatori dello stadio con formazioni in anteprima e ogni qualsiasi tipo di news che la società decide di divulgare, e in aggiunta aprendo una nuova strada al mondo della sponsorizzazione che prevede l'acquisto di spazi dove le società pubblicitarie possono mandare notifiche ad hoc all'interno dello stadio [21]. Un altro esempio italiano è il museo di Palazzo Farnese a Piacenza. L'uso che ne viene fatto è quello classico dei musei: posti in prossimità delle opere d'arte, con un lavoro altamente qualitativo, vengono descritte le opere a cui si è dinnanzi, anche sfruttando l'aiuto di una radio guida. Soffermandoci sui musei, in Belgio, la casa-museo di Rubens offre un'esperienza altamente coinvolgente. Al suo interno, l'app riporterà la storia del sagrato che si imbocca inizialmente. Andando avanti, saranno

[2] All'interno di questo sito sono tenute aggiornate tutte le innovazioni italiane in questo campo: http://www.beaconitaly.it/

narrate curiosità riguardanti le varie opere in cui ci si imbatte attraverso giochi a quiz. L'elemento ludico è alla base della visita in modo da stimolare continuamente il visitatore [43]. Volendo menzionare un esempio di utilizzo all'interno di un aeroporto, la nota rivista scientifica *The Verge* menziona l'aeroporto di San Francisco il quale sfrutta il segnale proveniente dai Beacon per poter definire una mappa utilizzata dai viaggiatori per muoversi all'interno del gate. Ogni Beacon si connetterà all'app predisposta alla ricezione e manderà informazione di ogni genere. Per i non vedenti viene sfruttata la tecnologia *VoiceOver* di Apple la quale indicherà i punti di interesse [42].



(a) Beacon all'interno degli ereoporti

(b) Beacon all'interno di musei

(c) Beacon all'interno di store

Figura 3.4: Esempi di utilizzo dei Beacon

Capitolo 4

Test sperimentali sui beacon

Testare ciò che sono in grado di offrire rappresenta il passo successivo per conferire autorevolezza e garanzia delle loro performance. Qui di seguito saranno mostrati alcuni dati ottenuti da esperimenti che descrivono il comportamento che assumono in circostanze normali. In primis, saranno presentati i dati che sanciscono come varia l'RSSI in relazione alla distanza reale. Successivamente saranno mostrati i dati che mettono a confronto la distanza reale da quella proposta dalla libreria AltBeacon per capire di quanto ci si discosta e infine, a tal proposito, sarà calcolato, il *miss rate* con il quale si intende il numero di pacchetti che non arrivano a destinazione. Per effettuare i test eseguiti sia in ambiente indoor che outdoor è stato utilizzato un Samsung Galaxy S6 che monta la versione Android Lollipop 5.1.1, munito di chipset bluetooth Broadcom BCM4773. Per la loro realizzazione inoltre è stata implementata un'applicazione apposita con l'ausilio dell'IDE Android Studio 1.4 pensata per l'esecuzione delle seguenti operazioni:

- scansione dei beacon visibili:
- acquisizione delle informazioni necessarie
- stampa su un file.txt

Per poter operare sui beacon, si è utilizzato due librerie: quella nativa del beacon Jaalee [1] in uso, più in aggiunta una seconda, la quale sfrutta il protocollo AltBeacon [2]

4.1 Test 1: Variazione dell'RSSI in funzione della distanza

Il primo esperimento, come detto, prevede di analizzare la variazione dell'RSSI in funzione della variazione della distanza sia in ambiente chiuso che in uno spazio aperto. In entrambe le circostanze, ad ogni spostamento sono stati presi n valori per poter meglio gestire l'andamento oscillatorio dell'RSSI; successivamente calcolata la media aritmetica:

$$\frac{\sum_{k=0}^{number\_of\_trials} RSSI}{number\_of\_trials} \tag{4.1}$$

In questo esperimento *number_ of_ trials* è settato a 20 in considerazione del numero di misurazioni effettuate. Per quel che riguarda lo spazio indoor, i dati che sono stati ottenuti e il relativo grafico sono i seguenti:

[1] Libreria Jaalee scaricabile dal seguente indirizzo, che contiene oltre alla libreria disponibile contenuta nella cartella bin, anche una demo: https://github.com/jaalee/Jaalee-Beacon-Android-SDK-master

[2] Libreria AltBeacon. Fornisce una guida per installarla in Android Studio o in Eclipse più una serie di guide su come utilizzarla: https://altbeacon.github.io/android-beacon-library/configure.html

Distanza (m)	Media RSSI (dB)
2	-76
4	-91
6	-91
8	-89
10	-93
12	-96
14	-89
16	-96
18	-96
20	-94

Tabella 4.1: Dati indoor primo esperimento

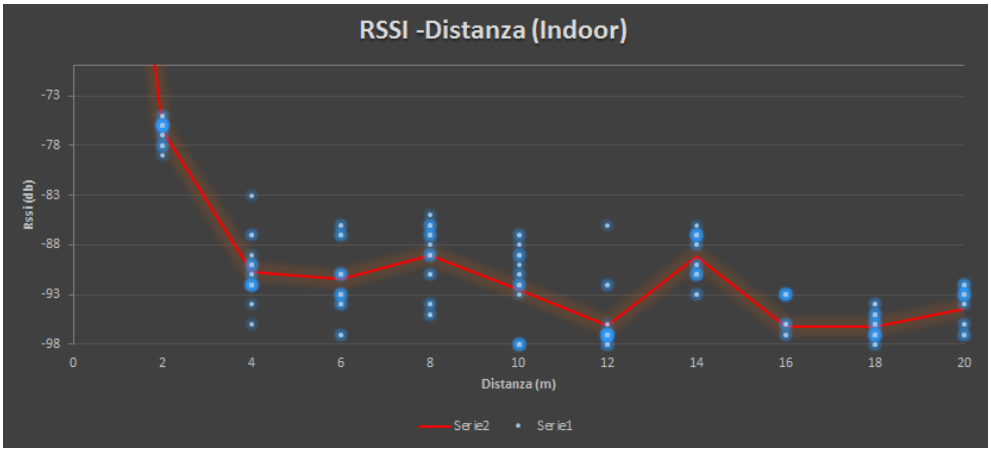


Figura 4.1: Grafico che mostra il valore RSSI in funzione della distanza per gli scenari indoor

Mentre per gli spazi outdoor i dati e i grafici ad esso associati sono:

Distanza (m)	Media RSSI (dB)
2	-90.3
4	-91
6	-91
8	-91
10	-91
12	-91
14	-91
16	-91
18	-91
20	-91

Tabella 4.2: Dati outdoor primo esperimento

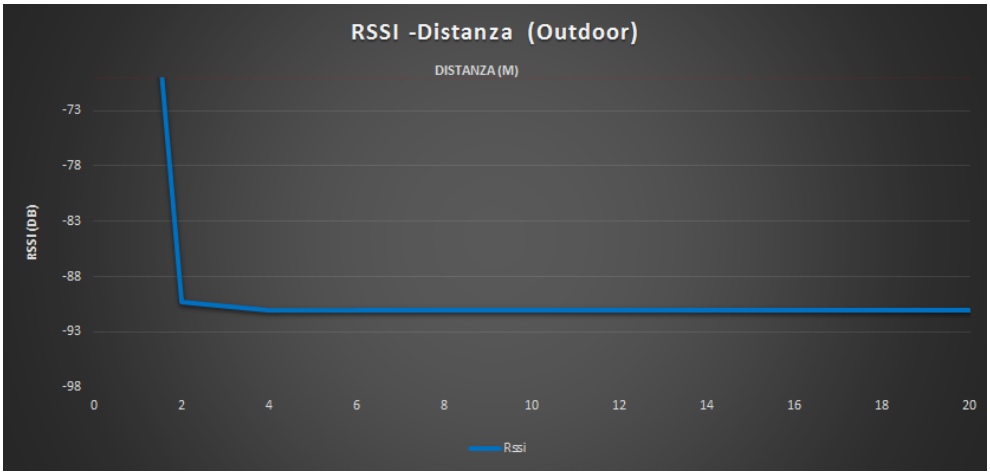


Figura 4.2: Grafico che mostra il valore RSSI in funzione della distanza per gli spazi aperti

In entrambe le circostanze, il periodo di osservazione per ogni distanza è stato di 2 minuti. I due grafici sono nettamente diversi tra loro. Ciò può dipendere, nel caso degli scenari outdoor, dalla dispersione del segnale che ha luogo già dai due metri restando debole e costante per tutto il resto dell'espe-

rimento; situazione che non si presenta per gli scenari indoor dove andando nello specifico l'andamento risulta essere oscillatorio e mostri picchi, soprattutto tra i 12 e i 14 metri.

4.2 Test 2: Analisi dell'errore

Nel secondo esperimento si è utilizzata la libreria AltBeacon la quale permette la scansione di tutti i tipi di Beacon a prescindere di chi sia il produttore, semplicemente settando un parser. Essendo una libreria diversa rispetto quella natia, è sembrato opportuno effettuare l'esperimento precedente per verificare se la libreria incidesse in qualche modo sul segnale RSSI. Dopo di che sono state messe a confronto le distanze: quella proposta dalla libreria e quella calcolata sperimentalmente e per concludere calcolato l'errore ossia di quanto differiscono le due distanze, secondo la seguente formula:

$$\frac{|d_{\text{reale}} - d_{\text{segnata}}|}{d_{\text{reale}}} \tag{4.2}$$

Da premettere che tale distanza in uso in questo secondo test, viene definita sulla base del seguente calcolo [3]:

$$d = A \cdot (\frac{r}{t})^{B+C} \tag{4.3}$$

dove:

- R = RSSI
- t = Riferimento RSSI a 1 metro di distanza
- A, B, C = costanti

I seguenti dati racchiudono le sperimentazioni effettuate secondo gli obiettivi prestabiliti.

[3]Formula della distanza suggerita da: https://altbeacon.github.io/android-beacon-library/distance-calculations.html

Distanza (m)	Distanza Indicata (m)	RSSI (dB)	Errore distanza (m)	Errore (%)
1	6,61	-78	5,61	561
2	6,97	-80	2,49	249
3	7,59	-88	1,53	153
4	7,85	-83	0,96	96
5	8,2	-94	0,64	64
6	9,57	-91	0,60	60
7	11,66	-97	0,67	67

Tabella 4.3: Dati relativi l'errore delle distanze

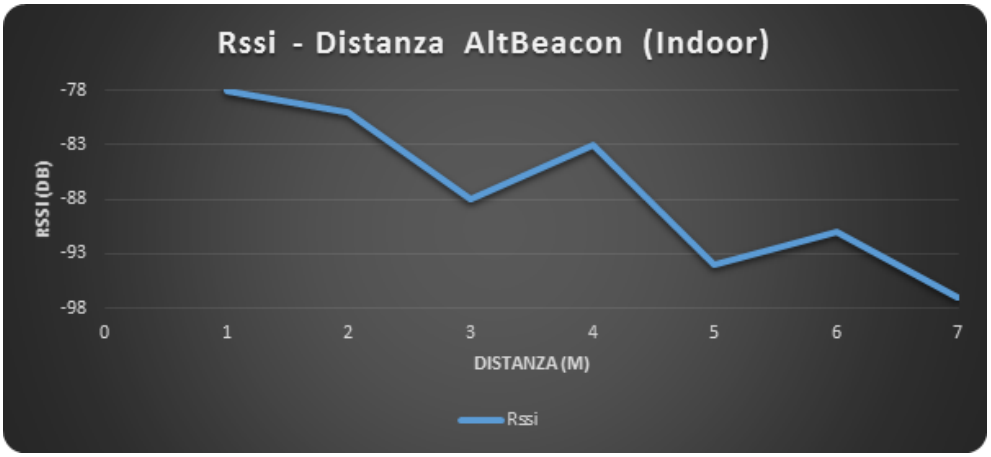


Figura 4.3: Variazione della distanza al variare dell'RSSI con il formato Altbeacon

A prima vista dall'andamento del grafico il segnale RSSI non sembra che si discosti troppo dal segnale esaminato nel test precedente. Ciò che balza agli occhi è la distanza proposta. In ambiente chiuso, sebbene ci si trovi a pochi metri lontani dal beacon, viene settata una distanza nettamente maggiore di quella reale, la quale sembra crescere in modo proporzionale al crescere della

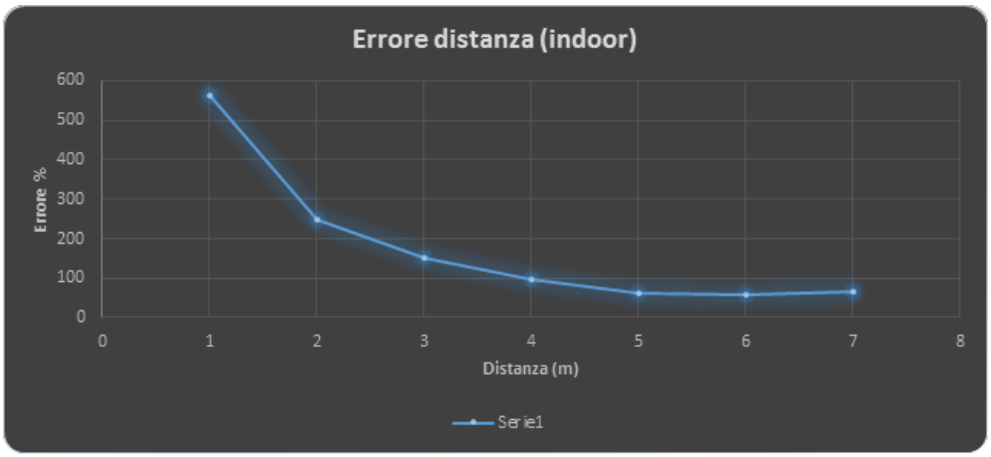


Figura 4.4: Variazione dell'errore al variare della distanza

distanza reale. Il grafico dell'errore mostra come in generale l'errore sia alto, ma tende ad assottigliarsi quando ci si allontana come è prevedibile.
Per gli esperimenti outdoor, i dati sono i seguenti:

Distanza reale (m)	**Distanza Indicata (m)**	**RSSI (dB)**	**errore distanza**	**errore (%)**
1	6,02	-96	5,02	502
2	9,46	-97	3,73	373
3	10,4	-95	2,47	247
4	10,22	-100	1,56	156
8	10,22	-100	0,28	28
12	30,67	-98	1,56	156
16	30,67	-98	0,92	92
20	30,67	-98	0,53	53

Tabella 4.4: Dati relativi l'errore delle distanze outdoor

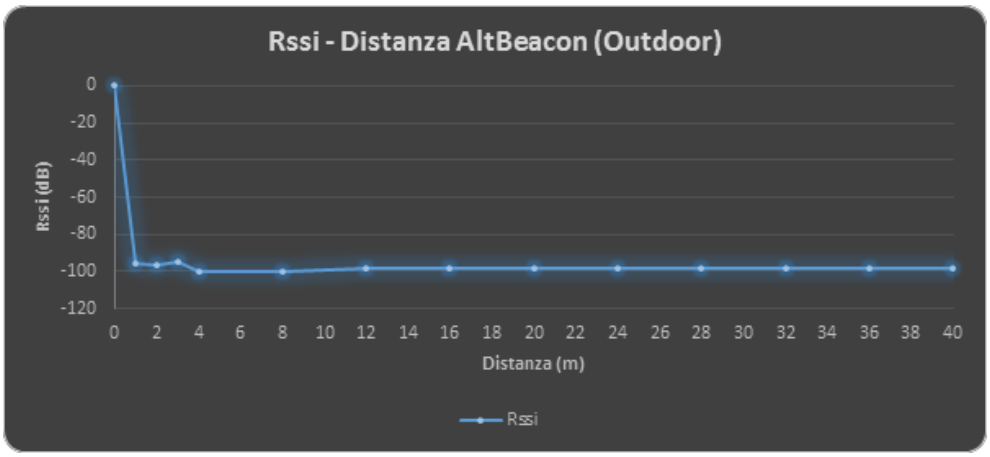


Figura 4.5: Variazione della distanza al variare dell'RSSI con il formato Altbeacon outdoor

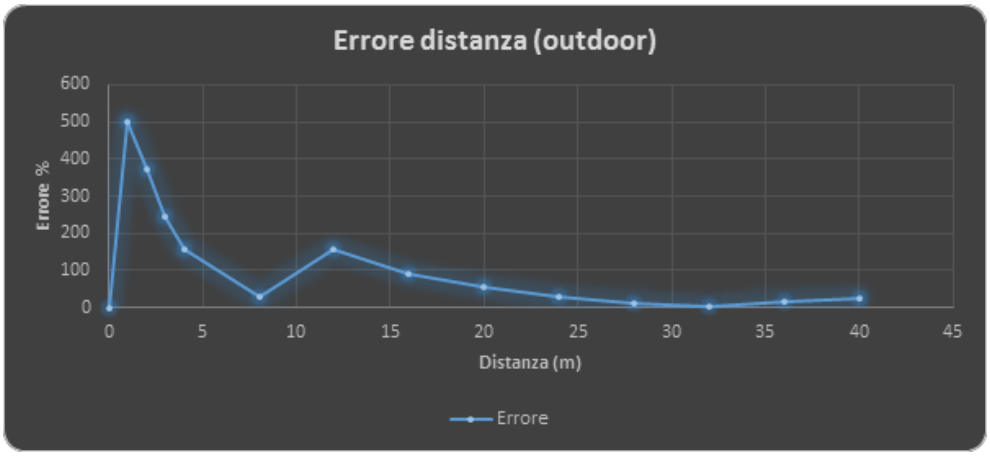


Figura 4.6: Variazione dell'errore al variare della distanza outdoor

La situazione all'esterno è la medesima: nel complesso vale la stessa condizione del primo esperimento e cioè che il segnale dopo pochi metri diminuisce notevolmente (sempre per la questione della mancata propagazione da parte dei muri). L'errore invece è alto tra 0 e 5 metri per poi gradualmente diminuire.

Da questo secondo test viene fuori che i segnali RSSI delle due librerie sono

simili ma che il calcolo della distanza discosti di molto da quello effettivo sia in ambienti chiusi che in ambienti aperti, sebbene questa differenza tenda ad assottigliarsi nelle lunghe distanze. Da studi ulteriori è emerso che l'accuratezza del calcolo migliora quando c'è una distribuzione maggiore di beacon entro i due metri ma allo stesso tempo ne consegue una minore per distanze superiori ai 9 metri considerando i settaggi consigliati di T_s e f_D [14].

4.3 Test 3: Studio della variazione del *Miss Rate*

L'ultimo esperimento invece prova spiegare se e come viene influenzato nell'emissione dei pacchetti il *Packet Delivery Ratio*: la sommatoria tra i pacchetti ricevuti, diviso quelli inviati. Il calcolo è stato effettuato sulla base del *Duty cycle* visto precedentemente. Per ottenere dei risultati attendibili è stato preso come frequenza di trasmissione il valore di default del dispositivo: 1s; lo stesso per il periodo di scansione, il quale generalmente come detto è settato a 5 secondi. Per cui l'obiettivo dell'esperimento è stato verificare come varii, in un'analisi di 2 minuti ad ogni spostamento, il Packet Delivery Ratio. Per questo tramite l'applicazione ideata per l'occasione, si è agito secondo il seguente modo:

- ogni qualvolta veniva riscontrato un segnale, una ListView sul dispositivo aggiornava il numero di Item;
- stampati i risultati su un file.txt.

Alla fine, considerando che il duty cycle settato fosse di 1 s, ad ogni distanza si sarebbero dovuti ricevere 120 pacchetti, ma in realtà il numero dei pacchetti è risultato essere inversamente proporzionale alla distanza, per cui più la distanza risultava essere maggiore, meno pacchetti venivano ricevuti. Prendendo in considerazione il *miss rate* ottenuto dalla differenza in percentuale tra:

$$MissRate = (100 - PacketDeliveryRatio)\% \tag{4.4}$$

si nota come a distanze crescenti, i pacchetti persi aumentino.

Distanza (m)	P. Inviati (1s)	P. Ricevuti	Packet Delivery Ratio	Packet Delivery Ratio (%)	Miss Rate (%)
2	120	92	0,77	76,67	23,33
4	120	91	0,76	75,83	24,17
6	120	85	0,71	70,83	29,17
8	120	84	0,70	70,00	30,00

Tabella 4.5: La tabella mostra come a distanze crescenti aumentano il numero di pacchetti persi

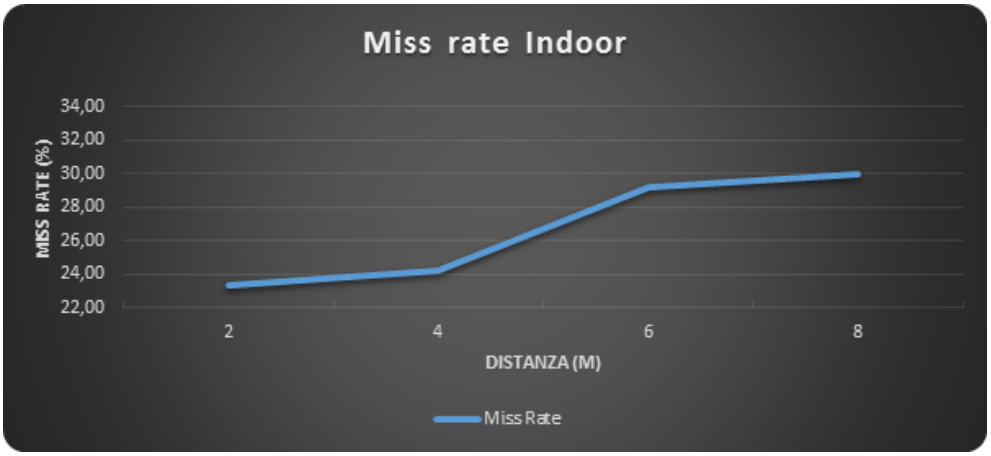


Figura 4.7: Il grafico mostra qual è il numero dei pacchetti persi al variare della distanza

Per quel che riguarda gli scenari outdoor ci si aspetta, considerando anche i risultati visti precedentemente, come il Miss Rate debba essere maggiore per le stesse distanze in dipendenza della dispersione del segnale:

Distanza (m)	P. Inviati (1s)	P. Ricevuti	Packet Delivery Ratio	Packet Delivery Ratio (%)	Miss Rate (%)
2	120	101	0,84	84,17	15,83
4	120	86	0,72	71,67	28,33
6	120	77	0,64	64,17	35,83
8	120	54	0,45	45,00	55,00

Tabella 4.6: La tabella mostra come a distanze crescenti aumentano il numero di pacchetti persi per gli spazi outdoor

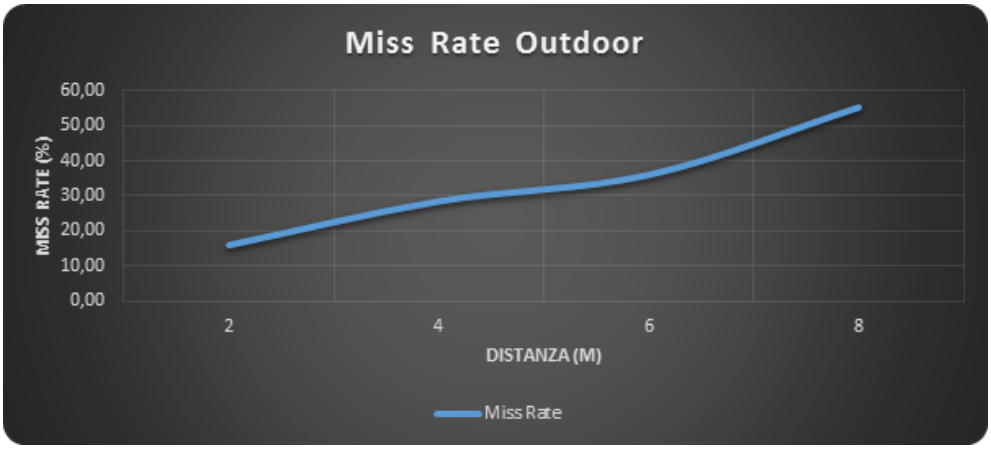


Figura 4.8: Il grafico mostra qual è il numero dei pacchetti persi al variare della distanza per gli spazi outdoor

A dare un responso in riferimento a quest'ultimo esperimento, ci viene in aiuto uno studio analogo [14] dove viene mostrato la variazione del miss rate per distanze crescenti fino ai 5 metri. In quel caso fino ai due metri il miss rate si stanzia su $\approx 15\%$, mentre invece dai due metri in su è $\approx 27\%$. In questo caso invece la media del miss *rate-indoor* è diversa: $\approx 23\%$ fino ai 4 metri, $\approx 29\%$, dai quattro metri in su. Per quel che riguarda il *miss rate-outdoor*, considerando le stesse distanze, fino ai 4 metri è addirittura più bassa

rispetto quella indoor $\approx 22\%$, dopo i quattro medi aumenta vertiginosamente: $\approx 45\%$. La discrepanza tra i risultati dei due esperimenti può essere adibita ad una serie di fattori come i device utilizzato (Samsung Galaxy S3, Samsung Galaxy S5 vs. Samsung Galaxy s6) e beacon usato (Estimote^{TM} Beacon vs. Jaalee). Ciònonostante hanno in comune il seguente responso: il valore del miss rate è indipendente dall'intervallo di scansione T_s, ma dipendente dal f_D Lo stesso studio approfondisce anche il miss rate di beacon multipli e mostra come i valori maggiori di pacchetti persi si hanno tra un numero di beacon compreso tra 1, 5, 9

Capitolo 5

FoundOut Beacon

Con questo ultimo capitolo, si cercherà di descrivere nel dettaglio l'implementazione di un progetto realizzato con la tecnologia Beacon per la localizzazione di corpi fisici. L'applicazione è stata implementata utilizzando l'IDE Android Studio 1.4, testata su un Samsung Galaxy S6 dotato di Android Lollipop 5.1.1 ed utilizzando due librerie: una, AltBeacon, per il monitoraggio, e allo stesso tempo, la libreria nativa Jaalee, che consente di poter interagire con il Beacon attraverso la lettura dei seguenti parametri:

- Il nome
- La password;
- Il Tx Power
- Lo stato (se attivo o disattivo);

In aggiunta è stato necessario considerare un'altra serie di informazioni, che non sono legate al beacon stesso ma che hanno costituito insieme ai parametri appena elencati, i record del database.

5.1 Descrizione del progetto

FoundOut Beacon è un'applicazione che si prefigge come obiettivo il monitoraggio di oggetti associati a dei Beacon, sfruttando i vantaggi che offre la tecnologia Bluetooth 4.0.

5.1.1 Funzionalità

L'obiettivo dell'applicazione è quello di fornire una serie di funzioni all'utente che gli consentano in maniera semplificata e con un'interfaccia chiara, un controllo costante, dopo l'avvenuta registrazione, sui propri dispositivi (si fa riferimento ai Beacon per semplicità ma in realtà ci si riferisce all'oggetto associato ad esso) e allo stesso tempo dare la possibilità ad ognuno di riscontrare eventuali smarrimenti. Volendo essere più precisi, esse comprendono:

1. Visualizzazione dei Beacon disponibili;

2. Possibilità di registrazione e/o modifica delle sue opzioni;

3. Possibilità di cercare il proprio dispositivo sfruttando in segnale Bluetooth di cui dispone[1];

4. Visualizzazione dell'ultimo avvistamento effettuato;

5. Visualizzazione di possibili ritrovamenti;

6. Possibilità di ricezione di una notifica nel momento in cui il proprio Beacon è settato come perso e si entra nuovamente nel suo range;

[1]Sarebbe stato più utile impostare un sistema di triangolazione per sapere esattamente all'interno di uno spazio chiuso dove si colloca il beacon, ma dovendo far riferimento esclusivamente ad uno unico, non è stata una realizzazione possibile

Applicazioni preesistenti

FoundOut Beacon non è l'unica applicazione che utilizza il segnale dei Beacon per il rilevamento degli oggetti. Sicuramente una delle più importanti e affermate è *XYFind* [2]. Accedendo al sito web è possibile vedere vari video, ognuno dei quali mostra un possibile caso d'uso. Per poter utilizzare l'applicazione è necessario acquistare i Beacon messi in vendita dall'azienda. Anche Samsung sta pensando di utilizzare la tecnologia Beacon con un'applicazione chiamata *Flybell*. Attualmente ancora non è presente sul Play Store, ma stando a quanto dichiara iSpazio [41] è in fase di lavorazione. Infine un ultimo esempio viene fornito da Apple, la quale con la sua applicazione *Find My Stuff* permette la ricerca di oggetti collegati ai Beacon. Come si apprende dal sito ufficiale [3] consente con un sistema di localizzazione indoor l'individuazione di oggetti smarriti.

5.1.2 Architettura *client-server*

Da come si può notare leggendo le funzionalità che mette a disposizione l'app, gran parte è basato su un'architettura *client-server*. Risulta essere molto importante questo aspetto, senza del quale alcune funzionalità risulterebbero impossibili, come ad esempio le richieste al database, oppure quelle effettuate ai server di Google per l'acquisizione delle mappe statiche. In generale, un sistema di questo tipo prevede due "attori": un *client* il quale identifica un terminale, rappresentato in questo caso dallo smartphone, che accede ad una risorsa, mediante un protocollo, posizionata sul un *server*. Con questo termine si descrive un elaboratore al cui interno generalmente è contenuta la risorsa richiesta da indirizzare al client richiedente. Al giorno d'oggi questa architettura si è altamente evoluta; i server soprattutto, e i protocolli che ne sono alla base, sono diventati estremamente potenti al tal punto che oggi sono in grado di gestire simultaneamente più richieste da mol-

[2] XYFind: http://www.xyfindit.com/
[3] https://www.fstuff.com/

teplici client. In questo caso si parla di *sistemi distribuiti* [38]. Basandoci su un architettura base, essa consta di una serie di fasi [39][40]:

- il client cerca di stabilire una connessione con il server richiedendo il nome del dominio sul quale il server risiede,la porta sulla quale il server è in ascolto, la directory del server nella quale è contenuta la risorsa e alcuni dettagli inerenti ad essa, il tutto attraverso un protocollo di comunicazione chiamato HTTP;

- il server una volta ricevuta la richiesta, verifica che il client ha accesso alle informazioni che richiede. Quest'ultima rappresenta una misura di sicurezza per eventuali malintenzionati.

- Nel caso di un riscontro positivo da parte del server, viene stabilita una connessione (in linguaggio tecnico si dice che il server manda un *ACK* che sta per *acknowledgment*) ;

- il client richiede la risorsa al server;

- il server gestisce la richiesta spedendo tale risorsa attraverso i nodi presenti nella rete utilizzando politiche di *routing* [4] e sfruttando un *un protocollo di comunicazione* chiamato *TCP/IP* [5].Quest'ultimo prevede la ricezione *"sicura"* di ogni pacchetto per evitare eventuali perdite. Per questo è più lento rispetto un protocollo alternativo *UDP* il quale permette la perdita dei pacchetti ma generalmente è più veloce ([18]iene usato generalmente per la visualizzazione in rete di filmati *streaming*)

- una volta che la risorsa sale lo stack dei protocolli e perviene a quello HTTP, è mandata al client il quale lo riceve e successivamente dealloca

[4]Le politiche di *routing* riguardano algoritmi diversi che cercano di "instradare" un pacchetto nel modo più veloce possibile

[5]Il protocollo TCP/IP si differenzia da quello precedente basato sul modello OSI. La differenza che intercorre tra i due riguarda i vari livelli di cui sono composti che nel primo caso sono 7 mentre nel secondo 4. In generale ad ogni livello ci si avvicina maggiormente all'interfaccia utente

le risorse. Nel caso in cui si necessiti fare una nuova richiesta vi è l'esigenza di ristabilire una nuova connessione perché il server non è tenuto a memorizzare informazioni preesistenti (si parla in questo caso di connessioni *stateless*)

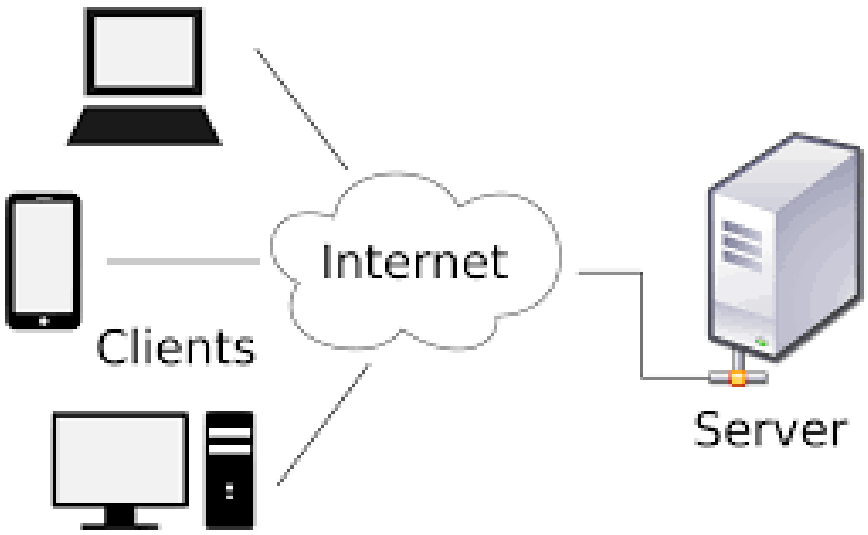


Figura 5.1: Esempio semplificato di una struttura client server

5.1.3 Interfaccia utente

Come già asserito, si è cercato di conferire all'interfaccia utente un aspetto chiaro e pulito al fine da non risultare fuorviante la navigazione per l'utente. Per l'utilizzo dei colori dell'interfaccia, è stato utilizzato il verde in varie gradazioni per non appesantire la vista degli assidui utilizzatori. Per cui:

ColorPrimaryDark: rappresenta il colore più intenso ed è quello presente nella barra di notifica. E' rappresentato in base esadecimale da: #1B5E20;

ColorPrimary: rappresenta il colore primario in background nell'app. Identifica generalmente la *toolbar*. E' rappresentato in base esadecimale da: #388E3C;

ColorAccent: rappresenta il colore con tonalità minore. E'usato per mettere in risalto alcuni particolari. E' rappresentato in base esadecimale da: #81C784;

Focalizzandoci sull'layout delle interfacce e sul funzionamento ad esso connesse, si può vedere come sono state strutturate. Dal *Navigation Drawer* si possono visualizzare tutte le funzioni implementate nell'applicazione. Ognuna è costituita da una relativa schermata identificativa.

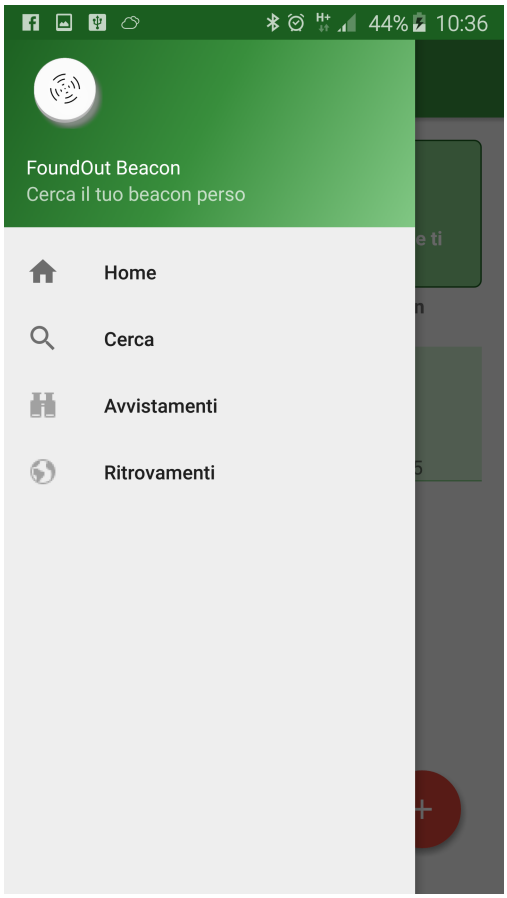
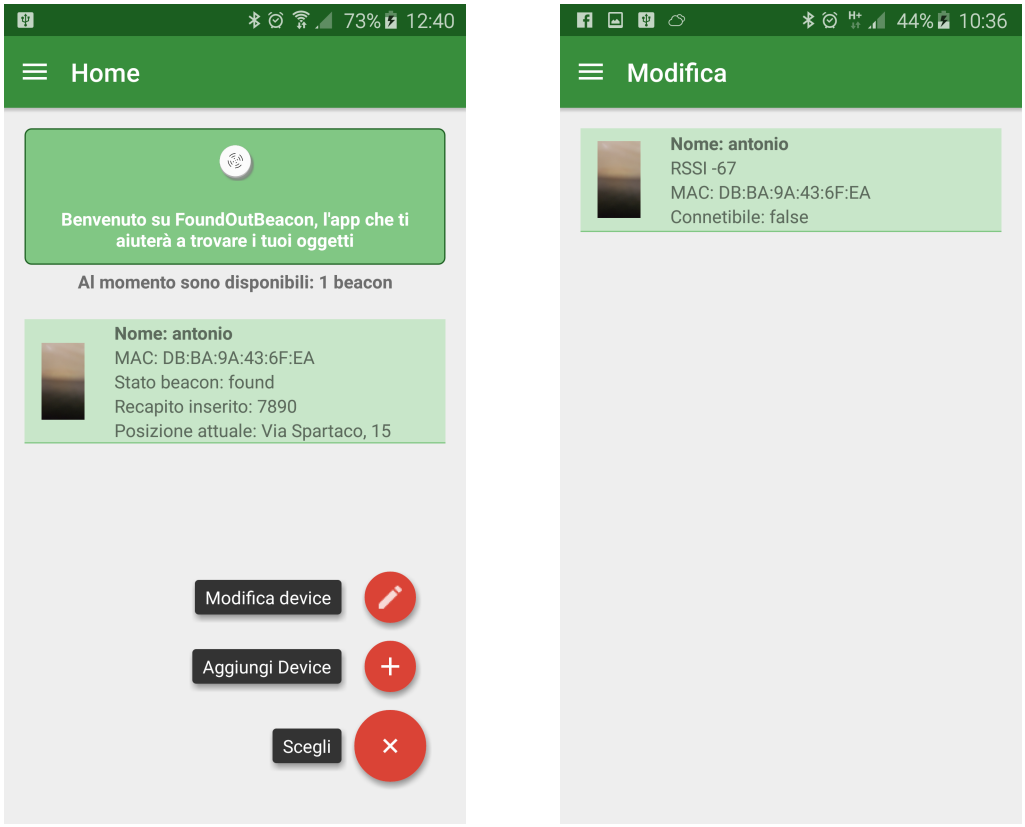


Figura 5.2: Visualizzazioni di tutti gli item a cui per ognuno è associata una funzione

Nella Home dell'applicazione, sono mostrati i Beacon di cui si dispone. Se non presenti, è possibile registrarne qualcuno, premendo sul relativo pulsante *Aggiungi Device* o in alternativa, modificarne uno, scegliendo *Modifica Device*. Entrambe le scelte rimanderanno alla schermata la quale mostra i dispositivi sottoposti a scansione. Se uno di questi è già registrato, non verrà visualizzato nel caso si sia giunti alla schermata dal pulsante di aggiunta, altrimenti invece si. Nell' ipotesi in cui si stia registrando un dispositivo per la prima volta, può verificarsi la mancata visualizzazione dello stesso. Tal evenienza indica che ci si trova in modalità *inattiva*. Per cui basterà semplicemente dargli dei

piccoli colpi affinché passi in modalità *attiva*, la quale ha una durata di 1 minuto circa dopodiché, il Beacon tornerà allo stato iniziale:

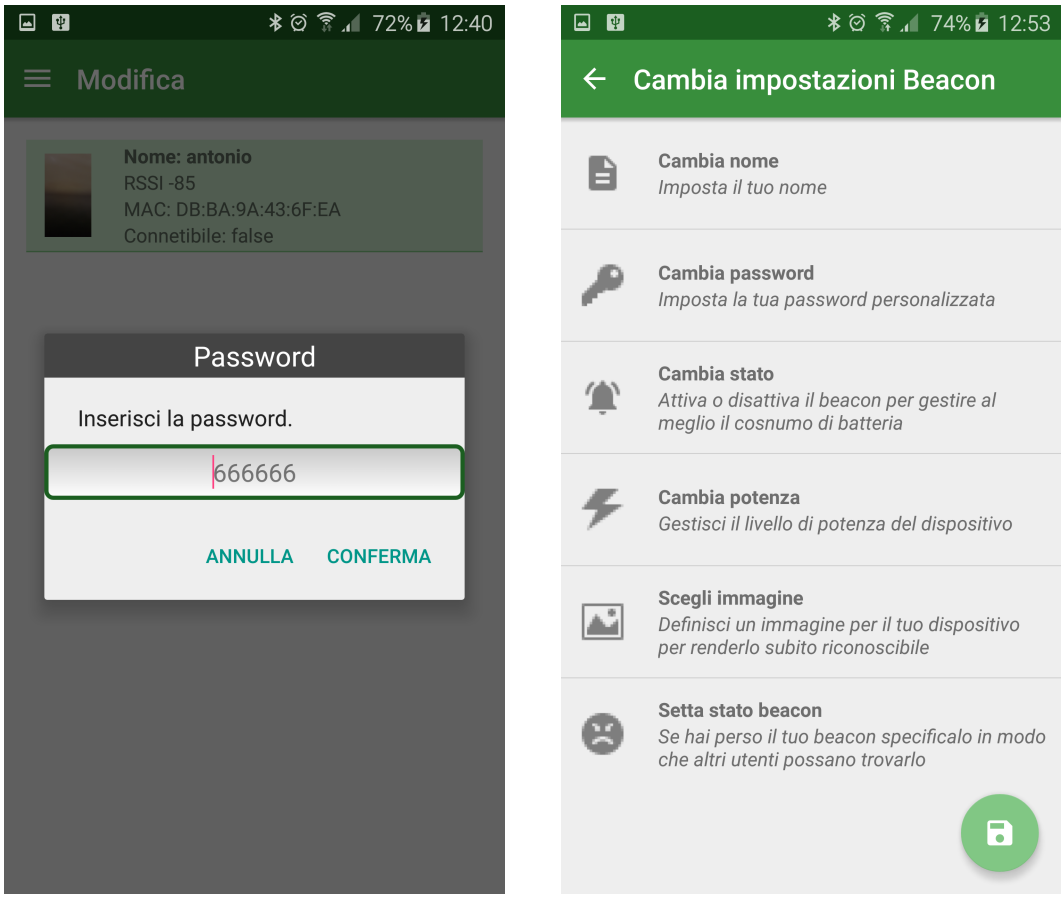


(a) Visualizzazione dei dispositivi disponibili

(b) Visualizzazione dei dispositivi già registrati

Figura 5.3: Visualizzazione di due delle prime operazioni possibili quando si avvia l'applicazione

Una volta effettuata la scelta potranno essere settate le impostazioni, previa inserimento della password, la quale di default è *666666*:



(a) Visualizzazione dell' AlertDialog per l'inserimento della password

(b) Item che presentano le impostazioni modificabili

Figura 5.4: Schermate relative il settaggio delle impostazioni

In riferimento a quanto detto prima, per far si che il Beacon risulti essere costantemente attivo, come prima impostazione, sarebbe utile settare l'item *Cambia Stato*. In questo modo sarà possibile trovarlo costantemente nelle successive scansioni.
Se tra le impostazioni, si scegliesse di settare il Beacon come perso, allora nel momento in cui si entra nuovamente nella sua area di competenza, sarebbe visualizzata la seguente notifica:

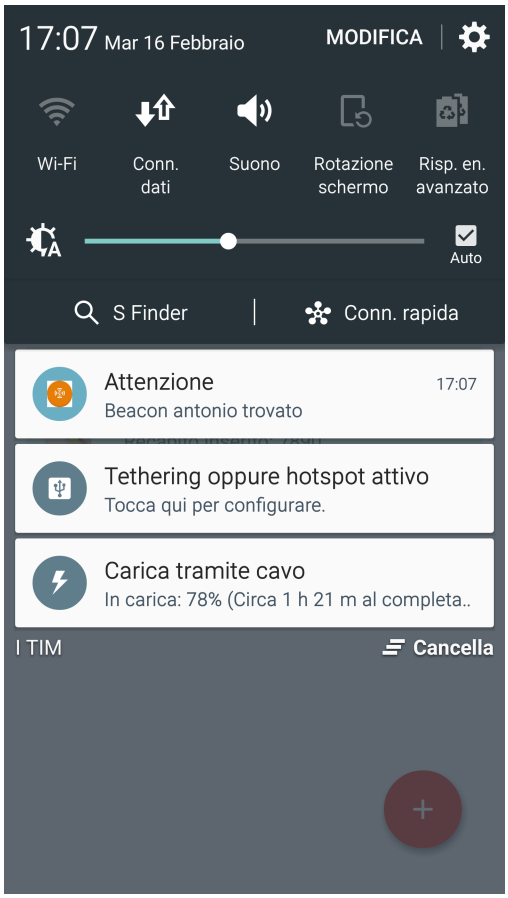
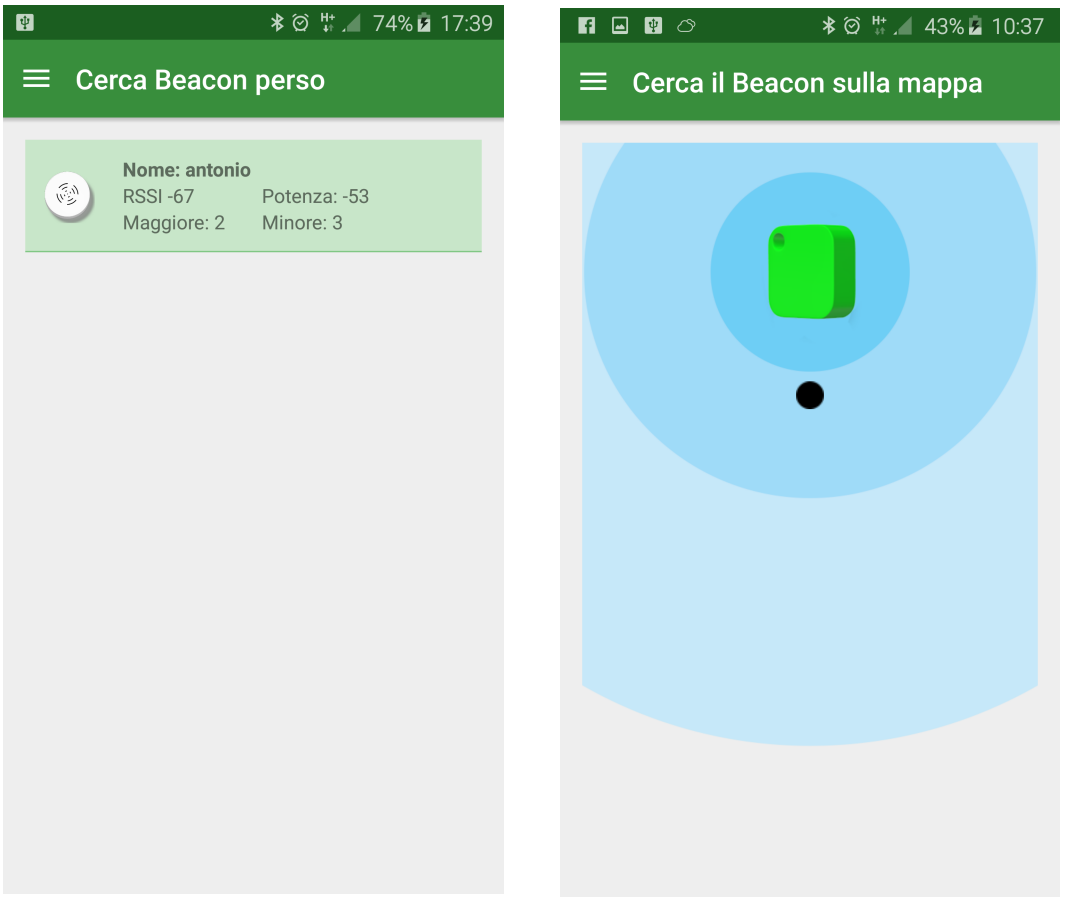


Figura 5.5: Viene notificato all'utente che è stato trovato il proprio dispositivo

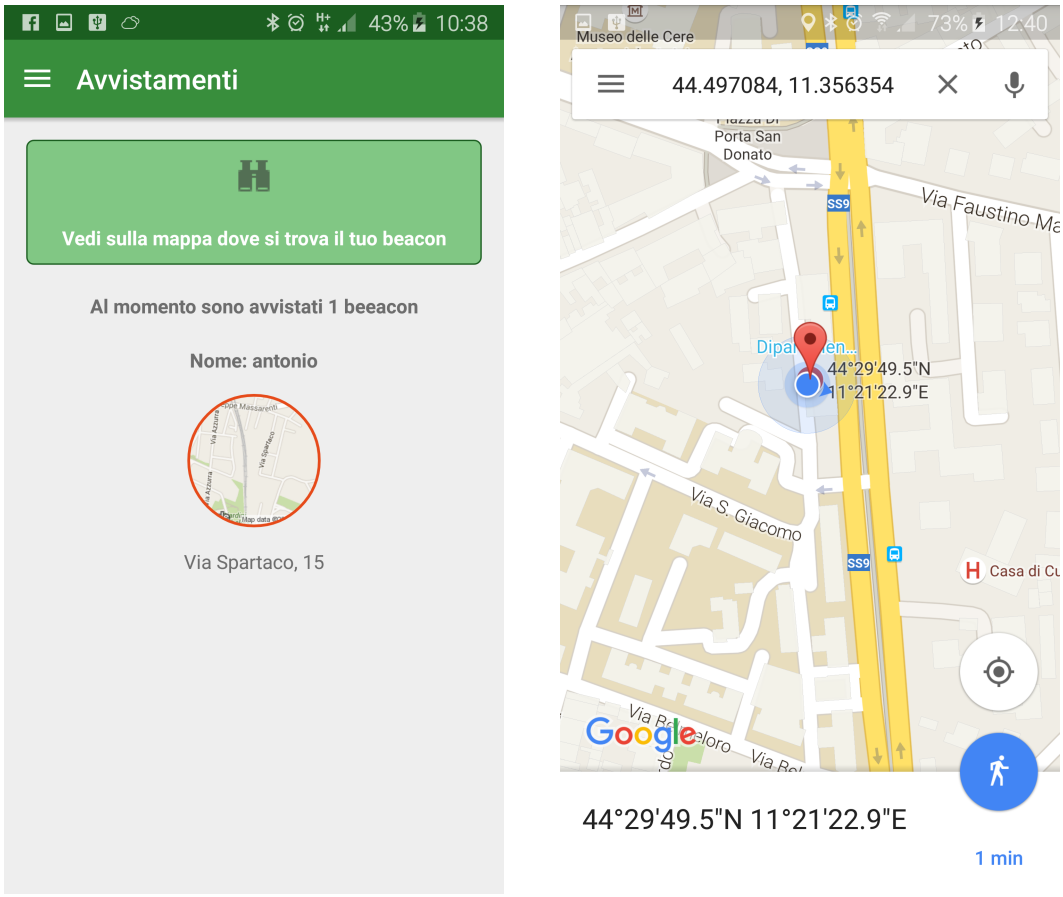
In questa circostanza, cliccando sulla notifica si viene rimandati alla Home. Qui mediante il Navigation Drawer visto precedentemente, scegliendo la seconda voce: *Cerca* è possibile avvistare tutti i device. Questa volte però questo tipo di scansione è eseguita con la libreria AltBeacon:



(a) Scansione device (AltBeacon) (b) Mappa che mostra uno scenario Indoor

Figura 5.6: Percorso da compiere per cercare il proprio dispositivo

Nell'ipotesi secondo la quale un utente ha smarrito il proprio Beacon, è disponibile una schermata adibita alla visualizzazione dell'ultimo avvistamento effettuato. Nel caso si volesse approfondire la ricerca si viene rimandati alle mappe di Google:



(a) Visualizzazione dell'ultima posizione in cui è stato visto il proprio dispositivo

(b) Posizione del Beacon mostrato nelle mappe Google

Figura 5.7: Posizione del Beacon su mappa statica e su quella Google

Infine l'ulitma operazione consentita riguarda il ritrovamento dei Beacon persi:

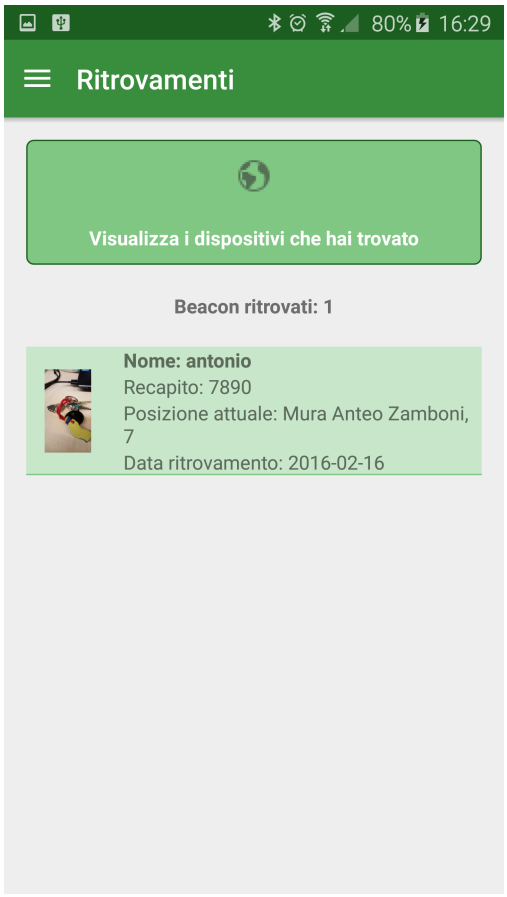


Figura 5.8: Visualizzazione dei ritrovamenti compiuti

Casomai ci si imbattesse in un Beacon smarrito sarà possibile contattare il proprietario qualora quest'ultimo metta a disposizione un contatto.

5.2 Requisiti del Progetto

Di qui in avanti saranno mostrati solo i frammenti di codice ritenuti importanti al fine di poter dare una visione globale completa. Nell'*Appendice A* sono stai inserite solo le classi chiave del progetto, mentre nell' *Appendice B* le query al database relazione, realizzate in PHP. Qui di seguito l'organizzazione e la divisione della struttura in package:

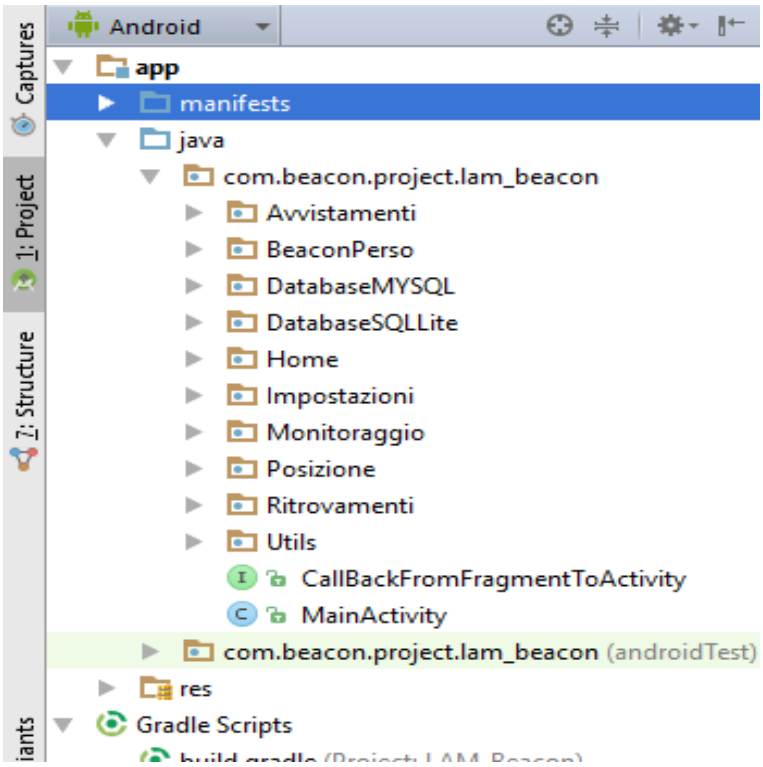


Figura 5.9: Struttura del progetto in Android Studio

Prima di addentrarsi nella descrizione del codice, è importante precisare che il progetto è stato basato interamente su Fragment fatta eccezione per due Activity: la MainActivity e quella di settaggio del beacon, più una classe a parte che estende *Application*.
Per l'implementazione, dapprima si è specificata la versione minima necessaria affinché possa essere utilizzata sul proprio smartphone. Le API fornite dalla versione Android 4.3 *Jelly Bean* sono risultate essere le più congeniali, in quanto i dispositivi che dispongono di questa versione rappresentano circa il 50% della totalità dei dispositivi Android. Per cui nell' *AndroidManifest.xml* è stato indicato:

```
1 <!-- minSdk specifica la versione minima mentre targetSdk la
     versione piú congeniale all' applicazione. In questo caso
     coincidono-->
```

```
<uses-sdk
        android:minSdkVersion="18"
        android:targetSdkVersion="18" />
```

Successivamente se si considera che l'app sfrutta, la locazione GPS e i Bluetooth Low Energy sono stati determinati i permessi necessari affinché tutti i dispositivi i quali risultino esserne sprovvisti, non possano scaricarla in quanto inutile.

```
<!-- Permessi BLE obbligatori -->
<uses-feature android:name="android.hardware.bluetooth_le"
    android:required="true" />

<uses-feature android:name="android.hardware.camera"
        android:required="false" />
```

L'utilizzo della fotocamera è sì un requisito ma non è indispensabile. Ciò perché può essere lasciata l'immagine di dafault in fase di registrazione. Il resto del manifest definisce l'activity principale che verrà lanciata al primo avvio, il *service* che servirà per la scansione dei beacon Jaalee e per finire la chiave Google per poter utilizzare le mappe:

```
<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyAa1JjuDmTZ_LUKLFtEoqvC0LurFuJOsuE"/>
```

All'interno del *tag* Application è definito il seguente attributo:

```
        android:name=".Monitoraggio.Monitoring
```

Verrà vista nel dettaglio più avanti, per adesso basta sapere che identifica il nome di una classe molto importante per il progetto che serve per eseguire il monitoraggio costante dei Beacon.

5.3 Database

Come già anticipato, l'applicazione sfrutta due database: uno locale utilizzato per vedere i dispositivi di cui si dispone e per verificare se sono stati scovati dei Beacon persi (compreso il proprio), mentre l'altro è il database remoto, identico a quello locale fatta eccezione per il fatto che manca la tabella ritrovamenti, presente solo nel primo. Per cui riassumento, in quello locale ci sono due tabelle, una chiamata *setting* e l'altra invece *ritrovamenti*. Nel db remoto solo la tabella *setting*. Ovviamente quello remoto ha una valenza maggiore, soprattutto per quanto riguarda tre campi: la latitudine, la longitudine e l'indirizzo. Grazie ad essi infatti, si può mantenere aggiornata la posizione e permettere il ritrovamento di un dispositivo.

5.3.1 SQLite

Si parte con la descrizione nel dettaglio del database locale. La classe responsabile della definizione dello schema del db è *DbSchema*. Al suo interno si trova, un costruttore vuoto, di default e due classi astratte, una per ogni tabella.

```
1  public static abstract class DbBeaconColumn implements BaseColumns {
2
3          // nome e colonne del mio database
4          public static final String TABLE_NAME = "setting";
5          public static final String COLUMN_NAME_UUID = "UUID";
6          public static final String COLUMN_NAME_MAC = "MAC";
7          public static final String COLUMN_NAME_NAME = "NAME";
8          public static final String COLUMN_NAME_PASS = "PASS";
9          public static final String COLUMN_NAME_MAJOR = "MAJOR";
10         public static final String COLUMN_NAME_MINOR = "MINOR";
11         public static final String COLUMN_NAME_SITUATION =
               "SITUATION";
12         public static final String COLUMN_NAME_IMAGE = "IMAGE";
13         public static final String COLUMN_NAME_CONTACT = "CONTACT";
```

```
14        public static final String COLUMN_NAME_LATITUDE= "LATITUDE";
15        public static final String COLUMN_NAME_LONGITUDE =
              "LONGITUDE";
16        public static final String COLUMN_NAME_POS = "POS";
17        public static final String COLUMN_NAME_NULLABLE = "NULLABLE";
18    }
```

Il frammento visualizzato, mostra come si è proceduto per definire una tabella. La classe astratta implementa *BaseColumns* indispensabile per fornire l'*ID* e il *count* sulla tabella. Dopo di che vengono settati il nome, e tutte le colonne. Si faccia attenzione all'ultima colonna *NULLABLE*. Questa non è di per se una colonna attinente alla tabella, ma Android come suggerisce la guida ufficiale [36] non permette l'inserimento di valori nulli avendo specificato il nome della tabella. Ciò significa che qualora tra i parametri passati non venga specificato un record da inserire, (magari per dimenticanza) il valore *NULL* viene inserito in questa colonna. Al contrario se volutamente si inserisse *"null"*, Android non lo inserisce affatto.
Stesso discorso per la tabella Ritrovamenti. Il resto della classe DbSchema descrive le query per la creazione e le query per l'eliminazione delle tabelle.
Le altre due classi importanti sono *BeaconDbHelper* e *OperationDb*. La prima ha il compito di creare il database vero e proprio fornendo la versione del database e il nome. E' importante sottolineare che qualora si decida di cambiare lo schema bisogna aggiornare il numero di versione, altrimenti le modifiche non saranno rese effettive. Tornando alla classe, oltre al costruttore il quale al suo interno ha il Context, il nome del db, la versione, un parametro null di default che si riferisce al cursore, ci sono tre metodi che sovrascrive la classe *SQLiteOpenHelper* estesa al momento della dichiarazione. I metodi in questione sono: onCreate, per la creazione, onUpgrade per l'aggiornamento dello schema e onDowngrade. Quest'ultimo è molto simile al precedente ma viene chiamato nel momento in cui la versione corrente è più recente di quella richiesta. Qui di seguito viene mostrato il metodo di creazione.

```
@Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DbSchema.SQL_CREATE_TABLE);
        db.execSQL(DbSchema.SQL_CREATE_TABLE_RITROVAMENTI);
    }
```

OperationDb è la classe dove vengono effettuate le query vere e proprie. Il costruttore ha al suo interno un oggetto Context più l'istanza della classe DbHelper. Infatti per qualsiasi operazione da effettuare vi è la necessità di specificare se si sta per compiere un operazione in scrittura:

```
        SQLiteDatabase db = dbHelper.getWritableDatabase();
```

oppure un operazione in lettura:

```
    SQLiteDatabase db = dbHelper.getReadableDatabase();
```

Il resto della classe implementa una serie di metodi: l'inserimento, l'aggiornamento, la cancellazione e una select che ritorna una lista con tutti i Beacon inseriti. Tutto quello che è stato detto per la classe *OperationDb*, vale anche per la classe *OperationDbRitrovamenti*.

5.3.2 MySQL

Per poter comunicare con un server remoto [6] si è creato un progetto parallelo a quello Android però questa volta in PHP. All'interno di tale progetto si è implementato una serie di script, uno diverso per ogni operazione sul db. La prima e anche unica classe è servita per poter stabilire la connessione, dopo aver definito tutti i settaggi relativi:

```
<?php
    // variabili per il nostro database
    define('DB_USER', "root"); // db user
```

[6] Essendo un prototipo, il server remoto è stato creato in locale con XAMPP

```
4     define('DB_PASSWORD', "root"); // db password (mention your db
          password here)
5     define('DB_DATABASE', "foundout_beacon"); // database name
6     define('DB_SERVER', "localhost"); // db server
7 ?>
```

Come si può vedere, è stato necessario specificare, come da prassi del resto, il nome del db, la password, lo user, e il nome del server.
Il file.sql crea il database che come detto è identico a quello locale. Per tutte le operazioni (si veda Appendice B) a prescindere dal tipo (cancellazione, inserimento o altro), oltre la normale query si è scelto per ragioni ovvie di ritornare anche l'esito dell'operazione effettuata. Per cui nel caso dell'inserimento per esempio, la situazione è la medesima:

```
1
2     $result = mysql_query($query);
3     // se la query éandata a buon fine, allora otteniamo un
          messaggio //di successo
4     if ($result) {
5         // inserimento avvenuto con successo
6         $response["success"] = 1;
7         $response["message"] = "Beacon inserito con successo.";
8         // trasformiamo in json
9         echo json_encode(\$response);
10    } else {
11        // inserimento fallito
12        $response["success"] = 0;
13        $response["message"] = 'Oops! errore.';
14        echo json_encode($response);
15    }
```

In questo modo nel momento in cui da Android Studio veniva effettuata una chiamata al server, nel caso in cui ci fossero degli errori, viene stampato un messaggio che sancisce l'errore oltre a stabilirne la causa. In Android chi si

occupa di comunicare con il Server è la classe *JSONParser* al cui interno sono implementati tutti i metodi contenenti chiamate HTTP. Di seguito un piccolo stralcio di codice relativo l'eliminazione di una riga dal database remoto.

```
1           HttpURLConnection conn = (HttpURLConnection)
                url.openConnection();
2           conn.setRequestMethod("POST");
3           conn.setDoInput(true);
4           conn.setDoOutput(true);
5           OutputStream os = conn.getOutputStream();
6           BufferedWriter writer = new BufferedWriter(new
                OutputStreamWriter(os,"UTF-8"));
7           writer.write(getPostDataString(postDataParams));
8
9           writer.flush();
10          writer.close();
11          os.close();
12          int responseCode=conn.getResponseCode();
13          if(responseCode == HttpURLConnection.HTTP_OK){
14              String line;
15              BufferedReader br=new BufferedReader(new
                    InputStreamReader(conn.getInputStream()));
16              while ((line=br.readLine()) != null) {
17                  response+=line;
18              }
19          }else {
20              response = "";
21          }
```

Riassumendo il codice, viene prima eseguita la connessione, definiti i parametri sulla base del metodo *getPostDataString(param)*, aperta la connessione e nel caso in cui la chiamata risulti essere andata a buon fine, letto il risultato. In qualsiasi evenienza, si leggerà l'esito della chiamata, per cui nel caso dell'eliminazione il responso darà: *"No beacon found"* se non è stato elimina-

to nulla, oppure *"Beacon successfully deleted"*. La classe JSONParser viene chiamata da molteplici altre. Ognuna di esse estende un *AsyncTask* perché la comunicazione con il server rappresenta un qualcosa di pesante da dover eseguire in background. Il responso della chiamata a uno dei metodi della classe JSONParser viene inserito in un JSONObject in questo modo:

```
1       final String result = jParser.deleteBeacon(
2                             DestinationURI.deleteBeacon,
3                             params
4                             );
5       JSONObject json = new JSONObject(result.toString());
6       int success = json.getInt(TAG_SUCCESS);
7       if(success==1){
8           Utils.showToast(activity, "Cancellazione effettuata con
                successo");
9           pDialog.dismiss();
10      }
11      else {
12          Utils.showToast(activity, "Cancellazione non eseguita");
13          pDialog.dismiss();
14
15      }
```

Il metodo *deleteBeacon* indica il nome del blocco di codice che esegue la chiamata HTTP appena vista. Al suo interno si percepisce che sono stati specificati sia l'URI dove è contenuto l'indirizzo del server e sia i parametri da inviare per la cancellazione. La riga $json.getInt(TAG\_SUCCESS)$; legge il valore del JSON risultante in PHP che informa sull'esito dell'operazione.

5.3.3 BeaconDb

Nell'introduzione al capitolo si è asserito riguardo la necessità di dover combinare informazioni legate nello specifico al Beacon (UUID, password etc) ad altre informazioni. E'stato necessario aggiungere queste informazioni per

poter meglio delineare il profilo di ogni utente che decide di utilizzare l'app. Esse riguardano:

- la definizione di un'immagine per poter riconoscere visivamente quale oggetto è legato ad un particolare beacon;
- la definizione di un recapito (telefonico o di qualche altro genere) per poter contattare il proprietario;
- la definizione dello stato del Beacon per poter settare se è in stato *perso* oppure *trovato* ;

Questa serie di informazioni dovevano essere raggruppate prima di poter fare una qualsiasi operazione su di esse, per cui in linea con i principi dell' *Object Oriented* [7] si è creata una classe "involucro" che identificasse un oggetto contenente al suo interno tutte le informazioni. In questo paragrafo si vuole descrivere come è stato strutturato l'oggetto le cui istanze sono state inviate al database. La classe *BeaconDb* che si premura di fare ciò ha un costruttore al cui interno sono racchiusi tutti i parametri. In totale essi sono:

- ID;
- UUID;
- mac;
- name;
- password;
- major;
- minor;
- situation;

[7] Le classi in Java sono usate o come contenitori o per plasmare un oggetto con delle caratteristiche precise

- image;
- contact;
- latitude;
- longitude;
- pos;

Successivamente sono stati implementati i metodi *getter* e *setter* per poter recuperare o settare le informazioni. Tra questi bisogna menzionare quelli dell'immagine. Nel Db infatti l'immagine è stata codificata come stringa di testo e all'occorenza trasformata in Bitmap attraverso il metodo:

```
1       public Bitmap getImageBitmap() {
2       String tempImage = getImage();
3       byte[] decodedString = Base64.decode(tempImage,
            Base64.DEFAULT);
4       return BitmapFactory.decodeByteArray(decodedString, 0,
            decodedString.length);
5   }
```

In aggiunta a questo, per l'utilizzo delle mappe si è utilizzato due metodi (sfruttano la latitudine, la longitudine e la posizione). Il primo apre semplicemente un *Intent* delle mappe di Google. Prende come parametro il seguente URL:

```
1       String STATIC\_MAP\_API\_ENDPOINT =
2       "http://maps.google.com/maps/api/staticmap?center=" +
            getLatitude() + "," + getLongitude() +
            "\&zoom=15\&size=200x200\&sensor=false";
3   }
```

Come si può vedere l'indirizzo è quello di Google (motivo per cui è stato necessario definire una chiave per poterlo utilizzare [8] L'altro metodo invece crea una mappa statica, anche con l'ausilio di una libreria definita nel file.xml di riferimento [9]. Qui come nel caso delle query al server, anche in questo frangente viene realizzata una chiamata Http al server di Google, la quale questa volta ritornerà un immagine Bitmap (si veda Appendice A).

5.4 MainActivity

Nel progetto Android la classe lanciata al primo avvio è *MainActivity.java*. Rappresenta una delle due Activity e in quanto tale estende AppCompatActivity. Inoltre essendo il "nodo" centrale su cui si appoggia l'intero progetto implementa due interfaccie: una per l'implementazione del NavigationDrawer e l'altra per consentire la comunicazione con i Fragment:

```
1  public class MainActivity extends AppCompatActivity
2  implements NavigationView.OnNavigationItemSelectedListener,
       CallBackFromFragmentToActivity {
3  // ......
```

Nell' onCreate invece sono verificati se i Bluetooth e il GPS sono accesi, implementato il NavigationDrawer e definito quale Fragment deve essere mostrato al primo avvio.
Nel momento in cui si decide di registrare o modificare un Beacon viene lanciata uno stesso Fragment, rappresentato dalla classe *ScannerBLEDevice*. Per evitare ridondanza di codice si è deciso di optare per il riuso delle classi

[8] Ci sono vari modi per definire una chiave di accesso per l'utilizzo delle API di Google. Una prevede l'uso del *Prompt dei comandi* digitando il seguente comando: **keytool -list -v -keystore "%USERPROFILE%åndroiddebug.keystore" -alias androiddebugkey -storepass android -keypass android** In alternativa si può far riferimento alla seguente guida: https://developers.google.com/maps/documentation/javascript/get-api-key

[9] Libreria Circle View: https://github.com/hdodenhof/CircleImageView

(ovviamente ove era possibile). La funzione del booleano definito nel metodo *addJaaleeDevice(boolean operation)* è quella di determinare "chi chiama cosa" ed in questo caso, se si clicca sul botton dell'aggiunta verranno mostrati solamente i dispositivi che non sono stati ancora inseriti nel database. Al contrario saranno mostrati quelli che già compaiono all'interno di esso. Andando con ordine, le operazioni effettuate nell'onCreate del Fragment sono le seguenti: per prima cosa si verifica se si è ricevuto come parametro dall'activity il valore booleano; in seguito creato l'istanza dell'oggetto che ritorna il risultato della query dal database locale. Il risultato della chiamata viene inserito all'interno di una lista e questo determina i beacon registrati. Il metodo di connessione serve per connettersi al metodo della libreria che consente le scansioni, mentre $TUTTE\_LE\_REGIONI$ serve per specificare se ci siano dei parametri di scansione, per esempio i dispositivi con un particolare UUID oppure con un valore Major o Minor particolare. In questo caso non serve, per cui sono settati tutti a *null*. Con questo diciamo all'applicazione di scansionare tutto ciò che vede, ma facendo attenzione (con un apposito metodo) a non mostrare dispositivi già visualizzati. Successivamente invece si verifica il valore del parametro proveniente dall'Activity. Se FALSE significa che si è arrivati al Fragment dal tasto di "modifica", per cui per ogni dispositivo scansionato, vengono mostrati solo quelli che sono contenuti nella lista ottenuta dalla query locale, e ciò è realizzato concretamente con l'ausilio del metodo *deviceAlreadyAdded(param)*.

```
1 if(resultBundle==true){
2           // configuriamo i device disponibili con una list View
3
4           beaconManager.setDeviceDiscoverListener(new
                DeviceDiscoverListener() {
5               @Override
6               public void onBLEDeviceDiscovered(BLEDevice
                    bleDevice) {
7
```

```
8                   if (!checkExist(bleDevice) &&
                        !deviceAlreadyAdded(bleDevice)) {
9                       listDevice.add(bleDevice);
10                      listBeaconJaalee.
11                      setAdapter(adapterBeaconJaalee);
12                      adapterBeaconJaalee.replaceWith(listDevice);
13                      listBeaconJaalee.
14                      setOnItemClickListener(clicItem());
15                  }
16              }
17          });
18      }
19      else if(resultBundle==false){
20          beaconManager.setDeviceDiscoverListener(new
                DeviceDiscoverListener() {
21              @Override
22              public void onBLEDeviceDiscovered(BLEDevice
                    bleDevice) {
23
24                  if (!checkExist(bleDevice) &&
                        deviceAlreadyAdded(bleDevice)) {
25                      listDevice.add(bleDevice);
26                      listBeaconJaalee.
27                      setAdapter(adapterBeaconJaalee);
28                      adapterBeaconJaalee.replaceWith(listDevice);
29                      listBeaconJaalee.
30                      setOnItemClickListener(clicItem());
31                  }
32              }
33          });
34      }
```

5.5 Adapter

All'interno della classe appena vista, esiste un riferimento, l'istanza di un oggetto: *adapterBeaconJaalee*.

```
BeaconJaaleeAdapter adapterBeaconJaalee = new
    BeaconJaaleeAdapter(getContext());
```

Essa fa riferimento ad una classe adapter. Bisogna sottolineare che gran parte del progetto è stato organizzato con ListView e per renderle maggiormente gradevoli, ognuna di esse è accompagnata da un Adapter: una classe che serve a personalizzare gli item della lista nel momento in cui vengono mostrate in un Fragment o in un Activity. In questo caso l'adapter di riferimento è la classe BeaconJaaleeAdapter. Essendo strutturalmente uguali, si riporterà nell'Appendice A il codice solo di questa appena menzionata, in quanto tutte si differenziano solo esclusivamente per le scelte estetiche adottate.
Gli Adapter hanno una valenza elevate nel progetto. Si illustra qui di seguito il suo funzionamento. All'inizio della classe viene esteso *BaseAdapter*. Tale operazione necessita della sovrascrittura di una serie di metodi. Ognuno di essi ritorna una qualche tipo di operazione su un *ArrayList* , il quale è dichiarato nel costruttore. Per esempio il metodo *getCount()* ritorna la lunghezza dell'arrayList:

```
@Override
public int getCount() {
    return bleDeviceJaalee.size();
}
```

Stessa cosa per un altra serie di metodi. Quello che lega il contenuto di ogni signolo item dell'arraylist al file.xml incaricato di definire il layout è:

```
@Override
public View getView(int position, View view, ViewGroup parent) {
    view = inflateLayoutRow(view, position, parent);
    collegaAllaView(getItem(position), view, position);
```

```
5         return view;
6     }
```

Qui, dapprima si associa il suddetto file.xml ad una classe interna all'adapter che ne definisce le componenti (textView, ImageView e qualsiasi altro componente grafico); dopo di che si collega il singolo item della lista alle componenti dell'xml (e quindi agli attributi della classe interna sopra citata).

5.6 Localizzazione Beacon Indoor

Avendo già asserito a proposito delle mappe, le quali sono implementate nella classe BeaconDb vista precedentemente, ora si spiegherà il sistema di localizzazione usato per scovare il punto esatto dove si trova un Beacon. Per cui ci si immedesimi nella situazione secondo la quale un utente perda il proprio dispositivo, aprendo l'applicazione visualizzi l'ultimo indirizzo in cui è stato visto e successivamente si diriga nel posto indicato. A questo punto, utilizza il sistema di localizzazione basato sul valore del segnale RSSI per verificare a quanti metri di distanza è situato. La classe principale è *CercaBeaconPerso* la quale fa riferimento ad una Adapter e la classe di monitoraggio della libreria AltBeacon. Andando con ordine, la classe *CercaBeaconPerso* è costituita dal sovente costruttore vuoto. Nell' *onCreate* del metodo si determina il parser utilizzato per fare in modo che qualsiasi Beacon possa essere letto:

```
1     private Region region;
2     private BeaconManager beaconManager;
3     @Override
4     public void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         beaconManager =
              BeaconManager.getInstanceForApplication(getActivity());
7         beaconManager.getBeaconParsers().add(new BeaconParser().
8                 setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,
```

```
9                 i:22-23,p:24-24,d:25-25"));
10          //....
11      }
```

Successivamente vengono presi i parametri del Beacon scelto i quali questa volta servono per definire la regione, in considerazione del fatto che devo localizzare esclusivamente un dato Beacon:

```
1   // in questo caso scansiono solo ed esclusivamente il Beacon che
        possiede le caratteristiche del mio Beacon
2   region = new Region("backgroundRegion", UUID, major, minor);
```

Alla fine eseguito il *bind*.
Nell' *onCreateView* si setta il layout della classe e si definisce la *View* che consente la visualizzazione dinamica dell'immagine che rappresenta la posizione rispetto il Beacon cercato. [10]

```
1   startY = (int) (POSIZIONE_INIZIALE_Y * view.getMeasuredHeight());
2   int stopY = (int) (POSIZIONE_DI_STOP_Y *
        view.getMeasuredHeight());
3   lunghezzaSegmento = stopY - startY;
4
5   imageProprietario.setVisibility(View.VISIBLE);
6   imageProprietario.setTranslationY((float)
        computeDotPosY(beacon));
```

Soffermandoci su queste poche righe di codice, si identifica la lunghezza di uno spostamento. *POSIZIONE_INIZIALE_Y* e *POSIZIONE_DI_STOP_Y* sono delle costanti definite arbitrariamente e ognuna è moltiplicata per la lunghezza dello schermo.
In questo modo l'immagine del punto che si sposterà al variare dell'RSSI, sarà inizialmente posta al centro.
Il metodo *computeDotPosY(params)* definisce lo spostamento ottenuto dalla

[10] Si ricorda che tutto è in funzione del segnale RSSI del Beacon

posizione iniziale più il calcolo della distanza.

```
1 private double computeDotPosY(Beacon beacon){
2       // qui praticamente viene considerato solamente distanze che
            sono inferiori ai 20 metri (secondo altBeacno) altrimenti
            la posizione resta la stessa.
3       double distance = Math.min(calculateAccuracy(beacon), 20.0);
4       return startY + lunghezzaSegmento * (distance/20);
5   }
```

Come suggerisce il commento, vengono considerate distanze che sono inferiori ai 20 metri. Nel caso in cui ci si trovi ad una distanza maggiore dei 20 metri l'immagine non si muove ma apparirà solamente un messaggio che indica che si è lontani. il metodo cardine della classe è *calculateAccuracy(params)*. In questo frangente viene effettivamente calcolata la distanza dal Beacon. Il calcolo è basato sulla formula della distanza descritta nel quarto capitolo. Nel caso in cui l'RSSI non sia uguale a 0 oppure il rapporto tra il Tx Power e l'RSSI non risulti essere minore di 1.0 per evitare che l'immagine oscilli troppo si è scelto di calcolare la media delle 10 distanze e solo successivamente ritornare il valore della distanza definitivo.

```
1   protected double calculateAccuracy(Beacon beacon) {
2       double rssi = beacon.getRssi();
3       int txPower = beacon.getTxPower();
4       if (rssi == 0) {
5           return -1.0; // if we cannot determine accuracy, return
                -1.
6       }
7
8       double ratio = rssi*1.0/txPower; // (r/t)
9       if (ratio < 1.0) {
10          //Toast.makeText(CercaBeaconPerso.this, " ratio < 1 " +
                Math.pow(ratio,10), Toast.LENGTH_LONG).show();
11          stampaMessaggio(ratio);
```

```
12          return Math.pow(ratio,10);
13      }
14      else {
15          double accuracy = (0.89976)*Math.pow(ratio,7.7095) +
                0.111;
16          double accuracy_migliorata = miglioraAccuracy(accuracy);
17          stampaMessaggio(accuracy_migliorata);
18          return accuracy_migliorata;
19      }
20  }
```

Resta da definire il punto in cui viene aggiornata continuamente la distanza. Ciò viene effettuato qui di seguito:

```
1  @Override
2      public void onBeaconServiceConnect() {
3          // quando c'é il collegamento (il bind) con il servizio,
               viene specificata la classe che dovrebbe essere chiamata
               ogni qual volta
4          // vengono scansionati i dati.
5          beaconManager.setRangeNotifier(new RangeNotifier() {
6              @Override
7              public void didRangeBeaconsInRegion(final
                   Collection<Beacon> beaconTrovati, Region region) {
8              // ...
9                              if (beaconDiInteresse != null) {
10                                 aggiornaDistanza(beaconDiInteresse);
11                             }
12             }
13         }
14         });
15
16         try {
17             // Indica al BeaconService di cercare i Beacon che fanno
```

```
                match con le caratteristiche specificate e fornisce
                una stima stimata della distanza ogni secondo finché
                il beacon nella regione évisibile
18              beaconManager.startRangingBeaconsInRegion(region);
19          } catch (RemoteException e) { // ...}
20
21      }
```

Essendo molto lungo sono state aggiunte solo le parti più significative. In generale il metodo di connessione parte quanto c'è il bind visto in precedenza. Ogni qual volta viene trovato un Beacon in linea con i parametri specificati nella *Region* (sempre lo stesso) viene ad essere aggiornata la posizione sulla mappa del Fragment.
Alla classe *CercaBeaconPerso* appena descritta si arriva tramite la classe che cerca tutti i Beacon visibili. Strutturalmente è identica alla classe appena descritta, ma differisce per il fatto che rispetto la precedente si limita a fare delle scansioni specificando questa volta una regione senza parametri.

5.7 Monitoraggio

All'inizio del paragrafo ci si è soffermato su una riga in particolare definita nell'*AndroidManifest.xml*. La riga in questione era la medesima:

```
1       android:name=".Monitoraggio.Monitoring
```

Questa rappresenta un attributo definito dichiarato nel tag **Application** Tale attributo esplicita che all'interno dell'applicazione è definita una classe, situata nel package *Monitoraggio* che estende la classe **Application**. Cosa significa estendere questa classe ? Generalmente i programmatori Android usano estendere, a seconda delle loro esigenze, le Activity, i Fragment nelle loro varianti, così come d'altronde è stato fatto per questo progetto (oltre ad un'altra serie di estensioni varie). In questo modo ogni operazione viene struttura nell'Activity o nel Fragment di competenza. Per cui per eseguire,

per esempio, un operazione *B* contenuta nel *FragmentB* si deve necessariamente passare attraverso un'*ActivityA* e magari prima eseguire un operazione *A* definita al suo interno. La classe Application invece permette di avere un accesso globale, intero, all'interno dell'app. Parte quando viene avviata la prima activity e cessa quando tutte le risorse sono state distrutte, ossia alla fine. Il motivo di questa premessa è quello di spiegare il motivo per cui il monitoraggio dei Beacon è stato implementato in questa classe, proprio perché vi è la necessità di poter vedere quali Beacon sono visibili e quali no a prescindere dall'Activity o dal Fragment in cui ci si trova. La classe *Monitoring* oltre ad estendere *Application* implementa l'interfaccia *BeaconConsumer*. Risulta essere molto simile alle classi CercaBeaconPerso e BeaconDisponibiliAltBeacon. Infatti hanno tutte in comune il settaggio di un parser universale.In aggiunta questa possiede nell'*onCreate* una serie di chiamate:

- una chiamata alla tabella ritrovamenti;
- una chiamata al database locale per poter recuperare tutti i device di cui si dispone;
- il settaggio del periodo di scansione in linea con quanto detto nel capitolo 3 paragrafo 3.4.2;
- la chiamata ad un metodo fornito dalla libreria AltBeacon per evitare i consumi eccessivi della batteria dello smartphone [11]

```
1       // query per salvarmi i miei beacon in locale
2       operationDB = new OperationDB(this);
3       beaconPersonal = operationDB.getBeaconDBList();
4
5       // query dove mi salvo le tuple della tabella ritrovamenti
6       operationToFind = new OperationDBRitrovamenti(this);
7       ritrovamentiConosciuti =
            operationToFind.getRitrovamentiDbList();
```

[11] https://altbeacon.github.io/android-beacon-library/battery_manager.html

```
8       // parser
9       // ....
10      // ....
11      beaconManager.setBackgroundScanPeriod(1100l);
12      beaconManager.setBackgroundBetweenScanPeriod(5000l);
13      beaconManager.bind(this);
14      saveBattery = new BackgroundPowerSaver(this);
```

La parte più importante del monitoraggio è situata nel metodo di scansione. Qui sono contenuti tre blocchi di codice (sono definiti dettagliatamente nell'Appendice A, qui saranno riportati solo dei frammenti)

5.7.1 Monitoraggio Beacon proprio

Il primo blocco di codice si riferisce al monitoraggio del proprio Beacon perso.

```
1       // controllo che ci entro solo una volta mediante lista
2       if( beacons.size() > 0 &&
            !verifyListener.contains(beacons.iterator().next())){
3           verifyListener.add(beacons.iterator().next());
4          /*
5           Verifico che:
6            --> il beacon iterato éuno dei miei,
7            --> il beacon ha stato "lost" quindi perso
8           */
9          Beacon nextBeacon = beacons.iterator().next();
10         if(personalBeacon(nextBeacon) &&
               personalBeaconLost(nextBeacon) && activity==null){
11            sendNotification(Monitoring.this, "Attenzione", "Beacon
                  " + nextBeacon.getBluetoothName() + " trovato");
12         }
13         else if(personalBeacon(nextBeacon) &&
               personalBeaconLost(nextBeacon) && activity!=null){
```

```
14              Utils.showToast(Monitoring.this, "Attenzione éstato
                    trovato un Beacon");
15          }
16      }
```

Da come è riportato nei commenti, il primo *if* verifica se ci sono beacon da leggere, successivamente se tra questi compare qualcuno settato come perso, allora viene mandata la notifica se l'app risulta essere in background, altrimenti compare un *Toast* con un messaggio il quale specifica il ritrovamento. In seconda battuta si è dovuto implementare un sistema che permettesse di tenere traccia di ogni singolo dispositivo. Si immagini la situazione in cui un utente decida di acquistare più Beacon e di associarli ognuno ad un oggetto diverso. Erroneamente accade che l'utente perda *solo* uno tra i dispositivi che aveva addosso mentre gli sono ancora tutti in suo possesso; per cui il sistema deve essere in grado di riconoscere distintamente quale tra questi è stato smarrito. Per poter implementare questo meccanismo ci si è serviti del listener che mette a disposizione la libreria. La logica è la seguente: con l'ausilio di due *HashMap* aventi ambedue come chiave il Beacon letto e come valore l'oggetto Posizione precedentemente istanziato [12], si aggiungono i Beacon scansionati. La differenza che intercorre tra le due riguarda l'impiego che se ne fa: la prima, inizialmente è vuota se vede un Beacon lo aggiunge, altrimenti se lo stesso Beacon è già presente si preoccupa di aggiornarne solo il valore. Non viene mai svuotata. L'altra lista invece è temporanea. Permette di aggiungere uno stesso Beacon una sola volta, senza che possa essere aggiornata la posizione. Per cui si può asserire che questa seconda lista funge da indice per la prima. Dopo 5 secondi infatti, vengono confrontate. Se hanno la stessa lunghezza allora significa che non sono stati persi Beacon, al contrario se hanno lunghezza diversa significa che durante il periodo di

[12] non si è discusso di questa classe nel capitolo, perché non attinente con la logica descrittiva del progetto. In generale si preoccupa di aggiornare la latitudine e la longitudine del Beacon sfruttando il GPS e quando necessario queste informazioni vengono salvate nel database in corrispondenza del Beacon relativo.

attesa dei 5 secondi uno dei beacon di cui si dispone è stato perso perché presente nella prima lista (quella che aggiorna la posizione) e non presente nella lista temporanea. Questo può accadere perché dopo lo svuotamento (avvenuto dopo i 5 secondi) non è stato più aggiunto quel dato Beacon. Di conseguenza viene estratto il Beacon perso, salvata la posizione e aggiornati entrambi i database, sia quello locale che quello remoto.

```
1   // risettiamo il listener come true
2   lastListener.set(true);
3
4   // Teniamo sempre aggiornata una hashmap che avrã come chiave il
        //beacon e come valore la posizione.
5   map.put(beacons.iterator().next(),
        map.get(beacons.iterator().next()));
6
7   // nel frattempo ci serviamo di una mappa temporanea uguale alla
        //precedente ma che conterrã un nuovo beacon solo qualora
8   // questi non lo contenga in precedenza
9   if (!mapTemp.containsKey(beacons.iterator().next())) {
10      mapTemp.put(beacons.iterator().next(), new
            Posizione(Monitoring.this));
11  }
12
13  // Dopo 5 secondi verifichiamo
14  if (System.currentTimeMillis() >= lastTime + 5000) {
15
16      // le due lunghezze sono uguali ? Se si allora puliamo la
            mappa //temporanea e si ripeterã il ciclo
17      workOnMap();
18
19      lastTime = System.currentTimeMillis();
20  }
```

Il metodo *workOnMap()* fa il confronto e salva le posizioni. All'interno di

questo frammento si può vedere il valore di una variabile booleana che viene settata a **true**. Il valore di tale variabile serve per verificare qualora tra i "molteplici" Beacon di cui si dispone, ne sia rimasto solo uno. Per capire al meglio come funziona è si riporta qui di seguito il codice:

```
1   // se dopo non ho piu nulla da leggere e quello ultimo e tra i
        miei, //verifico che ci
2   // sia qualcosa di rimasto insospeso
3   if (!beacons.iterator().hasNext() && lastListener.get() ) {
4           workOnMap();
5   }
```

Se non si hanno più Beacon successivi da leggere e quindi quello di cui si dispone è l'ultimo e allo stesso tempo il valore del booleano è true, allora si esegue lo stesso metodo visto prima. Qui nel caso in cui le due liste risultino essere diverse (per precedenti Beacon ora non pià visibili) oltre all'aggiornamento della posizione, viene settata la variabile a **false** per evitare che entri nuovamente nella condizione. Da notare che i booleani in questione sono *AtomicBoolean* il che significa che nel momento in cui si eseguono operazioni in lettura o scrittura non può essere eseguito nient'altro.

5.7.2 Monitoraggio Beacon altrui

L'ultimo tipo di monitoraggio è quello dei Beacon altrui. In questo frangente non è stato difficile fare un controllo che permettesse di verifica se qualche Beacon risultasse perso. A differenza del caso precedente infatti, si dovevano monitorare gli stessi Beacon per un periodo di tempo prolungato in un listening; invece per questo tipo di controllo si è operato nel seguente modo: prima si è verificato se qualche Beacon risultasse come **perso**. A quel punto aggiunto in una mappa, in modo da impedirgli di entrare sempre nella stessa condizione e sovraccaricare l'applicazione. Successivamente vengono eseguite le seguenti operazioni:

- aggiornamento della latitudine, longitudine e indirizzo del Beacon nel database remoto;
- con un metodo chiamato *getBeaconDb(beacon)*, acquisite tutte le informazioni del Beacon scansionato acquisite dal database remoto;
- aggiornata la tabella *Ritrovamenti* con la nuova posizione o aggiunta qualora tale Beacon non fosse ancora contenuto.

Conclusioni

In questo elaborato si è discusso sostanzialmente di due macro argomenti. Il primo riguarda il Mobile Advertising. All'inizio è stata fornita una una panoramica su cos'è e cosa vuole esprimere, cercando di porre ben in rilievo la sottile, ma rilevante differenza che lo contraddistingue dai messaggi spam. La struttura che costituisce un *advertising* è ben studiata e cerca di far leva sulla tipologia di utente su cui è incentrata cercando in generale di rendere ciò che si pubblicizza interessante e qualitativamente informativo; inoltre pone come primo obiettivo, come suggerisce il TAM *(Tecnology Acceptance Model)*, la facilità di utilizzo come uno dei basilari requisiti che deve possedere.
Il secondo macro argomento su cui si è dibattuto invece asserisce a proposito dei Beacon. L'adozione di questa nuova forma di tecnologia può essere la *"miccia"* ideale per erigere il Mobile Advertising come nuova forma predominante di marketing. Come dimostrato, in un articolo Forbes[37], gli introiti provenienti da questo nuovo mondo si aggirano sui $44 milioni. Sono stati forniti casi pratici di utilizzo e a tal proposito si è analizzata la tecnologia che ne è alla base: il Bluetooth 4.0, elencando le principali differenze rispetto alle versioni antecedenti. Infine, in comparazione con studi analoghi, sono stati effettuati dei test relativi alle performance che sono in grado di offrire. Nell'ultima parte,si è discusso di un caso dettagliato che sfrutta i Beacon, atto a testare le sue potenzialità. Da sottolineare che è stato basato su un obiettivo diverso rispetto il Mobile Advertising, proprio per mettere in rilievo i molteplici usi che se ne possono fare. Il progetto fornisce uno strumento

di tracciamento di oggetti associati ad un Beacon e salvati all'interno di un database. Mediante il segnale RSSI di cui il device dispone, si può rilevare la posizione che intercorre tra uno smartphone che utilizza l'applicazione e il Beacon. Inoltre l'applicazione fornisce un sistema di interazione con altri utenti con un fine comunicativo per poter informare dell'eventuale ritrovamento dell'oggetto perduto. Anche nel caso in cui un utente non comunica al proprietario dell'avvenuto ritrovamento, un sistema di monitoraggio permette di aggiornare l'ultimo avvistamento e prenderne visione nella schermata adibita. In questo modo risulta possibile verificare esattamente dov'è situato il proprio oggetto perso rispetto l'ultimo aggiornamento, utilizzando come supporto le mappe di Google con cui l'app è collegata.

Problemi riscontrati

Durante l'implementazione del progetto sono stati riscontrati una serie di problemi, come è sovente nella pratica di programmazione. La maggior parte di essi con cui si ha avuto a che fare sono dipesi essenzialmente da due fattori:

- problemi di compatibilità tra il le due librerie usate nel progetto (AltBeacon e Jaalee)
- impossibilità di utilizzo di un'interfaccia apposita per il monitoraggio

Per quanto riguarda il primo problema è emerso nel momento in cui si modificava una qualche impostazione al Beacon. Infatti in questo frangente il monitoraggio adoperato dalla libreria AltBeacon (libreria che si è sfruttata in cooperazione con quella nativa del Beacon utilizzato) veniva perso. Per far fronte a questa situazione si è deciso di aggiornare la posizione nel momento di "passaggio" tra le due librerie.
Il secondo problema invece riguarda l'impossibilità di poter sfruttare una classe ad hoc adibita a tenere traccia di un ipotetico avvistamento di un dispositivo. Il suo utilizzo avrebbe permesso un monitoraggio più consono alle

esigenze dell'applicazione. Ciò è dipeso dal fatto che tale interfaccia distingue i vari Beacon sulla base di alcuni valori di cui esso dispone: l' UUID, il Major e il Minor i quali in questo caso, non non sono differenziati tra i diversi dispositivi. In questo contesto per poterli distinguere, è sembrato più opportuno e allo stesso tempo logico avvalersi esclusivamente dell'indirizzo MAC. Alle condizioni esposte nell'ipotesi in cui un utente possa possedere più di un Beacon con se, rilevare esattamente il momento in cui ognuno di esso entra o esce dal range di competenza ed eseguire le conseguenti operazioni, risulta impossibile.

Sviluppi futuri

Nel complesso l'applicazione funziona bene; tutti gli obiettivi prefissati all'inizio del progetto sono stati raggiunti. L'idea è quella di continuare ad investire su quest'applicazione, cercandola di migliorarla sempre più. Per raggiungere tale intento l'obiettivo è annettere sempre maggiori funzionalità e renderla maggiormente fruibile. Inizialmente sarebbe utile implementare dei trigger all'interno del db che permettano di avvisare i diretti interessati quando il proprio Beacon, settato come perso, viene avvistato da qualcuno che non sia lui stesso e allo stesso tempo notificare gli utenti degli avvenuti ritrovamenti che hanno realizzato. Il conseguente passo successivo consisterebbe nel renderlo una piattaforma social, implementado una chat che permetta la comunicazione diretta fra di essi. Infine sarebbe vantaggioso acquistare un dominio e renderlo effettivamente funzionante, in questo modo le chiamate http potranno essere più veloci evitando così gli antiestetici *lag* che ogni tanto, seppur raramente, si verificano.

Appendice A

Android Java Class

Di seguito saranno riportate le classi del Progetto che sono state ritenute di maggiore importanza:

A.1 BeaconDb.java

```
1  public class BeaconDb {
2      private static final String TAG_ERROR = "TAG_ERROR";
3      private long _id;
4      private String uuid;
5      private String mac;
6      private final String name;
7      private String password;
8      private String major;
9      private String minor;
10     private String situation;
11     private String image;
12     private String contact;
13     private String latitude;
14     private String longitude;
15     private String pos;
16
```

```
17    private Bitmap googleImage = null;
18
19
20
21    public BeaconDb(long _id, String uuid, String mac, String name,
          String password, String major,
22                  String minor, String situation, String image,
                        String contact, String latitude, String
                        longitude, String pos) {
23
24        this._id = _id;
25        this.uuid = uuid;
26        this.mac = mac;
27        this.name = name;
28        this.password = password;
29        this.major = major;
30        this.minor = minor;
31        this.situation = situation;
32        this.image = image;
33        this.contact = contact;
34        this.latitude = latitude;
35        this.longitude = longitude;
36        this.pos = pos;
37
38    }
39
40    // metodi getter
41    public long getID() {
42        return _id;
43    }
44
45    public String getUUID() {
46        return uuid;
```

```
47      }
48
49      // ......... continua per tutti i parametri
50
51      public String bitmaToString() {
52          Bitmap bitmap = getImageBitmap();
53          ByteArrayOutputStream baos = new ByteArrayOutputStream();
54          bitmap.compress(Bitmap.CompressFormat.PNG, 100, baos);
55          byte[] b = baos.toByteArray();
56          String temp = Base64.encodeToString(b, Base64.DEFAULT);
57          return temp;
58      }
59
60      // ritorna l'immagine come testo
61      public String getImage() {
62          return image;
63      }
64
65      // ritorna l'Immagine come Bitmap
66      public Bitmap setBitmap(String image) {
67          byte[] decodedString = Base64.decode(image, Base64.DEFAULT);
68          return BitmapFactory.decodeByteArray(decodedString, 0,
                decodedString.length);
69      }
70
71      public Bitmap getImageBitmap() {
72          String tempImage = getImage();
73          byte[] decodedString = Base64.decode(tempImage,
                Base64.DEFAULT);
74          return BitmapFactory.decodeByteArray(decodedString, 0,
                decodedString.length);
75      }
76
```

```
77  public void openMaps(Context context){
78
79      String linkGoogle = "http://maps.google.com/?q=" +
            getLatitude() + "," + getLongitude() + "";
80      Intent intent = new
            Intent(android.content.Intent.ACTION_VIEW,
81              Uri.parse(linkGoogle));
82      context.startActivity(intent);
83  }
84
85  public Bitmap getMapPosition() {
86
87      String STATIC_MAP_API_ENDPOINT =
88              "http://maps.google.com/maps/api/staticmap?center=" +
                    getLatitude() + "," + getLongitude() +
                    "&zoom=15&size=200x200&sensor=false";
89
90
91      HTTPGoogleMap map = new HTTPGoogleMap();
92      map.execute(STATIC_MAP_API_ENDPOINT);
93      try {
94          map.get();
95          //map.get(3000, TimeUnit.MILLISECONDS);
96      } catch (InterruptedException e) {
97          e.printStackTrace();
98      } catch (ExecutionException e) {
99          e.printStackTrace();
100     }
101
102     return googleImage;
103
104 }
```

```
105   private class HTTPGoogleMap extends AsyncTask<String, Void,
          Bitmap>{
106
107       public HTTPGoogleMap(){
108           super();
109
110       }
111       @Override
112       protected void onPreExecute() {
113           super.onPreExecute();
114       }
115       @Override
116       protected Bitmap doInBackground(String... params) {
117           Bitmap imageMap = null;
118           URL url;
119           InputStream in = null;
120           HttpURLConnection conn = null;
121           try {
122               url = new URL(params[0]);
123               conn = (HttpURLConnection) url.openConnection();
124               conn.setRequestMethod("POST");
125               conn.setDoInput(true);
126               conn.setDoOutput(true);
127               conn.setChunkedStreamingMode(0);
128               conn.connect();
129               /*OutputStream out = new
                      BufferedOutputStream(conn.getOutputStream());
130               BufferedWriter writer = new BufferedWriter(new
                      OutputStreamWriter(out, "UTF-8"));
131               writer.write(out);*/
132
133               int responseCode = conn.getResponseCode();
134               if (responseCode == HttpURLConnection.HTTP_OK) {
```

```
135
136                 InputStream reader = new
                        BufferedInputStream(conn.getInputStream());
137                 googleImage = BitmapFactory.decodeStream(reader);
138
139             }
140             conn.disconnect();
141         } catch (Exception e) {
142             e.printStackTrace();
143         }finally {
144             if(conn!=null) {
145                 conn.disconnect();
146             }
147         }
148         return googleImage;
149     }
150 }
151 }
```

A.2 MonitoringBeacon.java

```
1 public class Monitoring extends Application implements
      BeaconConsumer {
2
3     // INT per notifiche
4     public static final int NOTIFICATION_ID = 123;
5     private static final String ALL_REGION = "ALL_REGION";
6
7     // ..... inizizzializzate tutte le variabili
8
9
10    @Override
```

```
11    public void onCreate() {
12        super.onCreate();
13        final boolean lastListener = true;
14
15        posizione = new Posizione(this);
16        if(!posizione.canGetLocation()){
17            posizione.showSettingsAlert();
18        }
19
20        mapLost = new ArrayList<>();
21
22        verifyListener = new ArrayList<>();
23
24        // query per salvarmi i miei beacon in locale
25        operationDB = new OperationDB(this);
26        beaconPersonal = operationDB.getBeaconDBList();
27
28        // query dove mi salvo le tuple della tabella ritrovamenti
29        operationToFind = new OperationDBRitrovamenti(this);
30        ritrovamentiConosciuti =
              operationToFind.getRitrovamentiDbList();
31
32        beaconManager =
              BeaconManager.getInstanceForApplication(this); // serve
              per realizzare una chiamata Singleton al servizio
33        beaconManager.getBeaconParsers().add(new BeaconParser().
34                setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,
35                i:22-23,p:24-24,d:25-25"));
36
37
38        notificationManager = (NotificationManager)
              getSystemService(Context.NOTIFICATION_SERVICE);
39
```

```
40        // array list hce conterrá i beacon che sono persi leggendo
              dal //db remoto
41        beaconPersi = new ArrayList<>();
42
43        // eseguo la query per ottenere tutti i dispositivi persi
              che sono sul server.
44        SelectBeaconFound select = new SelectBeaconFound();
45        try {
46            // si hanno tutti i beacon che sono stati persi !
47            beaconPersi = select.execute().get();
48            // Utils.consoleLog("listabeacon", " --> " +
                  beaconPersi.get(1).getMac());
49
50        } catch (InterruptedException e) {
51            e.printStackTrace();
52        } catch (ExecutionException e) {
53            e.printStackTrace();
54        }
55        //Utils.consoleLog("RESULT", ciao.toString());
56        // le seguenti due righe ci dicono 1 secondo di scansione
              ogni 5 secondi
57        beaconManager.setBackgroundScanPeriod(1100l);
58        beaconManager.setBackgroundBetweenScanPeriod(3000l);
59        beaconManager.bind(this);
60        saveBattery = new BackgroundPowerSaver(this);
61    }
62
63    @Override
64    public void onBeaconServiceConnect() {
65        beaconManager.setRangeNotifier(new RangeNotifier() {
66        @Override
67        public void didRangeBeaconsInRegion(final Collection<Beacon>
              beacons, Region region) {
```

```
68
69        // controllo che ci entro solo una volta mediante lista
70        if( beacons.size() > 0 &&
              !verifyListener.contains(beacons.iterator().next())){
71
72            verifyListener.add(beacons.iterator().next());
73           /*
74           // Verifico che:
75            // --> il beacon iterato éuno dei miei,
76            // --> il beacon ha stato "lost" quindi perso
77            */
78
79           Beacon nextBeacon = beacons.iterator().next();
80           if(personalBeacon(nextBeacon) &&
                 personalBeaconLost(nextBeacon) && activity==null){
81              sendNotification(Monitoring.this, "Attenzione", "Beacon
                    " + nextBeacon.getBluetoothName() + " trovato");
82           }
83           else if(personalBeacon(nextBeacon) &&
                 personalBeaconLost(nextBeacon) && activity!=null){
84              Utils.showToast(Monitoring.this, "Attenzione éstato
                     trovato un Beacon");
85           }
86       }
87
88        //  Monitoraggio beacon proprio
89        // se ci sono ancora beacon da leggere ed éuno dei miei
             //beacon
90        if (beacons.iterator().hasNext() &&
             personalBeacon(beacons.iterator().next())) {
91
92            // risettiamo il listener come true
93            lastListener.set(true);
```

```
94
95          // Teniamo sempre aggiornata una hashmap che avrá //come
                chiave il beacon e come valore la posizione.
96          map.put(beacons.iterator().next(),
                map.get(beacons.iterator().next()));
97
98          // nel frattempo ci serviamo di una mappa temporanea
                //uguale alla precedne ma che conterrá un nuovo
                //beacon solo qualora
99          // questi non lo contiene in precedenza
100         if (!mapTemp.containsKey(beacons.iterator().next())) {
101             mapTemp.put(beacons.iterator().next(), new
                    Posizione(Monitoring.this));
102         }
103
104         // Dopo 5 secondi verifichiamo
105         if (System.currentTimeMillis() >= lastTime + 5000) {
106
107             // le due lunghezze sono uguali ? Se si allora
                    //puliamo la mappa temporanea e si ripeterá il
                    //ciclo
108             workOnMap();
109
110             lastTime = System.currentTimeMillis();
111         }
112     }
113
114     // se dopo non ho piu nulla da leggere e quello ultimo e tra
            i miei, verifico che ci
115     // sia qualcosa di rimasto insospeso
116     if (!beacons.iterator().hasNext() && lastListener.get() ) {
117             workOnMap();
118     }
```

```
119
120
121         // Monitoraggio beacon altrui
122         if (beacons.size() > 0) {
123             Beacon beacon = beacons.iterator().next();
124         if (verifyLost(beacon) && !mapLost.contains(beacon)) {
125
126             mapLost.add(beacon);
127             Double latitude = posizione.getLatitudine();
128             Double longitude = posizione.getLongitudine();
129             String pos = posizione.getAddress();
130
131
132             // qui viene aggiornata la posizione sul server con la
                    chiamata successiva
133             new
                    UpdatePosition().execute(beacon.getBluetoothAddress(),
                    Double.toString(latitude),
                    Double.toString(longitude), pos);
134
135             // mi salvo l'oggetto beacon persi
136             BeaconDb infoBeaconDb = getBeaconDb(beacon);
137             // acquisisco tutte le informazioni del beacon
138             String mac = infoBeaconDb.getMac();
139             String name = infoBeaconDb.getName();
140             // ATTENZIONE: --> qui inserisco NON la //posizione con
                    relativa latitudine e //longitudine del Beacon salvato
141             // MA quella attuale affinché possa essere //visto
142
143             String image = infoBeaconDb.getImage();
144             String contact = infoBeaconDb.getContact();
145             String date = Utils.getActualeDate();
146
```

```
147             // verifico se égia presente nella tabella //ritrovamenti
148             if (checkLastFind(beacon)) {
149                 //double id = new
                        Long(infoBeaconDb.getID()).doubleValue();
150                 operationToFind.aggiornaRitrovamento(mac,name,
                        Double.toString(latitude),
                        Double.toString(longitude), pos, image, contact,
                        date);
151             } else {
152                 operationToFind.inserisciNuovoRitrovamento(
153                         mac, name, Double.toString(latitude),
                            Double.toString(longitude), pos,
                            image.toString(), contact, date);
154             }
155         }
156             }
157         }
158     });
159     try {
160         beaconManager.startRangingBeaconsInRegion(new
                Region(ALL_REGION, null, null, null));
161     } catch (RemoteException e) {
162     }
163     }
164
165     // ********************* METODI PER CLIENT **********************
166
167
168     // questo metodo fa in maniera atomica le seguenti operazioni:
169     // - confronto delle due mappe
170     // - aggiornamento
171     // - pulizia della mappa temporanea
172     private synchronized void workOnMap() {
```

```
173         if (map.size() == mapTemp.size()) { "lunghezza uguale");
174             mapTemp.clear();
175         }
176         // se non sono uguali invececompariamo le due hashmap e
                //otteniamo comerisultato una nuova hash map che conterrá
177         // i valori che differiscono tra le due mappe (i beacon
                persi)
178         else {
179         Map<Beacon, Posizione> result = compareMap(map, mapTemp);
180         Double latitude = posizione.getLatitudine();
181         Double longitude = posizione.getLongitudine();
182         String pos = posizione.getAddress();
183         // per ogni valore della mappa "risultante" aggiorniamo la
                posizione sul server in modo che possano essere
184         // visualizzati poi in Avvistamenti.
185         for (Map.Entry<Beacon, Posizione> entry : result.entrySet())
                {
186             try {
187                 new UpdatePosition().execute(entry.getKey().
188                 getBluetoothAddress(),
189                         Double.toString(latitude),
                            Double.toString(longitude), pos).get();
190
191             } catch (InterruptedException e) {
192                 e.printStackTrace();
193             } catch (ExecutionException e) {
194                 e.printStackTrace();
195             }
196         }
197         mapTemp.clear();
198         lastListener.set(false);
199         }
200     }
```

```
201
202     // metodo che mi confronta le due liste nel caso in cui fossero
            diverse di lunghezza (map e mapTemp)
203     // e successivamente mi aggiunge il valore che differisce in un
            altra mappa
204     private Map<Beacon, Posizione> compareMap(Map<Beacon, Posizione>
            map, Map<Beacon, Posizione> mapTemp) {
205         Map<Beacon, Posizione> result = new HashMap<>();
206         for (final Beacon key : map.keySet()) {
207             if (!mapTemp.containsKey(key)) {
208                 result.put(key, map.get(key));
209             }
210         }
211         return result;
212     }
213
214     // alcuni metodi "di contorno" sono stati omessi.
215 }
```

A.3 BeaconJaaleeAdapter.java

```
1  public class BeaconJaaleeAdapter extends BaseAdapter {
2      private final OperationDB operationDb;
3      private ArrayList<BLEDevice> bleDeviceJaalee; //
4      private LayoutInflater inflater;
5      ArrayList<BeaconDb> dispositivi_aggiunti;
6
7      public BeaconJaaleeAdapter(Context context) {
8          this.inflater = LayoutInflater.from(context); // serve per
               riciclare le view che dipendono dal contesto per non
               doverne creare altre
9          this.bleDeviceJaalee = new ArrayList<>();
```

```
10          dispositivi_aggiunti = new ArrayList<>();
11          operationDb = new OperationDB(context);
12          dispositivi_aggiunti = operationDb.getBeaconDBList();
13
14      }
15
16      /*
17          Metodo che serve per rinfrescare la view qualora ci sono
                cambiamenti. rimuovendo e poi aggiungendo tutti i nuovi
18          beacon passati come parametri
19       */
20      public void replaceWith(Collection<BLEDevice> newBeacon) {
21          this.bleDeviceJaalee.clear();
22          this.bleDeviceJaalee.addAll(newBeacon);
23          notifyDataSetChanged(); // viene notificato se ci sono
                cambiamenti alla view e rinfrescata la pagina
24
25      }
26
27      private boolean deviceAlreadyAdded(BLEDevice device){
28          Boolean alreadyAdd = false;
29          if(!dispositivi_aggiunti.isEmpty()){
30              for(BeaconDb item: dispositivi_aggiunti){
31                  if(device.getMacAddress().equals(item.getMac())){
32                      alreadyAdd = true;
33                      return alreadyAdd;
34                  }
35              }
36          }
37          return alreadyAdd;
38      }
39
40      @Override
```

```
41    public int getCount() {
42        return bleDeviceJaalee.size();
43    }
44
45    @Override
46    public BLEDevice getItem(int position) {
47        return bleDeviceJaalee.get(position);
48    }
49
50    @Override
51    public long getItemId(int position) {
52        return position;
53    }
54
55    /*
56      restituisce la View che costituirá una riga della ListView
57     */
58    @Override
59    public View getView(int position, View view, ViewGroup parent) {
60        view = inflateLayoutRow(view, position, parent);
61        collegaAllaView(getItem(position), view, position);
62        return view;
63    }
64
65    // con questo metodo collego l'item alla lista dell'adapter
          passandogli come parametro la view
66    private void collegaAllaView(BLEDevice beacon, View view, int
          position) {
67
68
69        ViewItemDevice item = (ViewItemDevice) view.getTag();
70
71        if(deviceAlreadyAdded(getItem(position))){
```

```
72          int id = (int) getItemId(position);
73          item.immagine.setImageBitmap(dispositivi_aggiunti.
74          get(id).getImageBitmap());
75      }
76
77      item.nome.setText("Nome: " + beacon.getName());
78      item.rssi.setText("RSSI " + beacon.getRssi());
79      item.mac.setText("MAC: " + beacon.getMacAddress());
80      item.connettibile.setText("Connetibile: " +
            beacon.getConnectable()) ;
81  }
82
83  private View inflateLayoutRow(View view, int position, ViewGroup
        parent) {
84      // verifica se la riga non esiste, solo in queso caso quindi
            verrá create una nuova
85      // altrimenti no e ritornerá semplicemente la view
86      if (view == null) {
87          view = inflater.inflate(R.layout.item_bledevice, null);
                // nessun View Parent
88          //view.setTag();
89          view.setTag(new ViewItemDevice(view)); // aggangiamo la
                view all'adapter
90      }
91      return view;
92  }
93  static class ViewItemDevice{
94      TextView mac;
95      TextView connettibile;
96      TextView nome;
97      TextView rssi;
98      ImageView immagine;
99
```

```
100
101         public ViewItemDevice(View view){
102
103             immagine = (ImageView)
                    view.findViewById(R.id.image_device);
104             mac = (TextView) view.findViewById(R.id.mac);
105             connettibile = (TextView)
                    view.findViewById(R.id.connettibile);
106             nome = (TextView) view.findViewById(R.id.nomedevice);
107             rssi = (TextView) view.findViewById(R.id.rssi);
108         }
109     }
110 }
```

Appendice B

PHP Script code

Qui di seguito saranno mostrati gli script PHP che si è usato durante il progetto responsabili dell'effettuazione delle query:

B.1 Connessione db

```
class DB_CONNECT {

    // constructor
    function __construct() {
        // connecting to database
        $this->connect();
    }

    // destructor
    function __destruct() {
        // closing db connection
        $this->close();
    }

    /**
     * Connessione al database
```

```
17      */
18     function connect() {
19         // importiamo le variabili di connessione
20         require_once __DIR__ . '/db_config.php';
21 
22         // connessione al database
23         $con = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or
               die(mysql_error());
24 
25         // selezioniamo il database
26         $db = mysql_select_db(DB_DATABASE) or die(mysql_error()) or
               die(mysql_error());
27 
28         // returing connection cursor
29         return $con;
30     }
31 
32     // una volta connessi chiudiamo la connessione
33     function close() {
34         mysql_close();
35     }
36 
37 }
```

B.2 Inserimento Beacon

```
1 // JSON che prenderá il risultato della nostra query
2 $response = array();
3 // verifica di requisiti dei campi
4 if (isset($_POST['UUID']) && isset($_POST['MAC']) &&
      isset($_POST['NAME']) &&
```

```
5     isset($_POST['PASS']) && isset($_POST['MAJOR']) &&
          isset($_POST['MINOR']) &&
6     isset($_POST['SITUATION']) && isset($_POST['IMAGE']) &&
          isset($_POST['CONTACT']) &&
7     isset($_POST['LATITUDE']) && isset($_POST['LONGITUDE']) &&
          isset($_POST['POS']))
8 //if(isset($_GET['MAC']))
9     {
10    // assegnazione delle nostra variabili
11    $uuid = $_POST['UUID'];
12    $mac = $_POST['MAC'];
13    $name = $_POST['NAME'];
14    $pass = $_POST['PASS'];
15    $major = $_POST['MAJOR'];
16    // continua per gli altri parametri.......
17
18    //includiamo la connessione al nostro database
19    require_once __DIR__ . '/db_connect.php';
20    $db = new DB_CONNECT();
21    $query = "INSERT INTO foundout_beacon.setting(UUID, MAC, NOME,
          PASS, MAJOR, MINOR, SITUATION, IMAGE, CONTACT, LATITUDE,
          LONGITUDE, POS) "
22            . "VALUES ('$uuid', '$mac', '$name', '$pass' , '$major', '
                $minor', '$situation', '$image' , '$contact', '
                $latitude', '$longitude', '$pos')";
23
24
25    $result = mysql_query($query);
26
27    // se la query éandata a buon fine, allora otteniamo un
          messaggio di successo
28    if ($result) {
29        // successfully inserted into database
```

```
30          $response["success"] = 1;
31          $response["message"] = "Beacon inserito con successo.";
32
33          // trasformiamo in json
34          echo json_encode($response);
35      } else {
36          // inserimento fallito
37          $response["success"] = 0;
38          $response["message"] = "Oops! errore.";
39
40          echo json_encode($response);
41      }
42  }
43  else {
44      // required field is missing
45      $response["success"] = 0;
46      $response["message"] = "Verificare che i campi siano inseriti";
47      // echoing JSON response
48      echo json_encode($response);
49  }
```

B.3 Cancellazione Beacon

```
1  $response = array();
2
3  // check for required fields
4  if (isset($_POST['MAC'])) {
5      $mac = $_POST['MAC'];
6
7      // include db connect class
8      require_once __DIR__ . '/db_connect.php';
9
```

```
10      // connecting to db
11      $db = new DB_CONNECT();
12
13      // mysql update row with matched pid
14      $result = mysql_query("DELETE FROM foundout_beacon.setting WHERE
            MAC = '$mac'");
15
16      // per la gestione della riuscita della query si veda il caso
            //dell'inserimento
17  }
```

B.4 Aggiornamento Beacon

```
1   $response = array();
2
3   // check for required fields
4   if (isset($_POST['UUID']) && isset($_POST['MAC']) &&
        isset($_POST['NAME']) &&
5       isset($_POST['PASS']) && isset($_POST['MAJOR']) &&
            isset($_POST['MINOR']) &&
6       isset($_POST['SITUATION']) && isset($_POST['IMAGE']) &&
            isset($_POST['CONTACT']) &&
7       isset($_POST['LATITUDE']) && isset($_POST['LONGITUDE']) &&
            isset($_POST['POS'])){
8
9        // assegnazione delle nostra variabili
10      $uuid = $_POST['UUID'];
11      $mac = $_POST['MAC'];
12      $name = $_POST['NAME'];
13      $pass = $_POST['PASS'];
14      $major = $_POST['MAJOR'];
15      // continua per anche per gli
```

```
16      // altri parametri
17
18      // include db connect class
19      require_once __DIR__ . '/db_connect.php';
20
21      // connecting to db
22      $db = new DB_CONNECT();
23
24      $query = "UPDATE foundout_beacon.setting SET UUID = '$uuid',
            NOME = '$name', PASS = '$pass', MAJOR = '$major', "
25              . "MINOR = '$minor', SITUATION = '$situation', IMAGE = '
                    $image' , CONTACT = '$contact', LATITUDE = '
                    $latitude', "
26              . "LONGITUDE = '$longitude', POS = '$pos' WHERE MAC = '
                    $mac' ";
27      //per la gestione della riuscita della query
28      // si veda il caso dell'inserimento
29
30  }
```

Bibliografia

[1] Lei Deng; Gao, J.;Vuppalapati, C., Building a Big Data Analytics Service Framework for Mobile Advertising and Marketing, in 'Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on', marzo 2015.

[2] K Varnali, AEL Toker., Mobile marketing research: The-state-of-the-art, in 'International Journal of Information Management, - Elsevier', aprile 2010.

[3] Coursaris, Constantinos K.; Sung, Jieun; Swierenga, S.J, Effects of Message Characteristics, Age, and Gender on Perceptions of Mobile Advertising _ An Empirical Investigation among College Students , in 'Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on', 2010.

[4] Doug Olenick, Apple iOS And Google Android Smartphone Market Share Flattening: IDC, Forbes - http://www.forbes.com/sites/dougolenick/2015/05/27/apple-ios-and-google-android-smartphone-market-share-flattening-idc/; ultima visualizzazione: 18/12/2015.

[5] Vatanparast, R.; Piercing the Fog of Mobile Advertising in 'Management of Mobile Business, 2007. ICMB 2007. International Conference on the', 2007.

[6] Simi Bajaj K.,Egbufor F.;Pieprzyk J.; Critical analysis of spam prevention techniques in "Security and Communication Networks (IWSCN), 2011 Third International Workshop on", maggio 2011

[7] American Marketing Association, 2003, AMA Dictionary of Marketing Terms. https://www.ama.org/resources/Pages/Dictionary.aspx -ultima visualizzazione: 21/12/2015.

[8] Steve Olenski, Is Mobile Advertising Dead? Forbes - http://www.forbes.com/sites/steveolenski/2015/09/24/is-mobile-advertising-dead/ 24 settembre 2015 -ultima visualizzazione: 22/12/2015.

[9] Alexei Oreskovic, Mobile devices are now the main source of Google search traffic, Business Insider, http://uk.businessinsider.com/google-search-traffic-mobile-passes-desktop-2015-5?r=US&IR=T 5 maggio 2015 -ultima visualizzazione 22/12/2015.

[10] Bar A.; Google Rolls Out New Ads as Mobile Searches Top PCs in 10 Countries, Wall Streat Journal, http://blogs.wsj.com/digits/2015/05/05/google-rolls-out-new-ads-as-mobile-searches-top-pcs-in-10-countries/ 5 maggio 2015 - ultima visualizzazione 22/12/2015

[11] Zhang x.; Xiong K., A Conceptual Model of User Adoption of Mobile Advertising in 'Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on', 2012

[12] Evelyne Beatrix Cleff , Privacy issues in mobile advertising in 'International Review of Law, Computers & Technology', 2007;

[13] Channel access method https://en.wikipedia.org/wiki/Channel_access-_method _ ultima visualizzazione 19/02/2016;

[14] Rajesh K. B., Meera R., Archan M., Youngki L; Smartphones & BLE Services: Empirical Insights, in 'Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on', 2015.

[15] Elke M., Matthias L. , Thomas M. W.; Performance analysis of an Bluetooth Low Energy sensor system, in Wireless Systems (IDAACS-SWS), 2012 IEEE 1st International Symposium on, 2012.

[16] The Internet of Things: A Survey L Atzori, A Iera, G Morabito - Computer networks, 2010 _ Elsevier.

[17] How beacons _ small, low_cost gadgets _ will influence billions in US retail sales, http://uk.businessinsider.com/beacons-impact-billions-in-reail-sales-2015-2?r=US&IR=T, ultima visualizzazione: 20/02/2016.

[18] Macy's rolls out retail's largest beacon installation, http://www.zdnet.com/article/macys-rolls-out-retails-largest-beacon-installation/, ultima visualizzazione: 20/02/2016.

[19] Lord & Taylor expands iBeacon program to all US stores http://www.mobilecommercedaily.com/lord-taylor-expands-ibeacon-program-to-all-us-stores, ultima visualizzazione: 20/02/2016.

[20] Major League Baseball's Beacons; http://ibeaconsblog.com/mlb-beacons/ - ultima visualizzazione: 20/02/2016.

[21] Beacon e sport: il successo del Perugia Calcio, http://www.beaconitaly.it/beacon-e-sport-il-successo-del-perugia-calcio/ _ ultima visualizzazione: 20/02/2016.

[22] Getting Started with iBeacon - https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf -ultima visualizzazione: 20/02/2016.

[23] Kohne, M., Sieck, J.; Location-Based Services with iBeacon Technology, in 'Artificial Intelligence, Modelling and Simulation (AIMS), 2014 2nd International Conference on', novembre 2014.

[24] Short Range iBeacon, IBEACON DEFAULT SETTINGS, http://www.avvel.co.uk/short-range - ultima visualizzazione: 21/02/2016.

[25] Platform Overview, https://developers.google.com/beacons/overview, - ultima visualizzazione 22/02/2016.

[26] Z HE; B CUI, W; ZHOW; S YOKOY, A proposal of interaction system between visitor and collection in museum hall by iBeacon, in Computer Science & Education (ICCSE), 2015 10th International Conference on, luglio 2015.

[27] AltBeacon - The Open and Interoperable Proximity Beacon Specification, http://altbeacon.org/ - ultima visualizzazione 23/02/2015.

[28] AltBeacon Protocol Specification v1.0, https://github.com/AltBeacon/spec - ultima visualizzazione 23/02/2015.

[29] Proximity Beacon API, https://developers.google.com/beacons/proximity/guides - ultima visualizzazione 23/02/2016.

[30] Nearby Messages API, https://developers.google.com/nearby/messages/-overview - ultima visualizzazione 23/02/2016.

[31] Eddystone, https://github.com/google/eddystone - ultima visualizzazione 23/02/2016.

[32]] Eddystone-UID,https://github.com/google/eddystone/tree/master/-eddystone-uid - ultima visualizzazione 23/02/2016.

[33] Eddystone-URL,https://github.com/google/eddystone/tree/master/-eddystone-url - ultima visualizzazione 23/02/2016.

[34] Eddystone-TLM,https://github.com/google/eddystone/tree/master/-eddystone-tlm - ultima visualizzazione 23/02/2016.

[35] Bluetooth Specification Version 4.0, Volume 6, Part B, Capitolo 2, sezione 2.1

[36] SQLite Database, http://developer.android.com/reference/android-/database/sqlite/SQLiteDatabase.html - ultima visualizzazione 26/02/2016

[37] Beacon Technology: The Where, What, Who, How and Why - http://www.forbes.com/sites/homaycotte/2015/09/01/beacon-technologythewhatwhohowwhyandwhere/#6ce6131e4fc1, ultima visualizzazione 27/02/2016

[38] Y.M. Glushan, P.Y. Lavrik, M.Y. Rybalchenko, Hypergraph model of hierarchical client-server architecture for distributed computing in "Application of Information and Communication Technologies (AICT), 2015 9th International Conference on", ottobre 2015

[39] S. Chinnappen-Rimer, G. P. Hancke; An XML model for use across heterogeneous client-server applications in "Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS '03. 2003 IEEE International Symposium on", luglio 2003

[40] A. Di Iorio, F. Vitali, Introduzione ad HTTP e REST, http://vitali.web.cs.unibo.it/TechWeb15/LezioniDelCorso, ultima visualizzazione mercoled' 02/05/2015;

[41] L'iBeacon di Samsung si chiamerà Flybell, http://www.ispazio.net/465127/libeacon-di-samsung-si-chiamera-flybell, ultima visualizzazione 02/03/2016;

[42] San Francisco Airport testing beacon system for blind travelers; http://www.theverge.com/2014/7/31/5956265/san-francisco-airport-testing-beacon-system-for-blind-travelers; ultima visualizzazione 03/03/2016

[43] Musei, http://www.beaconitaly.it/musei/; ultima visualizzazione 03/03/2016

Ringraziamenti

Grazie. Grazie è una parola che molto spesso viene detta a cuor leggero, senza pensarci, senza rendersi conto dell'importanza che esprime. Con *grazie* si vuole rendere partecipi colui o coloro che con un gesto, una parola, hanno fatto qualcosa per noi, piccola o grande che sia, che può svariare da un semplice favore, da una semplice gentilezza, fino ad arrivare ad un qualcosa di più importante, come l'aiuto fornito per raggiungere un traguardo, un obiettivo, come in questo caso. La cosa importante è che queste persone ci hanno fatto sentire in quei momenti *appagati*, *felici* perché hanno fatto loro un momento di difficoltà nostro. Durante questo periodo di intenso studio, di difficoltà, di stress, di tensioni, posso dire che ho avuto un sacco di persone al mio fianco che mi hanno aiutato, persone che io considero: *amici*. Innanzitutto, vorrei ringraziare la mia famiglia, mia sorella e la sua famiglia, mia nonna e mio nonno Matteo che adesso non c'è più, i miei zii, in particolare mia zia Luigia, che in questi tre anni mi ha sempre sopportato anche quando ero particolarmente nervoso e insopportabile. In seconda battuta tutti i miei amici, da quelli di una vita che io considero fratelli: Michele, Attilio, Andrea, Giuseppe, Antonio, Gianni, più Raffaele e Antonio che sono entrati di diritto in questa cerchia con cui ho condiviso mille esperienze e mille bevute (più con Antonio quest'ultimo punto) fino ad arrivare a quelli conosciuti in questi quattro anni. Quando abbandoni il tuo paese ti sembri spaesato, pensi a quando possa essere difficile in quel momento, dover riniziare da capo, hai mille paure e pensi che forse hai fatto la scelta sbagliata, e ti chiedi spaventato chissà come sarà da li in poi il futuro, ora che non hai più la

tua routine, il tuo mondo definito. Personalmente per fortuna questo non è successo, o se anche fosse è passato tutto subito, grazie a tutti coloro che mi hanno circondato e che mi hanno fatto sentire "a casa". Ho conosciuto un sacco di persone, alcune sono state di passaggio, altre sono rimaste e spero rimarranno per sempre. Andando con ordine (non di importanza perché sono tutti importanti allo stesso modo, ma conoscitivo) penso ai miei mitici coinquilini storici Albert e Davide (unici e insostituibili) più quelli che ora vivono da tutt'altra parte come Marco, Tommy.Un grazie anche alla new entry Ale. In università invece meritano un ringraziamento speciale in primis Riccardo, (che mi sembra di conoscerlo da 20 anni) ma anche Il Poggio (ha anche un nome vero ma è solo di facciata), Coppolino, Peco, Sarza, Fiore, Rappo e "le Silvie" (Silvia, Silvia e Alice). Per quanto riguarda gli amici "extra-universitari", amici che si sono dimostrati veramente tali, con cui ho condiviso momenti di divertimento senza eguali, che considero punti fermi, imprescindibili a cui sono veramente legato penso a Fabio, Silvia, Anna, Fabio, Vittorio, Paola, Alessandro, Simona, Poggi. Un ringraziamento è doveroso anche al mio relatore, il professore Marco di Felice. Lo ringrazio perché oltre ad essere un professore estremamente preparato e competente, è una persona umanamente fantastica: non ha fatto mai mancare il suo supporto, il suo sostegno, e tutto l'aiuto necessario sempre con il sorriso. Infine un ultimo ringraziamento, lo voglio riservare a tutte quelle persone, in particolare una di cui non voglio far nome, che hanno preso strade diverse per ragioni svariate, giuste o sbagliate che siano non importa, ma che ora non sono più al mio fianco. Purtroppo la vita mette di fronte a queste situazioni e a volte nonostante si cerchi di cambiarle e migliorarle non va come ci si aspetta. Voglio ringraziarle perché nonostante i problemi vari, se io sono arrivato a questo traguardo è anche grazie a loro. Spero che abbia menzionato tutti, e se non l'ho fatto (cosa molto probabile) non me ne vogliate, ma rischierei che i ringraziamenti diventino più lunghi della tesi stessa. Potrei dire mille cose su ognuno di voi, anche su quelli che ho sicuramente dimenticato di citare, ma una cosa ancora la voglio dire, ancora una volta, a tutti: *grazie*.