Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Sviluppo di una Piattaforma di Crowdsensing per l'Analisi di Dati

Relatore:
Chiar.mo Prof.
Luciano Bononi

Presentata da:
Simone Masini

Co-relatore:
Dott. Luca Bedogni

Sessione III
Anno Accademico 2013/2014

A Sandro,
Enza,
Monica
e Melissa

Indice

Elenco delle figure

Listati di codice

Elenco delle tabelle

Capitolo 1

Introduzione

Il progetto di tesi è consistito nello sviluppo di una piattaforma Client-Server in grado di registrare i dati ambientali. Tali dati possono essere raccolti da vari dispositivi, che possono essere dispositivi mobili, oppure schede Arduino o Raspberry PI con i relativi sensori collegati. Gli smartphone odierni, anche se non sono costruiti appositamente per il rilevamento, sono muniti di diversi sensori che possono essere utilizzati per raccogliere i dati dell'ambiente. Il sistema che andrò ad implementare dovrà essere un sistema indipendente dal tipo di dispositivo utilizzato, in modo che qualunque device potrà raccogliere i dati per questa piattaforma.

Il paradigma di raccolta dati è quello del Crowdsensing. Come vedremo meglio nel Capitolo 2, il crowdsensing consiste nella raccolta di dati condivisa tra numerosi utenti. Il crowdsensing permette di creare sistemi in cui tutti possono inserire dati senza troppa fatica, in questo modo si può generare una buona base di dati senza troppo lavoro da parte di un singolo individuo. La potenzialità del crowdsensing si dimostra dal fatto che una persona che vuole partecipare alla raccolta dati, spesso deve solo avviare un'applicazione nel proprio telefono, e questa si occupa autonomamente di raccogliere dati.[13, ACP] L'utente quindi non ha grosse difficoltà, a parte alcuni casi in cui la batteria avrà un consumo esagerato. Ci sono diverse distinzioni di crowdsensing, possiamo infatti voler gestire informazioni di diversi fenomeni,

come quelli ambientali, sociali oppure infrastrutturali. In questa piattaforma si cerca di gestire principalmente i dati di natura ambientale. Inoltre il crowdsensing può essere automatico o meno. Nel caso in cui è tutto automatico, l'utente non si rende conto di quali dati sta registrando e di quali dati vengono inviati, mentre in questo sistema, è consapevole di quali dati vengono raccolti. L'individuo infatti può scegliere quali sensori utilizzare, e in più può decidere di inviare i dati in forma anonima oppure associandoli al suo account. La parte più complessa è quella di creare un sistema che possa funzionare con tutti i tipi di sensori, e tutti i tipi di dispositivi. Per questo progetto mi limito a creare un client in Android per raccogliere i dati, ma con questo sistema si possono utilizzare molti altri dispositivi che abbiano i sensori opportuni. Se per esempio parliamo di temperatura, gli smartphone al giorno d'oggi non hanno il termometro per la temperatura esterna, ma se utilizzassimo un Arduino potremmo collegare un termometro ad esso ed avere una raccolta dati anche sulla temperatura ambientale.

Come verrà descritto meglio nel capitolo 3, questo sistema deve essere in grado di riconoscere quindi ogni tipo di dato e riconoscere il tipo di sensore che lo ha raccolto. Il progetto sarà diviso in tre applicazioni: il Server, il client Android e il client Web. Il server si occupa di registrare nel database i dati che il client android ha raccolto. L'applicazione Android quindi deve occuparsi di raccogliere i dati. Il client web invece ci permette di analizzare i dati presenti nel database. L'applicazione di raccolta, deve riconoscere i sensori installati nel dispositivo ed essere in grado di utilizzarli tutti. Oltre ai sensori si è pensato di aggiungere il WiFi e il Microfono, per misurare la rumorosità e il livello di onde wireless dell'ambiente. Il server quindi deve riconoscere che tipo di dati gli sta inviando l'applicazione, e registrarli opportunamente nel database. Il client invece ha accesso al database in sola lettura e permette di ricercare i dati e di visualizzarne i risultati all'interno di una mappa.

Nel capitolo 4 verranno spiegati i dettagli implementativi, in particolare lo sviluppo è iniziato dal server, in quanto è la parte più importante, poi

si è passato all'applicazione Android. Una volta che queste due strutture erano funzionanti ho potuto incominciare a raccogliere i dati per avere poi una base su cui fare le valutazioni. L'ultima parte è stata quella del client web, in quanto per crearla dovevo avere dei dati su cui lavorare.

Quando questa piattaforma era funzionante e avevo una buona base di dati raccolti, ho cominciato a fare ulteriori test, come verrà descritto nel capitolo 5. Con la fase di test è stato possibile valutare il sistema e verificare quali miglioramenti apportare.

Il client web è disponibile alla pagina: http://simone.masini3.web.cs.unibo.it mentre l'apk dell'applicazione Android si può scaricare da questo indirizzo: https://www.mediafire.com/folder/v3cz3721v6aq5/Crowdsensing_Client_Android

Capitolo 2

Stato dell'Arte

In questo capitolo sarà inizialmente introdotto il concetto di Crowdsensing. Il crowdsensing presenta un nuovo paradigma di rilevamento basata sul potere di vari oggetti mobili, ad esempio, smartphone, dispositivi indossabili, i veicoli equipaggiati con sensori, ecc...

2.1 Crowdsensing

Con il termine Crowdsensing si intende una raccolta condivisa di dati utilizzando i sensori di vari dispositivi con lo scopo di misurare un fenomeno di interesse comune.[16, MCFC] Questi dati non sono raccolti da un singolo individuo o organizzazione, ma vengono condivisi da molti utenti sparsi in varie zone. Ci possono essere vari tipi di categorie di Crowdsensing in base a diversi criteri, il primo criterio è il coinvolgimento diretto dell'utente nel processo, in questo caso si parla di Partecipatory Crowdsensing, altrimenti quando l'utente non è coinvolto parliamo di Opportunistic Crowdsensing. Nel Partecipatory Crowdsensing[17, PSMC] gli utenti dei dispositivi di rilevamento inviano attivamente i dati dei sensori ad un server. Nell'altro caso invece, l'invio di informazioni è automatico, con il minimo coinvolgimento dell'utente. Il secondo criterio è il tipo di fenomeno misurato, si possono distinguere tre categorie, che sono crowdsensing ambientale, infrastrutturale o sociale.

Il crowdsensing ambientale viene utilizzato per misurare l'ambiente naturale (ad esempio, il livello delle acque, l'inquinamento atmosferico). Il crowdsensing infrastrutturale viene utilizzato per misurare le infrastrutture pubbliche (ad esempio, la congestione del traffico e condizioni della strada).[19, BTBC] Infine il crowdsensing sociale viene utilizzato per misurare i dati sulla vita sociale degli individui (ad esempio, i cinema visitati da un individuo).

2.1.1 Applicazioni che utilizzano crowdsensing

Esistono diverse applicazioni che utilizzano questo meccanismo del crowdsensing, vediamone alcune.

1. **Creek Watch**, un'applicazione iPhone sviluppata da IBM Almaden Research Center. L'applicazione controlla i livelli di acqua e la qualità della zona circostante. Creek Watch permette agli utenti dell'applicazione di presentare le seguenti informazioni:

   - La quantità di acqua che vedono,
   - La velocità del flusso,
   - La quantità di spazzatura che vedono,
   - Una foto del corso d'acqua.

   L'IBM Almaden Research Center aggrega i dati raccolti e li condivide con le istituzioni che sono responsabili della gestione delle risorse idriche. I dati forniti dagli utenti vengono visualizzati su una mappa interattiva. Creek Watch richiede agli utenti di inviare manualmente i dati ambientali, perciò utilizza il participatory crowdsensing di tipo ambientale.

2. **Nericell**, Il progetto è sviluppato da Nericell Microsoft Research e permette alle persone di inviare automaticamente i dati sul traffico stradale utilizzando i loro telefoni smartphone. Una persona disposta a partecipare a Nericell ha solo bisogno di avviare un'applicazione installata

sul suo telefono. Lo smartphone inizia a monitorare automaticamente le condizioni della strada e del traffico e invia le informazioni raccolte al cloud per l'analisi. Le condizioni della strada e del traffico sono monitorati attraverso una serie di sensori disponibili su smartphone, come l'accelerometro, il Bluetooth, il GPS, e il microfono. Ad esempio, il microfono può essere utilizzato per rilevare clacson. L'accelerometro può essere utilizzato per rilevare buche. Nericell raccoglie automaticamente i dati dagli utenti quindi utilizza un opportunistic crowdsensing di tipo infrastrutturale.

3. **DietSense**, un sistema software sviluppato presso UCLA (University of California, Los Angeles). DietSense consente alle persone di condividere informazioni sulle loro abitudini alimentari con altri individui. Così, le persone possono confrontare le loro abitudini alimentari. Coloro che intendono utilizzare DietSense dovranno rispondere a un sondaggio chiedendo informazioni come: foto di cibo, la motivazione per la scelta di questo e il luogo della sua preparazione. Gli individui devono inviare manualmente le informazioni sulle loro abitudini alimentari, quindi DietSense utilizza un participatory crowdsensing di tipo sociale.

2.1.2 Crowdsensing e Privacy

C'è anche un aspetto da tenere conto quando si parla di crowdsensing che è quello della Privacy. I dati raccolti infatti, possono contenere informazioni personali, e tramite queste si può risalire all'identità di una persona. Inoltre quando si parla di opportunistic crowdsensing l'utente non ha controllo sui dati che invia, perciò il microfono per esempio potrebbe registrare conversazioni private senza che l'utente se ne accorga. Con il partecipatory crowdsensing invece, il tema della sicurezza è meno accentuato, perché l'utente decide quali dati condividere. I creatori di applicazioni crowdsensing devono trovare un modo per proteggere i dati delle persone, ci sono tre ap-

procci principali su come proteggere la privacy degli utenti di piattaforme crowdsensing.

1. **Anonymization**, può essere usato per rimuovere le informazioni identificative raccolte, ma solleva due problemi. In primo luogo, la semplice rimozione di informazioni identificative come i nomi e gli indirizzi non possono garantire l'anonimato. Per esempio, nei casi in cui le applicazioni crowdsensing raccolgono dati di posizione, l'anonymization non è in grado di impedire l'identificazione dell'individuo. Infatti i dati di localizzazione anonimizzati mostreranno ancora le posizioni più visitate di un utente, la quale a sua volta può portare all'identificazione di quella persona. In secondo luogo, nel contesto di trasformazione in forma anonima dei dati, l'utilità dei dati e la privacy dei dati sono obiettivi contrastanti. Di conseguenza, la trasformazione in forma anonima dei dati aumenterà la tutela della privacy, ma diminuisce l'utilità di essi.

2. **Crittografia**, terzi non autorizzati non saranno in grado di utilizzare i dati personali, anche se acquistano l'accesso ai dati crittografati. Va notato che la cifratura di un grande volume di dati può richiedere notevoli risorse informatiche.

3. **Perturbazione dei Dati**, ci si riferisce all'aggiunta di interferenze nei dati subito dopo che questi sono stati inviati dagli individui. A seguito della perturbazione, i dati presentati da persone non saranno identificabili, tuttavia consentirebbero un buon funzionamento delle applicazioni crowdsensing. Una forma popolare di perturbazioni dei dati è la micro-aggregazione. Il termine micro-aggregazione si riferisce alla sostituzione di un campo specifico con un valore aggregato o un valore più generale. Ad esempio la sostituzione di un codice di avviamento postale con il nome di una regione o uno stato. La micro-aggregazione può essere operativamente definita in due fasi, vale a dire, la partizione e aggregazione. La partitione si riferisce al partizionamento del set di dati in

più parti (gruppi, cluster). L'aggregazione si riferisce alla sostituzione di ogni record in una parte con il record medio.

2.1.3 Riepilogo

Il crowdsensing è un paradigma che presenta certamente un ottimo potenziale.[18, OMC] Gli utenti possono avere diversi incentivi per la presentazione dei dati, come ad esempio evitare ingorghi, evitare il consumo di cibo non preparato bene, e contribuire alla tutela dell'ambiente. Il conferimento dei dati crowdsensing non richiede sforzi notevoli da parte degli utenti. Spesso devono solo attivare un'applicazione crowdsensing per avviare l'invio automatico dei dati.

Il gran numero di utenti di dispositivi mobili e la loro mobilità intrinseca consente una nuova e rapida crescita di rilevamento: la capacità di acquisire conoscenze a livello locale attraverso sensori con capacità di telefonia mobile (per esempio, l'ubicazione, livello di rumore, le condizioni del traffico, e in futuro dati più specifici come l'inquinamento) e la possibilità di condividere questa conoscenza nella sfera sociale, ci fa capire le grandi potenzialità che presenta questo paradigma.

Tipi di crowdsensing:

- In base al coinvolgimento dell'utente:
  1. Participatory crowdsensing
  2. Opportunistic crowdsensing
- In base al tipo di fenomeno:
  1. Crowdsensing Ambientale
  2. Crowdsensing Infrastrutturale
  3. Crowdsensing Sociale

Capitolo 3

Progettazione

Il problema principale del sistema che si sta andando a creare è quello di dover registrare dati di natura diversa provenienti da molti tipi di sensori, che si riferiscono a fenomeni diversi. Esistono infatti applicazioni che registrano i dati da sensori stabiliti, e il tipo di fenomeno che si va a misurare è di un tipo preciso. In questa piattaforma invece si vuole dare la possibilità all'utente di scegliere di quale fenomeno vuole registrare i dati e di conseguenza di utilizzare i sensori relativi. Ad esempio il soggetto può essere interessato a voler monitorare la luminosità dell'ambiente e quindi di utilizzare il sensore di luminosità, oppure di voler misurare la rumorosità dell'ambiente e quindi di utilizzare il microfono. Il sistema quindi deve essere in grado di riconoscere quale sensore sta inviando i dati e registrarli nel modo opportuno, e dall'interfaccia Client Web, si deve poter analizzare i dati in base alle ricerche opportune. Gli esempi trovati in rete erano tutti basati su sensori collegati alle schede Arduino o a Raspberry PI, i quali comunicavano con un servizio web. Le schede avevano la funzione di utilizzare i sensori per registrare i dati e in un secondo momento inviarli in un server. Il server prendeva i dati inviatogli e li registrava in un database. L'idea di base da cui partire è questa, eccetto il fatto che in questi casi il server sapeva che il sensore era di un tipo specifico e li salvava per un fenomeno specifico. I casi maggiormente trovati erano termometri collegati alle schede, perciò il server

sapeva che i dati che gli arrivavano erano solo relativi alla temperatura e li salvava nel database come gradi centigradi. Possiamo notare che non è mai stata realizzata una struttura in grado di contenere dati riguardanti fenomeni eterogenei. A questo punto per riuscire a riconoscere i diversi tipi di dati sarà necessario realizzare una buona infrastruttura di base con la quale registrare i record nel database.

Il progetto è diviso in tre parti: server, client web e applicazione client mobile. L'applicazione mobile è il client che si occupa della raccolta dati, può essere configurato dall'utente, che potrà scegliere quali sensori usare e quindi quali dati inviare. Il server invece si occupa di ricevere i dati dall'applicazione e deve salvarli in un database. Il client web permette una ricerca dei dati registrati e restituisce un risultato dei dati analizzati.

3.1 Server

La parte fondamentale del progetto è il server che salva i dati. Il server si interfaccia con un database MySQL, dove ci saranno varie tabelle in grado di contenere tali dati. È necessario che questo componente sia solido, esso deve essere indipendente dalla parte client, e deve poter gestire ogni dato che viene inviato. Se il server dovesse dipendere dal client, il sistema sarebbe limitato, perché non si potrebbe utilizzare qualsiasi dispositivo per inviare i dati, ma solo i dispositivi con un applicazione creata ad hoc. Al contrario invece in questo sistema si vuole dare la possibilità di poter utilizzare qualsiasi dispositivo per raccogliere i dati. Per esempio si può utilizzare un Arduino, e ogni tipo di sensore disponibile che si colleghi ad esso. La responsabilità riguardanti il riconoscimento del sensore e il salvataggio dei dati va gestita quindi nella parte server, e non bisogna dare molte responsabilità al client.

A questo punto per riuscire a riconoscere i diversi tipi di dati è necessario dividere le tabelle del database facendo distinzione fra la tabella dei sensori e la tabella delle letture. In questo modo il server può raccogliere i dati sulla lettura tutti in un'unica tabella, associando ad ogni dato un identificativo del

sensore. Nella tabella dei sensori invece vengono raccolti tutti i sensori con le relative informazioni con cui analizzare il dato. Questo meccanismo permette di gestire lato server i dati ricevuti e di registrarli correttamente anche se provengono da diversi dispositivi e si riferiscono a fenomeni diversi. In questo caso rimane un solo problema, i sensori infatti devono essere già tutti registrati nel database, altrimenti quando viene inviato un dato potrebbe non esserci il sensore associato. Il database quindi sarà cosi composto:

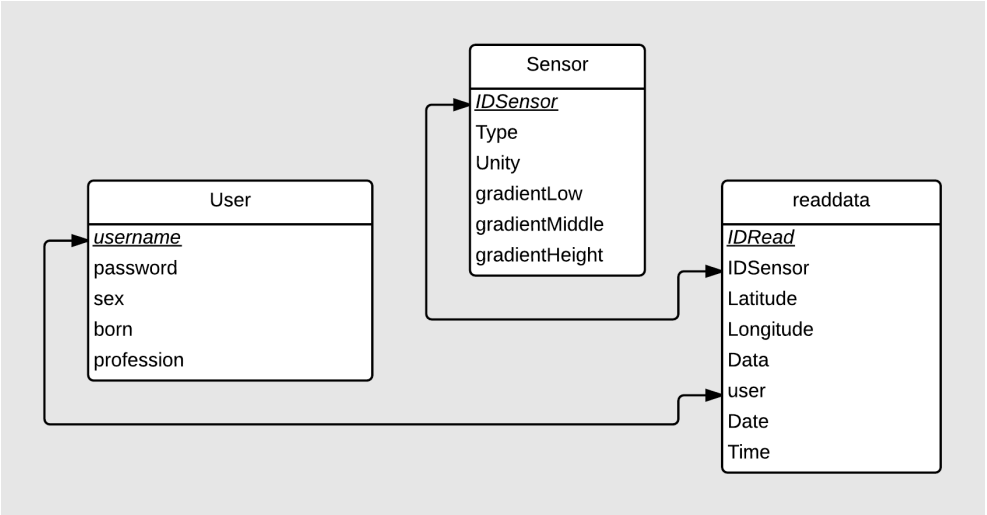

Figura 3.1: Diagramma Entità-Relazioni del database

3.2 Applicazione Mobile

L'applicazione mobile è stata sviluppata in ambiente Android. Essa è la parte che si occupa principalmente della raccolta dei dati. L'utente può configurare a suo piacimento l'applicazione per utilizzare i sensori che preferisce. L'applicazione mostra un elenco di tutti i sensori presenti nel device, e una volta scelti permette di avviare un servizio che registra i dati nella memoria interna del dispositivo. L'utente poi può inviare i dati al server. L'applicazione mobile non ha particolari responsabilità, in quanto la gestione dei dati

rimane tutta nel server. L'unico controllo che effettua l'applicazione è quello di verificare che un sensore selezionato sia già registrato nel database. In caso contrario non permette di utilizzare quel sensore. Ogni volta che viene scelto un sensore nelle impostazioni quindi, l'applicazione contatta il server.

Oltre ai sensori è stato deciso di inserire altri due componenti hardware per la raccolta dei dati: il microfono e il WiFi. Il microfono non è riconosciuto da Android come un sensore, perciò è stato inserito in fondo alla lista dei sensori per poter registrare il livello del suono. Il WiFi non è un sensore vero e proprio, ma può essere interessante registrare anche la potenza del segnale della connettività wireless, quindi è stato inserito anche questo componente nella scelta dei sensori. L'applicazione deve raccogliere anche le informazioni riguardante la posizione dell'utente, senza questa funzione la raccolta dei dati sarebbe inutile, perché nel database ci sarebbero dei dati di cui è impossibile risalire alla zona a cui si riferiscono.

3.3 Client Web

Il client web si occupa della ricerca e l'analisi dei dati registrati precedentemente. Il client web si interfaccia anch'esso con il database MySQL contenente le informazioni, ma a differenza del server che può leggere e scrivere i dati, questo client ha la possibilità solo di leggerli. L'utente potrà inserire dei parametri per ricercare i dati, i cui risultati saranno restituiti all'interno di una mappa. Per questa parte di progetto sono state fondamentali le librerie di Google Maps, senza le quali sarebbe difficoltoso rappresentare i risultati di ricerca.

Sono state previste diverse tipologie di ricerca, in modo da avere diversi output sulla mappa. Il primo tipo di ricerca è quello con le Heatmap, dove l'utente potrà vedere i dati rappresentati da regioni evidenziate nella mappa tramite gradienti di colore. Le zone saranno colorate da diversi gradienti in base al tipo di ricerca fatta. Ogni gradiente contiene tre colori che indicano dal minimo al massimo il livello di concentrazione. Un altro metodo di ricerca

è quello con i marker, numerosi marker verranno inseriti in vari punti e cliccando su un marker si visualizza il dato registrato in quel preciso punto. Per facilitare la lettura della mappa, i marker inoltre saranno raggruppati nel caso in cui siano presenti molti marker vicini, e saranno distinguibili solo zoomando la mappa.

Ogni volta che si effettua una ricerca, questa viene inserita in una tabella dove si possono effettuare controlli su di essa. Per esempio si può cancellare la ricerca, visualizzare gli utenti che hanno raccolto quei dati (ed eventualmente escludere certi utenti dalla ricerca), e visualizzare dei grafici che nerappresentano i dati.

3.4 Utilizzo

L'utilizzo del sistema può essere inteso da vari punti di vista. Il primo caso è quello dell'utente che registra i dati dall'applicazione Android, per prima cosa deve configurare i sensori che vuole utilizzare. Nelle impostazioni dell'applicazione c'è la voce ***Sensori***, composta da una lista di sensori e delle checkbox.

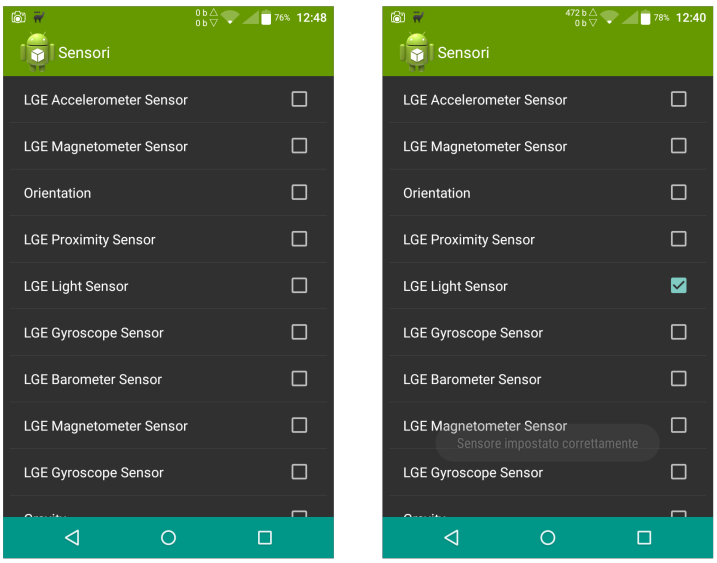

Figura 3.2: Impostazione di un sensore

Cliccando una checkbox viene effettuata una chiamata al server, che informa il client se quel sensore è possibile utilizzarlo oppure no e di conseguenza viene impostato il sensore oppure viene restituito un messaggio di errore.

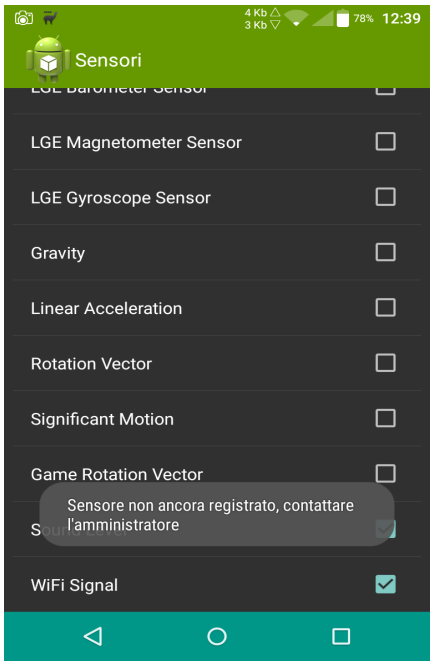

Figura 3.3: Impostazione di un sensore

Nelle impostazioni è anche possibile inserire informazioni riguardati l'account, utilizzando la voce ***Login***.

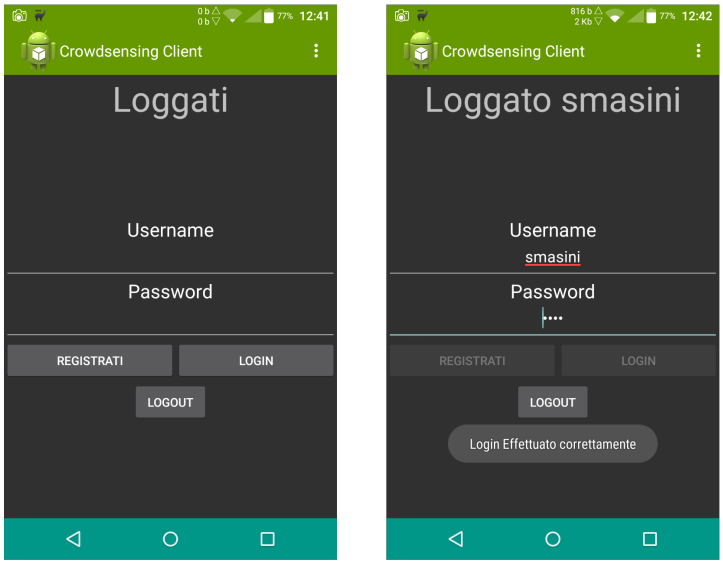

Figura 3.4: Login di un utente già registrato

È stata inserita questa funzionalità per associare i dati registrati ad un utente. Nel caso in cui l'utilizzatore non vuole fornire al sistema i suoi dati può tranquillamente saltare questa procedura, e inviare i dati senza registrarsi, in questo caso il sistema associa i dati ad un utente anonimo. Quindi l'utente può registrarsi o effettuare il login nel caso sia già registrato. La registrazione richiede i seguenti dati:

- Username
- Password
- Professione
- Sesso
- Data di Nascità

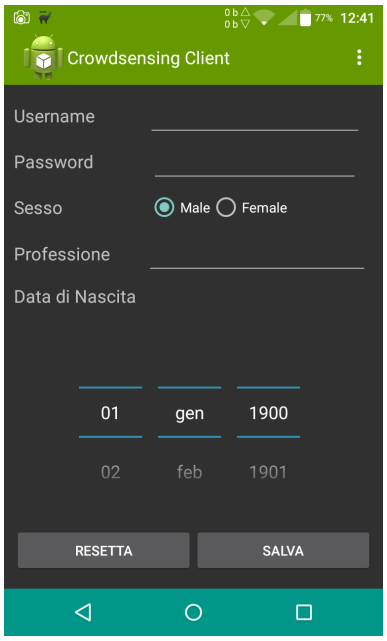

Figura 3.5: Registrazione di un nuovo account

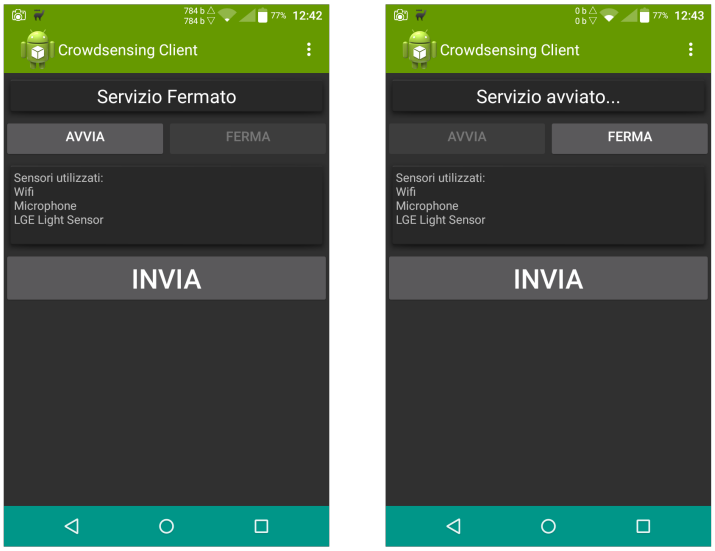

Figura 3.6: Schermata di avvio

Una volta finita la configurazione, dalla schermata principale si avvia o termina il servizio di raccolta dati. Una volta che i dati sono stati raccolti l'utente può decidere di inviare i dati al server che li registrerà. La raccolta di dati è stata implementata come un service n background, per questo motivo

non è necessario che si tenga aperta l'applicazione dopo aver avviato la registrazione dei dati, e potrà poi riaprirla quando vorrà terminare il processo o inviare i dati.

Nel caso in cui si voglia analizzare i dati presenti nella piattaforma, si dovrà visitare il client web, dove si potrà scegliere prima di tutto il sensore di cui visualizzare i dati, poi il range dei valori che si vuole visualizzare. Opzionalmente si può scegliere anche un periodo o un orario per filtrare ulteriormente la ricerca. Di default il tipo di ricerca è Heatmap ma si può cambiare selezionando il Marker.

Figura 3.7: Ricerca dei dati

Il risultato viene visualizzato nella mappa, se è stata scelta la heatmap, ci saranno delle regioni o punti della mappa che saranno colorate da gradienti diversi di colore. I gradienti sono diversi per ogni tipo di sensore, quindi per il sensore di luce avremo una scala di colori, per il microfono un'altra scala e cos via. Questi gradienti hanno 3 colori, dove il primo indica i punti con i valori più bassi, l'ultimo indica i punti con i valori più alti, mentre quello in mezzo indica i punti con i valori medi. Sono presenti nella mappa due pulsanti che permettono di modificare il radius e l'opacità dell'heatmap, in questo modo si rende più comoda la navigazione.

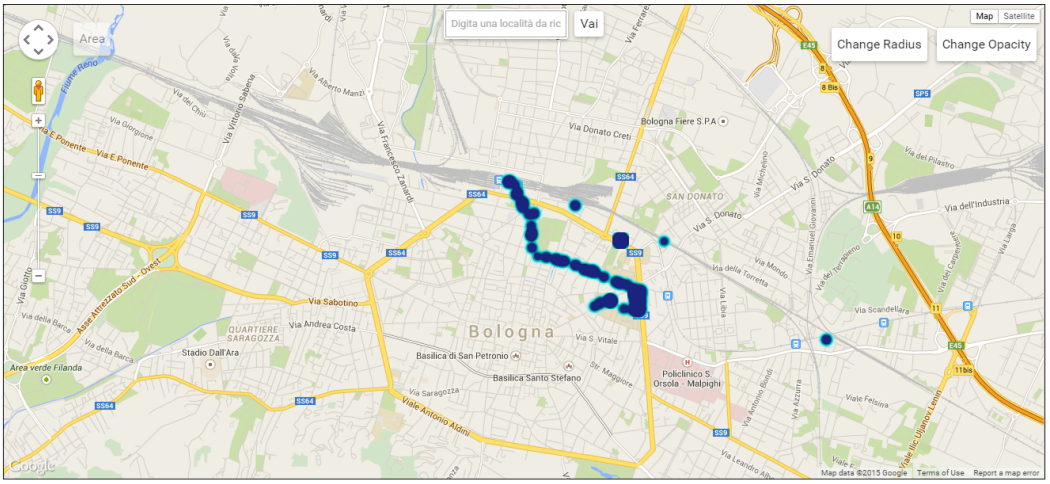

Figura 3.8: Visualizzazione dei Risultati mediante Heatmap

Il risultato con i marker invece sarà visualizzato in gruppi. I gruppi vengono creati automaticamente se i marker sono molto vicini tra di loro, questo meccanismo permette di consultare la mappa in maniera comprensibile. Andando a zoomare su un gruppo i marker si separano e si visualizzano le zone precise in cui sono collocati. I marker hanno colori diversi in base al tipo di sensore.

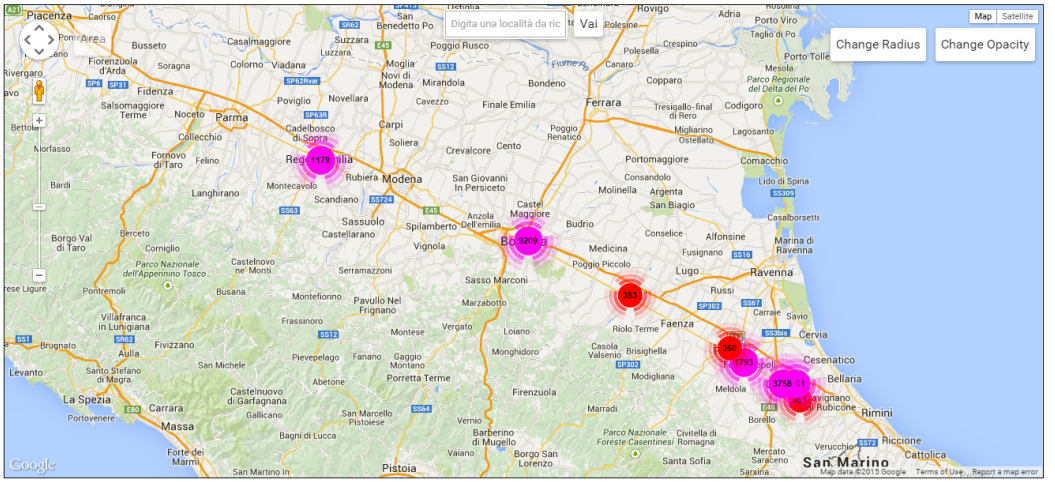

Figura 3.9: Visualizzazione dei Risultati mediante Gruppi di Marker

Cliccando su un marker si visualizzano i dati che sono stati raccolti in quel preciso punto.

Nella mappa è presente anche un controllo per ricercare un determinato luogo, si può digitare un indirizzo qualsiasi e la mappa sarà reindirizzata in quella località. Se si vuole limitare la ricerca in una precisa area, c'è il pulsante ***Area***. Selezionandolo compare un rettangolo editabile nella mappa, che l'utente può modificare. Oltre a questi controllo sono presenti i soliti controlli di Google Maps per zoomare o cambiare la vista della mappa.

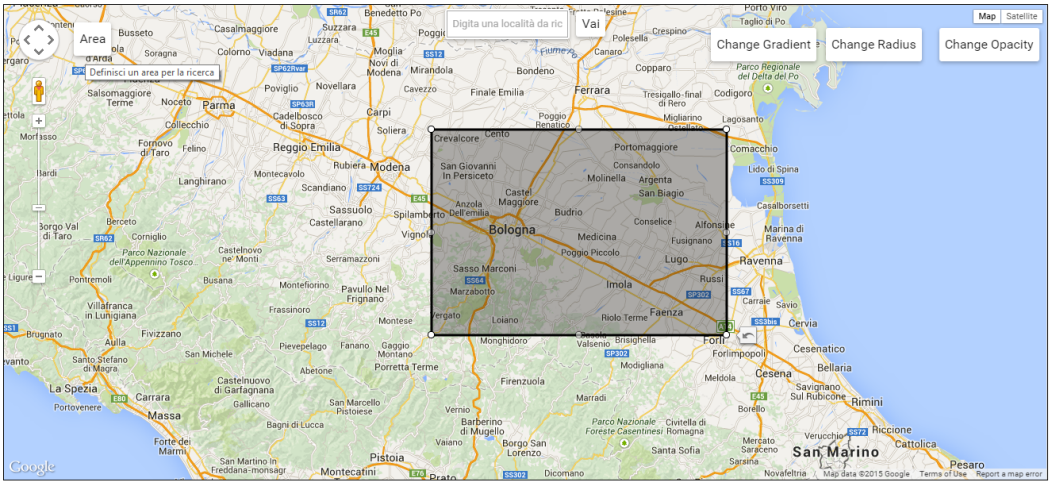

Figura 3.10: Selezionare un'area per limitare la Ricerca

Una volta effettuata una ricerca, questa viene inserita in una tabella riassuntiva di tutte le ricerche da cui è possibile rimuoverne una in un secondo momento. Così facendo è possibile effettuare ricerche incrociate, inserendo nella mappa risultati di diversi sensori, o diversi orari. È anche possibile effettuare ricerche di diverso tipo, inserendo nella mappa sia le regioni, sia i marker, ottenendo una vista più accurata. Da questa tabella inoltre sono presenti tre tasti, uno con il numero di utenti, e cliccandolo si apre una finestra modale con la lista degli utenti che hanno raccolto i dati relativi a quella specifica ricerca. Si possono escludere degli utenti rimuovendo i dati condivisi dal medesimo. Un'altro permette di scaricare i dati che fanno parte della ricerca in formato csv. L'ultimo pulsante ci permette di visualizzare

dei grafici. In particolare si vede un Istogramma che mostra il numero dei rilevamenti rapportati per valore, un grafico temporale che rappresenta il numero di rilevamenti nel tempo, e infine il grafico della CDF (Cumulative distribution function - Funzione di Ripartizione). Per ogni grafico si possono visualizzare i dati specifici che lo compongono.

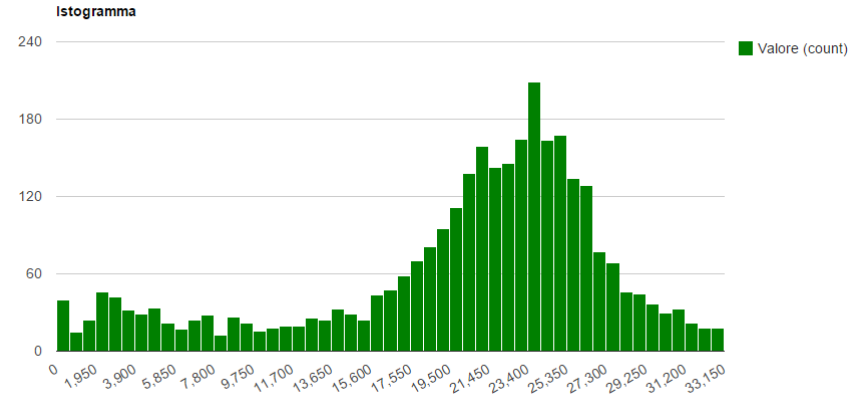

Figura 3.11: Istogramma

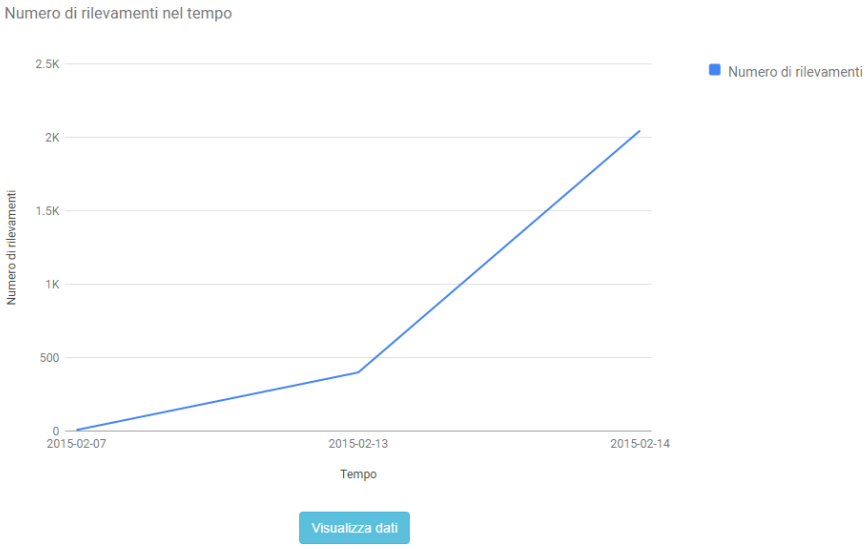

Figura 3.12: Grafico Temporale

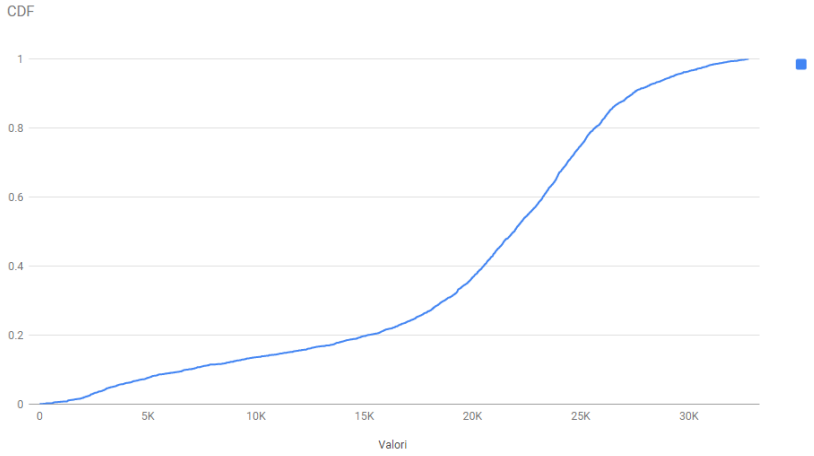

Figura 3.13: Grafico CDF

Tabella 3.1: Esempio di dati dell'Istogramma

Valore Rilevato	**N Rilevamenti**
0	28
74	2
193	2
278	2
309	2
318	2
1033	2
1135	2
1266	2
1268	2
1273	2
1286	2
1298	2
...	...
25151	2
25157	3
25169	2
25172	2
25173	2
25176	2
25177	2
25178	3
25185	2

Tabella 3.2: Esempio di dati dell'Istogramma

Valore Rilevato	**Percentuale**
0	0.00036603221083455345
74	0.0007320644216691069
193	0.0010980966325036604
278	0.0014641288433382138
309	0.0018301610541727673
318	0.0021961932650073207
319	0.002562225475841874
584	0.0029282576866764276
597	0.003294289897510981
...	...
32339	0.9959736456808199
32350	0.9963396778916545
32408	0.996705710102489
32409	0.9970717423133236
32498	0.9974377745241582
32573	0.9978038067349927
32590	0.9981698389458272
32597	0.9985358711566618
32637	0.9989019033674963
32642	0.9992679355783309
32717	0.9996339677891655
32767	1

Capitolo 4

Implementazione

Il progetto è stato implementato in diversi step, uno per ogni applicazione del progetto. Il primo step è stato quello di implementare il server, per prima cosa bisognava decidere in che linguaggio svilupparlo. Una volta creata questa prima parte era necessario avere un'applicazione che raccogliesse i dati e si interfacciasse con il server, quindi il secondo step è stato quello di creare l'applicazione Android. Completata l'infrastruttura di raccolta dati, serviva soltanto il client web per gestire l'analisi dei dati raccolti. Il terzo step consisteva nel creare una pagina interattiva per la ricerca.

4.1 Server

La parte più importante è la parte server, perciò è necessario partire da questa per sviluppare tutto il progetto. Inizialmente era stato pensato di svilupparlo tramite pagine PHP, ma in seguito questa soluzione si è rivelata poco flessibile, e scomoda da usare. La seconda opzione è stata quella di utilizzare node.js. Node.js è stato creato e pubblicato per l'uso di Linux nel 2009. Il suo sviluppo e la manutenzione è stato guidato da Ryan Dahl e sponsorizzato da Joyent, l'azienda dove lavorava Dahl. Successivamente, nel 2011, è stato introdotto NPM, un gestore di pacchetti per le librerie di Node.js. Node.js è un framework event-driven per il motore JavaScript V8, utilizzato

nei browser chrome e chromium. Si tratta quindi di un framework relativo all'utilizzo server-side di Javascript e ci permette di utilizzare questo linguaggio tipicamente client-side per lo sviluppo server. Il modello event-driven, o programmazione ad eventi, si basa su un concetto piuttosto semplice: si lancia una azione quando accade qualcosa. Ogni azione quindi risulta asincrona a differenza dei pattern di programmazione più comuni in cui una azione succede ad unaltra solo dopo che essa è stata completata. Node.js è ideato per girare su un apposito server HTTP e di utilizzare un unico thread eseguendo un processo alla volta. Le applicazioni Node.js sono events-based ed eseguono processi asincroni, non sfruttando quindi il classico modello basato su processi o thread concorrenti. Un vantaggio di node.js è quello di non bloccare input e output, infatti le richieste possono continuare ad arrivare per poi essere eseguite in un secondo momento. Essendo processi asincroni, le risposte vengono gestite con le callback. Per ogni evento che viene chiamato vengono eseguite delle callback, dove saranno specificate le azioni da compiere.

Con nodejs si è potuto creare un server completo con estrema facilità, inoltre con l'ausilio dei moduli messi a disposizione da NPM, sono stati resi facili alcuni processi insidiosi. Per utilizzare dei moduli di NPM è necessario richiamarli all'inizio del server.

```
1 var mysql = require('mysql');
2 var express = require('express');
3 var bodyParser = require('body-parser');
```

Codice 4.1: Importazione moduli

Il modulo mysql permette di collegare il server con un database MySQL. L'unica cosa da fare è creare la connessione con il database, basta inserire l'host e il nome del DB, insieme all'username e password di un utente con i permessi per collegarsi al database. In seguito si possono effettuare query SQL con il comando: ***query()***.

```
connection = mysql.createConnection({
        host : "stevie.heliohost.org",
        user : "my1503",
        password: "*****",
        database: "my1503"
    });
connection.connect();

var strQuery = "select MAX(IDRead) from readdata";
connection.query(strQuery, callback);
```

Codice 4.2: Connessione con il Database ed esecuzione di una query

Per la creazione del server è stato utilizzato il modulo express. Express è un framework per applicazioni web Node.js semplici e flessibili che fornisce un robusto set di funzionalità per il web e le applicazioni mobili. Tramite questo modulo sono riuscito a impostare un comportamento diverso in base all'url che viene passato alla richiesta. Per effettuare una richiesta infatti è necessario utilizzare un url del tipo:

```
http://indirizzo:porta/richiesta
```

Codice 4.3: Schema dell'indirizzo per effettuare una richiesta

Dove indirizzo sarà l'ip su cui è attivo il server, la porta sarà quella che viene impostata nello script del server (in questo caso 1337), e la richiesta sarà una stringa tra:

- insert
- requestValidSensor
- login
- insertUser

Esempio di indirizzo completo: ***http://melot.cs.unibo.it:1337/login***

```
app.use(function (req, res) {
    switch(req.url) {
        case '/insert':
            insertHandler(req, res);
            break;
        case '/requestValidSensor':
            requestValidSensorHandler(req, res);
            break;
        case '/insertUser':
            insertUser(req, res);
            break;
        case '/login':
            login(req, res);
            break;
    };
});
```

Codice 4.4: Gestione dell'url

Come possiamo notare nel Codice 4.4, in base all'url che viene passato alla richiesta viene chiamato un metodo diverso. In base al metodo chiamato ci sarà una gestione diversa dei dati che vengono inviati. Per ricevere i dati ci si è servito del modulo body-parser, grazie al quale, si possono ricevere dati con il metodo POST. Questo modulo è il motivo principale per cui ho deciso di utilizzare express. Body-parser infatti è un modulo complementare ad express, e grazie a questo si riescono, tra le altre cose, a ricevere dati di grandi dimensioni. Di default i dati che riceve il server possono essere grandi fino a 100 KB, e se viene inviato un file con peso maggiore ci restituisce un errore. Con body-parser si possono impostare diversi parametri che riguardano la ricezione dei dati. Ho utilizzato questo modulo per cambiare il limite della grandezza del json inviato dalla richiesta HTTP tramite metodo POST, e lo ho impostato a 50 MB.

```
app.use(bodyParser.json({limit: '50mb'}));
```

Codice 4.5: Cambiato il limite da 100 KB a 50 MB

Il server gestisce molti tipi di richieste, in base all'url che viene chiamato e i dati che vengono passati, il server chiama vari metodi con diversi comportamenti. Ora andrò ad analizzare le varie funzionalità del Server.

4.1.1 Richiesta del Sensore

Il metodo ***requestValidSensorHandler*** viene chiamato per effettuare un controllo su un sensore. Un client prima di poter inviare i dati deve accertarsi che il sensore che vuole utilizzare sia disponibile dal sistema. Per questo motivo il client invia un file json insieme alla richiesta con il nome del sensore, e questo metodo si occupa di leggere il json e di passare il valore ottenuto chiamando un'altro metodo: ***querySensor***. Questo metodo esegue una select nel database cercando il tipo che gli viene passato. Se la risposta esiste invia al client il codice identificativo di quel sensore, mentre nel caso in cui quel sensore non esiste nel database restituisce come risposta la stringa "no". Nel sistema sono già stati precaricati i tipi di sensori più comunemente utilizzati in Android. Nel caso in cui un utente voglia utilizzare un sensore non presente nel database, deve comunicarlo all'amministratore del sistema affinchè lo inserisca lui. Fino a che il sensore non è presente, l'utente non può utilizzarlo.

Il parametro da inviare con questa richiesta dev'essere una stringa con il tipo del sensore. L'applicazione android è stata implementata in modo tale che la stringa passata sia accettata correttamente dal server. Se si vuole implementare altre applicazioni in altri ambienti per la raccolta dei dati, allora queste devono prevedere un meccanismo in grado di inviare la stringa corretta in base al tipo di sensore, secondo le esigenze del server. I sensori presenti nel server al momento sono:

Quindi se si vuole inviare per esempio i dati relativi al sensore di luce, è necessario che l'applicazione invii i dati passando come parametro la stringa

Tabella 4.1: Sensori già inseriti nel database

IDSensore	Stringa Identificativa	Unità di Misura
1	light	lumen
2	sound_level	amplitude
3	wifi_signal	decibel

"*light*". In particolare il json deve avere questa forma:

```
{
  sensorType: "light"
}
```

Codice 4.6: json da passare alla richiesta di sensore

4.1.2 Richiesta di Login

Il metodo ***login*** richiede che siano stati passati alla richiesta l'username e la password dell'utente che intende effettuare il login. Il json deve essere composto in questo modo:

```
{
  user: "smasini",
  pass: "pass"
}
```

Codice 4.7: json da passare alla richiesta di login

Una volta letti questi dati effettua una query che va a contare il numero di righe che hanno sia l'username che la password scelti, e controlla che il risultato sia uguale a uno. Nel caso di login corretto restituisce la risposta "ok" altrimenti restituisce "no".

4.1.3 Registrazione di un Utente

Per inserire un utente deve essere passato dall'applicazione un json di questa forma:

```
{
  username: "smasini",
  password: "pass",
  sex: "male",
  born: "YYYY-MM-DD",
  profession: "studente"
}
```

Codice 4.8: json da passare alla richiesta di registrazione utente

Il metodo ***insertUser***, passerà questi valori al metodo ***insertData***, che si occuperà di inserirlo nel database. Questo metodo effettua l'inserimento, e se tutto va a buon fine restituisce la risposta "yes", altrimenti se si generano errori, (per esempio esiste già un utente con lo stesso username) viene restituita la stringa "no".

4.1.4 Inserimento Dati

Il metodo ***insertHandler*** riceve un json di queste caratteristiche:

```
{
  values = [
        {
          type: "light",
          data: 25.6,
          lat:  44.10038,
          lng:  12.301961,
          date: 2015-02-07,
          time: 12:01:00,
          user: "smasini"
        },
        {
          type: "light",
          data: 24.5,
          lat:  44.100395,
          lng:  12.301856,
          date: 2015-02-07,
```

```
            time: 12:21:00,
            user: "smasini"
        },
        .
        .
        .
        {
            ...
        }
    ]
}
```

Codice 4.9: json da passare alla richiesta per inserire i dati

Per prima cosa si ottiene l'ultimo IDRead più alto, e viene usato per ogni inserimento incrementandolo di volta in volta. Poi vengono generati i parametri di inserimento, che dovranno essere passati al metodo ***insertData***. Prima di chiamare ***insertData*** però, viene chiamato ***querySensor***, in quanto il tipo di sensore à definito come stringa, mentre nel database va salvato l'identificativo del sensore. Per questo quindi è necessario cambiare la stringa con il numero intero corrispondente, dopodiché si può effettuare l'inserimento.

4.2 Applicazione Mobile

L'applicazione Android ha il compito di raccogliere i dati. È necessario che questo processo venga eseguito in background senza dover lasciare aperta l'applicazione. Per questo il procedimento di raccolta dati è stato previsto come Service, in questo modo l'applicazione si può terminare, ma il processo rimane attivo. Questo permette anche un consumo limitato della batteria. L'applicazione si basa su una sola Activity che permette di avviare/fermare il service, oppure di inviare i dati al server. Dalla schermata principale si può accedere anche alle impostazioni da cui impostare i sensori o effettuare il login/registrazione. Le varie impostazioni sono stati implementati come Fragment che vengono sostituiti nel contenitore principale.

Nella voce sensori delle impostazioni, si visualizzano delle checkbox con il rispettivo sensore, la classe **SettingFragment** si occupa di generare questa lista. Premendo una checkbox si avvia un AsyncTask che effettua il collegamento con il server. Se il server risponde positivamente, il codice del sensore viene inserito nelle SharedPreferences.

Ho creato una classe **RequestTask** che eredita da AsyncTask, in modo da utilizzare sempre questa per inviare richieste al server. Quando si crea la classe bisogna passare una stringa con il tipo della richiesta, e creare l'oggetto json che poi sarà quello che dovrà essere inviato.

Il login è gestito dalla classe **LoginFragment**. Anche qui viene utilizzata la classe RequestTask per contattare il server. Premendo Registrati invece si chiama un altro Fragment, **AccountFragment**. Si possono inserire i dati che poi verranno utilizzati per la registrazione di un account, e vengono inviati al server, nel caso tutto vada a buon fine vengono salvati i dati dell'utente nelle SharedPreferences.

È stata implementata la classe **Geolocalizzazione**, che ha il compito di controllare la posizione e di tenerla in memoria. Questa implementa un LocationListener, con la quale viene sempre notificata ogni volta che l'utente cambia posizione. Questa classe è un Singleton, infatti ne esisterà solo un instanza, che verrà utilizzata da più parti del sistema.

La classe che definisce il Service è **DataSensorService**, che estende appunto la classe Service. Quando viene avviato, ricerca nelle SharedPreferences i codici dei sensori salvati, e crea per ognuno di loro una classe **SensorStart**, con cui incomincia a registrare i dati. SensorStart implementa SensorEventListener, al momento della creazione viene associato ad un sensore e viene creato un file json. Questa classe rimane in attesa e ogni volta che il sensore scelto cambia valore, viene aggiunto il valore correlato da tutti i dati nel file json salvato nella memoria interna del dispositivo. In questo momento per inserire il dato viene presa l'ultima posizione salvata in Geolocalizzazione, l'utente (oppure "unknown" se non loggato), il timestamp e il tipo di sensore.

Nel caso in cui venga scelto il WiFi o il Microfono, non viene creato un SensorStart, ma vengono create rispettivamente **WifiMeter** e **SoundMeter**. Queste due classi ripropongono lo stesso comportamento della classe SensorStart, ma a differenza di questa, non registrano dati di un sensore (il WiFi e il microfono non sono riconosciuti da Android come sensori), perciò creano un AsyncTask, che ogni secondo registra un valore.

Appena viene avviato il Fragment principale **MainFragment**, si effettua un controllo per vedere se il Service è attivo oppure no. Viene notificato in una label lo stato del Service, e tramite due pulsati si può agire per fermarlo o attivarlo. Il tasto **Invia** invece effettua un controllo nella memoria interna, se ci sono file json salvati dai sensori, li legge uno per uno e li invia al server, sempre utilizzando RequestTask. Se il server accetta e inserisce i dati, allora viene poi cancellato questo file json.

Una classe importante che mette a disposizione delle utility è **SensorConfiguration**. Qui ci sono dei metodi statici accessibili da qualsiasi classe per effettuare diversi compiti:

- controllare lo stato del service
- cancellare un file json
- aggiungere/rimuovere codici del sensore nelle SharedPreferences

4.2.1 Riepilogo

Elenco delle classi implementate:

- Fragment:
  - AccountFragment
  - LoginFragment
  - MainFragment
  - SettingFragment

- util:
  - Geolocalizzazione
  - RequestTask
  - SensorConfiguration
  - SensorStart
  - WifiMeter
  - SoundMeter
- main:
  - DataSensorService
  - MainActivity

4.3 Client Web

Per analizzare i dati bisogna usufruire del client web. Presenta un interfaccia molto semplice, che è stata sviluppata in HTML e CSS, servendosi anche delle librerie di bootstrap per facilitare il lavoro. La ricerca viene eseguita mediante JavaScript, che si collega con il database tramite delle chiamate ajax a degli script in PHP.

Al caricamento della pagina vengono recuperati tutti i sensori disponibili, e inseriti dentro la prima select. Selezionando un elemento qualsiasi della select, vengono recuperati anche i dati per riempire le select del range, e vengono già impostati nel minimo e nel massimo. Premendo **Ricerca** vengono effettuate due chiamate ajax: la prima va a prendere i dati da inserire nella tabella di ricerca, l'altra effettua la ricerca vera e propria e inserisce tutti i dati nella mappa. Per semplicità, ad ogni ricerca viene associato un numero identificativo che si può utilizzare per operare in un secondo momento.

Quando arrivano i risultati di ricerca, vengono salvati in degli array globali, e sono suddivisi utilizzando l'id della ricerca, e il nome dell'utente. Tra le variabili globali ce ne sono alcune che necessitano di spiegazioni:

```
var searchForUser=[];
var userExclude = [];
var dataForChart =[];
var heatmapSearch = [];
```

Codice 4.10: Client Web - Variabili globali

La variabile searchForUser è un array associativo, che associa ad ogni id di ricerca un altro array associativo. Ci sarà quindi per la prima ricerca nella posizione 1 un array che contiene negli indici gli utenti che fanno parte della ricerca 1, e per ogni utente i dati che questo ha raccolto. Nell'elemento searchForUser[1]['NomeUtente'] ci sarà un heatmap oppure un'array di marker in base al tipo di ricerca eseguita. Quando viene eliminato un utente da una ricerca invece viene inserito nell'array userExclude nella posizione con l'id della ricerca. Ci sarà quindi in userExclude[1] un array con tutti i nomi degli utenti esclusi. Per generare i grafici i dati vengono salvati in dataForChart. Nella posizione dell'id della ricerca ci sarà un array che contiene nell'indice dell'utente un array di oggetti json, con i dati della ricerca di un singolo utente. Quando viene effettuata un ricerca di tipo heatmap viene aggiunto l'heatmap nell'array heatmapSearch, nella posizione dell'id della ricerca e nella posizione dell'utente.

```
searchForUser[indexReserch][user]= //contiene un heatmap o un
  array di marker
userExclude[indexReserch]= //array di stringhe con i nomi
  degli utenti
var dataForChart[indexReserch][user] = {
  "IDRead":"5",
  "IDSensor":"1",
  "Latitude":"44.100373",
  "Longitude":"12.3019628",
  "Data":"1",
  "user":"unknown",
  "Date":"2015-02-07",
  "Time":"12:01:02",
  "Type":"light",
```

```
13      "Unity":"lumen",
14      "gradientLow":"#76FF03",
15      "gradientMiddle":"#FFEB3B",
16      "gradientHeight":"#F44336"
17    }
18    var heatmapSearch[indexReserch][user] = //heatmap
```

Codice 4.11: Client Web - Esempi del contenuto delle variabili globali

La divisione dei risultati per utente, permette di evidenziare quali utenti hanno raccolto quei dati. Dentro alla tabella di ricerca infatti sotto la voce **Utenti** compare un bottone con il numero di utenti, che se premuto, fa comparire una finestra modale con la lista degli utenti, e la possibilità di rimuovere i dati condivisi da un utente. In questo caso i risultati di quell'utente vengono eliminati dalla mappa e la tabella viene aggiornata.

Per inserire i risultati nella mappa sono state fondamentali le librerie di google maps (v3) in JavaScript. E' stato previsto fin da subito la possibilità di ricercare con i marker, la richiesta ajax infatti restituisce i dati che contengono le coordinate, oltre agli altri dati della lettura, quindi si utilizzano queste latitudini e longitudini ottenute per inserire i marker nella mappa. Si genera però un problema di leggibilità, perché i marker sono molti e posizionati tutti in zone molto vicine, per cui si è utilizzata un'altra libreria complementare a google maps, **MarkerClusterer**, che permette di raggruppare i marker in gruppi, e distinguerli soltanto zoomando in un preciso punto. Un secondo tipo di ricerca è stato quello del DataLayer, questo in base alle coordinate passate creava una regione all'interno della mappa colorata uniformemente. Questo tipo di ricerca non era molto accurato, perché creava una regione in cui non era distinguibile quali erano i valori raccolti. Per questo motivo si è deciso di abbandonare il DataLayer e di utilizzare invece l'HeatmapLayer. L'HeatmapLayer permette di creare zone intorno alle coordinate, in cui in base ai valori passati, crea zone colorate da gradienti diversi in base a quanto è alto il valore registrato in quel punto. Con questo meccanismo ci si rende conto subito ad occhio dove i valori sono più alti e dove sono più bassi. I

gradienti di default vanno dal verde per le zone con valori più bassi, al giallo per le zone con valori medi, al rosso per le zone con valori più alti, ma per ogni ricerca vengono caricati al posto di questi tre colori altre colorazioni salvate nel database, in modo da distinguere le diverse ricerca nella mappa. Per ogni ricerca ci sarà una legenda che mostra il gradiente.

Nella mappa sono stati inseriti dei controlli supplementari per rendere più piacevole la navigazione, google maps infatti mette a disposizione dei metodi per aggiungere controlli personalizzati all'interno della mappa, e diversi eventi per gestirne il comportamento. Il primo controllo è quello di ricerca di un indirizzo, infatti da qui la mappa viene centrata nelle coordinate dell'indirizzo ricercato. È utile quando si vuole cambiare posto, ed evita di doverlo cercare ad occhio. Quando viene cliccato, cerca le coordinate dell'indirizzo scritto attraverso dei metodi messi a disposizione da google maps, e poi imposta il centro della mappa in quelle coordinate. Altri controlli sono quelli che modificano l'heatmap, ci sono infatti due controlli che modificano rispettivamente:

- il radius dell'heatmap, che si può ingrandire o diminuire,
- l'opacità dell'heatmap, che si può rendere più trasparente o meno

Quando vengono selezionati vanno a prendere tutti gli heatmap inseriti, e ne modificano l'aspetto cambiando vari parametri. L'ultimo controllo aggiunto, e forse il più importante, è **Area**, questo controllo se selezionato fa comparire un rettangolo nella mappa, tramite il quale l'utente può selezionare un'area che delimita i risultati di ricerca. Quando viene effettuata una ricerca infatti vengono esclusi i punti che sono al di fuori dell'area selezionata. Se si vuole disabilitare questa funzione basta cliccare nuovamente il pulsante e l'area sparisce. Questa funzione è stata possibile implementarla grazie alla classe **Rectangle** della API di google maps, che permette di inserire un rettangolo e di renderla editabile o meno.

È possibile anche scaricare i dati che compongono la ricerca, infatti è presente un tasto **Download**, che scarica i dati in formato csv. Dalla tabella

delle ricerche è si trova anche un pulsante con l'icona dei grafici, premendo questo si nasconde il contenuto della pagina e compaiono tre grafici. Per generare il grafico vengono passati i dati già filtrati, contando anche eventuali esclusioni degli utenti. Per realizzare i grafici ho utilizzato la libreria *google chart*. Il primo grafico implementato è un istogramma, creato con il package **corechart** di google chart. Per realizzarlo bisogna passare un array contenente i valori della ricerca, e viene rapportato ogni valore con il numero di rilevamenti per quel valore. Il secondo grafico invece è un grafico temporale, che rappresenta il numero di rilevamenti nel tempo, creato con il package **line** di google chart. Per realizzarlo ho creato un contatore per ogni data, che viene impostato in base a quanti rilevamenti sono stati fatti quel giorno. L'ultimo grafico è quello della CDF, questo grafico è stato il più complesso da rappresentare, in quanto in google chart non è prevista una funzione per disegnare la CDF. Per farlo quindi ho trovato una formula, che ho poi implementato in JavaScript per calcolare i punti della CDF avendo come input i punti della x (i valori raccolti). Per questo calcolo però ho dovuto ordinare in maniera crescente i dati della x, dal più piccolo al più grande, altrimenti il calcolo sarebbe risultato errato. Una volta ottenuto i punti delle ascisse e delle ordinate ho utilizzato sempre google chart con il package line per rappresentare il grafico. Questo è il codice che calcola la formula utilizzata per il calcolo dei dati:

```
1  var dataLength = myData.length;
2    for (var i = 0; i < dataLength; i++) {
3        myData[i].x =  +myData[i].x;
4        myData[i].y = +((i+1)/dataLength);
5    };
```

Codice 4.12: Calcolo CDF

Sotto ad ogni grafico è presente un pulsante che se premuto visualizza i punti precisi che compongono il grafico.

Le librerie di google sono state utilissime per sviluppare questo client web, in particolare google maps e google chart hanno permesso di realizzare tutto

con estrema facilità.

Capitolo 5

Valutazioni e Miglioramenti

Per valutare l'efficacia di questo sistema sono state necessari diversi test. Sin dall'inizio ho cominciato a raccogliere i dati nei dintorni della zona in cui abito, per poi estendermi a raccogliere fino al centro di Bologna. Una parte di dati sono stati raccolti anche da altre persone, in questo modo abbiamo potuto testare la funzione di crowdsensing vera e propria. Nel crowdsensing infatti i dati sono raccolti da molteplici utenti, mentre fino ad ora avevo raccolto dati solo io. In fase di test ho potuto constatare che i sensori raccoglievano bene i dati, poiché venivano gestiti tramite le API Android e aggiornati ogni volta che cambiava il valore, mentre per il microfono e il WiFi, che erano stati gestiti a parte, era stato inserito un intervallo di dieci secondi tra un rilevamento e l'altro, che risultava essere errato. Questo intervallo infatti era troppo alto, in quanto dieci secondi di distanza creavano una raccolta in cui vi erano dei buchi tra un rilevamento e l'altro. Per questo è stato cambiato l'intervallo ad un solo secondo, in questo modo la raccolta dati è stata molto più affidabile, e i campioni raccolti erano molto più numerosi. Dal client web, è stata inserita la funzionalità di visualizzare quali utenti avevano raccolto i dati, ed eventualmente escludere degli utenti dalla ricerca. Questa funzionalità è stata possibile testarla solo quando nel sistema erano stati raccolti dati da più di un utente.

In fase di test si è verificato anche l'incorretta visualizzazione del Data-

Layer. Con poche misurazioni infatti veniva fuori un'area tutto sommato abbastanza vicina alla raccolta dati, ma quando il campione si allargava venivano fuori zone enormi e prive di significato. Questo è uno dei motivi che hanno portato all'abbandono del DataLayer a favore dell'HeatmapLayer.

5.1 Risultati

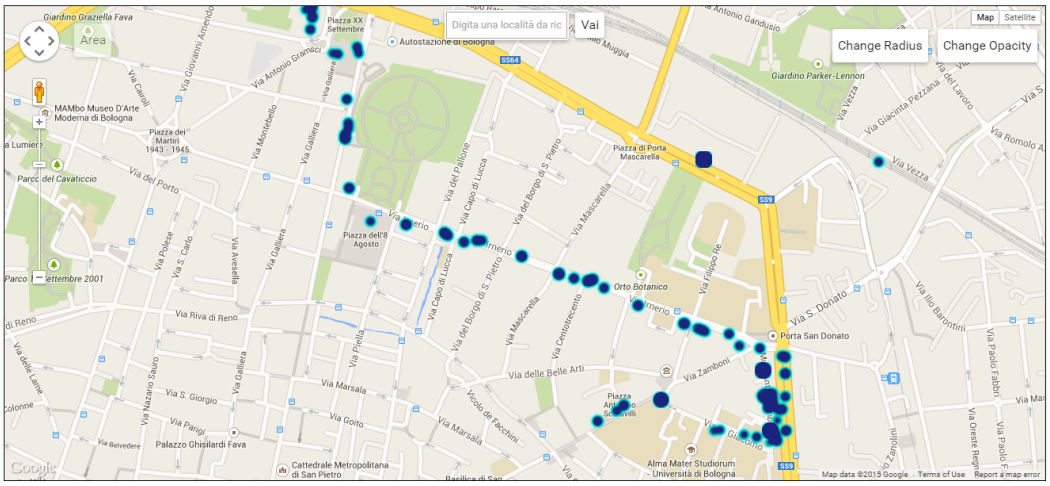

Figura 5.1: Analisi del Sound Level

Nella figura si può notare il livello di sound level nel tragitto tra la stazione di Bologna e il dipartimento di scienze. Di per sé i valori sembrano avere una colorazione più o meno uguale, e da questa distanza non riusciamo a distinguere bene le differenze di valori. Per un'analisi più accurata è necessario zoomare in un punto. Le seguenti immagini illustrano la differenza del sound level presa in due punti, uno è la strada del dipartimento di scienze, strada chiusa e poco trafficata, l'altro punto è la via Irnerio, strada principale e più trafficata. Nei pressi del dipartimento, seppure c'è una concentrazione alta di rilevamenti, questi risultano tutti con un raggio molto piccolo, ciò è dovuto al fatto che il livello di rumorosità in quella zona non era molto elevata. Se invece analizziamo dei punti presi dalla via Irnerio, notiamo che anche

se i punti sono meno concentrati, e abbiamo molta distanza tra un punto e l'altro, il raggio dell'area appare più grande. Questo sta a dimostrare che la via Irnerio, in quanto più frequentata ha un livello di rumorosità più alto.

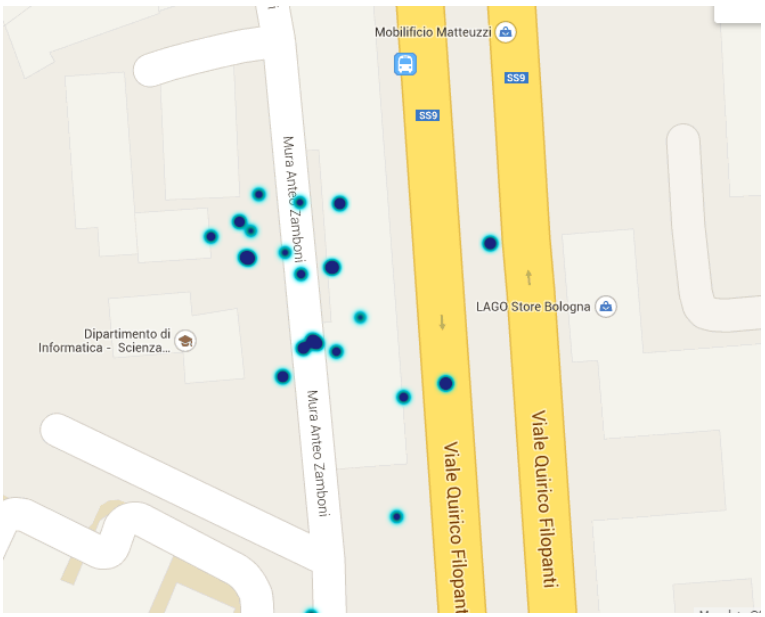

Figura 5.2: Analisi del Sound Level nei pressi del Dipartimento di Scienze

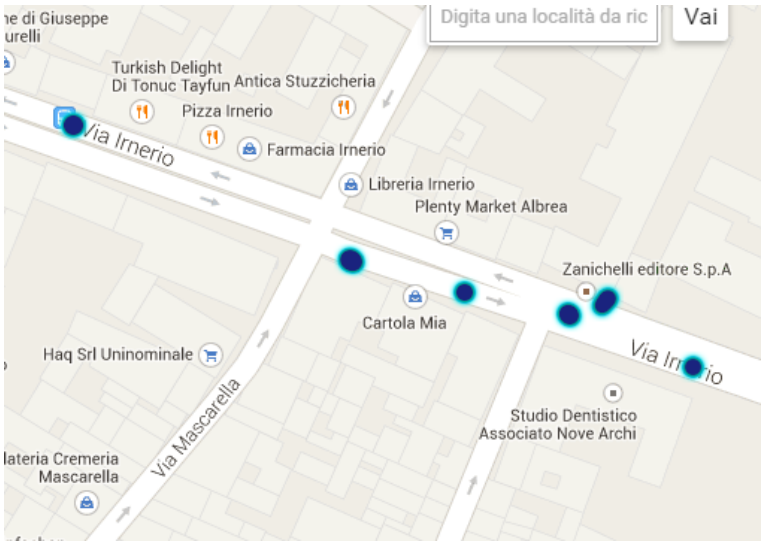

Figura 5.3: Analisi del Sound Level nei pressi della Via Irnerio

Come d'altronde ci aspettavamo, la mappa ci ha dimostrato quale delle due zone è la più rumorosa.

Dalle valutazioni si evince anche che con la visualizzazione tramite marker, ad occhio non si percepisce una lettura chiara. Sono utili infatti per capire dove il dato è stato preso, ma è necessario cliccare sul marker per capire che valore è stato raccolto in quel punto.

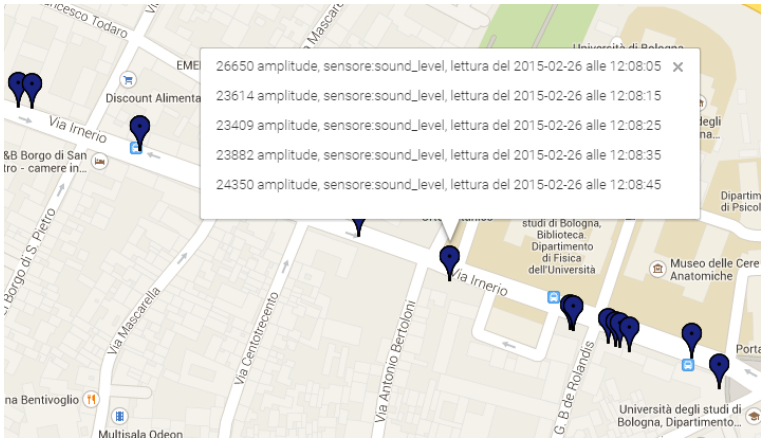

Figura 5.4: Analisi del Sound Level nei pressi della Via Irnerio tramite Marker

Rimane comunque una visualizzazione interessante, ma il marker diventa forse ancora più interessante se andiamo a sovrapporre i due tipi di ricerche, quella del marker e quella dell'heatmap. Così facendo possiamo vedere tramite l'heatmap la solita visualizzazione, ma cliccando sul marker avremo l'informazione specifica sul dato preciso raccolto in quel punto. Inoltre il marker a volte può essere ambiguo, mentre in questo caso viene disegnata un'area intorno al marker che ci fa capire meglio la sua posizione.

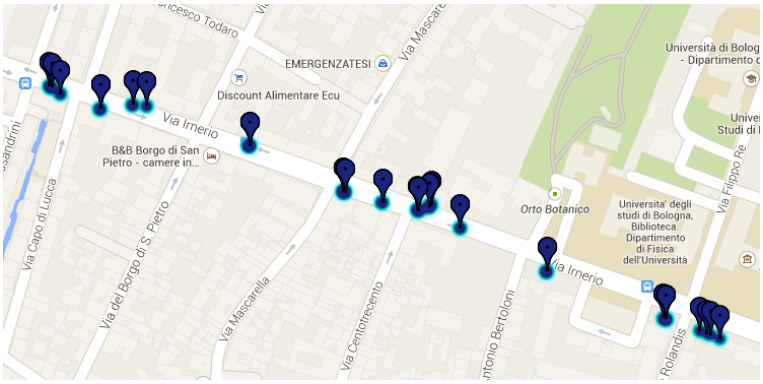

Figura 5.5: Analisi del Sound Level combinata nei pressi della Via Irnerio

Un altro aspetto interessante dell'analisi sono i grafici. Dai grafici possiamo notare meglio certe informazioni sui dati, per esempio possiamo notare la concentrazione delle raccolte nel tempo oppure il numero maggiore di rilevamenti per ogni valore. Nella Figura 5.6 per esempio possiamo notare che la maggior parte dei dati raccolti hanno tutti valori bassi.

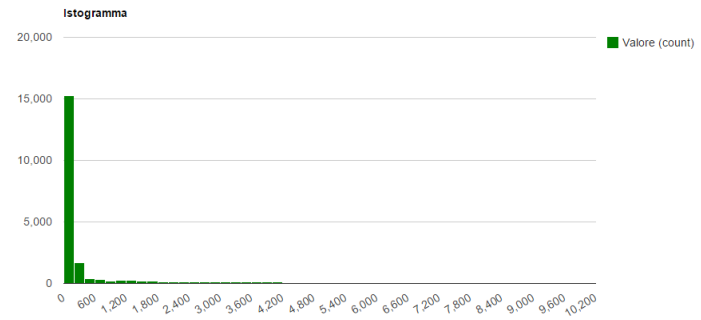

Figura 5.6: Analisi del Sound Level - Grafico Istogramma

Se guardiamo invece i rilevamenti nel tempo possiamo notare che la maggior parte dei rilevamenti sono stati fatti in un giorno unico, mentre negli altri giorni sono meno della metà. In base a questo si possono evidenziare dati più o meno affidabili, infatti se i dati appartengo più che altro ad un unico giorno non possiamo sapere se anche gli altri giorno i livelli di rumorosità siano simili.

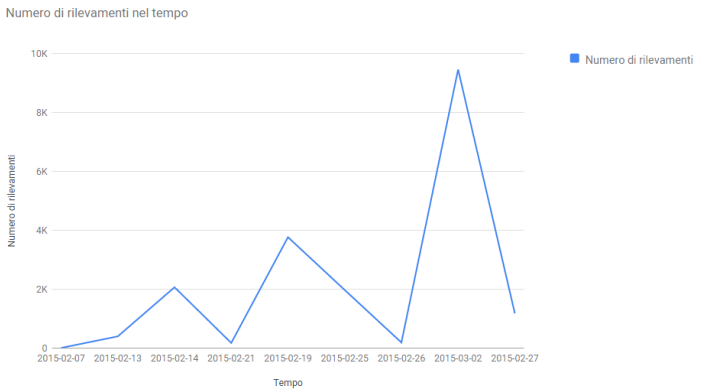

Figura 5.7: Analisi del Sound Level - Grafico Temporale

5.2 Sviluppi Futuri

Ci sono diversi aspetti che non sono stati presi in considerazione in questo progetto, ma che potrebbero ampliare questo sistema per generare una piattaforma più completa.[14, Medusa] [15, UMMC]

5.2.1 Affidabilità dei Dati

In questo sistema i dati sono stati raccolti solo da due utenti, che sapevano quello che stavano facendo. Ma nel momento in cui questa applicazione viene distribuita e diversi utenti diventano responsabili per la raccolta dati, possono emergere alcuni problemi. Non si può infatti avere la certezza sull'affidabilità dei dati. Un individuo per esempio potrebbe camminare con il telefono nella borsa, in quel caso la luminosità sarebbe quasi nulla, e invierebbe dati errati al server. In un altro caso l'utente potrebbe girare per la città in automobile con il volume della radio alto, e in ogni posto che visita registra un livello di volume molto più alto rispetto alla realtà. Questo è un problema di tutte le piattaforme si crowdsensing, infatti la raccolta di dati diventa poco difficoltosa in quanto distribuita tra gli utenti, ma più inaffidabile, poiché non possiamo avere certezze sui dati raccolti. Uno degli sviluppi futuri quindi è quello di cercare di escludere i dati inaffidabili, o comunque di analizzare i dati per capire se possono essere più o meno precisi. Per esempio una soluzione è quella di implementare nell'applicazione Android un filtro per i dati che vengono inviati al server. Quindi l'utente prima di inviare i dati, sapendo che ha tenuto il telefono in tasca durante l'ultima ora, può scegliere di inviare tutti i dati tranne quelli di luminosità raccolti durante l'ultima ora. La responsabilità è sempre dell'utente, ma in questo caso può esercitare un controllo maggiore sui dati che condivide.

5.2.2 Informazioni aggiuntive alle letture

Nel database si potrebbero aggiungere informazioni riguardanti non solo il sensore, ma anche quale modello specifico di sensore ha raccolto questi

dati e su quale dispositivo è stata effettuata la raccolta. Può essere utile in quanto ogni sensore ha una capacità diversa, e avere il modello preciso può aiutare ad individuare i limiti massimi e minimi che quel sensore può rilevare. Con queste informazioni quindi possiamo analizzare meglio i dati e renderci conto se sono più affidabili. In più possiamo avere anche una stima di quali dispositivi hanno raccolto i dati, quindi sappiamo se sono smartphone piuttosto che Arduino o Raspberry PI, e più nello specifico quale modello di dispositivo lo ha rilevato.

5.2.3 Crowdsensing sociale

Un altro aspetto ancora da implementare è quello dell'analisi di dati non relativi ai fenomeni ambientali ma riferiti agli aspetti sociali. I dati che vengono inviati infatti sono collegati ad un account. Tramite questo quindi si possono implementare delle ricerche anche in base ai criteri sulla persona. Per esempio voglio cercare i posti più frequentati dai quindicenni il sabato sera. Oppure vedere quante persone erano nei pressi di un cinema in un determinato orario il giorno in cui usciva un nuovo film. Il client web quindi si può ampliare in modo da effettuare ricerche con altri parametri oltre a quelli già esistenti. Eventualmente si possono aggiungere ulteriori informazioni legate all'account, e sviluppare nuovi tipi di parametri di ricerca per effettuare ricerche incrociate, o ricerche specifiche solo su questi aspetti.

5.2.4 Raccolta dai sensori non registrati

In questo sistema, quando l'utente seleziona un sensore dall'applicazione Android, viene effettuata una chiamata che lo informa se il sensore è presente nel database o meno. Questo vuol dire che se il sensore non è registrato l'utente non lo può utilizzare per raccogliere i dati. Questo meccanismo può essere un po' limitante, perché un individuo deve comunicare all'amministratore del sistema di voler utilizzare un determinato sensore, e questo lo deve aggiungere al database. Fino a quel momento non può essere utilizza-

to. Questo procedimento si potrebbe automatizzare, in modo che l'utente possa comunque utilizzare quel sensore fin da subito, anche se non è registrato. I dati potrebbero essere salvati in una tabella di letture temporanea e vengono registrati con un sensore temporaneo. Nello stesso momento viene notificato all'amministratore che vi è un nuovo sensore da registrare, e quando lo aggiunge vengono trasferiti i dati dalla tabella temporanea alla tabella effettiva collegati al sensore appena aggiunto. Questo procedimento permette all'utente di non dover aspettare tempi lunghi e di utilizzare subito l'applicazione.

5.2.5 Ampliare i Dispositivi

Fino adesso ci si serve di un'unica applicazione Android per la raccolta dei dati, ma sarebbe molto pi ù funzionale avere diversi dispositivi, anche diversi da uno smartphone in grado di eseguire questa funzione. Si potrebbero creare script per Arduino o Raspberry PI, in modo che un'utente può utilizzare questi device e piazzarli in giro per la città. Espandendo le funzioni di raccolta in una più vasta gamma di dispositivi si riesce ad ottenere una base di dati più completa, inoltre si espandono anche i tipi di sensori disponibili. Si riescono ad utilizzare anche dei sensori che sugli smartphone non sono presenti.

Capitolo 6

Conclusioni

La piattaforma che si è presentata, è un sistema di crowdsensing, in grado di gestire dati di ogni genere. Grazie a questo paradigma di raccolta dati, si riesce a costituire una buona base base di dati senza molti sforzi. La raccolta viene condivisa tra i diversi utenti, che senza fatica, devono solo abilitare un'applicazione e questa registra i dati. La potenzialità di questo sistema sta nel fatto che l'utente può configurare l'applicazione per raccogliere i dati che desidera a suo piacimento. Si possono utilizzare quindi tutti i sensori esistenti, e anche quelli che usciranno in futuro.

La raccolta dati viene eseguita da un'applicazione Android, e può essere configurata utilizzando tutti i sensori installati nel dispositivo. Tramite questa poi si avvia e ferma la raccolta, per poi inviare tutto al server.

Il server è stato scritto in node.js, e si connette con un database MySQL dove risiedono i dati. Il server si occupa di riceve le richieste e di operare nel database per leggere o scrivere i dati in base alla richiesta opportuna.

L'analisi dei dati invece viene effettuata da un client web, che ricerca i dati nel database in base ai parametri scelti dall'utente. I risultati vengono illustrati all'interno di una mappa sotto forma di Heatmap oppure di Marker, e oltre a questi sono visualizzabili anche tre grafici.

Come è stato già detto, non esisteva finora una piattaforma che raccogliesse tutti i tipi di dati da diversi tipi di sensori, ma i sistemi creati fino

adesso sono impostati per funzionare solo per determinati fenomeni. In questo progetto invece si è riusciti a creare una piattaforma che contenga dati diversi, per farlo è stato necessario dividere la tabella delle lettura dalla tabella dei sensori. In questo modo alla lettura viene associato un identificativo del sensore, tramite il quale si ottengono le informazioni di lettura. L'idea innovativa che differenzia questo progetto dalle altre piattaforme di crowdsensing, è quindi quella di concentrare in un unico posto i dati che si riferiscono a fenomeni diversi. Inoltre con qualche miglioramento, si possono includere anche dati di crowdsensig sociale oltre a quelli di natura ambientale.

Il client web è disponibile alla pagina: http://simone.masini3.web.cs.unibo.it mentre l'apk dell'applicazione Android si può scaricare da questo indirizzo: https://www.mediafire.com/folder/v3cz3721v6aq5/Crowdsensing_Client_Android

Bibliografia

[1] http://resources.infosecinstitute.com/crowdsensing-state-art-privacy-aspects

[2] http://www.comsoc.org/files/Publications/Magazines/ci/cfp/cfpcommag0814.html

[3] http://www.html.it/pag/32814/introduzione-a-nodejs

[4] http://whatis.techtarget.com/definition/Nodejs

[5] https://www.npmjs.com/package/mysql

[6] http://expressjs.com/

[7] https://www.npmjs.com/package/express

[8] https://www.npmjs.com/package/body-parser

[9] API Android

[10] https://developers.google.com/maps/documentation/javascript/tutorial

[11] https://google-developers.appspot.com/chart/interactive/docs/quick_start

[12] http://getbootstrap.com/getting-started

[13] Yohan Chon, Nicholas D. Lane, Fan Li, Hojung Cha, and Feng Zhao. 2012.
Automatically characterizing places with opportunistic crowdsensing using smartphones.

ACM, New York, NY, USA, 481-490. DOI=10.1145/2370216.2370288 http://doi.acm.org/10.1145/2370216.2370288

[14] Moo-Ryong Ra, Bin Liu, Tom F. La Porta, and Ramesh Govindan. 2012. Medusa: a programming framework for crowd-sensing applications. ACM, New York, NY, USA, 337-350. DOI=10.1145/2307636.2307668 http://doi.acm.org/10.1145/2307636.2307668

[15] Wanita Sherchan, Prem P. Jayaraman, Shonali Krishnaswamy, Arkady Zaslavsky, Seng Loke, and Abhijat Sinha. 2012. Using On-the-Move Mining for Mobile Crowdsensing. IEEE Computer Society, Washington, DC, USA, 115-124. DOI=10.1109/MDM.2012.58 http://dx.doi.org/10.1109/MDM.2012.58

[16] Ganti, R.K.; Fan Ye; Hui Lei, Mobile crowdsensing: current state and future challenges, Communications Magazine, IEEE , vol.49, no.11, pp.32,39, November 2011

[17] Bin Guo; Zhiwen Yu; Xingshe Zhou; Daqing Zhang, From participatory sensing to Mobile Crowd Sensing, Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on , vol., no., pp.593,598, 24-28 March 2014 doi: 10.1109/PerComW.2014.6815273

[18] Huadong Ma; Dong Zhao; Peiyan Yuan, Opportunities in mobile crowd sensing, Communications Magazine, IEEE , vol.52, no.8, pp.29,35, Aug. 2014

[19] Bedogni, L.; Di Felice, M.; Bononi, L., By train or by car? Detecting the user's motion type through smartphone sensors data, Wireless Days (WD), 2012 IFIP

Ringraziamenti

I ringraziamenti maggiori vanno alla mia famiglia, i miei genitori Sandro ed Enza per avermi dato la possibilità di intraprendere questo percorso grazie al quale sono diventato la persona che sono oggi, e mia sorella Monica che in tutti questi anni è sempre stata al mio fianco a sostenermi. Ringrazio Melissa, per avermi sempre supportato e incitato a raggiungere questo traguardo, tutti gli amici che mi sono sempre stati vicino, e i compagni di corso che hanno condiviso con me gli ostacoli che ci ha posto davanti questo percorso e le abbiamo superate insieme. Ringrazio tutti i professori incontrati in questi tre anni che mi hanno trasmesso passione e conoscenze, infine ringrazio il professor Bononi per essere stato relatore di questa tesi, ma soprattutto ringrazio Luca Bedogni, per l'idea, il sostegno e tutto l'aiuto ricevuto durante questo ultimo passo della mia carriera accademica.