UNIVERSITA' DEGLI STUDI DELLA BASILICATA

DIPARTIMENTO DI MATEMATICA, INFORMATICA ED ECONOMIA

Corso di Laurea in Scienze e Tecnologie Informatiche

Controllo georeferenziato per l'accesso a documenti digitali

Candidato:

Domenico Rossini

Matricola 56825

Relatore:

Ing. Domenico Daniele Bloisi

Anno Accademico 2021–2022

"C'è vero progresso solo quando i vantaggi di una nuova tecnologia diventano per tutti."
(Henry Ford)

Sommario

Oggigiorno tutti i dispositivi mobili sono dotati di modulo e antenna GPS in modo da captare i segnali inviati dai satelliti e determinare latitudine e longitudine del dispositivo e quindi dell'utente.

L'obiettivo della tesi e quello di adoperare in maniera intelligente queste caratteristiche dei dispositivi, in modo da poter offrire i più disparati servizi all'utente, come ad esempio, nel caso approfondito nella tesi, la possibilità per gli utenti di avere l'accesso a documenti e file diversi a seconda della loro posizione, con una precisione tale da poter capire se l'utente si trovi o meno in una determinata area di un edificio.

E' stato quindi progettato un applicativo mobile che, a condizione che l'utente consenta la localizzazione del dispositivo e che quest'ultimo si trovi in una determinata zona, permette di avere l'accesso ad una cartella di un server remoto contenente dati che diversamente non sarebbero accessibili.

Ringraziamenti

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale, da solo non sarei mai riuscito a compiere tutto ciò.

Un sentito grazie va al Professore Domenico Daniele Bloisi, il quale mi ha seguito in questo lavoro di tesi, per la sua infinita disponibilità e tempestività nel rispondere ad ogni mia richiesta o dubbio. Grazie per avermi fornito materiale e suggerimenti utili alla stesura dell'elaborato.

Un enorme grazie va ai miei genitori, senza di loro non sarei mai potuto arrivare fin qui. Grazie per esserci sempre stati.

Ringrazio il mio gruppo di studio: Antonio, Marco, Rocco Di Pierro, Rocco Galasso e Marika, amici e compagni di università, senza i quali le giornate in UniBas e fuori non sarebbero state le stesse.

Ringrazio inoltre tutti i colleghi del corso di informatica che si sono mostrati sempre disponibili e aperti al confronto in moltissime situazioni.

Indice

Capitolo 1

Introduzione

1.1 Definizione del problema

Andremo a trattare l'utilizzo e la definizione dei principali componenti utili alla creazione di un servizio LBS.

Fondamentali per il prosieguo della trattazione sono le differenze fra **geolocalizzazione** e **georeferenziazione** :

- **Geolocalizzazione -** l'identificazione della posizione sul pianeta di un dispositivo.
- **Georeferenziazione -** l' attribuzione di un metadato geografico ad un insieme di dati.

Grazie all'unione di queste due tecnologie infatti è possibile fornire un numero molto grande di servizi, con possibilità di fare geomarketing e integrazioni di vario genere con i social, con la preerogativa di rispettare la privacy e la sicurezza dei dispositivi e degli utenti che le utilizzano.

1.2 Obiettivo della tesi

L'obiettivo della tesi è quello di fornire un'applicazione che nella maniera più *semplice* e *immediata*, permetta all'utente l'accesso a file memorizzati su di un server remoto, previa la localizzazione dello stesso tramite le tecnologie di geolocalizzazione messe a disposizione dal sistema Android da gli SDK e le API per il *cloud storage remoto* fornite da Google tramite la piattaforma *Firebase*.

Un' applicativo del genere ha una moltitudine di possibilità di utilizzo nelle più disparate situazioni, noi vedremo un caso d'uso basato sulla necessità all'interno dell' UniBas di accedere a materiale differente a seconda dell'edificio in cui ci si trova.

L'applicativo realizzato, inoltre, è parte di un progetto più grande portato avanti

dal collega Enrico Rinaldi, il quale ha studiato una soluzione al medesimo problema basata su tecnologie browser.
I due progetti sviluppati sono complementari e sono stati presentati alla fiera Maker Faire di Roma che si è tenuta dal 7 al 9 Ottobre.

1.3 Contributi personali

E'stata progettata e sviluppata un'applicazione mobile con architettura *Model View Controller* scritta in java, in cui sono memorizzate le informazioni spaziali delle aree rappresentate in un file .JSON, mentre le informazioni a cui l'utente può accedere sono memorizzate e gestite dai server Firebase di Google.

L'utente interagisce con l'applicazione tramite una vista che gli permette di vedere dove si trova, e come il sistema riconosce l'area circostante grazie a una cartina con delle aree evidenziate in risalto.

La cartina è visualizzata grazie all'SDK di Google Maps che permette la visualizzazione della posizione dell'utente, oltre la possibilità di disegnare al di sopra della stessa, aggiungendo poligoni, punti e etichette.

Le aree in risalto invece sono importate da un file memorizzato all'interno del dispositivo in formato JSON, in seguito il file viene letto grazie alle librerie GSON e poi i poligoni vengono aggiunti alla mappa.

Il sistema è quindi in grado di confrontare la posizione dell'utente con le aree dei vari poligoni e determinare se si trovi all'interno o meno di una zona contenente dei file accessibili.

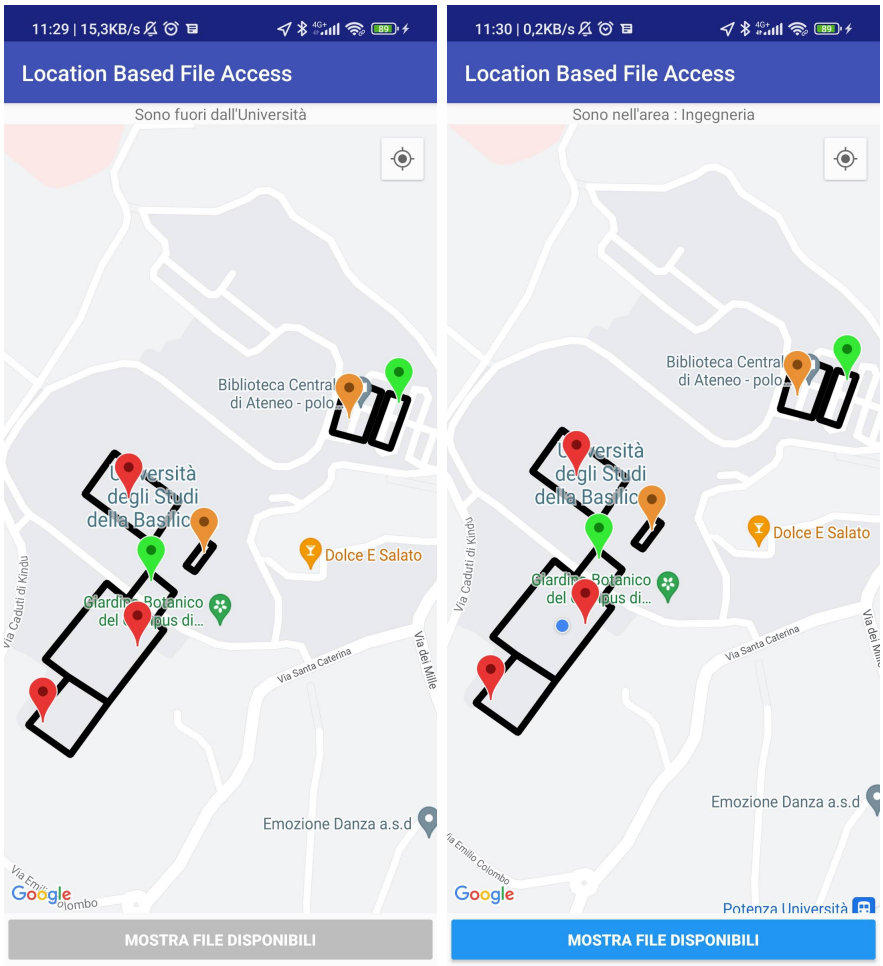


Figura 1.1: Schermate del Progetto

1.4 Struttura della tesi

Nel capitolo **Stato dell'Arte** vengono presentate le tecnologie attuali sulle quali si basa il funzionamento degli strumenti di geolocalizzazione, oltre che i componenti e il funzionamento di servizi che basano il loro funzionamento su queste tecnologie.
Viene presentato il formato JSON grazie al quale oggi vengono effettuate la maggior parte dell comunicazioni fra dispositivi.
Vengono inoltre presentate tutte le API e librerie utilizzate per la realizzazione del progetto.

Nel capitolo **Progettazione** viene approfondita la descrizione del problema e di come si è deciso di affrontarlo tramite scelte di grafica e di implementazione di funzioni.
Una parte sarà dedicata al *Design Thinking* a quello che rappresenta e alle scelte che porta in fase di progettazione.
Infine saranno presentati diagrammi e grafici riguardanti la progettazione del sistema stesso, sarà presente inoltre un'approfondimento su tutte le tecnologie secondarie utilizzate.

Nel capitolo **Implementazione** vengono discusse tutte le scelte implementative

e l'utilizzo delle tecnologie individuate durante la fase di progettazione.

Nel capitolo **Casi d'uso** viene spiegato il caso d'uso individuato per il funzionamento di un'applicazione del genere e in seguito viene fatta una piccola parentesi sulla fiera Maker Faire di Roma e sulla partecipazione del progetto congiuntamente a quella di altri colleghi.

Nel capitolo **Conclusioni** vengono fatte alcune considerazioni finali e personali sul progetto, oltre che la discussione di possibili sviluppi futuri dell'applicativo.

Capitolo 2

Stato dell'Arte

2.1 Georeferenziazione

Per *georeferenziazione* [1] si intende l'azione di creare coppie di dati formate da una posizione sulla Terra e dal relativo dato collegato, questi vengono poi utilizzati alla base del funzionamento di applicazioni del tipo **LBS** e **GIS**.

- **GIS -** servizi basati su mappe, che raccolgono i dati spaziali e li trasformano in visualizzazioni per l'utente finale. Sono software professionali utilizzati in ingegneria civile, in grado di mappare visivamente un luogo, utilizzi pratici sono ad esempio la mappatura di oleodotti tramite droni o il monitoraggio della recessione delle coste dovuta al riscaldamento globale.

- **LBS -** sono la forma consumer dei prodotti GIS, implementati all'interno di applicazioni di uso quotidiano come le mappe di Google o le applicazioni che tracciano i comportamenti dei consumatori tramite i social media, ma è con lo spopolare degli smartphone che i servizi di questo genere sono diventati parte integrante della nostra vita quotidiana.
  Le applicazioni di tipo LBS utilizzano nella maggior parte dei casi la posizione attuale dell'utente in modo da fornire informazioni e dati in *real-time* all'utilizzatore, basti pensare a applicazioni del tipo di **Uber** e **Glovo**.
  Un altro tipo di utilizzo di questi strumenti e il **geofencing**, letteralmente il creare delle recinzioni virtuali intorno ad alcuni luoghi, dove l'utente una volta entrato riceve messaggi o informazioni relative al luogo in cui si trova a fini commerciali e non. Inoltre molti motori di ricerca abbinano la localizzazione dell'utente con i dati delle località georeferenziate intorno a lui in modo da indicargli luoghi specifici raggiungibili a piedi o in auto facilmente.

2.2 Geolocalizzazione GPS

Per far si che tutti questi servizi funzionino in maniera adeguata è necessario conoscere con molta precisione la posizione del dispositivo che sta richiedendo un' LBS.

Per **Geolocalizzazione GPS** [2] si intende la tecnologia che si serve del sistema di posizionamento basato sui satelliti in orbita, in grado di fornire la posizione quasi esatta di ogni dispositivo dotato di un ricevitore.

Il sistema Global Positioning System [5], il cui nome completo è **NAVSTAR GPS** nasce nel 73 ma solo nel 91 viene aperto all'utilizzo civile, inizialmente limitato e portato a un'accuratezza di 1km per le rilevazioni civili.
Limitazione che poco dopo verrà rimossa e portata a 10 20 metri di imprecisione.

Principalmente il sistema GPS si compone di 3 componenti che lavorano insieme per offrire il posizionamento più accurato possibile :

- **Segmento spaziale -** i 31 satelliti in orbita, disposti su 6 piani distinti, che emettono continuamente segnali su 2 canali differenti, **L1** per usi civili e **L2** per scopi militari con precisione centimetrica.

- **Segmento di controllo -** la combinazione delle 8 stazioni terrestri e le 4 antenne che controllano i satelliti posizionate in punti strategici della terra.

- **Segmento utente -** l'insieme dei ricevitori GPS posizionati sul pianeta, militari e civili.
  Solitamente un ricevitore è composto da un processore, un'antenna e un'orologio per gestire le operazioni di sincronizzazione.

L'assegnazione di coordinate a un dispositivo si basa sul principio della **trilaterazione** [3] : sfruttando il segnale di 3 satelliti il ricevitore è in grado di capire, grazie al tempo di percorrenza dei 3 segnali dal satellite al ricevitore, il punto del pianeta in cui si trova il dispositivo.

L'orologio all'interno del ricevitore però non è preciso come quello all'interno dei satelliti e necessita di un quarto satellite per correggere i possibili errori.

Un ricevitore si collega a un massimo di 5 satelliti in modo da averne 1 di scorta in caso di problemi.

E'd'obbligo precisare che la tecnologia GPS non è l'unica in grado di determinare la posizione di un ricevitore sul pianeta, questa tecnologia è infatti di proprietà degli Stati Uniti.
Altre nazioni hanno sviluppato sistemi simili per la geolocalizzazione, come ad esempio il sistema **Galileo** di proprietà dell'Europa.

Le altre componenti fondamentali per identificare la posizione di un dispositivo sulla Terra sono :

- **Ricevitore -** che può essere uno smartphone o un dispositivo specializzato, che lavorando come un client è in grado di richiedere e ricevere la sua posizione attuale.
  Funzioni principali di un ricevitore sono quella di identificare un satellite attraverso una banca dati, e come detto prima quella di calcolare il *delta t* ovvero il tempo impiegato dal segnale per arrivare dal satellite al dispositivo.

- **Satellite -** nel caso della tecnologia NAVSTAR sono 31 i satelliti in orbita, disposti in gruppi da 4 per piano orbitale, in modo che su qualsiasi punto della terra si ricevano i segnali di almeno 5 satelliti.
  Ogni satellite a seconda della versione possiede un numero variabile di orologi atomici al *cesio* e *rubidio*. Le generazioni di satelliti che si sono susseguite fino ad oggi sono 5 con autonomie via via maggiori.

2.3 Altri servizi per la localizzazione

E'possibile identificare la posizione di un dispositivo su una mappa anche tramite altre tecnologie che utilizzano i seguenti metodi di localizzazione :

- **Trilaterazione -** come già visto il calcolo del tempo impiegato dal segnale per percorrere lo spazio dalla sorgente al ricevitore. L'individuazione della posizione avviene determinando 3 valori fondamentali per i quali si può costruire un solo triangolo.

- **Localizzazione tramite beacon -** forniscono la posizione del ricevitore in tempo reale con una precisione altissima con un raggio limitato. I beacon sono trasmettitori radio a bassa frequenza che sfruttano la tecnologia bluetooth a risparmio energetico per trasmettere dati entro un raggio che va dai 10 centimetri ai 70 metri, i dispositivi sono alimentati da batterie che, data la poca necessità di energia dei dispositivi, durano anche anni.

- **Crowdsourcing -** mediante ricevitori che rilevano i movimenti, ad esempio dispositivi indossati dal personale per facilitare la localizzazione di un oggetto dotato di tag.

Le tecnologie più utilizzate per la localizzazione di dispositivi sono le seguenti :

RFID - abbreviativo di *Radio Frequency Identification*, ovvero l'identificazione della radiofrequenza. E'una tecnologia di identificazione intelligente i cui dati vengono codificati in un tag e acquisiti da un lettore mediante onde radio, come succede per i codici a barre, solo che qui ci basiamo sulla scansione di onde radio invece che di una scansione ottica.
Altra caratteristica è che i dati memorizzati nei tag sono dinamici, è quindi possibile modificarli nel tempo.

Esistono due tipi di tag :

- **Tag passivi -** privi di alimentazione, formati da un'antenna che riceve le onde radio del lettore e alimenta il teg, questi dispositivi contengono informazioni molto basilari ma possono essere molto piccoli e durare molto nel tempo.

- **Tag attivi -** possiedono una fonte di alimentazione autonoma e un trasmettitore, possono memorizzare quantità di dati maggiori rispetto i passivi altre ad avere una portata maggiore.

Localizzazione tramite celle telefoniche - permette la localizzazione di un telefono anche se quest'ultimo non è dotato di nessun sistema di localizzazione in

particolare, senza che il proprietario si accorga di tutto ciò. La localizzazione è basata su concetti di *triangolazione radio* e *multilaterazione* in maniera molto simile a quello che avviene con i sistemi GPS, solo che vengono utilizzate metodologie molto più approssimative e imprecise.
Il tutto è reso possibile dalle antenne telefoniche che sono in grado di determinare a che distanza si trova un dispositivo collegato tramite la misurazione dell'intensità del segnale.
Per aumentare la precisione occorre avere una rete densa di antenne il più possibile vicine fra di loro, vengono infatti utilizzate 3 o più di loro dalle quali verranno presi i dati e schematizzati in uno o più triangoli che si intersecano fino a identificare una posizione con una tolleranza che va delle centinaia di metri ai kilometri.

Localizzazione tramite access point WIFI - essendo gli access point nella maggior parte dei casi dispositivi fissi, è possibile risalire alla posizione di un dispositivo tramite un enorme database formato dai dati pubblici di ogni rete (SSID e MAC) e la loro posizione, ottenuta ad esempio da Google durante il mapping delle strade con i veicoli street view, oppure più semplicemente tramite le informazioni anonime che condividono ogni giorno i nostri telefoni con chi ne gestisce il sistema. I veicoli di Google o gli smartphone infatti durante il movimento rilevano la presenza di reti WIFI e le registrano sulla mappa, come dichiarato nella informativa della privacy che tutti accettiamo.

Sistemi UWB - nati per sopperire a i limiti della tecnologia GPS quali una forte attenuazione del segnale in indoor, utilizzando come dice il nome segnali a banda ultra larga.
I sistemi di questo genere sfruttano quindi una tecnologia di telecomunicazioni basata su impulsi a banda molto larga (sopra i 500MHz) per le comunicazioni o la localizzazione che permette di avere i seguenti punti a suo favore :

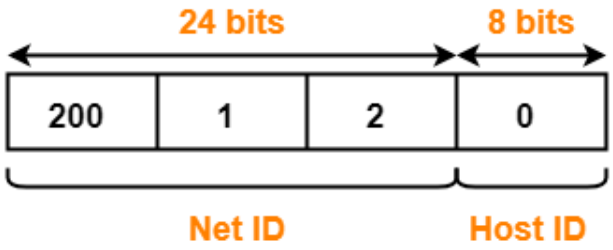
- **Alto bit rate -** nelle corte distanze infatti è in grado di raggiungere velocità di Gb/s.
- **Semplicità di implementazione -** le tecnologie di questo tipo permettono di modulare direttamente un'impulso su un'antenna, il che permette la realizzazione di ricevitori poco costosi.
- **Immunità ai cammini multipli dei segnali -** caratteristica fondamentale che permette un'alta risoluzione dei cammini multipli, grazie alla brevissima durata dell'impulso.
  Si è quindi in grado di rilevare con facilità un segnale riflesso e quindi da qui possiamo notare il principale motivo per il quale questa tecnologia è molto utilizzata in ambienti interni.
- **Comunicazione e localizzazione contemporanee -** solitamente queste azioni svolte contemporaneamente portano a una numerosa presenza di collisioni, nei sistemi UWB la grande larghezza di banda permette di separare i due flussi di dati e renderli indipendenti.

- **Capacità di penetrazione attraverso gli ostacoli -** i segnali UWB subiscono un'attenuazione dovuta agli ostacoli molto minore rispetto a quelli GPS, questo perchè i picchi di assorbimento della maggior parte dei materiali sono concentrati su bande di frequenza ristrette.

Geolocalizzazione basata su indirizzi IP - altro metodo molto approssimativo per la localizzazione di un dispositivo è il tracciamento del suo indirizzo IP pubblico. La posizione viene stabilita tramite il protocollo WHOIS interrogando i database che mantengono traccia del punto di connessione di ogni dispositivo alla rete.
L'indirizzo IP identifica univocamente un host su una rete e si compone di 2 parti :

- **NetID -** indica la classe di reti alla quale l'indirizzo appartiene, queste possono essere : classe A, classe B, classe C e così via.
  Le classi servono a dividere lo spazio di indirizzamento dello standard IPV4 in modo da raggruppare gli indirizzi per gerarchie di grandezza. Ad esempio le reti di uso privato/domestico sono di classe C, gli indirizzi sono quindi comporti da 3 Byte per il NetId e 1 Byte per l'HostId.

- **HostID -** indica precisamente la macchina collegata all'indirizzo IP



Gli indirizzi IP vengono generati dall'organizzazione *Internet Assigned Numbers Authority* (IANA), l'assegnazione degli indirizzi è responsabilità dell'Registro Internet Regionale, che controlla gli assegnamenti su base regionale effettuati dagli ISP e le richieste di indirizzi da parte delle organizzazioni.

Attualmente esistono 2 famiglie di indirizzi IP : l'**IPV4** che è anche il tipo più utilizzato, è formato da 4 Byte binari e permette 2^{32} combinazioni di indirizzi, la versione più recente dello standard di indirizzi IP è invece l'**IPV6** caratterizzato da una codifica esadecimale con la possibilità di comporre 2^{128} combinazioni.

Un altro modo per classificare gli indirizzi IP è la suddivisione in indirizzi **Statici** e indirizzi **Dinamici** :

- **Statici -** utilizzati per identificare dispositivi che dovrebbero essere permanenti sulla rete, per esempio stampanti e server.
  Questo tipo di indirizzo è utilizzato per rendere un dispositivo immediatamente raggiungibile e distinguibile tramite il suo indirizzo IP, questo però espone le macchine con questo tipo di indirizzo a più possibili attacchi tramite la rete.

- **Dinamici -** usati per identificare dispositivi non permanenti sulla rete, l'indirizzo verrà assegnato da una lista di indirizzi disponibili dal protocollo **DHCP**.

2.4 Google Maps

Google Maps è uno dei LBS più utilizzati al mondo che consente la ricerca e la visualizzazione di carte geografiche tramite pagina web o applicazione.
Tramite le mappe di Google è possibile cercare servizi, attrazioni o particolari luoghi, è inoltre possibile tracciare un percorso dalla posizione dell'utente al luogo individuato.

Lanciato nel 2005 inizialmente era disponibile solo negli Stati Uniti, in seguito è stato reso disponibile in tutte le altre nazioni, la prima città italiana a far parte delle mappe di Google fu Torino in occasione delle olimpiadi invernali.

Le mappe visionabili sulla piattaforma sono basate su una variante della proiezione di Mercatore ma utilizzano un sistema di coordinate del sistema geodetico mondiale. Poichè la proiezione di Mercatore e le sue varianti hanno valori infiniti ai poli, Google Maps non mostra i poli.

2.4.1 Google Street View

Altro progetto parallelo a Google Maps è Street View, questo permette di visualizzare fotografie a 360° di numerose località e strade del mondo.
Le foto sono raccolte tramite delle auto dotate di fotocamere e per questo le immagini che l'utente vede non sono i tempo reale.

Il servizio ha avuto molti problemi riguardanti la privacy delle persone che per caso venivano fotografate per strada, questi problemi sono stati risolti offrendo la possibilità a questi ultimi di essere oscurati semplicemente facendone richiesta. In seguito è stato integrato un'algoritmo che riconosce i volti e li offusca automaticamente.

2.5 Riferimenti legali

Fondamentale è essere consapevoli degli impatti normativi che l'utilizzo di strumenti di localizzazione incontra e, precisamente, dei limiti dettati dalla disciplina giuslavorista e dalla protezione dei dati personali.
Il trattamento dei dati relativi alla geolocalizzazione degli individui è disciplinato dal **GDPR** che pone le regole generali quando i dati sulla posizione sono riconducibili, anche in modo indiretto, ad un individuo.

2.5.1 GPDR e ePrivacy

Per regolamentare tutti i servizi che richiedono la posizione dell'utente è nata la direttiva **ePrivacy**, che estende il regolamento generale europeo.

La normativa prevede che solo i fornitori dei servizi di comunicazione o fornitori

di servizi a valore aggiunto possano trattare i dati relativi alla posizione.

I casi possibili sono due :

- I dati vengono trattati in forma **anonima**
- I dati vengono trattati in base al **consenso** dell'interessato, purche'l'eleaborazione sia necessaria al fine di erogare un servizio a valore aggiunto

Un servizio a valore aggiunto è un servizio che richiede il trattamento di dati di traffico o posizione oltre quelli necessari alla trasmissione di una comunicazione.
Il consenso non può valere per tutti i servizi e quindi ogni servizio che intende accedere a questi dati sensibili dovrà richiedere il permesso esplicitamente.
Oltre alla richiesta di accedere al dato sulla posizione ogni servizio deve anche indicare che tipo di dati tratterà, per quanto tempo rimarranno memorizzati e se sono comunicati a terzi.

Tutti gli utenti hanno il diritto di revocare il proprio consenso in qualsiasi momento, in questo caso è necessario interrompere immediatamente l'utilizzo dei suoi dati di localizzazione.

E'obbligatorio infatti fornire agli utenti un modo semplice e gratuito per revocare il proprio consenso al trattamento di questi dati in qualsiasi momento. E' inoltre possibile far si che l'utente possa revocare il proprio consenso in maniera temporanea, in questo caso invece è necessario mettere a conoscenza l'utente del funzionamento di questa pratica e di come e quando sarà riattivata la localizzazione nei suoi confronti

Fanno eccezione i dati relativi alla posizione raccolti in maniera anonima, questi infatti non richiedono alcun tipo di consenso da parte dell'utente, ma rimane il problema di raccogliere questi dati davvero senza tracciare minimamente la loro fonte. Una possibile soluzione per rendere davvero anonimi dei dati è quella di utilizzare dei dati aggregati, grandi insiemi di dati provenienti da migliaia di persone alle quali diventa difficile risalire singolarmente.

A trattare principalmente dati sulla posizione sono :

- **Operatori di telefonia mobile -** che utilizzano la posizione per poter instradare le comunicazioni attraverso le torri cellulari.
- **Sistemi Operativi -** principalmente quelli per dispositivi mobili, conoscono la posizione del dispositivo per poter erogare servizi in caso di richiesta da parte dell'utente.
- **Applicazioni e partner -** nel momento in cui l'uso di questi dati è necessario per l'esecuzione dell'app.
- **Servizi di analisi della posizione -** tramite ad esempio dispositivi IoT, possono effettuare analisi relative alla quantità di persone presenti in un luogo.

Ovviamente i dati relativi alla posizione sono accessibili in casi di emergenza dalle autorità pubbliche competenti.

2.6 Firebase

Firebase è un servizio *back-end* online fornito da Google che permette di salvare e sincronizzare i dati elaborati dalle applicazioni web e mobile.

La sua funzione principale è quella di **database NoSQL**, ma offre anche servizi di *cloud storage*, *autenticazione* e *hosting* facilmente integrabili in progetti software.

Permette agli sviluppatori di occuparsi principalmente della parte *front-end* dell'applicazione e si occupa di fornire i dati richiesti tramite **Websoket**.

Websoket è una tecnologia web che fornisce canali di comunicazione *full-duplex* (bidirezionale simultanea) attraverso una singola connessione *TCP*. Questo protocollo permette una maggiore interazione fra *client* e *server*, facilitando la realizzazione di applicazioni che forniscono contenuti in tempo reale in quanto molto più veloce del protocollo **HTTP** grazie al fatto che è necessaria la sola connessione al soket per effettuare delle comunicazioni.

Gli svantaggi di **Firebase** sono i seguenti :

- **Query limitate**
- **Modelli di dati relazionali non applicabili**
- **Nessuna installazione *on premise***

2.7 URI e URL

Un'**URI** è una sequenza di caratteri che identifica una risorsa globalmente in maniera universale e univoca.

Un'**URL** è una sequenza di caratteri che identifica univocamente il percorso per raggiungere una risorsa all'interno di una rete di computer.

2.8 JSON

Acronimo di **JavaScript Object Notation** è un formato di file adatto allo scambio di dati fra applicazioni *client - server* basato principalmente su due strutture, un'insieme di coppie chiave/valore e un'elenco ordinato di valori.
Essendo le due strutture che lo compongono universali, il formato JSON è supportato dalla maggioranza dei linguaggi di programmazione moderni.

I dati all'interno di un file JSON sono letti tramite un *parser* JSON e convertiti nel tipo di dato adatto comprensibile dal linguaggio in uso.

Un file JSON descrive una serie di oggetti già inizializzati, cioè con attributi già assegnati e pronti per essere utilizzati.

Il formato **JSON** nasce proprio dal modo in cui sono scritti array e oggetti in *JavaScript*, e permette di essere di facile comprensione per l'uomo e per la macchina, cosa per niente scontata.
Viene introdotto nei primi anni 2000, quando a ogni richiesta dell'utente il server forniva le informazioni in HTML e il browser ogni volta renderizzava una nuova pagina. Questo procedimento era molto lento e inefficiente, richiedeva infatti un nuovo rendering anche solo se ci si spostava da una sezione di una pagina all'altra.

A partire da Internet Explorer 5 venne poi implementato un nuovo modo di navigazione nel web grazie all'utilizzo di JavaScript come linguaggio di programmazione universale nei browser, la novità consistevano infatti nella possibilità di caricare in maniera incrementale una pagina a seconda dei movimenti dell'utente su di essa.

Inizialmente i dati erano scambiati tra client e server in formato **XML** tramite il protocollo **SOAP**, questo però non era nativamente supportato da JavaScript.
Solo in un secondo momento Douglas Crockford prese una parte del formato XML come base per creare il formato JSON, con l'obiettivo di essere più facile da leggere per le persone e più veloce da analizzare per i browser.

Durante lo stesso periodo un'altra tecnologia per trasferire iniziò ad essere sempre più utilizzata : le **API REST**, che permettevano, al contrario delle API SOAP, di utilizzare più formati di dati, tra i quali XML, JSON e HTML.

Oggi JSON è lo standard più utilizzato per lo scambio di dati tra client web e mobile e servizi di back-end.

Anche questa tecnologia non è però esente da limiti, di seguito alcuni :

- **Nessuno schema**
- **E'presente un solo tipo di numero**
- **Nessun tipo di data**
- **Non è possibile inserire commenti**
- **Verbosità**

2.9 GeoJSON

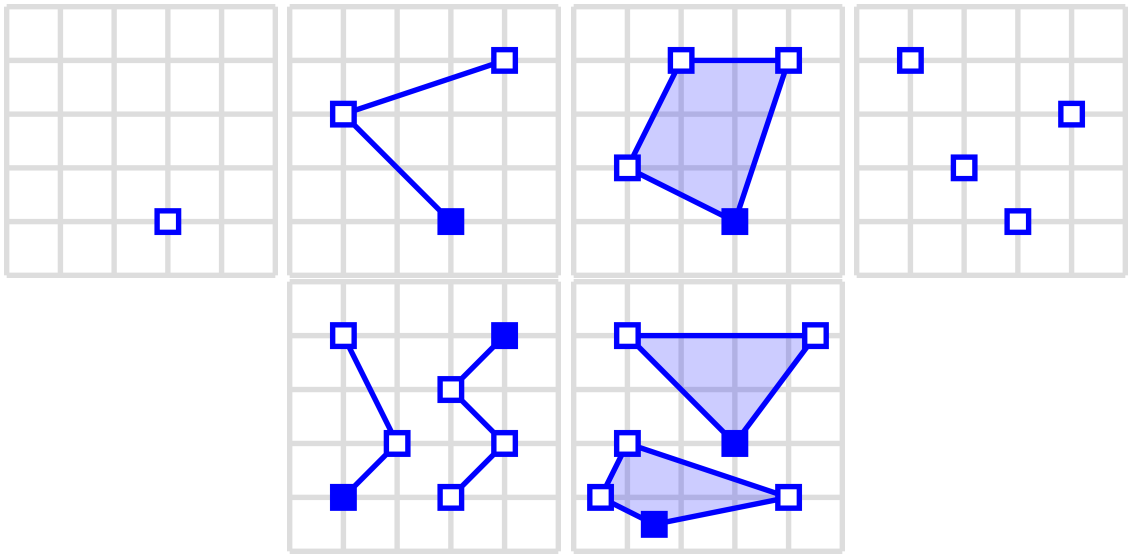
GeoJSON è un formato che permette l'interscambio di dati geospaziali con standard aperti, rappresentati tramite *feature* geografiche e i loro attributi non spaziali.

Questo formato si basa fortemente su **JSON** per il modo di rappresentare i valori tramite coppie chiave/valore e elenchi di valori.

Le geometrie rappresentabili sono le seguenti :

- **Punto**

- **LineString**
- **Poligono**
- **Multipoint**
- **MultiLineString**
- **MultiPolygon**



E' possibile inoltre dichiarare oggetti geometrici con proprietà aggiuntive denominati *feature*.

Capitolo 3

Progettazione

Questo capitolo contiene la descrizione del problema dal punto di vista dell'utente, e la discussione delle scelte progettuali.

Verrà inoltre fatto un'approfondimento sul design thinking e su quello che ne comporta.

3.1 Requisiti funzionali

In questa sezione si discute dei **requisiti funzionali** del progetto, cioè dei principi sulla base dei quali sono stati scritti uno o più componenti dell'applicazione, definendone gli *input* e gli *output*, nonchè il comportamento degli stessi.

Un'applicazione come quella in esame è pensata per funzionare come un **LBS** e ha quindi la necessità di accedere a una o più tecnologie di posizionamento fornite dal sistema Android. Una volta ottenuta la posizione il sistema permette all'utente di accedere, visualizzare e scaricare determinati file memorizzati su una macchina remota.

I requisiti da parte dell'utente per far si che l'applicazione funzioni a dovere sono i seguenti :

- **Connessione a internet -** l'utente deve connettere il dispositivo a una rete affidabile in modo che i dati riguardanti le mappe, i file disponibili, e in alcuni casi anche la posizione possano essere visualizzati in maniera adeguata.

- **Ricevitore GPS attivato -** la quasi totalità dei dispositivi mobili oggi è in possesso di un ricevitore GPS, per questo è stato reso necessario per la corretta localizzazione dell'utente e di conseguenza fondamentale per il funzionamento dell'applicazione.

- **Google Play Services -** questa suite di servizi forniti da Google sono resi necessari dal fatto che la mappa su cui si basa il funzionamento del sistema è basata su Google Maps.

I telefoni quindi che non hanno accesso a questo pacchetto, come ad esempio gli ultimi Huawei, non potranno eseguire l'applicazione.

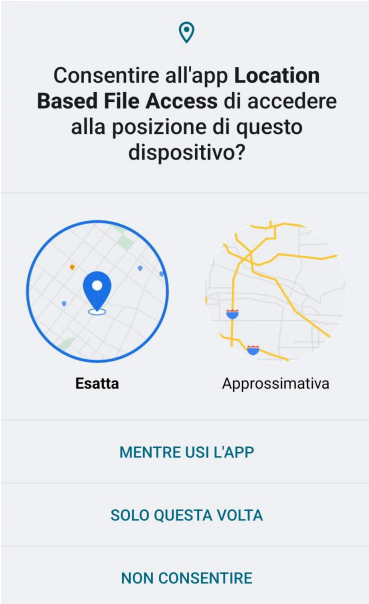
Una volta avviata l'applicazione, il sistema mostrerà un interfaccia molto familiare per chi utilizza dispositivi Android, formata da una grande area destinata alla visualizzazione della mappa con all'interno il pulsante per centrare la mappa nella propria posizione, e subito sotto un unico pulsante funzione che si attiva, cambiando colore, solo nel caso in cui l'utente si trovi all'interno di una delle zone designate.

L'interfaccia è realizzata con layout responsivi in modo che possano cambiare le proprie dimensioni a seconda del dispositivo sul quale l'applicazione venga eseguita.

Appena il dispositivo caricherà la mappa da Google Maps, questa verrà visualizzata centrata in un punto predefinito.

Per poter determinare la posizione dell'utente è necessario che questo permetta l'utilizzo della localizzazione tramite una finestra di richiesta permessi, resa obbligatoria in Android per tutelare la privacy degli utenti.

Oltre questo controllo, in caso il ricevitore GPS del dispositivo risulti essere disattivato verrà mostrato un *toast* che chiederà all'utente di abilitarlo.



A seconda delle risposte dell'utente potranno verificarsi tre casi :

- **Utente accetta tutte e due le richieste -** in questo caso l'applicazione provvede a recuperare la posizione nel modo più preciso possibile, tramite il ricevitore GPS o tramite la localizzazione via gli indirizzi IP delle reti circostanti.

  In seguito non appena il sistema riesce a determinare delle coordinate valide per la posizione dell'utente viene effettuato uno zoom nella posizione dell'utente, visualizzando inoltre un segnaposto nella posizione approssimativa e

un'area che indica il possibile errore relativo alla determinazione della posizione.

Se l'utente dovesse trovarsi in una delle aree contenenti file,il pulsante "Mostra File Disponibili" si attiva e premendoci sopra l'utente visualizza una nuova pagina contenete una lista di file ai quali è possibile accedere.

- **Utente consente la localizzazione approssimativa -** cosi'facendo l'applicazione non riuscirà a funzionare in maniera adeguata in quanto non sarà possibile stabilire all'interno di che area si trova l'utente.

  La posizione approssimativa infatti serve a localizzare l'utente in una determinata area metropolitana o città, fornisce quindi una posizione molto approssimativa del dispositivo.

- **Utente rifiuta una delle due richieste -** in questo caso l'applicazione non mostrerà nessuna informazione relativa alla posizione dell'utente e non permetterà quindi l accesso a nessuno dei file disponibili.
  L'unica cosa visualizzabile dall'utente sarà la mappa e le zone evidenziate su di essa, all'interno delle quali,a condizione di accettare le richieste dell'applicazione, potrà accedere ai file disponibili.

  Le aree vengono visualizzate in modo che l'utente veda che ci sono dei file disponibili e sia motivato a permettere l'accesso alla sua posizione.

Durante la fase di progettazione, quindi sono state identificate le informazioni che l'utente finale dovrebbe visualizzare in un'applicazione del tipo LBS.
Sulla base di questo si è scelto di implementare le seguenti funzionalità :

1. Controllare se il ricevitore GPS del dispositivo è attivo.
2. Richiesta e controllo concessione permessi per accesso alla posizione del dispositivo.
3. Mostrare una mappa e visualizzare la posizione attuale dell'utente.
4. Disegnare sulla mappa le aree individuate per permettere l'accesso ai file.
5. Mostrare messaggi sulla posizione attuale riconosciuta dal sistema oltre a messaggi informativi all'utente.
6. Se la posizione è all'interno di un'area evidenziata attivare il pulsante che permette di vedere i file disponibili in una nuova schermata.

3.2 Design Thinking

Il **Design Thinking** è un processo non lineare e iterativo per la realizzazione di un progetto, al fine di comprendere gli utenti, discutere le opzioni, ridefinire i problemi e creare soluzioni innovative in modo da poterle prototipare e testare.

Ultimamente il concetto di **Design Thinking** è sempre più accostato al processo di innovazione di prodotti e servizi nel mondo dello sviluppo di prodotti digitali.

L'approccio del **Design Thinking** è stato codificato per la prima volta intorno agli anni 2000 dalla **Stanford University**, che ha basato il processo di sviluppo su capacità analitiche e creatività.

Il Design Thinking basa le sue innovazioni su quattro principi fondamentali :

- **Creatività -** fare leva sulle capacità creative delle persone coinvolte nel progetto, utilizzando metodologie come l'*How Might We*.
  - **How Might We :** il metodo in questione è un modo di porre le domande per rendere più efficace il *brainstorming*, con l'obbiettivo di provocare nei partecipanti idee significative e pertinenti al problema affrontato.

    Partendo dal obiettivo si separano i problemi che ci impediscono di raggiungerlo, una volta fatto si trasformano tutti i singoli problemi individuati in domande che iniziano appunto con *"come potremmo..."* facendo attenzione ad ampliare i concetti evidenziati nel punto precedente.

    In seguito è necessario portare all'estremo ognuno di questi problemi e fare assunzioni e analogie con una conseguente focalizzazione su di esse.
- **Prototipazione -** in modo da comprendere in maniera rapida punti di forza e debolezza delle soluzioni da implementare tramite il confezionamento di prototipi che in seguito potrebbero concretizzarsi in *roadmap* o in modelli funzionanti.
- **User Contribution -** è fondamentale all'interno del Design Thinking il ruolo dell'utente finale, per questo si fa largo uso di *ricerche etnografiche* e *A/B Testing*.
  - **Ricerche Etnografiche -** recarsi tra coloro che si vuole studiare per un periodo di tempo utilizzando delle tecniche di ricerca per collezionare dati che una volta raccolti e interpretati rendono possibile la comprensione del pubblico in esame.

    Nel caso di sviluppo di software questo avviene interpellando e chiedendo continue opinioni o punti di vista da parte di possibili utilizzatori del servizio in sviluppo.
  - **A/B Testing -** è un test di ipotesi a due campioni, utilizzato nel *web design* e in particolare nella *User Experience*.

    L'obiettivo è quello di identificare la scelta migliore da fare fra due possibili alternative in modo da ottenere il migliore responso possibile da parte degli utenti.
- **Durata del Processo -** i progetti hanno una durata variabile, dovuta alle possibili idee o rallentamenti consecutivi alla generazione di numerose idee durante i momenti di brainstorming.

3.2.1 Modelli di Design Thinking

Dopo uno studio di molti casi di utilizzo di metodologie basate sul *Design Thinking* l'**Osservatorio del Design Thinking del Politecnico di Milano** ha individuato quattro forme principali che questo metodo di approccio può assumere :

- **Creative Problem Solving -** metodologia di innovazione basata sulla comprensione dei bisogni dell'utente immaginando il maggior numero di soluzioni, per poi in seguito identificare la più efficace tra queste.
- **Sprint Execution -** basata sulla realizzazione di prototipi per poi ricevere valutazioni e suggerimenti da parte degli utenti finali sulle modifiche da effettuare.
- **Cretive Confidence -** creare all'interno delle organizzazioni le condizioni che permettano a gli sviluppatori di essere innovativi e inclini al cambiamento stimolando empatia e tolleranza al rischio di fallimento.
- **Innovation Of Meaning -** strategia tramite la quale si punta a apportare valore sia all'organizzazione che all'utente finale.

3.2.2 Le cinque fasi nel Design Thinking

L'**Hasso Plattner Institute of Design at Stanford** descrive il *Design Thinking* come un processo formato da cinque fasi : *Empatizzare*, *Definire*, *Ideare*, *Prototipare* e *Testare*.

Figura 3.1: fasi del Design Thinking

Nota : *queste fasi non sono sempre sequenziali e spesso sono eseguite da più team che le possono eseguire in contemporanea, in ordine sparso, o ripetendole in maniera iterativa.*

Empatizzare con il problema - fase in cui si fa ricerca su gli utenti, mettendo da parte le supposizioni dell'organizzazione e acquisendo la visione dell'utenza

sul problema in questione.

Definire i problemi e i bisogni degli utenti - si raccolgono le informazioni e le si analizza, in modo da sintetizzare i principali problemi che si andranno ad affrontare.

Ideare - ora si mettono in discussione varie possibilità in modo da ottenere delle idee concrete pensando fuori dagli schemi e cercando modi alternativi di vedere il problema.

Prototipare - l'obiettivo di questa fase è quello di identificale la soluzione migliore per ogni problema analizzato tramite la produzione di prototipi semplici e economici da realizzare utili per studiare le soluzioni selezionate.

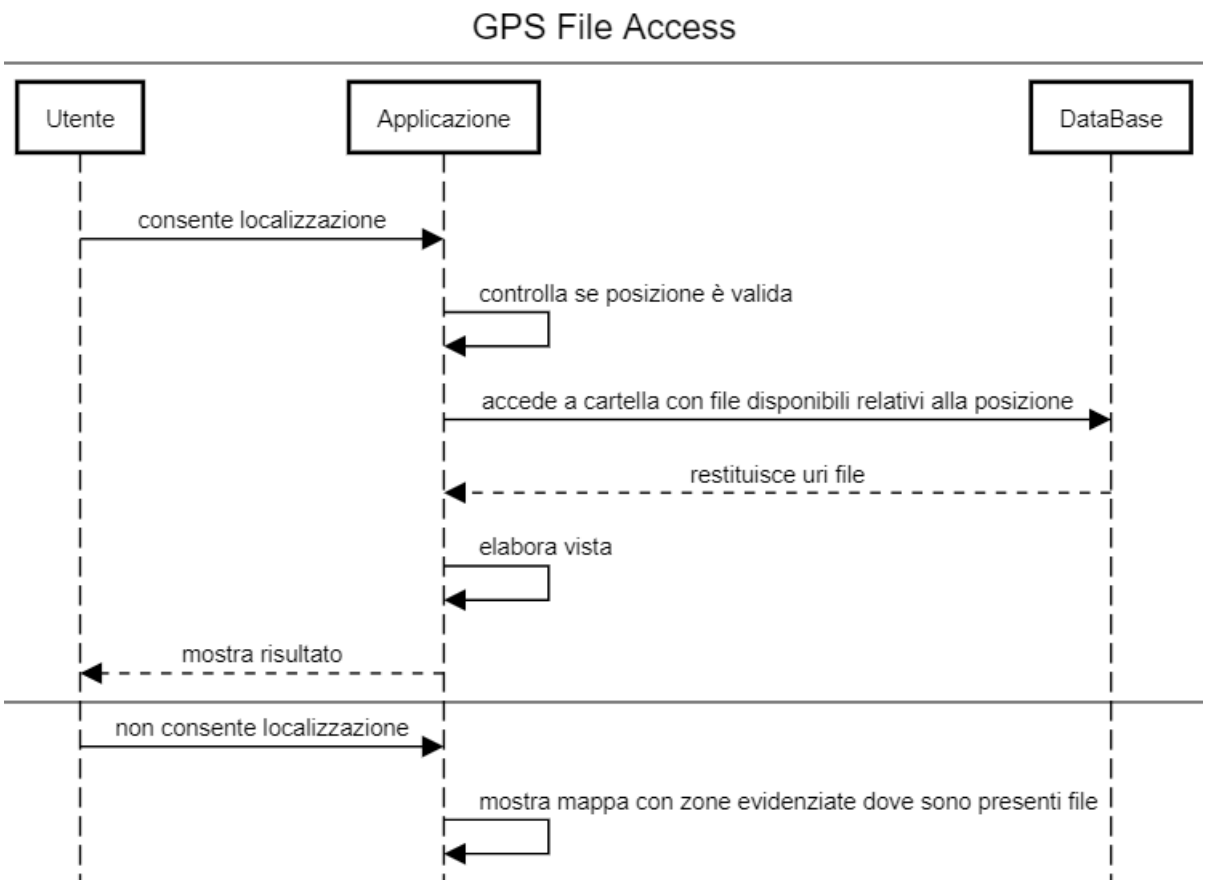
Testare - a questo punto vanno coinvolti dei *tester* che provino in maniera approfondita i prototipi realizzati. In base a i *feedback* di questi utenti selezionati sarà possibile decidere se la soluzione adottata è valida o meno.

3.3 Diagrammi UML

La fase di progettazione ha previsto la realizzazione di alcuni diagrammi UML che costituiscono un'astrazione di alto livello del sistema e non forniscono informazioni sugli strumenti adottati durante la fase implementativa.
I diagrammi proposti per descrivere il funzionamento del sistema sviluppato sono il diagramma di sequenza, il diagramma degli stati e il diagramma delle attività.

3.3.1 Sequence diagram

Un **diagramma di sequenza** è un diagramma UML *(Unified Modeling Language)* che illustra la sequenza dei messaggi tra oggetti in un'interazione.
Un diagramma di sequenza consiste in un gruppo di oggetti, rappresentati da *lifeline*, e nei messaggi che tali istanze si scambiano durante l'interazione.
Dal alto verso il basso è mostrata la sequenza temporale dei messaggi secondo l'ordine in cui sono scambiati.
Da sinistra verso destra invece è possibile distinguere le istanze tra le quali questi messaggi vengono scambiati.



3.3.2 State Chart Diagram

Lo **State Chart Diagram** o Diagramma degli Stati è un diagramma che, come previsto dall'**UML**, descrive il comportamento di entità o di classi in termini di stato.
Il diagramma mostra gli stati che sono assunti dall'entità in risposta ad eventi esterni.
Gli eventi rappresentabili sono :

- **Stato -** distinto in *iniziale*, *intermedio* e *finale*.
- **Evento -** azione che porta al cambiamento di stato.
- **Azione -** risposta a evento
- **Guardia -** condizione a fin che l'azione possa essere completata.

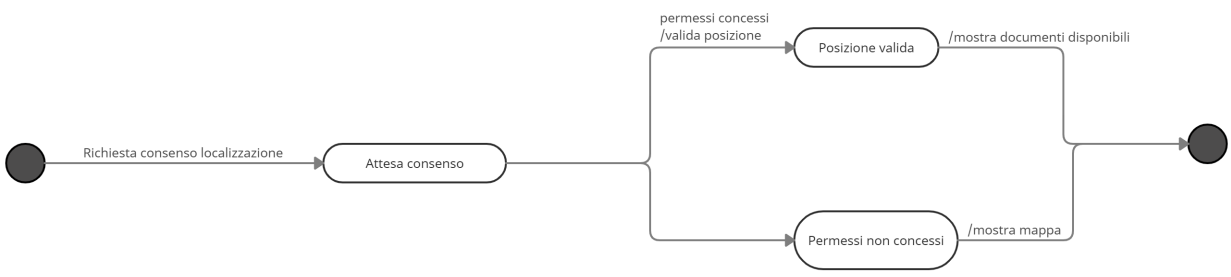


Figura 3.2: Posizione

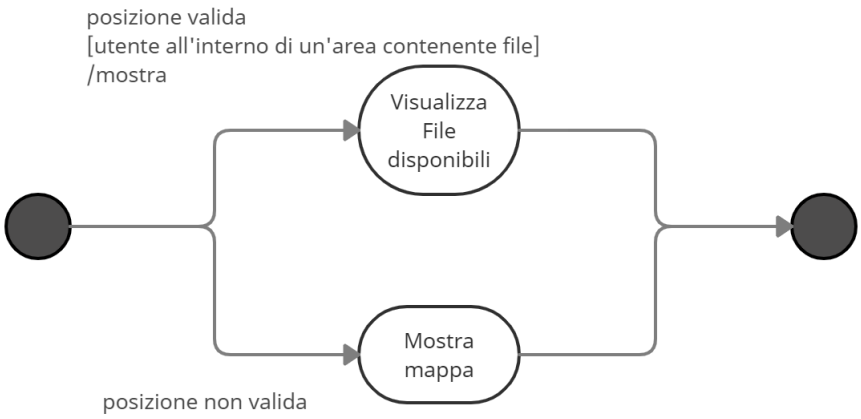


Figura 3.3: Documenti

3.3.3 Activity diagram

L'**Activity Diagram** permette di descrivere il processo attraverso grafi in cui i nodi rappresentano le attività e gli archi invece l'ordine con il quale sono eseguite.
Le attività sono rappresentate da un rettangolo smussato, accompagnate da una descrizione delle stesse, il flusso di esecuzione delle attività è rappresentato tramite delle frecce orientate.
Le attività possono essere eseguite in sequenza o in parallelo tramite *fork* che viene rappresentato tramite una freccia di flusso che si sdoppia, invece in caso di condizioni queste sono rappresentate da rombi dai quali partono flussi alternativi.
I punti in cui più flussi si ricongiungono sono chiamati *join* e sono realizzati tramite delle frecce che si ricongiungono.

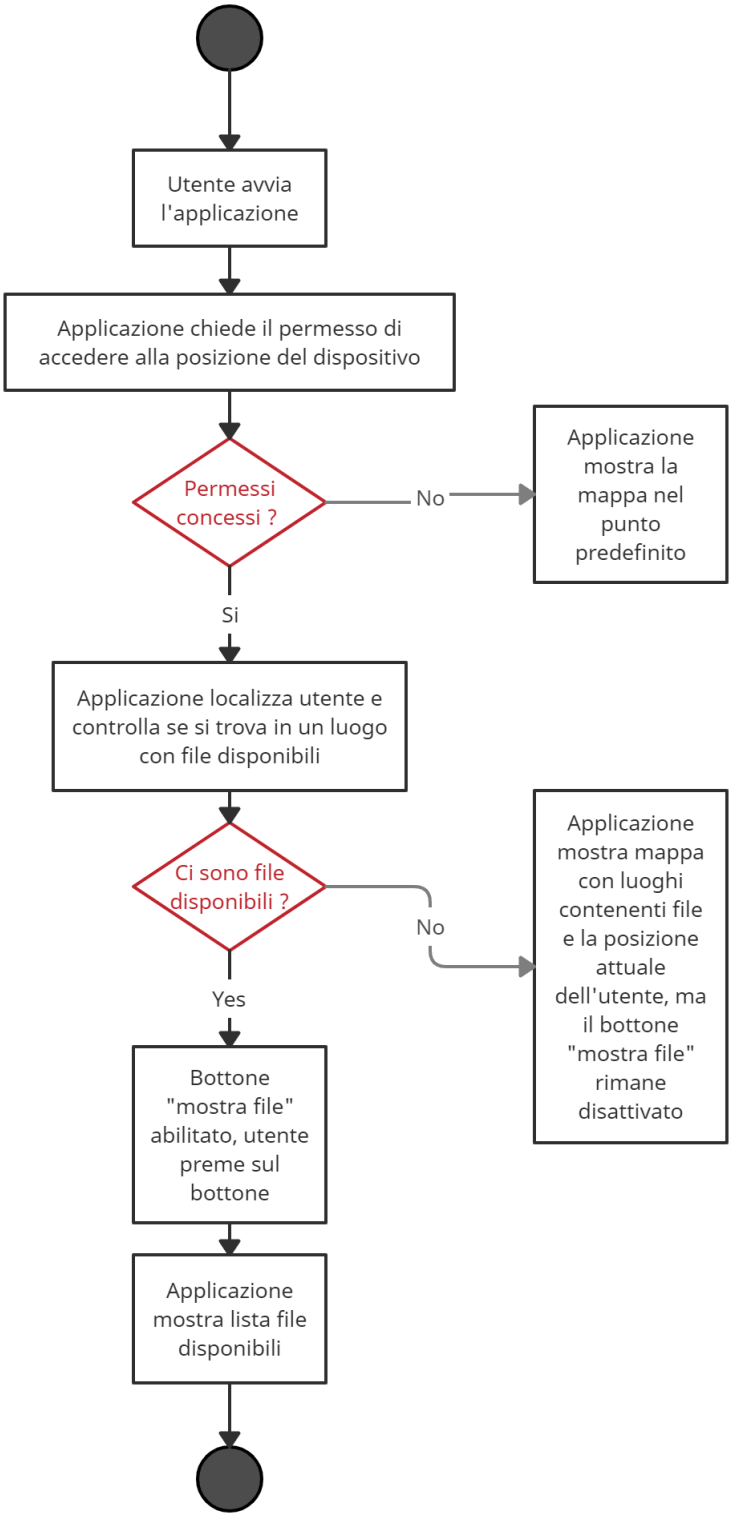


Figura 3.4: Activity Diagram

Capitolo 4

Implementazione

In questo capitolo vengono discusse le scelte implementative adottate per la realizzazione dell'applicazione, discutendo delle tecnologie utilizzate.

Vedremo in particolare come sono stati gestiti i permessi relativi alla posizione gestiti dal sistema Android e come sono stati gestiti i poligoni che vengono sovrapposti alla mappa nella schermata principale dell'applicazione.

Il codice sorgente è disponibile sotto licenza LGPL, al link :
https://github.com/Domendomen99/LocationBasedFileAccess

4.1 Interfaccia Utente

Nel pattern architetturale *Modello Vista Controllo* i componenti della *vista* sono quelli che forniscono una qualsiasi rappresentazione in *output* di informazioni, come una tabella o una lista.

I componenti della *vista* permettono di visualizzare i dati contenuti nel *modello* e di interpretare le interazioni dell'utente con il sistema.

Nel progetto l'**Interfaccia Utente** è rappresentata dalle possibili viste che l'applicazione può mostrare. Attualmente le viste utilizzate sono due :

- **Vista principale -** qui viene mostrata la mappa con evidenziate le zone dove sono presenti dei file accessibili al loro interno, oltre che i dati relativi alla posizione dell'utente e un pulsante per accedere alla lista di file disponibili in una zona.
- **Vista File Disponibili -** in questa vista invece viene visualizzata una lista di file, ognuno identificato dal nome e dalla sua estensione.

Per far si che l'interfaccia si ridimensioni automaticamente in caso l'applicazione venga eseguita su dispositivi di grandezza diversa, il layout delle viste è stato realizzato tramite l'utilizzo di **Relative Layout** in grado di cambiare la propria dimensione e

disposizione a seconda dello spazio disponibile a schermo.

Caratteristiche del Relative Layout

Il **Relative Layout** permette di disporre oggetti della vista in riferimento ad altri oggetti, possono quindi essere assegnati attributi che specificano rapporto che devono avere due elementi della vista tra di loro. Ad esempio :

- **layoutalignParentTop**, **layoutalignParentLeft**, **layoutalignLeft**, **layoutalignRight**, **layoutbelow** e **layoutCenterHorizontal** .

Elementi all'interno della Vista Principale

La vista principale è caratterizzata dalla presenza del riquadro dove è rappresentata la mappa con tutte le informazioni che contiene e un singolo bottone "Mostra File Disponibili".

E' stata inserita inoltre in cima a questi due elementi una *label* che riporta, nel caso in cui l'utente su trovi in una zona con file disponibili, il nome dell'area identificata dall'applicazione.
Così facendo l'utente può capire subito se l'applicazione ha ottenuto la sua posizione e se questa è valida al fine di accedere ai file destinati a quella specifica zona.

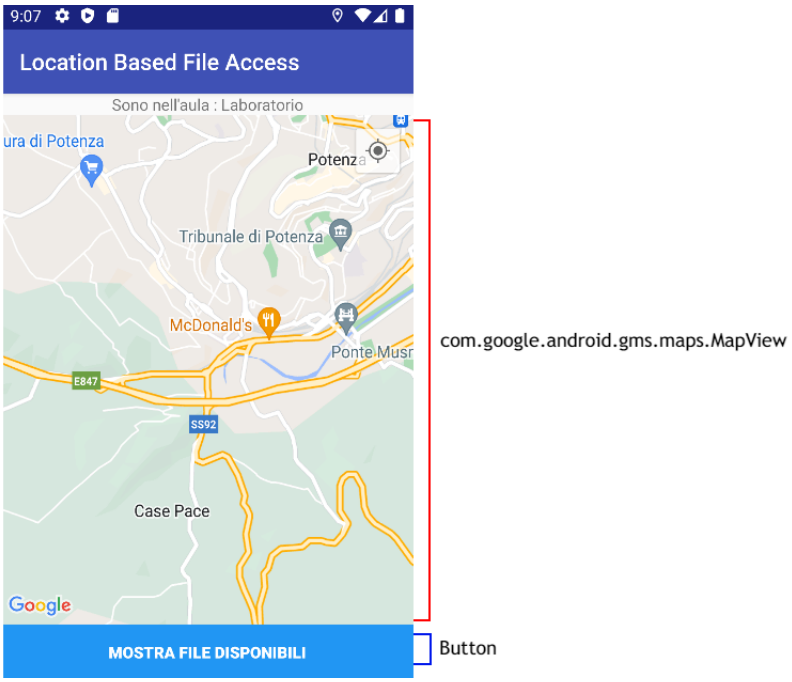


Figura 4.1: Vista Principale

Sono stati effettuati diversi prototipi riguardanti l'aspetto della vista principale, in maniera da individuare quella più familiare e semplice da utilizzare da proporre

agli utenti.

Alcuni di questi prototipi non mostravano la mappa, altri mostravano gia'nella vista principale un'area dedicata alla lista di file disponibili in una determinata zona. Si è in seguito deciso di optare per la versione proposta sulla base di 2 scelte ben precise :

1. **Mostrare all'utente la mappa -** in modo che questa indichi un chiaro funzionamento dell'applicazione e che quest'ultima mostri dove un'utente si deve recare per poter accedere ai file disponibili.

2. **Offrire un "family feeling" -** l'applicazione in oggetto infatti doveva affiancare quella di un collega, che ha sviluppato un'applicazione simile accessibile via browser, alla fiera **Maker Faire di Roma**. Il collega per la realizzazione della sua interfaccia ha infatti lasciato a tutto schermo la mappa e poi in caso ci si localizzasse in una posizione adeguata in una barra laterale vengono mostrati i file disponibili.

Elementi all'interno della Vista File Disponibili

La vista file disponibili è stata realizzata in maniera molto semplice tramite l'inserimento di una **ListView** che permette di scorrere i documenti accessibili da una determinata posizione e in seguito premendoci sopra permette di scaricarli o visualizzarli.

Una **ListView** è un'**AdapterView**, cioè un componente visuale che collegato ad un'**Adapter** raccoglie le **View** prodotte dall'Adapter e le mostra secondo la sua configurazione.

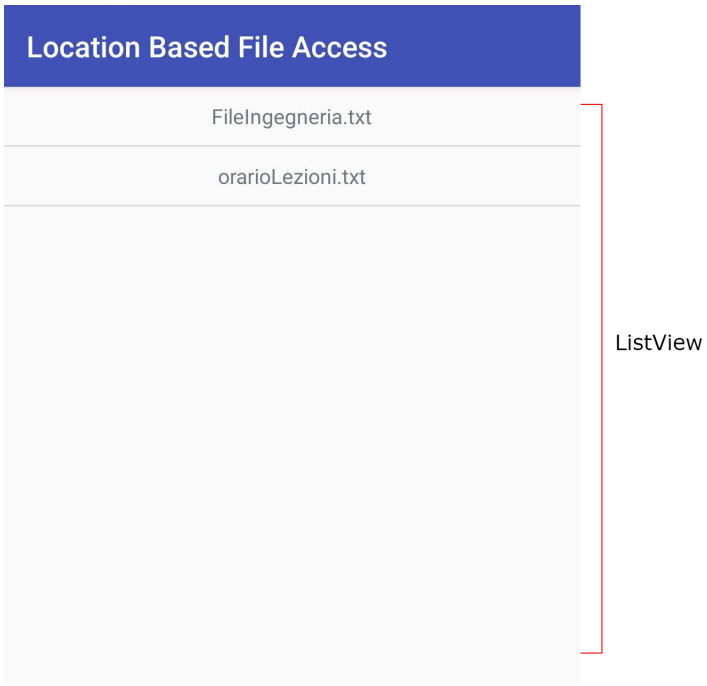


Figura 4.2: Vista File Disponibili

4.2 Google Map

L'inserimento della mappa è stato possibile utilizzando le *Maps SDK* [4] fornite da Google per l'implementazione in progetti basati su Android delle sue mappe.

Come primo passo per poter utilizzare le mappe di Google è stato necessario registrarsi sulla piattaforma *Google Maps Platform* e abilitare le API e l'SDK per il progetto indicato.
L'utilizzo di queste funzioni rimane gratuito finche' non si supera una soglia di richieste, dopo la quale verranno addebitati dei costi per l'utilizzo delle funzionalità offerte.

E'in seguito stato necessario ottenere una chiave per accedere alle API, questa serve come identificatore univoco per autenticare le richieste associate al progetto in sviluppo.
In seguito questa chiave viene inserita nelle *local properties* dell'applicazione e verrà letta ogni qual volta si avvia l'app.

Per sfruttare le funzionalità di Google Maps è inoltre richiesta la presenza dei **Play Services** sul dispositivo, questi sono dei software alla base della funzionalità del sistema di tutti i dispositivi Android Certificati.
Questi servizi forniscono : **Sicurezza e Affidabilità**, **API per sviluppatori**, **Servizi Principali per i Dispositivi**.

Aggiunta della mappa

L'SDK Maps offre diverse classi utilizzabili per gestire il ciclo di vita, le funzionalità e i dati di una mappa.

Le interfacce principali utilizzate sono le seguenti :

- **GoogleMap -** il punto di accesso per tutte le funzionalità e per i dati cartografici di base.
- **MapView -** una vista per la gestione del ciclo di vita e la visualizzazione di un oggetto **GoogleMap**.
- **OnMapReadyCallback -** interfaccia di callback che gestisce gli eventi e l'interazione dell'utente con l'istanza **GoogleMap**.

L'oggetto **GoogleMap** una volta istanziato grazie a una MapView esegue in maniera autonoma i seguenti task :

- Connessione al servizio Google Maps.
- Download e visualizzazione della grafica della mappa da visualizzare.
- Se abilitati mostra i controlli relativi allo zoom e al recupero della posizione attuale dell'utente.
- Rende la vista della mappa responsiva a tocchi e *pinch*.

All'interno di *OnCreateView* della Vista Principale oltre le classiche assegnazioni viene inizializzata la *mapView*. Dopodiché vengono chiamati i seguenti metodi di **MapView** :

- **onCreate (Bundle savedInstanceState) -** passando *savedInstanceState* permette alla view di ripristinare i suoi valori precedenti qualora questa fosse già stata inizializzata.
- **onResume () -** necessario per il ciclo di vita della MapView.
- **getMapAsync (OnMapReadyCallback callback) -** utilizzato nell'attesa che la mappa venga inizializzata completamente e che venga fornito un'istanza di GoogleMap tramite callback.

Per visualizzare e personalizzare la mappa il callback **OnMapReady** è stato sovrascritto, al suo interno vengono effettuate le seguenti operazioni riguardanti la visualizzazione della mappa :

Dopo aver inizializzato l'oggetto **GoogleMap** con l'istanza ottenuta da i parametri del callback vengono assegnate delle coordinate di partenza, in base alle quali la vista della mappa viene centrata e ingrandita.

Queste operazioni sono realizzate tramite le seguenti procedure della classe **GoogleMap** :

- **moveCamera(CameraUpdateFactory.newLatLng(<lat>,<long>) -** che permette di centrare la vista della mappa appunto in un determinato punto.
- **animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, <lvlZoom>) -** oltre a spostare la camera permette anche l'ingrandimento della mappa per avvicinarsi al luogo scelto.
- **setMyLocationEnabled(true) -** abilita il layer che permette di visualizzare e lrichiedere la posizione dell'utente in tempo reale.

4.3 Permessi Localizzazione

Per permettere il trattamento della posizione attuale dell'utente da parte delle applicazioni il sistema Android obbliga le singole applicazioni a chiedere all'utente di concedere i permessi per la localizzazione approssimativa e precisa del dispositivo.

La richiesta di questi permessi va‘ effettuata in due passaggi :

1. **Dichiarazione dei permessi necessari all'interno del Manifest -** ogni permesso necessario viene dichiarato tramite il tag *<uses-permission>* da collocare all'interno del nodo *<application>*.
   In questo modo il sistema conosce quali attività l'applicazione andrà a svolgere e potrà in seguito concedere il permesso per eseguirle.
   I permessi richiesti sono di tipo *dangerous* cioè quelli potenzialmente più lesivi per la riservatezza dell'utente e sono i seguenti :

- `android.permission.ACCESS_FINE_LOCATION`
- `android.permission.ACCESS_COARSE_LOCATION`

2. **Richiesta esplicita per la concessione dei singoli permessi da parte dell'utente -** i permessi del tipo *dagerous* devono poi essere accettati a runtime dall'utente in modo che sia pienamente consapevole dell'utilizzo di queste funzioni da parte dell'applicazione.

Richiesta e controllo permessi a runtime

La gestione e il controllo dei permessi a runtime viene fatta grazie alla classe **ContextCompat**, inclusa nella libreria di supporto.

ContextCompat permette di **controllare** se alcuni permessi sono stati accettati tramite la funzione :

```
ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION

ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION
```

Questa funzione restituisce un valore intero che indica se il permesso è stato già concesso, in questo caso il risultato sarà 0, o se il permesso risulta negato.

Invece per **richiedere** all'utente di concedere un permesso *dangerous* è necessario lanciare un'apposito *intent* tramite il seguente codice :

```
ActivityCompat.requestPermissions(this , new String [] Manifest.permission.
ACCESS_FINE_LOCATION , ID_RICHIESTA_PERMISSION);
```

In questo modo viene passato un'array contenente i permessi che dovranno essere accettate e un numero costante che identifica la richiesta permessi in particolare. Quando il codice viene eseguito si apre una finestra di dialogo in sovrimpressione che richiede di concedere o negare i permessi richiesti.

La **gestione delle risposte** a queste richieste di concessione dei permessi viene fatta tramite il metodo :

```
onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults)
```

Il metodo riceve il risultato della richiesta di permessi formato da tre elementi :

- **Array permessi -** elenco di tutti i permessi richiesti.
- **Array risultati -** elenco dei risultati di tutte le richieste effettuate.
- **Codice della richiesta -** identificata dal numero assegnato in fase di richiesta.

E'quindi possibile distinguere le varie richieste di concessione permessi e decidere cosa fare in caso vengano accettati o rifiutati.

4.4 Determinare la posizione dell'utente

Una volta ottenuti i permessi l'applicazione identifica la posizione dell'utente in due modi diversi :

- **Utilizzando le reti che circondano l'utente -** poco accurato, localizza l'utente tramite la rilevazione delle reti Wi-Fi e GSM intorno al dispositivo.
- **Utilizzando il sistema GPS**, molto preciso.

La posizione dell'utente è memorizzata in oggetti del tipo **LatLng** che mantengono al loro interno una proprietà per la latitudine e una per la longitudine.

Per ottenere le coordinate dell'utente si utilizzano le classi **Location Manager Location Listner**.

- **Location Manager -** provvede a fornire l'accesso ai servizi di localizzazione del dispositivo, in modo da ottenere aggiornamenti periodici della posizione grazie a un Location Listner.
- **Location Listner -** listner che provvede a notificare la posizione dell'utente in caso questa cambi o passi un determinato lasso di tempo dall'ultimo aggiornamento della posizione corrente.

Il metodo *onLocationChanged* di Location Listner è stato sovrascritto in modo che ogni volta che viene ottenuta una nuova posizione la mappa venga centrata nelle coordinate appena attenute. Inoltre ogni volta che viene identificata una nuova posizione viene eseguito il codice che permette al sistema di stabilire se l'utente si trova o meno all'interno di una delle aree che contengono file, il codice in questione lo esamineremo in seguito.

Subito dopo dichiarato il Listner questo viene registrato presso il Location Manager all'interno della vista principale, tramite il codice :

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
MIN_TIME, MIN_DST, locationListener);
```

Qui nel metodo *.requestLocationUpdates* vengono passati quattro parametri : il **primo** è il metodo che deve utilizzare il dispositivo per ottenere una nuova posizione dell'utente, il **secondo** e il **terzo** sono rispettivamente il tempo che deve passare tra una rilevazione e un'altra e lo spostamento che ci deve essere stato affinchè lo spostamento venga considerato, il **quarto** argomento invece è proprio il Location Listner.
I valori di MIN TIME e MIN DIST sono costanti e i loro valori sono rispettivamente 1 minuto per il tempo e 5 metri per il minimo spostamento.

Le richieste del Locaiton Manager per ottenere una nuova posizione vengono interrotte tramite il metodo `.removeUpdates ( locationListener)` chiamato quando ci si sposta in una nuova activity.
Le richieste di una nuova posizione vengono poi ripristinate, tramite il codice visto in precedenza, all'interno del metodo `avviaRicercaPosizione()` chiamato nell' *OnResume* della vista principale a condizione che il Location Manager sia stato già inizializzato.

4.5 Utilizzo di più tipi di localizzazione

Come visto nel codice presentato, il sistema richiede la posizione dell'utente tramite il GPS, quando questa però non disponibile è in grado di identificare una posizione approssimativa dell'utente tramite le reti circostanti senza richiedere esplicitamente l'utilizzo di questa tecnologia tramite codice.

Ciò è dovuto al fatto che una volta concessi i permessi per l'accesso alla posizione precisa, i permessi per l'accesso alla posizione approssimativa vengono automaticamente accettati perchè considerati inferiori rispetto i primi.

L'applicazione comunque quando viene per la prima volta richiesto di accettare i permessi per l'accesso alla posizione esegue la funzione *isGPSEnabled()* che controlla se la localizzazione del dispositivo sia stata attivata dall'utente.
Nel caso in cui la localizzazione non sia stata attivata la funzione mostra un *Toast* che chiede all'utente di attivare la localizzazione.

4.6 Disegnare sulla mappa i poligoni letti tramite file.JSON

Ottenuta la mappa di Google è stato necessario riuscire a disegnarci da sopra dei poligoni che rappresentino le zone contenenti dei file a cui l'utente può accedere.

Il file.JSON

All'interno della cartella *Row* del progetto è presente un file .JSON che contiene al suo interno una lista di punti che individuano i vertici dei poligoni e il nome dei luoghi che dovranno poi essere rappresentati sulla mappa.

Per assegnare i dati contenuti all'interno del file ad un'oggetto è stata definita la classe *Poligono*, caratterizzata da due proprietà :

1. Nome dell'area individuata dal poligono.

2. Lista di oggetti del tipo *LatLng*

Gli oggetti *Poligono* vengono inizializzati in cima al metodo *onMapReady* tramite la funzione *caricaPoligoniDaFile*.

La funzione caricaPoligoniDaFile()

La funzione caricaPoligoniDaFile() esegue due operazioni :

- Esegue la funzione **leggiDatiDaFileJSON()** che legge il file e JSON restituisce una stringa con i dati contenuti nel file.
- Restituisce al chiamante il risultato della funzione *convertiJsonInOggetti()* che tramite il metodo *fromJSON()* di *Gson* crea e inizializza gli oggetti Poligono presenti nella stringa e restituisce la lista.

  ```
  List<Poligono> listaPoligoni = new Gson().fromJson(testoFile, new
  TypeToken<List<Poligono»().getType());
  ```

Disegnare i poligoni sulla mappa

La lista dei poligoni ottenuti viene poi passata alla funzione *disegnaMappa(listaPoligoni)* che si occuperà di riportare sulla mappa le aree rappresentate dai poligoni.
Le aree vengono riportate sulla mappa grazie alla funzione *GoogleMap.addPolygon* con il seguente codice :

```
gMap.addPolygon(new PolygonOptions().addAll(poligono.getListaPunti())
.geodesic(true).strokeColor(Color.BLACK).strokeWidth(20));
```

La funzione *addPolygon* viene chiamata per ogni poligono contenuto all'interno della lista poligoni, la funzione assegna tutti i punti di un'oggetto Poligono a un nuovo oggetto del tipo *PolygonOptions*.
Vengono poi modificate le proprietà del nuovo oggetto impostando il colore nero per i bordi e lo spessore degli stessi.

4.7 Determinare se l'utente si trova in una area con file disponibili

A decidere se l'utente si trovi o meno all'interno di una delle aree contenenti file accessibili è la funzione *doveMiTrovo(location,listapoligoni)* chiamata all'interno del metodo *onLocationChanged()* che viene eseguita ogni qual volta viene richiesta una nuova posizione dell'utente.

La funzione riceve come primo argomento l'oggetto *location*, parametro del metodo *onLocationChanged* che contiene al suo interno le proprietà latitudine e longitudine attuali dell'utente, e come secondo argomento la lista dei poligoni che individuano le aree.

All'interno del metodo *doveMiTrovo* per ogni poligono presente nella lista viene verificata la seguente condizione :

```
if (PolyUtil.containsLocation(new LatLng(location.getLatitude(),
location.getLongitude()), poligono.getListaPunti(), false))
```

La funzione *.containsLocation()* ricevendo un punto indicato ,tramite latitudine e longitudine, e una serie di punti, in questo caso i vertici del poligono, è in grado di determinare se il punto sia all'interno o meno dell'area del poligono passato come argomento.

A questo punto ci sono due possibili scenari :

1. **L'utente si trova in una delle aree con file disponibili**
   In questo caso la label nella vista principale viene aggiornata con la scritta

   "ti trovi nell'aula : <nomeAula>"

   L'aula in cui viene individuato l'utente viene salvata come una costante e il bottone per accedere ai file disponibili diventa attivo cambiando colore.

2. **L'utente non si trova in nessuna area**
   In questo caso la label nella vista principale viene aggiornata con la scritta :

   "sei fuori dall'"università"

   La costante utilizzata per memorizzare il nome dell'aula viene salvata come *null*.

4.8 Mostrare i file disponibili

A condizione che l'utente si trovi in una delle aree selezionate che presentano file al loro interno, sarà possibile accedere alla vista "VistaFileDisponibili" tramite la pressione del pulsante "Mostra File Disponibili" posizionato subito sotto la vista della mappa nella Vista Principale.

Questa vista è formata da una listView che mantiene al suo interno una lista di stringhe che saranno i nomi dei file visualizzabili all'interno di un'area.

Alla fine del metodo *onCreateView* è stato chiamato il metodo "visualizzaDati(aulaAttuale)" che riceve come argomento l'aula in cui si trova l'utente.
All'interno del metodo vengono implementati utilizzando le funzioni di **Firebase**, approfondite in seguito, che permettono di ottenere i nomi dei file contenuti in una determinata cartella del *Cloud Storage* e i loro rispettivi uri.

Quando questi dati vengono recuperati la lista viene aggiornata e mostra i nomi dei file che sono stati trovati.

4.9 Utente seleziona un file dalla lista

Una volta che i dati sono stati caricati nella lista l'utente può selezionare uno dei file e scaricarli o aprirli, l'implementazione di questa funzione è stata eseguita come segue :

Nella classe *ControlloVistaFileDisponibili* è stata implementata l'azione "SelezionaFile" grazie al listner "OnItemClickListner" di AdapterView. L'azione è collegata alla lista mostrata nella vista.

Quando l'utente seleziona un'elemento della lista viene eseguita l'azione "SelezionaFile", al suo interno vengono recuperate le lista dei nomi di file e di tutti gli uri dei file.

In seguito in un ciclo vengono confrontati tutti gli uri con il nome del file che è stato selezionato. Il ciclo controlla se all'interno dell'uri in esame sia contenuto il nome del file.

Quando l'uri desiderato viene individuato, viene creato un'**Intent** del tipo *ACTION VIEW* e tramite il seguente codice viene effettuato un parsing dell'uri, che era un oggetto di tipo String, trasformandolo in un oggetto di tipo URI.

```
intent.setData(Uri.parse(uriFileSelezionato));
```

Avvenuto il parsing l'uri viene assegnato all'intent e in seguito lanciato.

Adesso si aprirà una finestra di dialogo del sistema che permetterà all'utente di selezionare in che modo desidera aprire il file selezionato.

4.10 Implementazione Firebase

L'utilizzo dei servizi offerti dalla piattaforma Firebase di Google è molto semplice e immediato per lo sviluppo di applicativi Android tramite Android Studio.

Per prima cosa è stato necessario collegare il servizio all'applicazione tramite dei passaggi guidati molto semplici e immediati.

Fatto ciò, tramite browser, è stato necessario selezionare il servizio di Firebase che si voleva utilizzare. Per memorizzare e in seguito richiamare dei file caricati sul server si è scelto il servizio **Cloud Storage**.

Lato Server

Per caricare i file sul server Firebase è stato necessario eseguire un upload dalla pagina web del servizio, facendo attenzione e suddividere i file in cartelle con il nome

dell'area all'interno della quale si trovano.

Lato Client

Tramite Android Studio è stato necessario selezionare il servizio di Firebase che si voleva utilizzare e in automatico sono state aggiunte le dipendenze all'interno del file *build.gradle* in modo da potersi servire degli strumenti messi a disposizione.

L'interazione con il server Firebase avviene tramite quattro principali oggetti :

- **FirebaseStorage -** contiene tutti i dati relativi alla connessione con il server cloud.
  Chiamando i metodi *.getIstance().getReference* è possibile ottenere il percorso del root accessibile tramite l'applicazione.
- **StorageReference -** permette di spostarsi partendo dalla cartella root in altre sottocartelle o singoli file.
  Memorizza al suo interno il percorso per raggiungere la risorsa selezionata.
- **ListResult -** Oggetto che si ottiene quando si fà una richiesta al server.
  Al suo interno contiene la lista dei nomi le chiavi e le posizione per ogni file contenuto nella cartella.
- **Uri -** identifica in maniera univoca il percorso per accedere a un file, in questo caso memorizzato su un server cloud.
  Permette di ottenere il file a cui si riferisce collegandosi all'indirizzo contenuto nell'uri.

Ottenere i nomi dei file

Nel metodo *visualizzaDati()* in VistaDatiDisponibili all'interno di un blocco try-catch viene eseguita la seguente richiesta :

```
riferimentoCartella.listAll()
```

Con `riferimentoCartella` che è stato inizializzato come il riferimento alla cartella che porta il nome dell'aula attuale.

Per ottenere i risultati di questa richiesta vengono utilizzati i seguenti listner :

- **OnSuccessListner -** permette di ottenere l'oggetto *ListResult* contenente i dati di tutti i file trovati all'interno della cartella selezionata.

  E'possibile accedere alla lista di `StorageReference`, contenente il riferimento di ogni file all'interno della cartella selezionata, chiamando la funzione `.getItems()` su `listResult`.

  Effettuando una scansione su tutti gli oggetti StorageReference contenuti all'interno di listResult è possibile ottenere il nome di ogni singolo file chiamando

la funzione `.getName` sull'oggetto StorageReference attualmente scansionato.

Questi nomi vengono poi aggiunti a una lista di stringhe chiamata `listaNomiFile`.

- **OnCompleteListner -** questo listner notifica la fine dell'operazione precedente.
  Qui la lista di nomi ottenuta viene salvata in una costante e passata, insieme alla lista di uri che viene popolata contemporaneamente, all'adapter che visualizzerà la lista.

Ottenere gli URI dei file

Contemporaneamente al popolamento della lista di nomi dei file, all'interno della scansione degli oggetti di tipo StorageReference, viene chiamata per ognuno di questi la funzione `.getDownloadUrl()`.

E'possibile ottenere i risultati di questa operazione tramite il *listner* `onSuccess` grazie al quale si può accedere all'oggetto di tipo URI che conterrà l'uri del file che sta venendo scansionato.

Viene eseguito un parsing da URI a STRING e le stringhe vengono memorizzate in una lista chiamata *listaUriFile*

Capitolo 5

Casi d'Uso

5.1 Utilizzo in UniBas

Il caso d'uso ideato per un'applicazione del genere è stato dell'utilizzo da parte degli studenti dell'UniBas.

L'applicazione può infatti essere utilizzata per permettere agli utenti del campus universitario di accedere a documenti riguardanti i luoghi che frequentano durante la giornata, che appunto sono le varie aree dell'università identificate tramite i poligoni inseriti nel sistema.

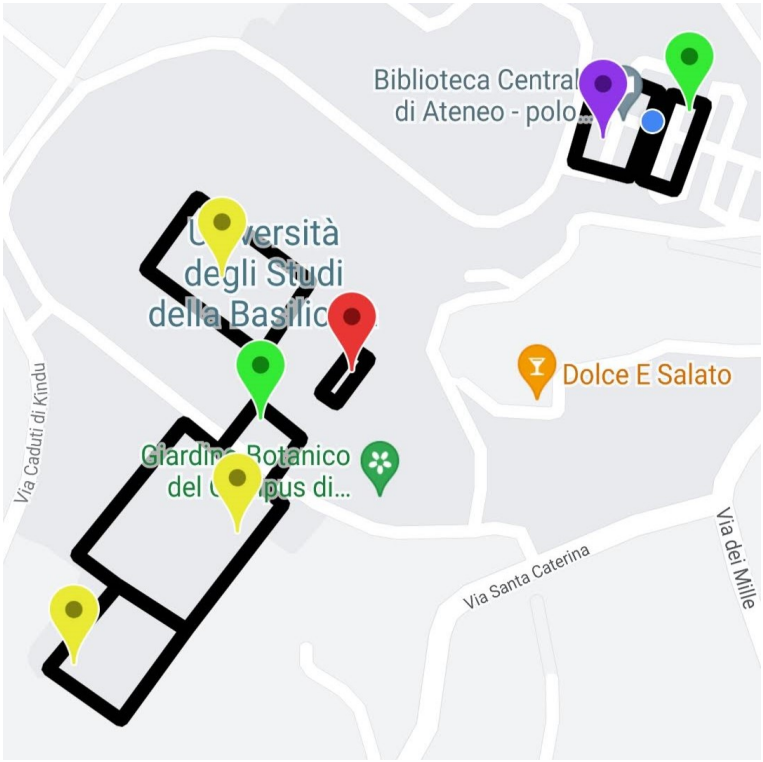
Le varie aree individuate sono state divise in quattro categorie principali :

- **Aree di Studio -** fanno parte di questa categoria l'area del DiMIE, le aule del dipartimento della S.I e le aule del dipartimento SAFE.
  All'interno di queste aree è possibile trovare documenti come gli orari delle lezioni per i vari corsi.

- **Aree per consumare cibi o bevande -** fanno parte di questa categoria l'area del bar e quella della mensa.
  Localizzandosi in queste zone è possibile accedere a menu, prezzi e orari delle attività.

- **Biblioteca -** utilizzando l'applicazione nei pressi della biblioteca è possibile accedere a documenti come la lista dei libri disponibili e gli orari di apertura.

- **Segreteria Studenti UniBas -** localizzandosi nella zona della segreteria è possibile ottenere documenti o visionare gli orari di apertura al pubblico.



5.2 Maker Faire Roma

L'applicazione è rientrata all'interno dei progetti portati all'evento **Maker Faire di Roma** al fine di essere utilizzata in maniera simile ad un applicazione di **geocaching**.

Il caso d'uso studiato è quello in cui viene chiesto ai visitatori della fiera di andare su una pagina internet e, a seconda della loro preferenza, di scaricare un'applicazione o continuare ad utilizzare il servizio web.

Ai visitatori viene chiesto di raggiungere un'area, dalla quale tramite l'applicazione o il sito internet potranno scaricare un file da mostrare tornando allo stand in modo da ottenere dei gadget.

Capitolo 6

Risultati Sperimentali

In questo capitolo viene mostrato il **Set-Up Sperimentale** oltre a discutere i risultati sperimentali relativi all'accuratezza della localizzazione dell'utente e al comportamento dell'applicazione quando la precisione della localizzazione non è molto alta.

I seguenti dati sono stati raccolti dopo essersi accertati del corretto funzionamento del sistema e della stabilità dello stesso.

Per verificare ciò sono stati svolti test sull'applicazione funzionante in modo da evidenziare lacune o malfunzionamenti nel più veloce dei modi.

6.1 Set up sperimentale

Verificata la stabilità del sistema, l'*APK* dell'applicazione è stata condivisa tramite un *link* a Google Drive, in modo da permettere facilmente l'istallazione su i dispositivi dei miei colleghi di università, intervenuti in maniera da potermi permettere di raccogliere velocemente molti feedback.

6.2 Obiettivo dei test

Per poter verificare il funzionamento dell'applicazione in un contesto realistico e poter raccogliere dati relativi alla localizzazione più numerosi possibili, una volta verificata la stabilità dell'applicazione si è proceduto ad effettuare una fase di testing sul campo.

L'obiettivo dei test è stato quello di verificare che in qualsiasi luogo, tra quelli selezionati per l'accesso ai file e non, i dispositivi fossero in grado di determinare la posizione dell'utente e gestire correttamente l'accesso ai file.

6.3 Consenso alla localizzazione concesso

Come detto in precedenza, il consenso dell'utente per accedere ai suoi dati sensibili è obbligatorio, per questo all'avvio dell'applicazione viene esplicitamente richiesto

l'accesso alla posizione.

Altro fattore determinante per permettere all'applicazione di localizzare l'utente è l'accensione della funzione di localizzazione del dispositivo, anche in questo caso se dovesse risultare spenta ne verrà esplicitamente richiesta l'accensione.

6.4 Consenso alla localizzazione negato

In questo scenario l'applicazione non è minimamente in grado di determinare la posizione dell'utente, in quanto in Android è fondamentale la concessione per permessi per poter effettuare operazioni su dati sensibili dell'utente.

Non è stato quindi possibile raccogliere nessun tipo di dato utile.

6.4.1 Accuratezza del rilevamento

Avendo concesso i permessi per accedere alla posizione più precisa possibile, il sistema Android automaticamente unirà i risultati ottenuti grazie alla tecnologia GPS a quelli basati sulla rilevazione di reti circostanti.

Non è possibile sapere precisamente in che modo la localizzazione è stata effettuata, ma lo si può dedurre osservando l'area mostrata intorno al segnaposto che contrassegna la posizione dell'utente. L'area infatti corrisponde al possibile errore che c'è stato in fase di localizzazione.

Quando quest'area sarà molto estesa, nell'ordine da ricoprire interi dipartimenti dell'università, è possibile dedurre che in quell'occasione sarà stata utilizzata una posizione approssimativa ottenuta unicamente tramite le reti circostanti.

6.5 Dati raccolti

Verranno quindi solo considerati i test eseguiti una volta aver concesso i permessi di localizzazione.

Non tutte le zone dove sono stati effettuati presentano caratteristiche morfologiche uguali, alcuni test infatti sono stati eseguiti in zone interrate a altri in su piani più elevati al fine di avere un'idea più chiara sul funzionamento del sistema.

Di seguito alcuni dei rilevamenti :

- **DiMIE -** all'interno del dipartimento DiMIE sono state effettuate due misurazioni :
  - Presso il **CISIT**, quindi un piano interrato, l'ottenimento della posizione è stato molto rapido e preciso. La posizione interrata del luogo non ha quindi comportato molti problemi per la localizzazione dell'utente,

quest'ultima che se con qualche imprecisione risulta sufficiente per il funzionamento dell'applicazione.
Il sistema rileva la posizione ottenuta come facente parte dell'area del DIMIE e concede l'accesso ai file disponibili.

– Anche presso il **terzo piano dell' edificio 3D** la determinazione della posizione dell'utente è stata molto precisa e si è quindi riusciti ad ottenere l'accesso ai file del DIMIE.

- **Segreteria UniBas -** recandosi sul posto l'applicazione rileva la posizione con qualche errore, probabilmente dovuto alla posizione degli uffici della segreteria.
Provando a spostarsi si è riusciti comunque a farsi localizzare all'interno dell'area della segreteria e ad accedere ai file disponibili.

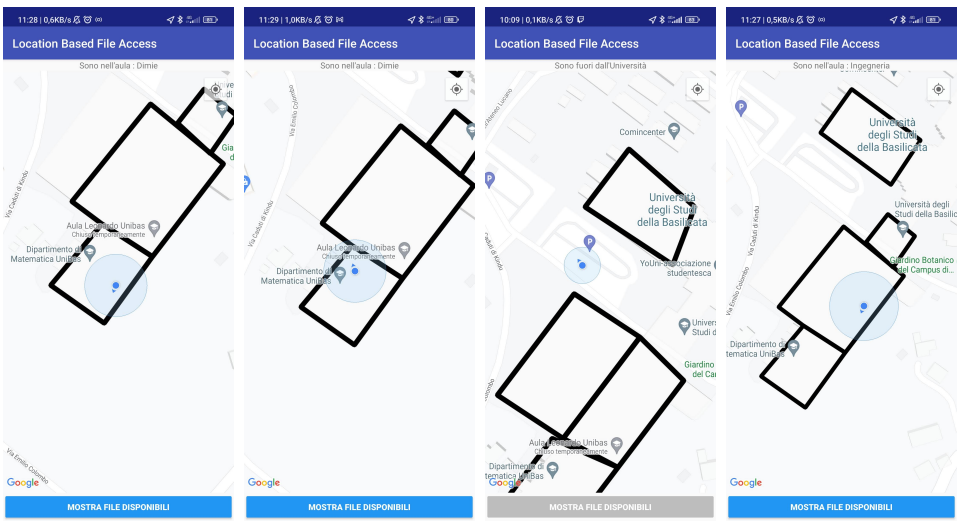
- **Ufficio ARDSU -** la posizione ottenuta localizzandosi nell'area del dipartimento SAFE non risulta molto accurata.
Il sistema comunque riesce a capire che ci si trova all'interno di un'area con file disponibili e permette l'accesso a questi ultimi.

- **Biblioteca -** anche in questa zona nessun problema riguardante la localizzazione e l'accesso ai file disponibili.

- **Mensa -** tralasciando qualche imprecisione dovuta alla vicinanza delle aree *Mensa* e *Biblioteca* anche qui ci si è riusciti a localizzare efficacemente.
Non semplicissimo è stato invece il riuscire a ottenere l'accesso ai file disponibili, ci si è dovuti infatti allontanare un po dall'area biblioteca per permettere al sistema di identificare con precisione in che zona ci si trovava.

- **S.I. -** posizionandosi sotto i porticati del dipartimento di ingegneria è stato possibile localizzarsi precisamente e accedere ai file disponibili per la zona.

- **Parcheggio interno UniBas -** localizzandosi nel parcheggio interno dell'UniBas, come da previsioni, la posizione dell'utente è stata identificata velocemente, e data l'assenza di file disponibili non è stato concesso l'accesso a nessun file.



6.6 Tabella Rilevamenti

Nella seguente tabella sono riportati in maniera schematica i rilevamenti effettuati e i relativi risultati ottenuti.

I vari rilevamenti sono distinti per precisione della localizzazione e corretto funzionamento dell'accesso ai file.
Quando nelle caselle riguardanti l'accuratezza è presente la croce vuol dire che sono stati necessari più tentativi al fine di farsi localizzare correttamente dal sistema.

Luogo	Accuratezza	Logica di acesso file funzionante
CISIT	✓	✓
3°Piano 3D	✓	✓
Segreteria UniBas	✗	✓
Segreteria Ardsu	✓	✓
S.I.	✓	✓
Biblioteca	✓	✓
Mensa	✗	✓
Parcheggio Interno	✓	✓

Capitolo 7

Conclusioni

In questo capitolo si tirano le somme sul lavoro svolto, evidenziando problematiche affrontate e possibili sviluppi futuri.

Con questo progetto ci si è impegnati a mostrare come sfruttare le possibilità offerte dalle moderne tecnologie di localizzazione che troviamo all'interno degli smartphone, in modo da costruire un servizio basato sulla localizzazione dell'utente.

Nel caso d'uso in esame si è reso possibile l'accesso ad alcuni documenti solo se l'utente si trova in una determinata zona, grazie all'utilizzo di diverse metodologie di localizzazione e di varie librerie e procedure messe a disposizione dal sistema Android a da altri SDK di Google.

Il sistema Android permette l'accesso alla posizione dell'utente in maniera molto efficace e permette di avere aggiornamenti in tempo reale su i suoi spostamenti.

Le mappe invece sono state implementate tramite l'utilizzo dell'SDK di Google Maps, in questo modo è stato possibile accedere a funzioni utilissime per lo sviluppo dell'applicazione.

Per il lato server si è invece optato per l'implementazione della piattaforma Firebase, strumento molto potente che semplifica l'implementazione di funzioni *back-end*.

Tutto ciò è stato sviluppato all'interno dell'IDE Android Studio, grazie al quale operazioni di compilazione, testing, emulazione e implementazione di funzionalità sono state molto semplici e veloci.

Grazie al **Sistema Android** e alla sua dinamica di gestione dei permessi, questi, una volta accettati dall'utente, permettono all'applicazione di localizzare il dispositivo nella maniera che viene ritenuta più efficace e precisa, a seconda delle condizioni dell'ambiente in cui si trova il dispositivo.

I test hanno mostrato che nella maggioranza dei casi la localizzazione avviene in maniera precisa e con margini di errore molto bassi, non ci si è mai imbattuti in posizioni completamente sbagliate o in situazioni che causavano lo scorretto funzionamento della logica dell'applicazione.

Tutto cio'è possibile grazie alla localizzazione tramite GPS, offerta dalla stragrande dei dispositivi mobili, che grazie ad una rete di satelliti dedicati riesce a fornire una posizione precisa in maniera indipendente dall'infrastruttura di rete, che si sarebbe costretti ad utilizzare in assenza di questa tecnologia.
L'utilizzo delle infrastrutture di rete rimane però uno degli strumenti più affidabili sotto l'aspetto della continuità di funzionamento. Queste ultime, anche se non permettono una localizzazione molto precisa vengono comunque utilizzate dall'applicazione nel caso in cui il segnale GPS non dovesse essere abbastanza potente.

7.1 Sviluppi futuri

Posizioni fittizie

L'applicazione in questo momento non è sicuramente in grado di distinguere una posizione reale da una fittizia fornita tramite un provider secondario, un possibile sviluppo futuro sarebbe quello di implementare questa funzione in maniera approfondita.

Esistono molte applicazioni che permettono all'utente di modificare la propria posizione scegliendone una qualsiasi dalla mappa. Sotto questo aspetto l'applicazione è inerme, e ogni tipologia di posizione che riceverà sarà considerata attendibile e utilizzata al fine di determinare se sia valida o meno per l'accesso ai file.

Un controllo efficace potrebbe essere quello di verificare che non siano stati concessi permessi per mascherare la posizione del dispositivo o che non ci siano in esecuzione altre applicazioni che utilizzano la posizione dell'utente.

7.1.1 Funzionamento in assenza dei permessi

Altra funzionalità della quale l'applicazione è sprovvista è il funzionamento senza la concessione dei permessi per la localizzazione da parte dell'utente.

Infatti nel caso in cui l'utente non conceda i permessi richiesti l'applicazione non sarà in grado di svolgere nessuno dei suoi compiti principali, rimane infatti solamente possibile visionare la mappa, con le relative zone che contengono file evidenziate, e muoversi su di essa.

Tramite uno studio approfondito del caso d'uso e magari l'implementazione di un back-end diverso si riuscirebbe a determinare la posizione dell'utente in maniera alternativa a quella prevista dal sistema Android.

7.1.2 Futuro dei L.B.S.

Secondo alcuni report del sito `businesswire.com` il mercato dei servizi basati sulla localizzazione è in costante crescita a livello globale e entro il 2027 dovrebbe arrivare

ad un valore di 97 miliardi di euro, attualmente ne varrebbe 28.

Questi servizi hanno ricevuto un'implementazione capillare in tantissimi ambienti e di conseguenza si sono create tantissime opportunità per utilizzare questi mezzi per sponsorizzare luoghi o attività in particolare, funzioni che attualmente rappresentano uno dei fattori chiave che guidano il mercato.

I principali attori del settore attualmente sono : Google, Telecom Technologies Co., Apple, Intel, Oracle, Microsoft e Qualcomm.

In un prossimo futuro con la maggiore diffusione del 5G, i servizi di questo genere potranno ricevere un forte boost grazie al potenziamento dei dispositivi IoT in quanto andrebbero a risolvere molte delle problematiche relative ai metodi di localizzazione utilizzati attualmente, facendo degli esempi : l'utilizzo del GPS richiede molta energia, mentre la localizzazione tramite reti wifi è possibile in aree ristrette e soffre di problemi di sicurezza.

Una rete di dispositivi IoT basata sul 5G utilizzata per la localizzazione è in grado di risolvere questi problemi garantendo un'elevata larghezza di banda, una connettività affidabile e maggiore sicurezza.

Bibliografia

[1] Georeferenziazione. https://it.wikipedia.org/wiki/Georeferenziazione

[2] Geolocalizzazione. https://it.wikipedia.org/wiki/Geolocalizzazione

[3] Trilaterazione. https://it.wikipedia.org/wiki/Trilaterazione

[4] Maps SDK for Android.
https://developers.google.com/maps/documentation/android-sdk/start

[5] Sistema di posizionamento globale.
https://it.wikipedia.org/wiki/Sistema_di_posizionamento_globale

[6] Documentazione Android. https://developer.android.com/docs

[7] Testo GPDR. https://www.garanteprivacy.it/il-testo-del-regolamento

[8] ePrivacy. https://www.cookiebot.com/it/regolamento-eprivacy/

[9] Documentazione Firebase. https://firebase.google.com/docs