Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Triennale in Informatica

Progettazione e integrazione di un sistema di Mobile Crowdsensing con una piattaforma IoT orientata ai servizi

Relatore:
Dr.
Federico Montori

Presentata da:
Guillaume Franzoni Darnois

Sessione I
Anno Accademico 2019/2020

Sommario

In questi ultimi anni l'uso di dispositivi intelligenti per qualsiasi tipo di attività è aumentato, si sfruttano ad esempio gli smartwatch per ottenere utili informazioni riguardanti il proprio battito cardiaco e monitorare le calorie consumate in base ai passi effettuati. Allo stesso tempo questi dispositivi sono capaci di connettersi fra di loro comunicando dati e dandoci l'opportunità di controllare da remoto determinati comportamenti. Tutti questi dati che viaggiano da un dispositivo all'altro sono estremamente utili e possono aiutare a comprendere meglio la realtà che ci circonda. Specialmente in ambito di monitoraggio ambientale tali dispositivi possono essere estremamente d'aiuto per catturare caratteristiche della realtà tramite i sensori di cui sono sempre più muniti in modo completo. Contemporaneamente tali informazioni che si muovo sulla rete trasportano dati sensibili ed è necessario assicurare che tali dati possano essere condivisi in tranquillità da parte degli utenti.
Per questo motivo il progetto di questa tesi verte in modo particolare sull'implementazione di un'architettura di Mobile CrowdSensing che sia capace di rendere disponibili ad una piattaforma di IoT collaborativo i dati misurati degli utenti, non solo assicurando loro la tutela della privacy ma fornendo un protocollo sicuro di scambio di dati pensato per permettere un meccanismo di reward, atto a incentivare alla partecipazione nelle campagne proposte dalla piattaforma.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni il numero di strumenti intelligenti di qualsiasi tipo è aumentato, dai contatori "smart" alle telecamere IP, nel contesto delle Smart Home, dai parcheggi "smart" alle stazioni di monitoraggio ambientale, legate invece al concetto di Smart Cities, e tanti di più sono i casi in cui oggetti quotidiani incominciano ad assumere caratteristiche smart, acquisiscono capacità di connessione alla rete, diventando dunque attivabili o monitorabili da remoto, ed in futuro sempre più saranno gli oggetti che si saranno trasformati per accomodare le nostre nuove abitudini. Questo sta già avendo una grande influenza sulla nostra realtà sotto diversi aspetti, facilitando alcuni compiti e migliorando la qualità della vita di moltissime persone. L'impatto dunque che questo mondo, quello dell'Internet of Things, sta avendo sulla società è innegabile.
Allo stesso tempo però una tale innovazione porta con sé diversi aspetti che ancora necessitano di migliorie e standardizzazione. Questo è infatti l'aspetto su cui questa tesi cerca di concentrarsi, partendo quindi da uno dei problemi individuati nel mondo dell'IoT, ovvero la diffusione di INTRAnet of Things, si è deciso inizialmente di partire da una piattaforma, SenSquare, che già aveva proposto una soluzione efficace ed ispirata al concetto di IoT collaborativo per poi espandere una delle sue sezioni andando a proporre un'architettura funzionante, seppur embrionale, di

Mobile CrowdSensing orientata ad aumentare il numero di misurazioni e dati disponibili su tale sistema mantenendo la privacy ed incentivando alla partecipazione da parte degli utenti.
Questa tesi quindi si articolerà in cinque capitoli oltre a quello corrente. Innanzitutto nel Capitolo 2 verrà descritta la situazione attuale esponendo quali sono le caratteristiche e quali i problemi principali fino ad ora individuati nel campo dell'Internet of Things. In seguito, all'interno del Capitolo 3 si procederà a dare una visione dall'alto di quella che è l'architettura del progetto che è stato utilizzato come base di partenza, ovvero SenSquare. Dopodiché due capitoli, Capitolo 4 e Capitolo 5, saranno dedicati alla spiegazione tecnica e implementativa rispettivamente del progetto lato server e dell'applicazione mobile. Infine il Capitolo 6, contenente le conclusioni, chiuderà il documento riassumendo e proponendo i possibili lavori futuri.

Capitolo 2

Stato dell'Arte

2.1 IoT

L'Internet delle Cose (in inglese, Internet of Things) può essere definito come un'estensione dell'internet atta a mettere in comunicazione dispositivi di calcolo dotati di sensori e della capacità di comunicare le misurazioni e le osservazioni effettuate in un ambiente, che può variare da quello locale ad uno globale, con lo scopo di fornire un insieme di servizi ad aziende e/o privati. È facilmente intuibile quindi il come ed il perché un concetto simile si sia sviluppato e continui a svilupparsi ad una velocità impressionante, le utilità possono variare dalla creazione di un Home Automation System, molto localizzato e di dimensione tutto sommato ridotte, e scalare fino ad interessante enti internazionali che basano i loro servizi su una raccolta dati di dimensioni globali.
La diversità delle funzioni che questo paradigma riesce ad offrire è enorme, un caso però particolarmente interessante è quello dell'ambito urbano all'interno del quale si cerca di sfruttare tale concetto per migliorare i servizi forniti ai cittadini rendendo più efficace l'uso delle risorse pubbliche attraverso l'ottimizzazione dei costi derivati dalla pubblica amministrazione ponendo le basi per il concetto di Smart City [1]. Altri esempi importanti sono, ad esempio, il caso delle Smart Home, dove inizialmente la tendenza

è stata quella di sfruttare dispositivi intelligenti e connessi per migliorare la qualità di vita delle persone controllando da remoto elementi presenti all'interno delle abitazioni, automatizzando compiti eseguiti quotidianamente, aumentando la sicurezza. Inoltre l'avvento delle Smart Home ha portato a progettare sistemi capaci di sfruttare la tecnologia già presente e sempre più a basso costo per rendere più facili le comuni azioni di gestione di elettrodomestici a persone disabili tramite l'uso di comandi vocali [2].
Si parla però di un sistema piuttosto giovane e che quindi presenta diverse difficoltà, infatti la ricezione di una grande mole di dati proveniente da fonti diverse fra loro genera insiemi eterogenei di informazioni che, al crescere di dimensione e varietà, sono incredibilmente difficili da gestire. Questo implica il sorgere di diverse "isole" dove, più che un Internet of Things, è possibile individuare una sorta di Intranet of Things [3]. Le conseguenze di questa chiusura (necessaria per semplificare la gestione dei dati e delle connessioni) sono ovviamente una minor partecipazione da parte di individui singoli e una minor scelta in termini di servizi offerti, limitati dalla minor quantità di risorse a disposizione [4]. Tale problema è frutto della mancanza di uno standard, mancanza dovuta principalmente ad una difficoltà da parte della ricerca e degli sforzi compiuti con l'obiettivo di creare protocolli efficienti di tenere il passo con il quale questo mondo si evolve. Questo, come quindi già accennato, porta alla creazione di sistemi chiusi che devono essere creati in brevi tempi e senza dare peso al concetto di interoperabilità che invece è fondamentale per l'IoT. Questo è evidente anche nei casi apparentemente più semplici come quello delle Smart Home, dove, nonostante si potrebbe essere portati e pensarlo come un ambiente facilmente controllabile, ci si accorge che la varietà dei dispositivi e delle diverse tecnologie da essi usate, anche solo per la comunicazione, è inaspettatamente grande. Sforzi in questo senso sono infatti stati compiuti per tentare di proporre protocolli che permettano la comunicazione fra dispositivi che usano sistemi di comunicazione diversi, uno di questi è ad esempio RouteX [5].

Più in generale la necessità di avere sistemi anche grandi ma con una buona capacità di collaborazione è nato il concetto di IoT Collaborativo (C-IoT), pensato come un paradigma orientato alla comunicazione e all'interoperabilità fra umani, imprese ed entità governative con l'obiettivo quindi di sopperire alla mancanza portata dai sistemi così sviluppati in senso verticale ma molto meno in orizzontale. Insieme ad esso poi si è sviluppato anche il concetto di Community-Based Monitoring che si divide in "Consultative", nel caso in cui i cittadini non siano resi partecipi dei risultati ottenuti dalla raccolta dati a cui hanno partecipato, e "Collaborative", in cui i cittadini sono nuovamente la risorsa primaria di raccolta dati ma hanno accesso ai risultati e possono decidere in che direzione muovere gli sforzi futuri relativi alla raccolta dati [6]. Questo nuovo concetto si è poi evoluto nell'interessante forma del MCS (Mobile Crowdsensing) come vedremo nella prossima sezione.
Una nota importante è quella riguardante la sicurezza in ambiente IoT, infatti, mentre l'aspetto privacy sarà trattato nel prossimo paragrafo in merito allo scambio di dati all'interno di un sistema di Crowdsensing, vi sono diversi problemi legati ad essa che è importante conoscere. In un sistema IoT infatti, come abbiamo già detto, tutto si basa su un insieme di dispositivi interconnessi che si scambiano informazioni riguardanti l'ambiente in cui sono stati inseriti, questo implica però che la sicurezza del sistema non possa essere più sicura dei dispositivi stessi o dei protocolli di comunicazione fra i dispositivi (soggetti dunque ai più classici attacchi quali DoS, Spoofing, etc.). Inoltre trattandosi di oggetti concreti non bisogna escludere danni non solo di tipo informatico ma anche concreti ai sensori che possono di conseguenza compromettere i servizi forniti da quel sistema [7].
Infine, un'altra possibile soluzione alla chiusura dei sistemi IoT, di notevole importanza, è quella degli Open Data che in realtà per determinati aspetti si avvicina al concetto del Community-Based Monitoring, infatti l'idea è quella di creare una repository di dati pubblici che possono essere

usati e ri-usati da tutti senza limiti, se non occasionalmente il dovere di attribuzione di tali dati. Questo permette quindi di liberarsi della chiusura dei sistemi di IoT aprendo quindi una scelta molto più ampia agli utenti che non hanno così più limitazioni dettate dall'uso di dati proprietari.

2.2 Crowdsensing

Come già accennato nella sezione precedente, a partire dalla necessità di un sistema più diffuso e capace di gestire un insieme di dati molto eterogenei, nasce il concetto di Community-Based Monitoring una cui estensione molto importante ed interessante è il Crowdsensing, al quale più spesso ci si riferisce con il nome di Mobile Crowdsensing. Si intende dunque un paradigma che basa la raccolta di dati non su un insieme di infrastrutture estremamente costose da costruire e mantenere, ma su qualcosa che è invece già presente nella vita quotidiana dei cittadini, ovvero tutti quei dispositivi, quali smartphone, tablets o wearables, che possiedono al proprio interno un set di sensori e sono capaci di comunicare i valori raccolti tramite una connessione (mobile o locale). Di conseguenza i costi e la manutenzione richiesti per questo tipo di sistemi si riducono banalmente alla produzione e al mantenimento di un'interfaccia attraverso la quale gli utenti finali possano, spontaneamente o non, condividere i dati utili a fornire servizi dei quali poi possono o meno beneficiare.
A seconda, quindi, di come questa raccolta dati avviene è stato possibile individuare due principali tipi di Mobile Crowdsensing: il primo vede la raccolta dati come uno sforzo attivo da parte dell'utente che attraverso l'uso di un'interfaccia comunica osservazioni sull'ambiente circostante (Participatory Crowdsensing, PCS), il secondo invece è considerato l'esatto opposto, l'utente non ha quindi un ruolo attivo nella comunicazione dei dati ma viene sfruttata un'applicazione che, venendo eseguita in background, si occupa di raccogliere i dati e comunicarli in modo automatico (Opportunistic Crowdsensing, OCS). Questo nuovo approccio però implica una

nuova difficoltà, derivante della mancanza di rigorosità nell'assunzione dei dati, in quanto la raccolta viene affidata agli utenti finali ed ai loro dispositivi, che possono essere estremamente diversi. Ciò può causare, per tale motivo, un problema già conosciuto nel campo dei Big Data come Curse of Data (in questo caso si parla però di Curse of Sensing), ovvero la possibilità che si presentino due situazioni non desiderate: quella della presenza di Sparse Data da un lato o di Dense Data dall'altro. Nel primo caso ci si riferisce ad una scarsa quantità di dati che impedisce il raggiungimento dell'obiettivo prefissato, nel secondo invece si tratta di della situazione opposta, dove una mole di dati troppo grande, con una finissima granularità nell'acquisizione di osservazioni può portare ad ottenere uno spreco di risorse computazionali e ad un'analisi generale fuorviante, inoltre il grande numero di dati implica anche un grande numero di premi da riconoscere ai partecipanti [8].
L'altro aspetto fondamentale è infatti il concetto di premio. É necessario, qual ora il risultato della raccolta di dati non sia di interesse diretto dell'utente, dare una ragione sufficientemente valida che lo spinga a voler partecipare alla campagna. Spesso questo consiste nell'assegnazione di premi in denaro, come buoni o pagamenti, che rappresentano anche il modo più immediato di incentivare l'utente finale alla condivisione dei propri dati [9]. Allo stesso tempo però il premio richiede una prova che tali informazioni siano stato inviate alla piattaforma e questo implica che l'utente comunichi alla piattaforma quali misurazioni egli ha condiviso. Questo meccanismo evidentemente però va a minare la privacy in quanto le informazioni richieste spesso prevedono la presenza di dati riguardanti la posizione geografica e la data. Tali informazioni possono, a lungo andare, rivelare routine e dettagli personali di un utente ed è per questo fondamentale che nella creazione di una campagna di crowdsensing venga dato il giusto peso a questo aspetto, in primis per garantire sicurezza ai propri utenti e poi per attrarre più individui altrimenti non disposti a condividere i propri dati.

Per risolvere questo problema diverse soluzioni sono state pensate, tra le quali una si propone di minimizzare la quantità di informazioni cercando di scegliere un sottoinsieme di misurazioni fra di loro il meno correlate possibili sia in senso geografico che temporale [10]. E' importante però sottolineare l'esistenza di un altro tipo di raccolta dati, in parte simile a quello già esposto, si tratta del paradigma Open Data, basato sull'utilizzo di dati presenti su repository liberamente accessibili che possono contenere due tipi di informazioni: reliable e non-reliable. Questo perciò rappresenta un modello ibrido dove è possibile sfruttare dati di strutture o enti che forniscono misurazioni precise e allo stesso tempo sfruttare i dati che possono essere forniti dagli utenti [4].
Il motivo per cui questi due modelli di collezione dati sono importanti, specialmente nell'ambito di questa tesi, è dato dal fatto che sono i due approcci usati dalla già esistente piattaforma SenSquare, proposta inizialmente nel 2016 [9] come architettura di Mobile Crowdsensing orientata a fornire servizi all'interno di una Smart City e poi trasformata, nel 2017, in un architettura più generale orientata ai servizi che permettesse a enti e singoli di partecipare fornendo dati e creando servizi [11]. Come vedremo meglio nella sezione successiva quindi, questa rappresenta la base di partenza alla quale questa tesi si propone di apportare un'estensione.

2.3 Contributi

L'obiettivo di questa tesi è dunque quello di fornire una proof of concept, un modello, che implementi un sunto dei concetti sopra esposti. Gli sforzi sono quindi stati concentrati nell'espansione di un modello già esistente di una piattaforma di IoT orientata sia alla raccolta basata su infrastrutture governative ma aperta alla collaborazione di singoli individui, SenSquare.
La motivazione per cui questa piattaforma è stata scelta è dovuta principalmente al fatto che essa propone un sistema già piuttosto completo e ben

implementato. Da qui ci si è quindi potuti muovere senza dover affrontare le diverse difficoltà di creare un'intera infrastruttura solo allo scopo di creare un protocollo di Mobile Crowdsensing. Inoltre la piattaforma non presentava un'implementazione concreta del sistema di ricezione e gestione dati provenienti da dispositivi mobili, in questo modo ci si propone di non solo rendere maggiormente completo un sistema già ben formato ma anche di proporre un'implementazione reale di un sistema di Mobile Crowdsensing collaborativo. In una fase preliminare è stata aggiunta la possibilità di visualizzare una serie temporale dei dati calcolati relativi al servizio richiesto da un utente. Questo per dare alla piattaforma una funzione particolarmente utile per la visualizzazione dei dati ottenuti e desiderati. In seguito la piattaforma ha subito modifiche al database e al server in modo tale da poter permettere la gestione dei due possibili diversi tipi di misurazione, statica o mobile. Infine la parte più importante e consistente della tesi, ovvero lo sviluppo dell'applicazione per la raccolta dati e la conseguente gestione del meccanismo di reward.
Ciò che è stato creato però non è da intendersi come un prodotto del tutto nuovo per sé, la raccolta dati attraverso un'applicazione e protocolli che prevedono la preservazione della privacy nello scambio di dati contenenti informazioni geografiche e temporali sensibili esistono già, come accennato precedentemente. Quello che questa tesi si propone di fare è di assemblare tali concetti in un'architettura che possa fornire una base di sviluppo e un esempio di come tutti questi aspetti possano, in un contesto reale, lavorare insieme efficacemente.

Capitolo 3

Progettazione

In questa sezione verrà discussa l'architettura generale di SenSquare, che ricordo essere un piattaforma già esistente, il cui risultato dunque non è totalmente frutto del mio operato. In Figura 3.1 è possibile vedere un diagramma di come il progetto è stato diviso. L'architettura di SenSquare può quindi essere divisa in tre aree principali: Open Data, Web Server & Database e Mobile CrowdSensing & SenSquare Web Services. Come già annunciato nella Sezione 2.3 il mio contributo si è principalmente limitato all'area riguardante l'MCS ed i WebServices, inoltre è stato anche necessario, come vedremo, un aggiornamento nell'area di Open Data.
In generale però questo capitolo avrà il ruolo di dare un'idea di come la piattaforma funziona, chi agisce all'interno di questo ambiente e quali sono i componenti principali. Per tale motivo si è dunque deciso di dividere il capitolo in tre sezioni: **Attori**, **Componenti** e **Interazioni**.
Prima di tutto però è necessario esporre brevemente alcuni concetti per rendere più comprensibili certi punti che saranno presenti nella spiegazione di queste figure. SenSquare si pone come obiettivo quello di proporsi come un'efficiente soluzione di IoT Collaborativo dove più figure possono collaborare non solo nella raccolta dati ma anche nella creazione di servizi che possano essere creati dai più e poi condivisi con tutti i partecipanti sulla piattaforma. In questo modo si dà la possibilità non solo di crea-

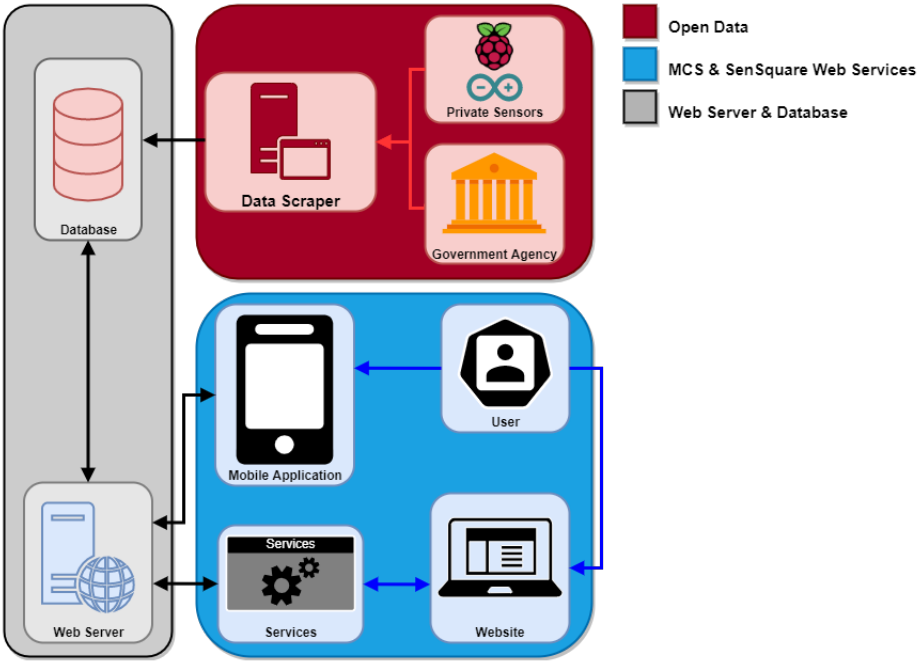


Figura 3.1: Diagramma architetturale di SenSquare

re da zero servizi non precedentemente esistenti ma, sfruttando questo concetto a moduli, viene anche fornito un modo per poter implementare servizi anche di complessità maggiore riutilizzando quelli già creati dalla community.

3.1 Attori

Questa sezione si impegnerà ad elencare tutti gli attori presenti nella scena di SenSquare specificando per ognuno quali sono i servizi di cui può usufruire e quali invece i contributi che può apportare al progetto. È importante sottolineare che per come il progetto è stato pensato, il concetto che qui viene esposto con il nome di Attore corrisponde a quello di originale di Participant, ovvero colui, privato o ente, che decide di partecipare alla piattaforma potenzialmente fornendo supporto in termini di raccolta dati, usufruendo dei servizi messi a disposizione o entrambe.

Ciò che si vuole sottolineare dunque è come sulla piattaforma ciascuno possa essere in realtà indicato come semplice partecipante, quella fatta in questa sezione è una suddivisione atta a rendere più comprensibile la divisione logica del progetto e di come quest'ultimo sia quindi un esempio di IoT Collaborativo. Inoltre ci permette di contestualizzare l'architettura in un ambiente quantomeno realistico.

3.1.1 Utente finale

Per utente finale si intente banalmente il privato, il singolo, che ha la possibilità, fornita dalla piattaforma, di registrarsi, possedere un account ed accedere a tutti i servizi che vengono messi a disposizione da SenSquare. In un senso l'utente finale non è diverso dall'attore del Paragrafo 3.1.3, talvolta infatti i due possono anche coincidere. La differenza logica però è chiara e sarà resa ancora più esplicita quando ne parleremo. Per ora è sufficiente comprendere che il primo ha un ruolo di fruitore dei servizi mentre il secondo agisce più da provider in modo indiretto.
Il ruolo dell'utente finale è dunque determinato dalla possibilità di creare templates, ovvero modelli, attraverso l'utile strumento messo a disposizione da Blockly. Questi templates quindi vengono pubblicati al momento della creazione e possono poi essere resi istanze nel momento in cui un utente registrato decide di attivarli in una zona precisa.

3.1.2 Ente

L'attore in questione ha un ruolo prettamente passivo, si tratta infatti più in generale di tutte quelle infrastrutture che possiedono una certa affidabilità o credibilità, come ad esempio un ente governativo. Il ruolo dunque, pur essendo di primaria importanza e necessario affinché tutto possa funzionare, non è attivo allo stesso modo di come un Utente agisce

sulla piattaforma. In questo caso infatti più che un participant in senso proprio si può pensare all'ente come a una repository pubblica di dati accessibili che vengono aggiornati generalmente di giorno in giorno e resi conseguentemente disponibili a chiunque ne possa o ne debba aver bisogno. SenSquare quindi va a raccogliere i dati di questi enti e li mette a sua volta a disposizione della piattaforma attraverso un componente che vedremo nel Paragrafo 3.2.3.
Un esempio concreto è dato dall'ARPA (Agenzia Regionale per la Protezione dell'Ambiente) che, fra le altre cose, si occupa anche di monitoraggio ambientale e i dati raccolti, ottenuti dalle varie centrali installate nelle principali città in 19 regioni Italiane, vengono poi messi a disposizione attraverso delle Web API che permettono quindi l'interrogazione dei loro server filtrando i dati in base all'area geografica, alla data e altri parametri.

3.1.3 Participant

Come accennato nel Paragrafo 3.1.1, il Participant può talvolta coincidere con l'utente finale. Infatti nonostante sia presente all'interno del sistema un'App dedicata al Mobile CrowdSensing, è possibile per un utente partecipare alla raccolta dati anche mettendo a disposizione i sensori posseduti. Il motivo per cui questo è stato reso possibile è principalmente il fatto che, nonostante il Mobile Crowdsensing rappresenti il modo più facile per ottenere una grande mole di dati, mantenendo i costi relativamente bassi, ha il grande di difetto di affidarsi a sensori che, per quanto buoni, sono molto meno precisi ed affidabili rispetto a sensori ad-hoc che invece molti meno soggetti possiedono. In questo modo chi dovesse averne disponibilità può contribuire all'inserimento di dati sul database con una maggiore affidabilità.

3.2 Componenti

In questa sezione cercherò, allo stesso modo in cui è stato fatto per gli Attori, di elencare e spiegare quali sono i componenti principali dell'architettura di SenSquare ed il loro ruolo. Mi pare però importante specificare che per componenti in questa sezione si intende componenti software, per questo motivo non si farà riferimento a tutti i sensori privati, come potrebbero esserlo quelli presenti su un Arduino o una Raspberry Pi, o quelli pubblici, come le stazione dell'ARPA.

3.2.1 Web Server

Il Web Server è, ovviamente, l'elemento più importante di tutto il progetto. Infatti non si può individuare un unico ruolo, vi è piuttosto una pletora di servizi che è capace di offrire. Come è possibile vedere in Figura 3.1 il server fa da intermediario fra diversi componenti ed attori, si posiziona infatti fra il database e qualsiasi altro componente presente nell'area dei servizi web e dell'MCS.
In modo molto classico il server si pone ad esempio come l'intermezzo che fornisce al sito web le informazioni da mostrare agli utenti che sono presenti sul database ma è anche, e forse soprattutto, un'unità di calcolo responsabile della gestione dei templates e delle istanze di tali template. È infatti il responsabile del calcolo dei valori desiderati per una determinata istanza su una precisa serie temporale, come vedremo più avanti nel Capitolo riguardante l'Implementazione. Inoltre il Server ha il ruolo fondamentale di implementare la comunicazione con i dispositivi mobili del MCS e questo include da un lato la raccolta dei dati inviati dall'applicazione da un Utente e dell'altro le gestione del meccanismo di reward che è di fondamentale importanza soprattutto sotto l'aspetto della privacy. Infatti l'aspetto più importante è proprio la comunicazione App-Server, il cui protocollo è mostrato in Figura 3.2. Quello che tale protocollo garantisce è ovviamente l'anonimato dell'utente, infatti quando le misurazioni vengo-

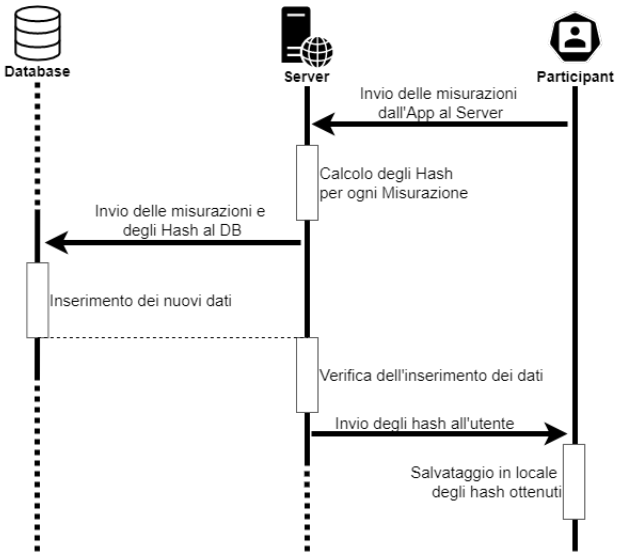


Figura 3.2: Protocollo di comunicazione App-Server

no salvate nel Database non vi sono riferimenti all'utente, ogni misurazione relativa a ad un determinato sensore sarà identificata unicamente dal tipo del sensore. Questo però ancora non evita la possibilità di un Man-in-the-middle in fase di reward, per questo è stato necessario implementare un algoritmo che limitasse, almeno a livello concettuale, la quantità di informazioni sensibile trasportata dall'insieme di dati inviati. In Figura 3.3 è possibile osservare invece come si svolge il processo di reward, il calcolo del subset ottimale verrà discusso più nel dettaglio all'interno del capitolo relativo all'Implementazione.

3.2.2 Website

Il sito web di SenSquare, sviluppato utilizzato il framework Angular2, sfruttando quindi tecnologie come TypeScript, HTML5 e SCSS, si presenta con un design che segue le regole del Material Design proposto da Google nel 2014, questo è dovuto al fatto che essendo un progetto in Angular2 possiede la capacità di trasformarsi in Progressive Web App, ovvero un'applicazione che può essere usata senza dover essere installata sul dispositivo

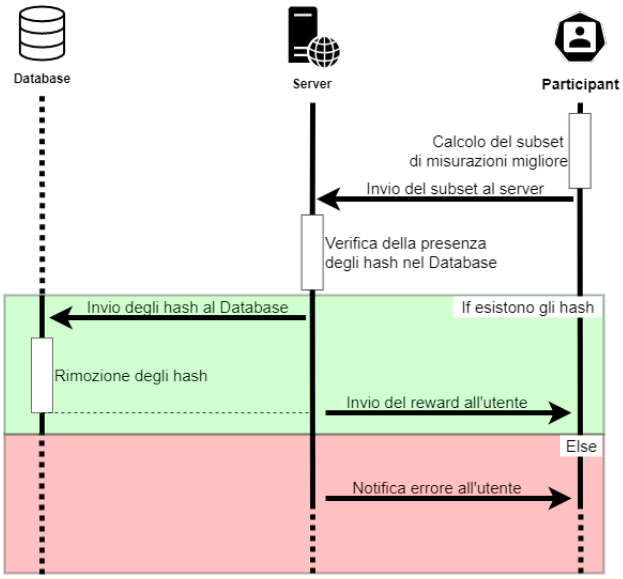


Figura 3.3: Protocollo di Reward nel sistema MCS di SenSquare

dell'utente. Questo componente, come è ovvio che sia, rappresenta il mezzo tramite il quale un utente può accedere ai servizi a crearne di nuovi. Vi sono infatti due pagine principali: la prima elenca i templates pubblicati dagli altri utenti presenti sulla piattaforma, la seconda invece mostra le istanze create a partire da tali template. È poi possibile, cliccando su un'istanza, visualizzare la pagina di dettaglio che mostra la zona in cui è stata dispiegata, gli ultimi parametri ottenuti per i valori desiderati e l'ultimo valore calcolato con quei parametri. Qui è stato deciso di aggiungere all'interno del progetto originale la possibilità di visualizzare anche una serie temporale che a scelta dell'utente può variare da una giornata a una settimana ad un mese intero. Questo poi avrà implicazioni a livello implementativo che vedremo nel capitolo sull'Implementazione.

3.2.3 Data Gathering

Il Data Gathering è il componente che si occupa di fare quello che in gergo viene chiamato "Scraping", ovvero l'azione di ottenere dati, cosa alla quale generalmente si dà una connotazione negativa. Infatti spesso si

intende il fatto di estrarre dati che non sono stati messi a disposizione tramite un servizio di Web API ed è quindi necessario trovare vie traverse per ottenerli, come ad esempio cercarli direttamente all'interno del sorgente HTML di un sito web. In realtà qui il lavoro svolto dal componente di Data Gathering è l'insieme dello Scraping e dell'uso di API messe a disposizione da enti affidabili. Per contestualizzare può essere utile prendere da esempio quindi il servizio fornito dall'ARPA che tramite delle Web API mette a disposizione i dati che vengono raccolti dalle loro stazioni urbane. Questa è per SenSquare al momento la fonte più rilevante, regolare ed affidabile di informazioni.
Più in generale però oltre a collezionare le informazioni si preoccupa anche di metterle assieme, classificarle secondo determinati criteri per poi aggiungere le misurazioni sul database in modo tale che diventino disponibili agli utenti. È importante però introdurre qui una delle modifiche che poi saranno approfondite discutendo l'implementazione del Database, infatti è fondamentale comprendere che il Data Gathering si concentra nell'ottenimento di misurazioni **statiche**, ovvero dati che provengono da sensori che non cambiano mai la loro posizione nel tempo. Dal lato opposto invece vi sono le misurazioni **mobili** che come sarà chiaro più avanti sono necessarie per distinguere l'Open Data dal MCS.
Infine ritengo importante sottolineare che la rappresentazione di questo componente è un Applicazione Server, infatti è costituito da un applicazione che viene eseguita in background periodicamente per ottenere i dati e questo avviene parallelamente al Server Web.

3.2.4 Applicazione Mobile

Questo componente potrebbe, nel quadro d'insieme, non risultare essenziale a differenza dei precedenti, ma è in realtà il componente che verrà approfondito più di tutti in quanto rappresenta la sezione del progetto su cui questa tesi si è concentrata maggiormente, pertanto questa sarà la parte su cui mi soffermerò di più specialmente nel capitolo dell'Implementazio-

ne.
L'applicazione è in realtà un componente estremamente importante all'interno dell'architettura poiché permette al progetto di diventare di raggiungere il suo massimo potenziale e diventare un esempio in campo di IoT collaborativo. Come già sottolineato precedentemente infatti la peculiarità di SenSquare è la sua capacità di ottenere dati da più fonti e molto diverse fra loro, mettere assieme i dati e renderli disponibili ad un grande pubblico, costruendo un sistema IoT che sia capace di spezzare il trend legato all'esistenza di Isole delle Cose.

3.3 Interazioni

All'interno di questa sezione descriverò le relazioni principali fra i diversi Attori e Componenti che sono presenti all'interno dell'architettura di SenSquare.

3.3.1 Utente-WebSite

Questa relazione, di tipo Attore-Componente, è quella che avviene chiaramente fra un Utente Finale e il sito Web (disponibile all'indirizzo `https://sensquare.disi.unibo.it`). Consiste, come già accennato, nell'interazione dell'utente con i servizi, la creazione di modelli di servizio e il loro stanziamento su una determinata area, entrambe le cose avvengono poi in senso bilaterale poiché i modelli e le istanze possono essere condivise con tutta la comunità presente su SenSquare. In questo modo quindi, nonostante l'uso principale sia quello di sfruttare il servizio, l'utente può anche creare un contenuto che diventa utile per altri utenti.

3.3.2 Utente-Applicazione Mobile

La relazione fra utente finale e applicazione mobile è molto semplice a differenza di altri approcci dove vi sono regole rigorose e precise per la

raccolta dati[9]. Infatti l'obiettivo, come scritto nella Sezione 2.3, è quello di proporre un semplice modello che potesse fornire un'implementazione reale all'interno di un sistema concreto. Non ci sono regole dunque che decidono come e quando un utente debba inviare dati, vi è invece totale libertà dell'utente di inviare molteplici volte misurazioni, ammesso che si trovi all'interno di un'area e che i sensori usati siano richiesti all'interno di quell'area di interesse. Questo quindi è un esempio di Participatory Crowdsensing dove è l'utente a decidere le modalità della sua partecipazione ed inoltre può potenzialmente fare uso di quegli stessi dati da lui inviati trasformando in determinati casi la partecipazione stessa in un incentivo abbastanza forte di per sé.

3.3.3 Server-Data Gathering

Questa relazione di tipo Componente-Componente è quella su cui si basa gran parte del sistema e come già accennato nelle sezioni precedenti in questa relazione vengono implementati tutti i processi che si occupano di raccogliere dati da diverse fonti, diverse in numero e diverse nella modalità. Lo sforzo compiuto è quindi quello di unificare sorgenti diverse portandole tutte sotto uno stesso standard implementato nel Database.

3.3.4 Server-Applicazione Mobile

Questa relazione è quella che in modo più diretto rispetto a quella fra Utente e WebSite mette in relazione l'utente con il server, infatti come già accennato estensivamente spiegato in Sezione 4.2.2 il Server principale è quello che si occupa non solo del calcolo delle istanze e di fornire un collegamento fra sito e database, ma ha il compito di ascoltare le richieste provenienti dalle applicazioni mobili che, come verrà spiegato in maggior dettaglio più avanti, possono potenzialmente comunicare con grande frequenza le osservazioni. Inoltre, l'avvio dell'applicazione genera anche una richiesta al server che è necessaria per fornire all'applicazione le diverse

aree di interesse per le quali sono richieste misurazioni e i tipi di sensori che sono necessari in quelle aree. Successivamente, ciascuna volta che l'utente decide di mandare dati, l'applicazione si mette in comunicazione con il server inviando tutte le misurazioni di quell'istante di tempo, dopodiché il server aggiunge i dati al database e invia all'applicazione gli hash relativi alle misurazioni ricevute. Questo permette, in fase di reward, di poter controllare che effettivamente l'utente abbia partecipato alla campagna attivamente. Infatti come è possibile vedere in Figura 3.3 nel momento in cui un utente richiede il premio viene inviato un insieme di hash dei quali il server verifica l'esistenza e comunica all'applicazione, a seconda dell'esito di tale verifica, se ha accesso al premio o se invece si è verificato un errore dovuto alla presenza di hash non contenuti nel database.

Capitolo 4

Implementazione

Questo capitolo sarà orientato alla spiegazione dei dettagli tecnici e implementativi delle aree che sono state modificate o create. Non mancheranno ovviamente i doverosi richiami a specifiche del progetto originale qualora siano necessarie a comprendere meglio il funzionamento e la motivazione di determinate scelte. Come scritto nel capitolo precedente, l'area in cui vi è stato un maggiore impegno è stata quella relativa al MCS e al Web Server. Allo stesso tempo è stato necessario aggiornare parte dell'area di Open Data. Poiché più in generale ogni area di questo progetto è stata toccata, seppur in misure diverse, il capitolo si suddividerà in tre sezioni, una per ciascuna area individuata precedentemente in Figura 3.1. Ciascuna sezione poi sarà divisa nei suoi principali componenti sempre richiamando solo quelli su cui vi è stato un'effettiva modifica o che sono stati aggiunti. Ritengo fondamentale specificare che non è presente in questo capitolo un'introduzione iniziale sui linguaggi e frameworks utilizzati per il progetto in quanto essendo già stato precedentemente sviluppato in grande parte essi non rappresentano una scelta implementativa bensì è stato necessario adattare anche le proprie esigenze a ciò che era già presente di questa piattaforma. Ciononostante, le scelte fatte in fase di progettazione hanno garantito un'elasticità sufficiente all'introduzione di novità e modifiche.

4.1 Open Data

Questa sezione non sarà particolarmente approfondita in quanto è stata inserita a seguito della necessità di un aggiornamento di parte di questo componente. In particolare infatti sarà presente unicamente la spiegazione della parte che si occupa di raccogliere dati dall'ARPA il cui aggiornamento è stato necessario in seguito all'aggiornamento delle API da loro messe a disposizione.

4.1.1 ARPA Data Gathering

L'ottenimento dei dati dall'ARPA è stato implementato tramite l'esecuzione di uno script python che viene schedulato sul server per essere eseguito una volta al giorno. Questo script in realtà non è il vero e proprio scraper, collabora invece con un altro script Python che è il diretto responsabile dell'interrogazione dei server ARPA. Più nello specifico quello che accade è che il primo script possiede una lista di codici, ciascuno corrispondente ad una provincia, e attraverso l'uso di un ciclo for il secondo script viene richiamata più volte andando a raccogliere i dati per ogni stazione presente all'interno di quella provincia.
In Figura 4.1 è possibile osservare uno schema riassuntivo di questo processo. Come è quindi possibile osservare, la comunicazione fra gli script e le API dell'ARPA non è unidirezionale ed ovviamente questo perché l'interrogazione risulta nella restituzione di un file JSON che contiene tutte le misurazioni dell'ultimo anno per una specifica stazione di monitoraggio. Quest'ultima deve essere specificata come argomento della Query ed è rappresentata da un codice univoco, tale codice è ottenuto in una fase preliminare dallo script che ci crea la lista con tutti i codici relativi ad una provincia e di conseguenza poi richiede i dati all'ARPA. Una volta ottenute le misurazione lo script si occupa di inserirle all'interno del database, più precisamente nella tabella relativa alle misurazioni che richiede l'inserimento di: coordinate (quelle della stazione di monitoraggio), valore

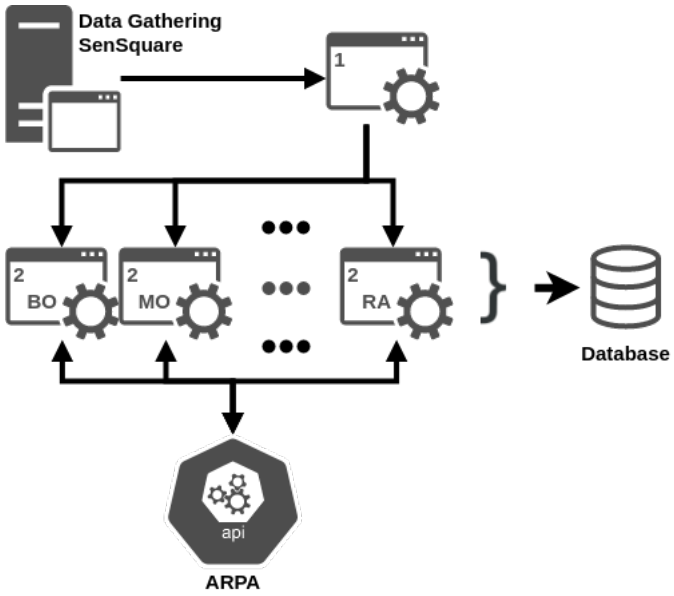


Figura 4.1: Diagramma dello Scraper ARPA

misurato, tipo di sensore, timestamp e DataStream_ID (questo campo sarà più chiaro in Sezione 4.2.1 dove descriverò più nel dettaglio la struttura del DataBase).

A questo punto è necessario approfondire il motivo per cui ho precedentemente dichiarato, in Sezione 3.2.3, il Data Gathering un ibrido fra uso di Web API e Scraping. Infatti, a seguito dell'aggiornamento del sistema dell'ARPA, le API usate precedentemente hanno cessato di funzionare, questo perché il servizio è stato modificato e i dati possono essere richiesti ora solo tramite mail e non possono essere richiesti tramite lo stesso endpoint presente prima. Facendo qualche ricerca all'interno dei sorgenti inviati dal server al caricamento della pagina per la visualizzazione dei dati ambientali (`https://www.arpae.it/v2_rete_di_monitoraggio.asp?p=BO`, nello specifico relativa a Bologna ma valido anche per altre province) mi è stato possibile trovare il nuovo endpoint. In questo senso è stato quindi necessario un workaround per ottenere i dati desiderati.

4.2 Back-end (Web Server & Database)

All'interno di questa sezione descriverò dal punto di vista tecnico le componenti essenziali e più interne di SenSquare dal punto di vista tecnico. L'obiettivo è in questo caso quello di dare un'idea dettagliata di come i diversi componenti possano essere messi in comunicazione gli uni con gli altri. Verranno quindi analizzati il Web Server, di cui molto è già stato detto in Sezione 4.2.2, e il DataBase, colonna portante del progetto. Entrambi, come sarà poi chiaro, sono stati modificati, il primo in maggior misura rispetto al secondo ma quest'ultimo richiede comunque un approfondimento tecnico per poter avere una migliore comprensione del progetto. Inizierò infatti a descrivere il DataBase.

4.2.1 DataBase

Il Database rappresenta lo scheletro dell'intero sistema SenSquare. Infatti contiene, come è ovvio che sia, tutte le strutture dati ed i relativi dati necessari al funzionamento del site, dell'applicazione ed in generale essenziali per poter fornire un servizio. Le relazioni, seppur indirette, vi sono con tutti i componenti presenti all'interno dell'architettura, di queste solo una è diretta. Tutto infatti, come vedremo nella Sezione successiva, è filtrato dal Server che si occupa di inserire ed ottenere informazioni dal Database.
Le tabelle presenti sono diverse ma l'obiettivo di questo capitolo è quello di soffermarsi principalmente sugli aspetti su cui il mio lavoro si è concentrato, per questo e per il fatto che solo alcune tabelle sono state modificate verrano qui descritte solo una parte:

- **Measurements**: In Figura 4.2 è possibile osservare la struttura della tabella. Le colonne sono tante ma non tutte necessarie allo stesso modo. Infatti al momento il progetto ancora non fa uso delle coordinate MGRS (Military Grid Reference System) che infatti viene sempre inizializzato con un valore pari a **-1**. Alcuni campi sono autoesplicativi,

Measurements	
ID	int(11)
data_stream_ID	int(11)
GPS_latitude	double
GPS_longitude	double
MGRS_coordinates	varchar(16)
value	double
timestamp	timestamp
Sensor_type	varchar(256)
hash	varchar(65)

Figura 4.2: Struttura della tabella *Measurements*

l'ID ad esempio è chiave interna per la tabella ed è rappresentata da un intero di 11 cifre che viene auto-incrementato ed assegnato dal Database ogni qualvolta venga inserita una nuova misurazione. Le coordinate GPS, il valore e il timestamp chiudono invece la lista dei campi esistenti nel progetto originale. Nel corso dell'implementazione del progetto di tesi però si è rivelato necessario aggiungere due campi: **Sensor_type** e **hash**, quelli evidenziati in azzurro in Figura 4.2. Il primo può contenere il valore *STC* nel caso in cui la misurazione sia relativa ad un sensore proveniente dall'area degli Open Data, in caso contrario contiene l'acronimo del tipo di sensore relativo a quella misurazione. Il secondo invece può essere *NULL* (valore di default) oppure contenere l'hash relativo alla misurazione.

- **DataStreams**:La Figura 4.3 mostra la struttura di questa tabella che non ha subito modifiche durante l'aggiunta dei nuovi moduli. Nonostante ciò è comunque importante dare qualche dettaglio riguardante questa struttura in quanto rappresenta uno dei punti chiave del sistema di servizi forniti tramite sito web.

  Infatti l'entità qui rappresentata è il sensore o la stazione di monito-

DataStreams	
ID	int(11)
name	varchar(32)
data_class	varchar(255)
creation_timestamp	timestamp
last_update_timestamp	timestamp
description	varchar(64)
elevation	float
url	varchar(32)
last_entry_ID	int(11)
device_ID	varchar(96)

Figura 4.3: Struttura della tabella *DataStreams*

raggio che fornisce quindi un *flusso di dati* e di conseguenza quindi si vuole salvare informazioni riguardanti il nome, il tipo di dati che forniscono, la data in cui è stato registrato, la data dell'ultimo aggiornamento ed altre informazioni che nel contesto di questa tesi hanno una minor rilevanza.
Come è possibile osservare dalla tabella il campo ID è l 'unica chiave interna della tabella, questo implica dunque che due DataStream possano avere lo stesso nome, il che non è un errore bensì una scelta progettuale. Come è già stato detto infatti buona parte dei valori affidabili sono forniti dall'ARPA, che li raccoglie tramite un numero limitato di stazioni di monitoraggio presenti in ciascuna provincia; poiché ciascuna di queste stazioni può essere fornita di molteplici sensori, è molto spesso vero che una stessa stazione (con lo stesso nome dunque) fornisca più dati di tipo differente. Per il funzionamento però è ovviamente necessario fare una distinzione separando logicamente i provider di dati in base al tipo di dati che forniscono e non unicamente al loro nome. Il caso di due DataStream con stesso nome e data_class viene evitato attraverso gli opportuni controlli in fase di inserimento.

Per quanto riguarda però il caso dei DataStream relativi ai dispositivi mobile è stata fatta una scelta orientata a preservare la privacy degli utenti, si è dunque deciso di mantenere un unico DataStream per tutti i sensori dello stesso tipo, in questo modo tutte le misurazioni relative alla temperatura registrata da uno smartphone tramite l'applicazione avranno lo stesso datastream_ID, nascondendo in questo modo ogni informazione riguardante l'utente.
Un'altra nota importante riguarda il campo **ID** che essendo la chiave per i DataStream rappresenta il campo **data_stream_ID** della tabella delle misurazioni ed è quindi la relazione che lega queste due tabelle. Dal punto di vista logico ovviamente c'è quindi una relazione di appartenenza delle Misurazioni ai DataStream.

- **CustomServices**: In Figura 4.4 viene presentata la struttura della tabella che logicamente rappresenta i servizi di cui gli utenti possono usufruire all'interno del sito. Come per le altre tabelle sino ad ora descritte è presente un campo che ha lo scopo di identificare in modo univoco un servizio, oltre a questo poi vi è invece la colonna che contiene l'ID dell'utente che ha istanziato il servizio, quest'ultimo è un codice univoco presente all'interno di una tabella che non verrà qui descritta ma che, in breve, raccoglie le informazioni degli utenti che si sono registrati sulla piattaforma.
  Continuando poi si possono notare tre campi relativi al deployment del servizio, i primi due forniscono le coordinate del centro dell'area dello schieramento dell'istanza, il terzo invece contiene un valore decimale che corrisponde al raggio di quest'area.
  Successivamente il template_ID dà indicazioni su quale sia il template corrispondente all'istanza corrente e permette quindi di recuperare informazioni necessarie per l'esecuzione dell'istanza che sono però presenti unicamente all'interno della tabella dei template. Il campo **json_args** infine contiene un testo in formato JSON che fornisce informazioni relative al tipo di dati desiderati per la specifica istanza

CustomServices	
ID	int(11)
participant_ID	int(11)
deployment_center_lat	double
deployment_center_lon	double
deployment_radius	double
template_ID	int(11)
title	varchar(64)
json_args	longtext
description	varchar(128)

Figura 4.4: Struttura della tabella *CustomServices*

presa in analisi, qui oltre al tipo di sensore desiderato vi è anche un'informazione relativa alla modalità in cui le misurazioni devono essere considerate, ovvero se l'utente abbia richiesto la misurazione specifica, il massimo, il minimo o la media dei dati richiesti. Tali opzioni sono espresse tramite 4 diverse costanti: *DSID*, *MAX*, *MIN* e *AVG*.

- **CustomServicesTemplates**: La Figura 4.5 mostra la struttura della tabella e come si nota alcuni campi sono identici a quelli presenti nella tabella descritta precedentemente, vi è infatti un identificativo univoco, ma soprattutto viene ripetuto il campo *participant_ID* e *json_args*. I campi principali però sono *xml_template*, che contiene il template creato con Blockly, *python_code*, che contiente il corrispondente codice in python tradotto dal Blockly, ed il campo *share_with_others* che contiene un semplice intero ad una cifra che rappresenta un booleano indicante lo stato di condivisione del template. Ciascun template in fase di creazione infatti può essere reso pubblico o mantenuto privato. Come nel caso misurazioni-datastream vi è anche qui una relazione di appartenenza dei CustomServices ai CustomServicesTemplate, infatti il template, pur essendo un'implementazione concreta di un ser-

CustomServicesTemplate	
ID	int(11)
participant_ID	int(11)
output_type	varchar(16)
expression	varchar(128)
title	varchar(50)
xml_template	longtext
json_args	longtext
python_code	longtext
description	varchar(128)
share_with_others	tinyint(1)

Figura 4.5: Struttura della tabella *CustomServicesTemplate*

vizio rappresenta anche un oggetto astratto che viene concretizzando attraverso lo schieramento su un'area e quindi nella trasformazione in un CustomService. Questo è sottolineato dal fatto che più CustomServices possono corrispondere ad un solo template, quello che generalmente varia è invece l'area di deployment ed il raggio.

4.2.2 Web Server

Ora che le strutture dati principali sono state chiarite procederò a descrivere il funzionamento del Server che mette in comunicazione i diversi componenti della piattaforma. Nello specifico però mi limiterò a spiegare solo la parti coinvolte dal lavoro svolto per questa tesi, in particolare quindi saranno presentati i moduli relativi all'esecuzione delle istanze e alla gestione della comunicazione con l'applicazione mobile. Quest'ultima sarà analizzata solo lato server lasciando la parte descrizione della parte client al capitolo relativo all'Applicazione Mobile.

- **InstanceExecutor**: Questo modulo si occupa, come suggerito dal nome stesso, di calcolare i valori richiesti da una determinata istanza.

Per calcolo si intende l'esecuzione dello script python relativo al template al quale l'istanza appartiene. Questo generalmente consiste nel calcolare operazioni elementari su diversi dati che gli vengono forniti. I dati però gli vengono forniti dal server, dunque quest'ultimo deve occuparsi di recuperarli dal database. Come descritto nella Sezione precedente, al punto riferito alla tabella dei CustomServices, i dati necessari possono essere di 4 tipi, raggruppati in 2 macro tipologie. Vi è il caso in cui viene richiesto il valore specifico di un datastream, oppure quello in cui bisogna trovare un valore all'interno di un insieme di misurazioni (massimo, minimo o media). All'interno del server quindi vi sono due classi che si occupano di gestire ciascuna il proprio caso: **SpecificDataStream** e **MaxMinAvgOfDataStream**. Prima di passare a descrivere il modo in cui queste due classi operano è però necessario ricordare che l'utente può richiedere il calcolo dell'istanza su un periodo di tempo che può variare dagli ultimi 30 giorni alle ultime 24 ore, con l'opzione intermedia dell'ultima settimana. Questo viene indicato nella query e il server lo passa come argomento ad entrambe le classi che quindi ne hanno un riferimento. Questo permette di fare un calcolo su serie temporali a differenza della versione precedente.
Entrambe presentano gli stessi metodi e le stesse variabili, la differenza sostanziale vi è nella modalità in cui però i dati vengono recuperati dal Database. Infatti nel caso di SpecificDataStream le diverse misurazioni vengono recuperate filtrando la tabella delle misurazioni con l'ID del datastream desiderato, che è indicato all'interno del campo *json_args* del CustomService. Per quanto riguarda MaxMinAvgOfDataStream invece i dati vengono recuperati filtrando le misurazioni sulla base della loro posizione e il tipo di sensore a cui corrispondono. In entrambe le situazioni tutto viene salvato in una lista che contiene le misurazioni ulteriormente filtrate per data coerentemente con il periodo di tempo richiesto dall'utente. Una terza classe poi associa

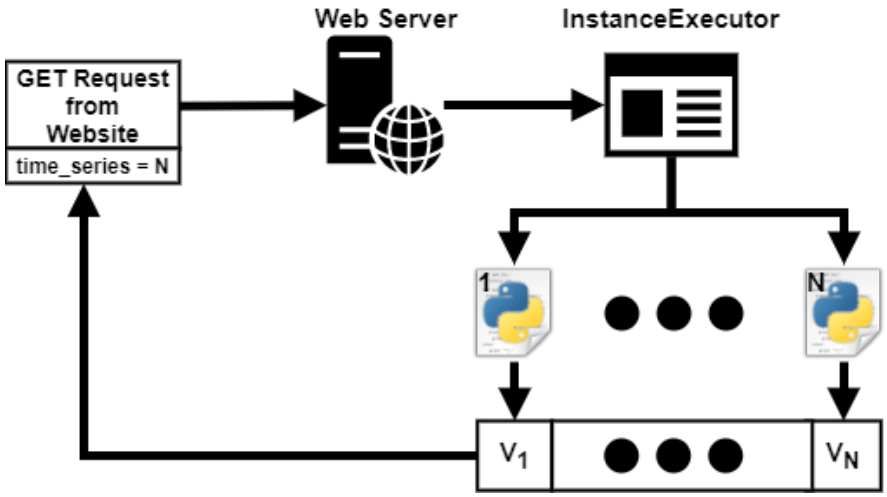


Figura 4.6: Ciclo di vita del calcolo di un Istanza

ciascun parametro dell'istanza alla corrispettiva classe e si occupa, tramite un ciclo, di calcolare N volte (con N pari al periodo di tempo richiesto) i valori da fornire all'esecuzione dell'istanza, la quale viene a sua volta eseguita N volte. Quest'ultima viene eseguita ogni volta con un set di dati che va da 1 ad N (il primo giorno del periodo usa una misurazione, il secondo due e così via fino ad N misurazioni per l'N-esimo valore). In questo modo si calcola il valore con i parametri relativi unicamente ad un giorno o, nel caso non siano presenti valori, al primo giorno passato che dispone di valori. In Figura 4.6 è possibile vedere uno schema riassuntivo del calcolo delle istanze, si noti che il valore N può corrispondere a tre costanti, come già accennato precedentemente, che sono 30 per il mese, 7 per la settimana e 1 per il giorno.

- **App Back-End**: Non precedentemente presente in SenSquare, permette la comunicazione con i dispositivi mobili che decidono di partecipare alla raccolta dati. È un modulo relativamente semplice, costituito di una funzione che gestisce sia le richieste POST che GET provenienti dall'applicazione mobile:

- **GET**: Nell'intero processo vi è solo un possibile tipo di richiesta di questo tipo, ovvero quella che è necessaria ai dispositivi mobili per ottenere la lista di aree di interesse. Dunque il server interroga il database richiedendo tutte le istanze create dagli utenti ed invia per ciascuna le coordinate e le informazioni relative ai sensori richiesti.

- **POST**: In questo caso si possono distinguere due chiamate, una per la ricezione di misurazione dall'app e una per la gestione del reward. La prima riceve una lista di misurazioni sotto formato JSON dall'app. Per ogni misurazione si controlla se esiste il DataStream relativo per il sensore, se non esiste viene creato e viene assegnato alla misurazione. Qui è necessaria una digressione in merito alla gestione dei DataStream per i sensori Mobile che, come già descritto in Sezione 4.2.1 a differenza del generale concetto legato all'entità in questo contesto è stato deciso di crearne uno unico semplicemente per il tipo di sensore. In questo modo ogni dato dello stesso tipo viene raggruppato insieme e si perde ogni riferimento con l'utente che l'ha inviato. Per ogni misurazione poi viene calcolata la sha256 della string composta da **{Latitudine} -{Longitudine}-{valore}-{timestamp}-{sensore}**, il digest ottenuto viene salvato in una lista temporanea che alla fine del processo di inserimento viene restituita al client in modo tale che possa avere prova di aver inviato dati al server ed aver contribuito alla campagna.
  Il secondo tipo di POST Request invece riguarda la gestione del reward. Il server riceve in questo caso un lista di hash e si occupa di controllare che tutti gli hash siano presenti all'interno del database, se la verifica va a buon fine allora modifica ognuna delle righe contenenti gli hash ricevuto annullando il campo **hash** dopodiché avvisa l'utente che l'operazione è andata a buon fine restituendo il premio. Nel caso contrario invece l'utente viene

notificato che il processo di reward ha riscontrato delle incongruenze fra i dati inviati e quelli presenti sul database e che il tentativo di ottenere il premio è fallito.

4.3 Front-end (MCS & Web Services)

In questa sezione sarà presentato più nel dettaglio il sito web di SenSquare (corrispondente ai Web Services sul piano logico) mentre la parte relativa al Mobile CrowdSensing, ovvero l'applicazione mobile, sarà affrontata in modo più superficiale, lasciando spazio per maggiori spiegazioni al capitolo dedicato. Il sito web non sarà comunque approfondito in modo estremamente dettagliato poiché le modifiche apportate non sono state sostanziali e l'intenzione non è quella di ripetere il contenuto già presente nella tesi del Dr. Gianluca Iselli, creatore della piattaforma [11].

4.3.1 Web Site

Del sito web verrà descritto solo ciò che è stato modificato e dunque nello specifico si tratta della presenza del grafico all'interno della pagina di dettaglio di un istanza.
Come riferimento in Figura 4.7 è possibile vedere una schermata presa dal vecchio sito. Ogni elemento è stato quindi mantenuto ma sopra all'ultimo valore calcolato è stato inserito il grafico accompagnato da un selettore che permette all'utente di scegliere quale periodo di tempo visualizzare. Questa modifica è osservabile in Figura 4.8. A livello implementativo è stato molto semplice aggiungere il grafico, l'architettura a moduli caratteristica di Angular infatti ha permesso di creare tramite CLI un nuovo modulo. La presenza di librerie esterne installabili tramite il node package manager ha reso facile trovarne una dedicata alla creazione di grafici, in questo caso è stato usato il pacchetto Chart.js. I dati necessari per la creazione del grafico vengono in parte ottenuti dal modulo padre (corrispondono alla lista di

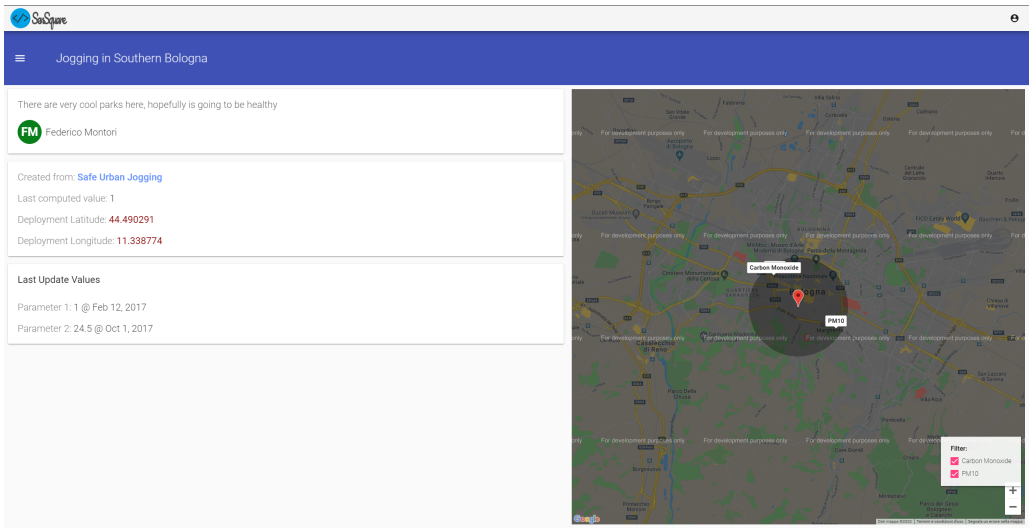


Figura 4.7: Schermata della vecchia pagina di dettaglio di un'istanza

valori ritornata dal server) e in parte calcolati. L'asse X del grafico infatti contiene la date/ore delle misurazioni, ma essendo queste ultime calcolate a partire dalla giornata corrente al momento dell'esecuzione è facilmente calcolabile localmente la lista di valori per questo asse. Sulle ascisse invece si trovano ovviamente i valori calcolati dal server. Nel momento in cui il selettore viene modificato, la pagina viene ricaricata con i nuovi dati inviati dal server ed automaticamente il grafico viene aggiornato mostrando i dati richiesti dall'utente.

4.3.2 MCS

Descriverò in questa parte qualche dettaglio in più a riguardo dell'applicazione mobile senza scendere troppo nei dettagli, in quanto quello sarà il compito del prossimo capitolo. Dunque l'obiettivo qui è quello di introdurre brevemente l'applicazione mobile. Alcune scelte implementative sono innanzitutto necessarie. L'applicazione è stata sviluppata in Android nativo usando come linguaggio di programmazione Java. Inizialmente era stata considerata la possibilità di svilupparla usando l'utile framework messo a disposizione da Google, Flutter, estremamente comodo, molto vi-

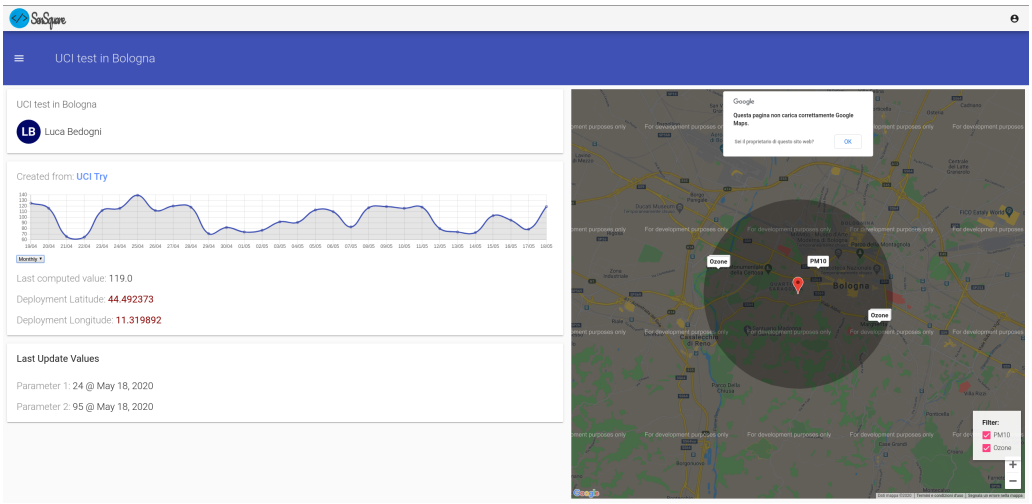


Figura 4.8: Schermata della nuova pagina di dettaglio di un'istanza

cino anche all'approccio di Angular di costruzione a moduli e componenti. Il punto di maggior forza di Flutter è il fatto di poter essere eseguito sia su iOS, sia su Android scrivendo solo una volta il codice che viene poi tradotto e compilato per il rispettivo sistema operativo. Questo ovviamente implica una maggior partecipazione, almeno in potenza. Lo svantaggio però, che deriva direttamente dalla caratteristica appena descritta, è il fatto di avere un minore accesso alle API di sistema e più nello specifico ai sensori. Sono comunque presenti librerie esterne che permettono di scavalcare questi ostacoli, ma più che di soluzioni affidabili si tratta di workaround. Dal momento che quest'applicazione si pone come obiettivo principale quello di proporre una proof-of-concept, la decisione è virata sulla soluzione che in un contesto reale non avrebbe massimizzato il numero di potenziali partecipanti, ma che in questa istanza ha permesso di avere un maggior facilità di sviluppo.

Altro aspetto importante che ritengo importante sottolineare è che maggiore priorità in ogni scelta è stata data alla facilità di sviluppo, ignorando alcuni aspetti che in un'altra sede sarebbero dovuti essere curati maggiormente. Non sarà quindi presente un discorso riguardante le performance dell'applicazione e allo stesso modo l'aspetto estetico non è stato curato

con l'impegno che normalmente il design di un'applicazione richiederebbe. Aspetto però che invece ha ricevuto la giusta attenzione, anche se rimane embrionale a livello implementativo, è il discorso privacy che si è ritenuto doveroso affrontare, soprattutto nel contesto storico attuale.
Come già affermato più volte dunque, questa applicazione è stata creata come supporto per l'implementazione dello scheletro dell'architettura per l'area riguardante il Mobile CrowdSensing in SenSquare.

Capitolo 5

Applicazione Mobile

Il capitolo si articolerà in una prima sezione principale che avrà il compito di descrivere il modo in cui l'applicazione funziona, le diverse opzioni che fornisce all'utente e la logica con cui è stata costruita. Non si entrerà però nel dettaglio, come vedremo, poiché la gran parte del codice relativo a questa sezione è d'uso comune e composto di blocchi spesso ripetuti all'interno delle applicazioni per costruire una base dalla quale partire. In seguito poi verranno analizzati i tre aspetti principali che richiedono un approfondimento maggiore poiché sia dal punto di vista tecnico, sia da quello concettuale sono più articolati e peculiari dell'applicazione.

5.1 Funzionamento

In questa sezione descriverò il funzionamento generale dell'applicazione spiegando come l'utente può muoversi all'interno dell'applicazione e quali sono le azione che gli sono concesse. I dettagli implementativi qui non saranno approfonditi eccessivamente in quanto le caratteristiche esposte corrispondono a codice di inizializzazione che non ha rilevanza nel contesto di questa tesi.

In Figura 5.1 viene mostrato il ciclo di vita dell'applicazione che verrà di seguito approfondito. All'avvio dell'applicazione infatti viene immediata-

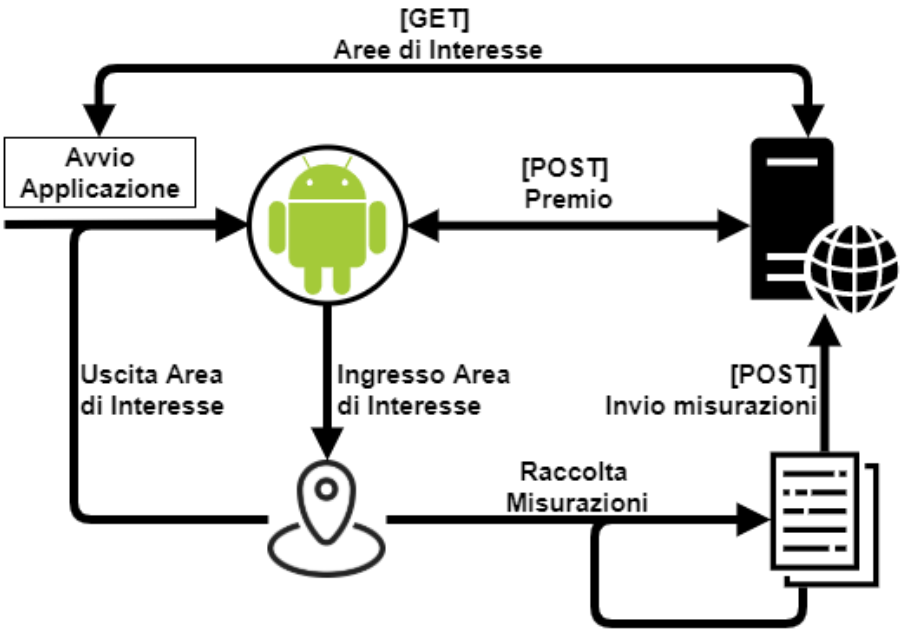
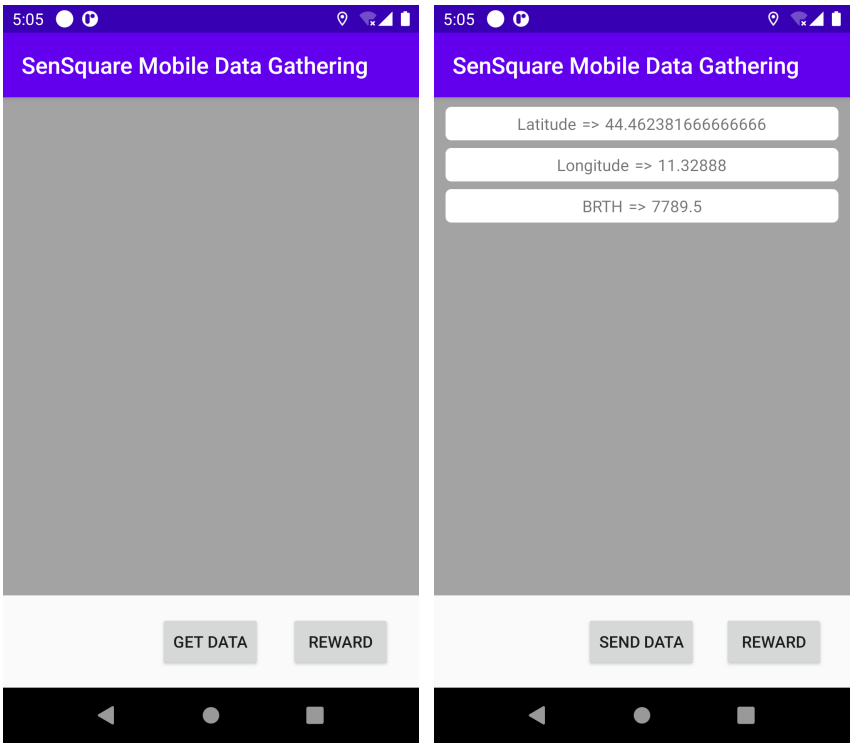


Figura 5.1: Ciclo di Vita dell'Applicazione

mente effettuata una richiesta di tipo GET al server che immediatamente, come già discusso nel capitolo relativo, interroga il DataBase e restituisce all'applicazione la lista delle aree con le informazioni riguardanti ai sensori richiesti per ciascuna di esse. Le informazioni, ricevute in formato JSON, vengono trasformate in una lista di *InstanceObject*, una classe creata ad-hoc per contenere le informazioni quali le coordinate del centro, i sensori richiesti ed il raggio dell'area relativa all'istanza. La posizione viene poi costantemente monitorata e nel momento in cui questa ricade all'interno di una delle aree viene data la possibilità all'utente di inviare i dati. Questo controllo viene fatto per evitare che l'utente invii costantemente i dati anche in luoghi dove le misurazioni non sono necessarie. In questo modo viene limitato il numero di misurazioni e ridotto il rischio di dense data, anche se è importante notare che non è stata applicata nessun'altra misura per prevenire tali casi. L'obiettivo infatti non è quello di creare un'applicazione per l'MCS che sia capace di ottimizzare la quantità e la qualità dei dati, questo tema è già stato affrontato in altri articoli [9]. Nel momen-

to in cui quindi l'utente entra in un'area per la quale all'interno del sito web sono state create istanze, un bottone nella parte bassa dello schermo viene reso cliccabile e permette di collezionare i dati dai sensori che sono disponibili sul dispositivo. I dati a questo punto però non sono ancora stati inviati, infatti, subito dopo aver raccolto le misurazioni, l'applicazione mostra a schermo le coordinate dell'utente e i dati raccolti, in modo da rendere l'utente consapevole dei dati che invierà al server. Se egli decide di inviare può farlo semplicemente cliccando una seconda volta il bottone che nel frattempo ha cambiato il proprio testo da "*GET DATA*" a "*SEND DATA*". Quello che internamente succede è che il dati raccolti vengono filtrati, scartando quelli che non sono richiesti, e viene creato un oggetto di tipo JSON che può infine esser inviato al server il quale poi salverà le informazioni restituendo gli hash relativi che vengono localmente salvati nel Database locale in SQLite che vedremo in una delle sezioni successive. Il reward invece può essere richiesto in qualsiasi momento dall'utente, partendo dal momento in cui l'applicazione è avviata fino all'attimo in cui non viene interrotta. Il bottone per richiedere il premio si trova anch'esso nella parte bassa dello schermo, alla sinistra del bottone per la raccolta dati. Nel momento in cui questo viene cliccato l'applicazione sfrutta l'algoritmo dedicato per decidere quali hash mandare al server in modo, come sarà poi spiegato più nel dettaglio in Sezione 5.2, da mantenere quanto più possibile la privacy dell'utente. Quando il server risponde, indifferentemente dal fatto che tale risposta risulti positiva o negativa, gli hash inviati vengono eliminati in modo tale che non possano essere inviati nuovamente. Al momento ovviamente l'applicazione non possiede nessun premio concreto da poter dare all'utente, questo aspetto è lasciato al caso in cui l'applicazione dovesse essere implementata per un caso reale.
In Figura **??** è possibile vedere un esempio di come l'applicazione si presenta nei due casi d'uso più comune, l'entrata all'interno di un'area di interesse, che come precedentemente descritto è caratterizzata dal bottone cliccabile nella parte bassa centrale, e la collezione dei dati che invece



(a) Entrata in un'Area (b) Display dei dati

Figura 5.2: Activity principale dell'Applicazione

permette all'utente di visualizzare gli esatti valori che saranno inviati al server.

5.2 Algoritmo Privacy

All'interno di questa sezione verrà discusso l'algoritmo che viene usato per scegliere le informazioni da inviare al server, dunque verrà spiegato in primis il motivo per cui è importante che sia presente anche all'interno dell'applicazione mobile e come si è deciso di implementarlo.
Il motivo dietro alla necessità di munire l'applicazione di un algoritmo atto a nascondere informazioni sensibili è dato dal fatto che innanzitutto back-end e front-end non sono necessariamente creati dagli stessi sviluppatori, questo dunque implica il fatto che, nonostante in questa istanza

ci sia la certezza che il server curi la privacy degli utenti, non è scontato che in un contesto reale la piattaforma back-end sia un servizio fornito da un individuo diverso rispetto a quello che decide di sviluppare l'applicazione per la raccolta di dati. Inoltre, più in generale un rischio potrebbe essere quello di subire un Man-in-the-middle Attack, ovvero un tipo di attacco che consiste nel porsi fra due sorgenti che scambiano messaggi e presentarsi a ciascuna di esse come la sorgente opposta, potendo così o modificare il contenuto delle informazioni scambiate o più semplicemente ottenere dati sensibili. In questo senso, poiché lo scambio di dati porta con sé informazioni riguardanti misurazioni che sono avvenute in un determinato luogo in una determinata data ed ora, è possibile che facendo un matching con i dati eventualmente estratti dal database potrebbe poter riuscire ad individuare quali set di misurazioni appartengono ad un determinato individuo. Per questo motivo si è voluta porre attenzione su questo aspetto che, pur rimanendo embrionale a livello implementativo, è concettualmente fondamentale per garantire affidabilità agli utenti e di conseguenza incentivare alla partecipazione nelle diverse campagne offerte. Il premio da solo rimane un utile incentivo ma non è l'unico aspetto da tenere in considerazione.

In Figura 5.3 viene mostrato un esempio banale di come l'algoritmo opera in un sistema fittizio di coordinate cartesiane. In verde vi sono le coordinate tra le quali scegliere il sottoinsieme. Quelle che possiedono una corona circolare rossa invece sono quelle scelte dall'algoritmo. In blu troviamo il centro "reale" che tutte le coordinate individuerebbero, al contrario in arancione c'è il centro fittizio che il sottoinsieme scelto genera. Come è evidente, l'obiettivo dell'algoritmo è quello di rendere il centro finto il più lontano possibile dal centro reale.

Ritengo doveroso sottolineare che questa non è una soluzione applicabile in un contesto reale, ci sono tanti aspetti che l'euristica presa in considerazione non gestisce, banalmente vi è il semplice caso in cui al posto che essere fortemente sparsi i punti si raggruppano in diverse aree, che potrem-

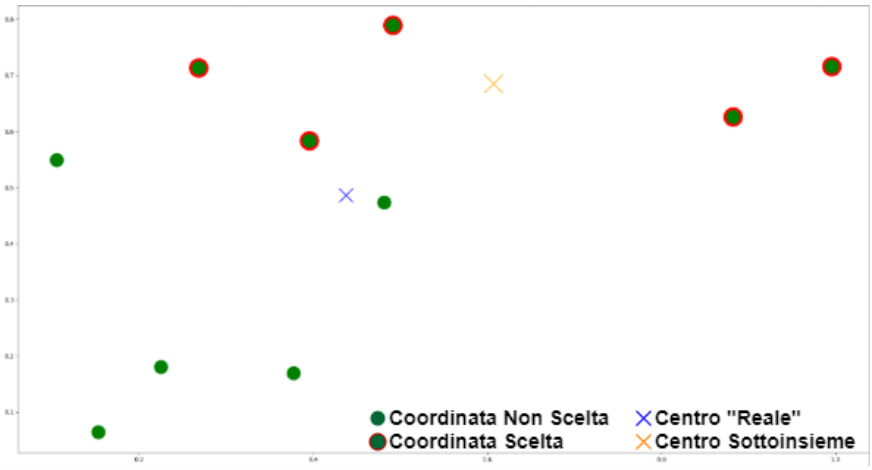


Figura 5.3: Esempio di come l'algoritmo lavora sulle coordinate

mo pensare come l'area di lavoro e l'area nella quale si trova l'abitazione, in questo caso l'algoritmo probabilmente fallirebbe nell'ottimizzazione del sottoinsieme. Come già però specificato più volte l'obiettivo non è quello di risolvere per intero il problema, in questo caso l'algoritmo è una black box che può essere sostituita a piacere con algoritmi più avanzati e capaci di tenere conto di molti più aspetti.

5.3 DataBase SQLite

Qui sarà descritto il database implementato in locale per permettere all'applicazione di memorizzare gli hash ricevuti dal server. La scelta è ricaduta sull'uso del database piuttosto che dello storage di file perché rende più semplice e facile la scrittura e lettura di informazioni, non è necessario un parsing in fase di lettura ed in generale tutto assume un aspetto più rigoroso e preciso.
Come descritto in Figura 5.4 la struttura della tabella è molto semplice, poche infatti sono le informazioni necessarie all'applicazione. Il campo hash che è di tipo testuale contiene l'hash relativo a una delle misurazioni

inviate al server. *LNG* e *LAT* invece contengono rispettivamente il valore della Longitudine e Latitudine che sono necessari per il calcolo dell'algoritmo descritto nella sezione precedente.
La tabella viene creata per la prima volta quando l'utente installa ed avvia l'applicazione, le volte successive il database è già presente e non è richiesta nessuna inizializzazione. Quando l'utente manda i dati al server l'applicazione attende la risposta e appena questa arriva la risposta viene tradotta e per ogni hash ricevuto viene inserita una riga nel database. Nel momento in cui invece il premio viene richiesto, indifferentemente dal tipo di risposta, gli hash ritornati dal server vengono eliminati. In un caso, quello di risposta positiva, l'eliminazione avviene per evitare il riciclo di hash, nell'altro invece ciò viene fatto per evitare che hash che per un qualche motivo non trovano corrispondenza nel database principale della piattaforma rimangano sull'applicazione e possano nuovamente causare il fallimento della richiesto del premio.
Un approccio che si potrebbe però implementare, diverso da quello correntemente usato dall'applicazione, potrebbe essere quello di non cancellare ogni riga ma di aggiungere un campo booleano che indichi se la misurazione è già stata usata, in questo modo si potrebbe mantenere un più corretto concetto di "centro", altrimenti usando l'approccio descritto in questa tesi l'algoritmo potrebbe non lavorare come desiderato. Rinnovo quindi il concetto importantissimo, ciò che viene proposto in questa tesi è un'architettura funzionante e concreta di come un sistema di MCS possa essere applicato in particolare a SenSquare ed in generale ad una piattaforma di Crowdsensing, i componenti sviluppati però hanno margine di miglioramento e non sono completi in senso implementativo.

5.4 Gestione Sensori

In quest'ultima sezione verranno descritti i sensori e la loro gestione, quali sono stati scelti e per quale motivo. Innanzitutto una nota fonda-

Hashes	
Hash	TEXT
LNG	DOUBLE
LAT	DOUBLE

Figura 5.4: Tabella del Database locale dell'applicazione

mentale è quella che riguarda la creazione, a livello implementativo, di una classe che permettesse di gestire tutti i sensori indipendentemente dal tipo, particolarmente utile per poter gestire tutti i sensori come un unico oggetto dello stesso tipo. Non è stato però possibile far rientrare tutti i sensori all'interno della stessa classe. Due infatti sono stati gestiti in maniera differente: sensore di rumore e di potenza di segnale. Il primo dei due richiede infatti la registrazione del microfono per poter percepire il livello del suono, il secondo invece l'accesso alle informazioni relative al segnale GSM che sono state ottenute tramite la libreria built-in *CellInfoLTE*.
I sensori scelti sono sei: Luce ambientale, Pressione, Temperatura ambientale, Umidità, Forza del segnale GSM e Intensità del suono. Il motivo per cui questi sono stati scelti fra i tanti sensori disponibili, nonostante alcuni non siano comuni all'interno degli smartphone, è il fatto che da un punto di vista di monitoraggio ambientale possono dare più indicazioni rispetto ad altri che possano anche essere di un determinato interesse ed eventualmente sfruttabili. Altri casi come quelli relativi al movimento e all'accelerazione sono sicuramente interessanti, ma richiedono una maggior pulizia dei dati ed una gestione un po' più complessa.

In Figura 5.4 viene mostrato un piccolo schema riassuntivo della gestione e come è possibile osservare tutti i sensori vengono raggruppati e di conseguenza gestiti allo stesso modo, nel momento in cui l'applicazione entra in **onPause()** i listener vengono stoppati per evitare un consumo eccessivo di batteria, in **onResume()** invece l'ascolto viene ripristinato, tutto questo insieme al listener relativo alla posizione. Questo aspetto è molto importante perché come altre caratteristiche già descritte, contribuisce ad

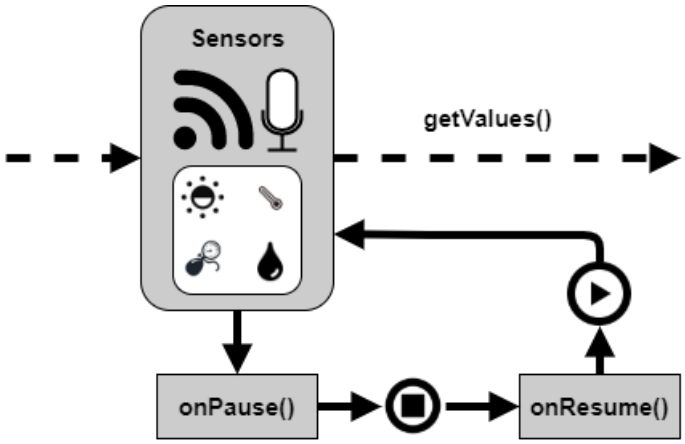


Figura 5.5: Schema gestione sensori

incentivare l'utente all'uso dell'applicazione e alla partecipazione, se non vi sono remore legate ad un consumo della batteria l'utente avrà meno problemi a partecipare.
Continuando con la gestione, al momento del click del bottone centrale viene attivata la funzione che, dopo aver filtrato i sensori necessari, recupera tramite un semplice metodo, appartenente ai listener dei sensori, l'ultimo valore da questi osservato. Come già accennato, il sensore per il livello di decibels necessita di un task asincrono che gli permetta per qualche secondo di ascoltare l'ambiente circostante per poter determinare il valore da comunicare all'applicazione, durante questo periodo quindi il bottone non è cliccabile ed è necessario attendere il completamento dell'operazione per poter inviare i dati. Completata questa azione i sensori sono nuovamente pronti per essere ascoltati e finché l'applicazione continua ad eseguire in foreground non verranno stoppati, questo per permettere all'utente di inviare dati senza limite. Il motivo per cui si è deciso di non porre un limite è prettamente dovuto ad una necessità di testing, l'azione può senza problemi essere ritardata a piacere decidendo quando è sensato che l'utente invii le misurazioni.

Capitolo 6

Conclusioni

Concludendo, quindi, quello che è stato presentato è una proposta di un'architettura che estende un progetto già esistente basato su un tipo di IoT collaborativo che attinge agli Open Data per fornire più servizi liberi ai propri utenti.
Lo sforzo è stato quindi inizialmente orientato ad aggiungere un ulteriore servizio al sito web, fornendolo quindi di un grafico relativo a serie temporali di dati che spaziano da un periodo di 24 ore ad un periodo di 30 giorni.
Il secondo passo è stato poi quello di aggiornare l'ottenimento dei dati dall'arpa, estremamente essenziali in fase di testing. Una volta aggiornato il Data Gathering il passaggio successivo è stato quello di apportare le necessarie modifiche al lavoro fino a quel momento svolto perché potesse gestire tipi di dati statici e mobili in modi differenti. Questo è stato quindi fatto per preparare la piattaforma all'introduzione dell'applicazione per il Mobile CrowdSensing.
Come è già stato accennato nel corso di questo documento, le possibilità sono tante e le migliorie che si potrebbero apportare altrettante, ad esempio si potrebbe implementare una logica che ottimizzi il modo in cui gli utenti possono inviare le singole misurazioni, decidendo anche quali tipi di sensori richiedano più dati e quali invece non siano necessari in un numero

alto. Inoltre l'applicazione è stata sviluppata in Android, il che implica una riduzione dei partecipanti alle campagne, un miglioramento potrebbe dunque semplicemente consistere nello sviluppare anche una corrispettiva app per il mondo iOS che quindi apra le porte anche a tantissimi altri utenti. Infine forse l'aspetto per il quale di più mi aspetto un cambiamento è l'algoritmo per la scelta del migliore sottoinsieme di coordinate, è probabilmente l'aspetto più interessante ed attuale della tesi anche se per ovvie ragioni non il più sviluppato. In ogni caso questo rimane estremamente semplice poiché, come ho già detto, si tratta di una proof of concept ma una miglior euristica nella scelta delle coordinate è fondamentale per davvero assicurare una buona protezione della privacy degli utenti.

Bibliografia

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[2] P. Mtshali and F. Khubisa, "A smart home appliance control system for physically disabled people," in *2019 Conference on Information Communications Technology and Society (ICTAS)*, pp. 1–5, 2019.

[3] S. M.Zorzi, A. Gluhak and A.Bassi, "From today's intranet of things to a future internet of things: A wireless- and mobility-related view," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 44–51, 2010.

[4] F. Montori, L. Bedogni, G. Iselli, and L. Bononi, "Delivering iot smart services through collective awareness, mobile crowdsensing and open data," in *5th IEEE International Workshop on Pervasive Context-Aware Smart Cities and Intelligent Transport Systems (PerAwareCity 2020)*, 2020.

[5] F. Montori, L. Bedogni, F. Morselli, and L. Bononi, "Achieving iot interoperability through a service oriented in-home appliance," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, 2017.

[6] F. Montori, L. Bedogni, and L. Bononi, "A collaborative internet of things architecture for smart cities and environmental monitoring," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 592–605, 2018.

[7] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[8] F. Montori, P. P. Jayaraman, A. Yavari, A. Hassani, and D. Georgakopoulos, "The curse of sensing: Survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for internet of things," *Pervasive and Mobile Computing*, vol. 49, pp. 111 – 125, 2018.

[9] F. Montori, L. Bedogni, A. Di Chiappari, and L. Bononi, "Sensquare: A mobile crowdsensing architecture for smart cities," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 536–541, 2016.

[10] F. Montori and L. Bedogni, "A privacy preserving framework for rewarding users in opportunistic mobile crowdsensing," in *7th International Workshop on Crowd Assisted Sensing, Pervasive Systems and Communications (CASPer 2020)*, 2020.

[11] G. Iselli, "Sensquare: una piattaforma iot di crowdsensing e sviluppo collaborativo di servizi personalizzati," *Alma Mater Studiorum, Bologna*, 2017.