Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

SVILUPPO DI YOUCROWD: UNA PIATTAFORMA PER CAMPAGNE MOBILE CROWDSENSING ON-DEMAND

Relatore:
Dott.
FEDERICO MONTORI

Presentata da:
DOMENICO RINALDO

Sessione III
Anno Accademico 2020/2021

Il web ci ha insegnato il potere
dell'"effetto di rete": quando connettete
le persone e le idee, esse crescono.
(Chris Anderson)

Sommario

Con il termine "crowdsensing" si intende una tecnica in cui un folto gruppo di individui aventi dispositivi mobili acquisiscono e condividono dati di natura diversa in maniera collettiva, al fine di estrarre informazioni utili. Il concetto di Mobile Crowdsensing è molto recente e derivante dalle ultime innovazioni tecnologiche in materia di connettività online e cattura di dati di vario genere; pertanto non si trova attualmente una vera e propria applicazione in campo reale, la modellazione solo teorica e fin troppo specifica pone un limite alla conoscenza di un ambito che può rivelarsi molto utile ai fini di ricerca. *YouCrowd* è un piattaforma web che va ad implementare un sistema di crowdsourcing completo, in grado di leggere dati dai numerosi sensori di uno smartphone e condividerli, al fine di ottenere una remunerazione per gli utenti che completano una campagna. La web application vede la sua implementazione di base supportata da NodeJS e si configura come una piattaforma dinamica che varia la propria interfaccia con l'utente in base alle richieste di dati da parte degli administrators. Il test di *YouCrowd* ha coinvolto un buon numero di partecipanti più o meno esperti nell'utilizzo degli strumenti informatici, rivelando delle buone prestazioni in relazione alla difficoltà del task e alle prestazioni del device in test.

Introduzione

La nascita e l'evoluzione di internet sono eventi che hanno sconvolto il mondo sotto ogni punto di vista, in quanto le necessità relative alla trasmissione e condivisione di dati hanno generato un valore immenso per l'economia. La sfida moderna non riguarda la crescita di internet ma, bensì, la semplificazione dell'accesso aumentando la velocità generale delle connessioni alla rete e allargando la banda a disposizione, ovvero il quantitativo di dati trasmissibile in maniera simultanea. Chiaramente quella esposta si tratta di una evoluzione tecnologica che può progredire solo dal momento in cui ogni tipologia di utente può averne accesso attraverso dispositivi abilitati. Gli strumenti necessari per poter mettere in atto tutto ciò hanno visto una diffusione molto rapida e sono in continuo sviluppo per ciò che riguarda le proprie funzionalità: una categoria emergente di dispositivi ai margini di Internet sono i dispositivi mobili di rilevamento e di calcolo incentrati sul consumatore, come gli smartphone, i lettori musicali e i sensori a bordo dei veicoli. Questi dispositivi alimentano sempre di più l'evoluzione dell'*Internet of Things* in quanto contribuiscono ad una creazione di una rete di sensori su scala sociale e globale, andando ad alimentare anche altri contesti di ricerca che pongono al centro i dati grezzi, come quello dei *Big Data*. In questo articolo si ha l'esposizione di una piattaforma web denominata *YouCrowd* che copre perfettamente il concetto di *Mobile Crowdsensing*, il quale ha una stretta correlazione con l'IoT [2] in quanto tutti gli utenti che possiedono dispositivi di rilevamento e di calcolo possono condividerne collettivamente i dati ed estraggono informazioni per misurare e mappare fenomeni di inte-

resse comune. Tutto ciò va a delineare un'attività di crowdsensing che deve essere alimentata da una buona campagna di incentivazione, necessaria per far si che un buon quantitativo di utenti riescano a condividere dati ma anche a ricevere un *reward* personale: si ipotizza la retribuzione monetaria come uno degli strumenti più efficaci per aumentare l'interesse della popolazione. Il Mobile Crowdsensing può apparire come un ambito piuttosto di nicchia che però verrà approfondito nei vari capitoli, attraverso la descrizione della piattaforma *YouCrowd* e su come è stata realizzata. In particolare, uno degli aspetti che verrà maggiormente evidenziato riguarda l'integrazione totale che *YouCrowd* mette al centro rispetto alle poche altre piattaforme esistenti, per quanto riguarda tutte le fasi cruciali della classica attività di un *crowdsourcer*. Il testo non si concentrerà solamente su una mera presentazione dell'applicazione ma si andrà a "validare" il sistema attraverso un test vero e proprio effettuato direttamente su un campione di clienti, i quali hanno fornito diverse indicazioni in merito all'attività di crowdsensing e al loro rapporto con la piattaforma nello specifico.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Stato dell'arte

In tutti i contesti di ricerca è di fondamentale importanza accedere ad una varietà di dati, provenienti da test in campo reale oppure da simulazioni, al fine di poter estrarre delle informazioni utili a supporto di un determinato caso di studio. Un esempio diretto riguarda gli algoritmi di intelligenza artificiale, la cui bontà può essere testata solo ed esclusivamente su un set di dati possibilmente veritiero. Diventa quindi fondamentale la collaborazione tra utenti, anche inconsapevoli ed esterni ai casi di studio, in modo da poter raccogliere il maggior numero possibile di dati grezzi.

Per *Crowdsensing* si intende un concetto molto ampio, volto alla collaborazione tra individui al fine di ricercare e donare dati di vario tipo, con lo scopo di ottenere un elemento di valore, come una remunerazione. Nel caso di studio il *Mobile Crowdsensing* si pone come obiettivo l'incentivazione di un gruppo di individui nel rilevare e condividere dati attraverso l'utilizzo di dispositivi mobili. I dati raccolti sono quindi utili per l'estrazione di informazioni di vario tipo, soprattutto se accompagnati da indicazioni relative al target dei donatori.

1.1 Concetti e terminologia

I dispositivi mobili moderni sono dotati di vari sensori ed ormai il loro pricing li rende accessibili a tutti. Pertanto la società moderna fa largo uso di smartphone in grado di rilevare diversi dati come la luce ambientale, il rumore (attraverso il microfono), la posizione (attraverso il GPS) e altri ancora. La diffusione di dispositivi mobili è divenuta quindi sempre più capillare: secondo le statistiche riportate da Gartner [6] il numero di smartphone venduti nel mondo nel 2018 è stato di 1,55 miliardi di unità e il numero di dispositivi indossabili spediti nel 2018 è stato di 178,91 milioni, che si prevede raggiungerà i 453,19 milioni nel 2022. Smartwatch, occhiali, anelli, guanti e caschi sono i dispositivi indossabili più popolari attualmente disponibili sul mercato, corrispondenti a un fatturato in forte aumento che si stima aumenterà fino a 95,3 miliardi di dollari entro il 2021.

Il Mobile Crowdsensing è pertanto un concetto che nasce in contemporanea allo sviluppo delle tecnologie di comunicazione più recenti, come il 5G, che rendono molto più immediato il trasferimento di una grande mole di dati a partire dai vari tipi di dispositivi smart. Soprattutto per quanto riguarda il contesto urbano, il Mobile Crowdsensing può migliorare significativamente la vita quotidiana dei cittadini e fornire nuove prospettive e servizi: un esempio diretto è quello delle *Smart Cities*, in cui possono nascere degli spunti importanti per il futuro [3] volti all'ottimizzazione e all'innovazione dei servizi pubblici, mettendo direttamente in relazione le infrastrutture delle città con il capitale umano.

Il concetto fondamentale che, in generale, va a ricoprire le nozioni già presentate e fornisce un forte significato al Mobile Crowdsensing è quello dell'*Internet Of Things (IoT)*, il quale definisce un'ulteriore aggregazione di dati in supporto all'applicazione specifica. La "reazione" di dispositivi smart in relazione alla diversità di dati che essi vanno ad analizzare è un concetto chiave dell'internet delle cose.

L'ambito generale descritto pertanto fornisce un background informativo su cosa si intende per Mobile Crowdsensing e come si inserisce nel contesto

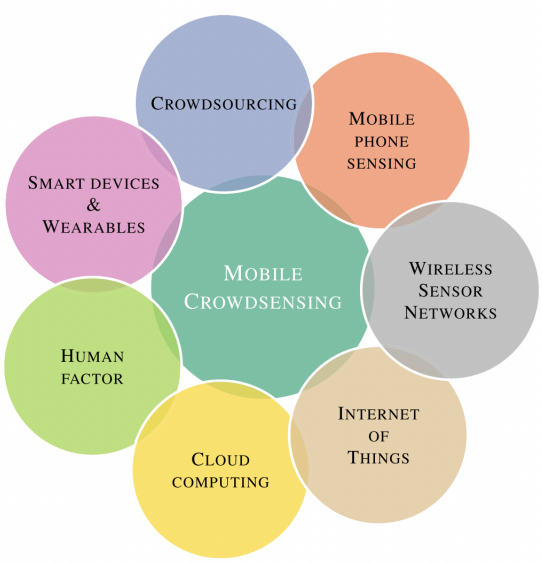

Figura 1.1: Insieme di tutti i concetti chiave e fattori che contribuiscono alla crescita del Mobile Crowdsensing

sociale odierno. Vi sono però un insieme di definizioni e attributi specifici da considerare per capire meglio cosa comprende un ambito così generico. Dalla Fig. 1 andiamo a riassumere punto per punto.

- ***Mobile Phone Sensing***: Area che racchiude la raccolta di vari dati da smartphone, i quali implicano diverse applicazioni. Ad esempio, si può sfruttare l'accelerazione verticale di uno smartphone (con il giroscopio) per rilevare se il proprietario è caduto. Ancora, si può sfruttare l'accelerometro per rilevare il proprio stile di guida della bicicletta o la propria camminata [5].

- ***Wireless Sensor Networks***: Si intende delle infrastrutture in grado di rilevare movimenti e dati di vario genere, sfruttando l'interconnessione tra sensori. Molto utili ad esempio nei contesti indoor

- ***Internet of Things***: Come già descritto, concetto per cui si modifica il comportamento di dispositivi smart sulla base dei dati forniti.

- ***Cloud Computing***: Considerando la capacità di storage piuttosto limitata da parte dei dispositivi smart, diventa fondamentale salvare tutti i dati raccolti in una grande entità centrale. Spesso i database adibiti al trattamento di dati IoT sono di natura sparsa, vista la differente tipologia di informazioni

- ***Human Factor***: Si tratta forse della componente più importante per via dell'impatto che ha sul Mobile Crowdsensing [1]. La mobilità e l'intelligenza degli esseri umani garantiscono una maggiore copertura e una migliore consapevolezza del contesto rispetto alle tradizionali reti di sensori. La progettazione di un'efficiente architettura human-in-the-loop può essere impegnativa.

- ***Smart Devices and Wearables***: L'intelligenza e la capacità di raccolta dati degli smartphones è stata inclusa, negli ultimi anni, anche nei dispositivi indossabili. In particolar modo gli smartwatch, oggi sono in grado di raccogliere praticamente gli stessi dati degli smartphone. Pertanto la loro inclusione nel Mobile Crowdsensing, vista la loro semplicità d'uso e il prezzo spesso contenuto, può essere molto utile.

- ***Crowdsourcing***: Vi sono diverse correnti filosofiche dietro a questo termine. La definizione più accreditata è quella di Howe [10] che definisce il crowdsourcing come l'atto di un'azienda o di un'istituzione di esternalizzazione dei compiti precedentemente svolti dai dipendenti a una rete indefinita di persone sotto forma di una chiamata aperta. Il lavoro collettivo tende a rafforzare la produzione e quindi a restituire un risultato di qualità. Il Crowdsourcing, per essere effettivo, necessita però di una campagna di *incentivization* per raccogliere consensi; in relazione a ciò l'*Amazon Mechanical Turk* è un chiaro esempio.

L'insieme dei concetti principali aiutano a capire il contesto di cui si parlerà. Avviare un progetto di Mobile Crowdsensing significa, oltre a definire un'architettura coerente e funzionante, anche stabilire un campagna di incentivizzazione in modo tale da avere un certo consenso tra gli utenti: senza

un lavoro collettivo difficilmente si ottiene un buon risultato. Questo aspetto verrà definito nella prossima sezione.

1.2 Area di interesse

Il contesto del Mobile Crowdsensing va descritto secondo le due macroaree che lo caratterizzano: stiamo parlando dell'incentivazione della componente client, ovvero il fattore umano che dà vita al sistema stesso, e dell'architettura vera e propria del sistema di acquisizione dati.

1.2.1 Incentivazione dei clienti

L'importanza del fattore umano all'interno di un progetto di Mobile Crowdsensing, come già sottolineato, è scaturita dalla mole di dati che si intende raccogliere [13]. Pertanto, in base alla tipologia di progetto, deve essere messa in atto una diversa campagna di incentivizzazione per convincere gli utenti a raccogliere dati, quindi stimolare il loro interesse. Diversi studi recenti si sono concentrati sui punti cardine che contribuiscono all'incentivizzazione del Mobile Crowdsensing. Prendendo spunto da uno di questi [12] si possono descrivere le caratteristiche fondamentali per incentivare un progetto di crowdsensing.

Fattibilità economica : Si tratta di una questione delicata, spesso decisiva per l'andamento totale del progetto di ricerca. Vi sono una serie di approcci che limitano una spesa elevata, mantenendo così il budget basso, andando soprattutto a incoraggiare la partecipazione alla campagna di crowdsourcing in qualcosa di divertente. Ad esempio, al completamento del task di raccolta dati, si può fornire all'utente una sorta di ricompensa sociale.

Qualità dei dati : Come fare per incentivare gli utenti a raccogliere dati di qualità? Spesso si utilizza un approccio basato sul ranking, in

cui si assegna un punteggio ad ogni partecipante. In genere, è possibile calcolare la reputazione degli utenti dalle performance passate, dalle valutazioni degli altri utenti o da una combinazione di entrambi i fattori. Pertanto, sulla base delle prestazioni passate, seguendo quest'approccio diventa possibile anche effettuare una predizione su una eventuale ricerca futura.

Area di copertura : Affrontare il problema della copertura dell'area nel Mobile Crowdsensing è un compito complesso per l'incentivazione. Supponiamo che la variabile di interesse sia la temperatura e che l'obiettivo sia quello di stimarla in una città. Il piano d'azione sarebbe quello di acquisire campioni da utenti che sono uniformemente distribuiti nell'area. Tuttavia ciò non è sempre possibile in quanto la disponibilità di samples è piuttosto sbilanciata geograficamente: il prezzo può differire in base alla regione, oppure semplicemente vi è uno squilibrio di partecipanti tra le varie aree di acquisizione. Nel primo caso d'esempio, la soluzione sarebbe quella di comprare solo il cluster con i campioni più economici (cioè, scarsa copertura), mentre nel secondo caso ci si deve arrendere in base alla disponibilità locale dei samples. Pertanto è richiesta una valutazione della variabilità spaziale dei samples, quindi richiedere quantitativi di campioni differenti in base alla regione.

Equità : Questo è un concetto più complesso ed è inteso come pari opportunità per tutti i partecipanti. Stiamo parlando di un contesto economico in cui diventa inevitabile che gli utenti meno costosi siano da preferire rispetto ai più cari, anche se ciò espone la problematica dell'abbandono della piattaforma da parte di alcuni individui. Per ovviare a tale problema si può richiedere un effort maggiore agli utenti più economici, anche se diversi approcci suggeriscono di considerare la sua qualità piuttosto che il prezzo del singolo [11].

Numero adeguato di partecipanti : Bisogna sempre stabilire un numero minimo di partecipanti che garantisca all'administrator di mantenere

un flusso di dati costante, soprattutto in termini quantitativi. Questa caratteristica è dipendente dalle altre in quanto il numero di utenti attivi nella piattaforma resterà più o meno costante in base alla campagna di incentivazione messa in atto.

Scalabilità : Secondo una definizione più generica, con questo termine si va a definire la proprietà di un sistema di continuare a funzionare correttamente nonostante l'espansione dello stesso in termini dimensionali. Nel contesto del Mobile Crowdsensing bisogna garantire un costante flusso di utenti attivi all'aumentare, ad esempio, dell'area di copertura oppure in base alla modifica dei task da eseguire. Mantenere un numero costante di utenti significa anche sostituire coloro che abbandonano il progetto con nuove reclute. All'aumentare della richiesta di dati del progetto di crowdsourcing si può incentivare gli utenti con nuovi bonus in base all'esecuzione della novità posta in gioco.

Funzionalità indipendente : Si intende la capacità del sistema di funzionare e continuare a raccogliere dati indipendentemente dal tipo di interazione utente-device. In linea generale vi sono due modalità di raccolta dati per un progetto di crowdsensing: modo *partecipatorio* se l'utente deve attivamente inviare un sample (magari cliccando un bottone); modo *opportunistico* se invece il device reagisce in maniera completamente autonoma. Dettagli relativi a questi due aspetti verranno dscritti a fondo nei prossimi capitoli, ma la loro importanza in un contesto di incentivazione è decisiva, perchè si definisce come avviene l'interazione tra l'utente e lo smartphone.

Tutte le caratteristiche descritte ruotano chiaramente attorno ad un discorso economico: un'utente deve poter essere gratificato attraverso l'esecuzione di un task. Il successo di una campagna di crowdsensing si basa su una grande partecipazione e contributo dei cittadini. A tal fine, gli incentivi sono una componente fondamentale.

Tutti i concetti esposti relativamente a i fattori che influenzano una campagna di incentivazione ruotano intorno al contesto economico. Pertanto la miglior ricompensa per l'utente finale resta una remunerazione in denaro, senza dubbio il mezzo motivatore più efficace. La proposta di Reddy [18] espone due tipologie di ricompense monetarie: statica o dinamica.

- ***Ricompensa statica***: In questo approccio, l'ammontare della ricompensa è determinato in anticipo per mezzo di un criterio prestabilito, pertanto questo importo resta lo stesso per tutto l'esperimento. Si può ipotizzare, ad esempio, un numero di livelli di ricompensa statica con un valore crescente in base alla prestazione erogata. Nell'articolo di Mushtag [17] sono esposti tre tipi di incentivi statici sotto forma di micropagamenti: uniforme, variabile e nascosto. Nello schema uniforme, un importo fisso è stato ricompensato per il completamento del task, mentre nello schema variabile importi casuali vengono assegnati sulla base di una distribuzione precedente. Lo schema nascosto pone anch'esso delle ricompense variabili, tuttavia l'importo non viene rivelato fino a quando il task non sia stato completato.

- ***Ricompensa dinamica***: Gli incentivi monetari dinamici stabiliscono un budget variabile per ogni compito, seppur esso sia in dipendenza dalle condizioni in tempo reale del sistema. La variabilità della ricompensa può essere dettata, ad esempio, dalla necessità da parte del crowdsourcer di acquisire dei samples in una certa area con scarsa copertura: proprio in quella zona la ricompensa potrebbe aumentare anche del doppio. Rispettare una scadenza temporale potrebbe rappresentare un altro fattore influente sulla remunerazione finale in maniera incrementale.

L'incentivazione da sola potrebbe però non essere sufficiente qualora non fidelizzi un cliente all'utilizzo di una valida architettura software. D'altronde anche quest'ultimo aspetto rappresenta una componente molto importante per spronare l'utente alla partecipazione di un progetto di crowdsourcing.

Quante volte ci siamo ritrovati a dover disinstallare un'applicazione dal nostro smartphone perchè lenta o malfunzionante rispetto alle promesse? Il principio è molto simile.

1.2.2 Architettura di un sistema MCS

Determinare la struttura di un progetto di Mobile Crowdsensing è molto importante, in quanto si vanno a raccogliere un quantitativo piuttosto elevato di dati, anche sensibili. Per cui l'architettura deve necessariamente essere definita nel suo insieme, attraverso una struttura a strati che racconta come e dove avviene il passaggio dei dati.

Come presentato dalla Fig. 1.2 siamo di fronte ad uno sviluppo verticale in cui, dall'alto verso il basso, il primo livello è l'*Application Layer*, il quale si riferisce all'utente e al compito assegnato. Il secondo livello è il *Data Layer*, riguardante la gestione di tutti i dati raccolti. Il terzo livello è il *Communication Layer* che caratterizza le tecnologie di comunicazione e le metodologie di reporting. Infine, alla base dell'architettura a strati, c'è il *Sensing Layer* che comprende tutti gli aspetti legati al processo e le modalità di rilevamento. Andiamo a descrivere i vari strati nel dettaglio.

- ***Application Layer***: Strato di applicazione che coinvolge aspetti di alto livello delle campagne MCS. In particolare si ha la gestione delle fasi di progettazione e organizzazione della campagna [7], come le strategie per il reclutamento degli utenti e la programmazione dei vari task, oltre agli approcci relativi alla realizzazione dei compiti. Ad esempio è qui che si decide la selezione dei contributi degli utenti al fine di massimizzare la qualità delle informazioni.

- ***Data Layer***: Il livello dei dati comprende tutti i componenti responsabili di memorizzare, analizzare ed elaborare i dati ricevuti dai collaboratori. I samples genericamente risiedono in un'architettura cloud, accessibile o meno dagli utenti. Solitamente questo è il layer in cui

vengono raccolti i dati grezzi. Per esempio, in questo livello le informazioni sono dedotte dai dati grezzi e l'admin potrà affinarli oppure calcolare l'utilità nel ricevere un certo tipo di samples, per eventualmente modificare i tasks.

- ***Communication Layer***: Esposizione di tutte le tecnologie per fornire i dati acquisiti dai dispositivi mobili attraverso i loro sensori al collettore cloud. I moderni devices sono tipicamente dotati di diverse interfacce radio, ad esempio, cellulare, WiFi, Bluetooth, e diverse ottimizzazioni sono possibili per sfruttare meglio le interfacce di comunicazione. Ovviamente si sceglie la tecnologia di comunicazione in base alle proprie esigenze spaziali, più che di dato.

- ***Sensing Layer***: Riguarda il livello dei sensori, ovvero il core concept del Mobile Crowdsensing. I dispositivi mobili di solito acquisiscono dati attraverso sensori incorporati, ma per campagne di rilevamento specializzate possono esserne collegati altri tipi. I sensori incorporati nel dispositivo sono fondamentali per il suo normale utilizzo (ad esempio, l'accelerometro per l'orientamento del display o il sensore di luce per regolare la luminosità del monitor), ma possono essere sfruttati anche per l'acquisizione dei dati. Si includono sensori popolari e diffusi come il giroscopio, il GPS, la fotocamera, il microfono, temperatura, pressione, ma anche altri di ultima generazione, come l'NFC. Vi sono poi i sensori specifici, i quali possono anche essere collegati ai dispositivi mobili tramite cavo o wireless, come i sensori di radiazione, di glutine e di qualità dell'aria. I dati acquisiti sono trasmessi alla piattaforma MCS sfruttando il Communication Layer, come spiegato in precedenza.

Sulla base di questa architettura descritta si può progettare un generico sistema di Mobile Crowdsensing, il quale poi sarà orientato ad una raccolta dati di un certo tipo. Si va quindi a stabilire una tipologia di *framework* sulla base dell'obiettivo della campagna e, anche in questo caso, abbiamo differenti possibilità ed esempi che andremo a definire nella prossima sezione.

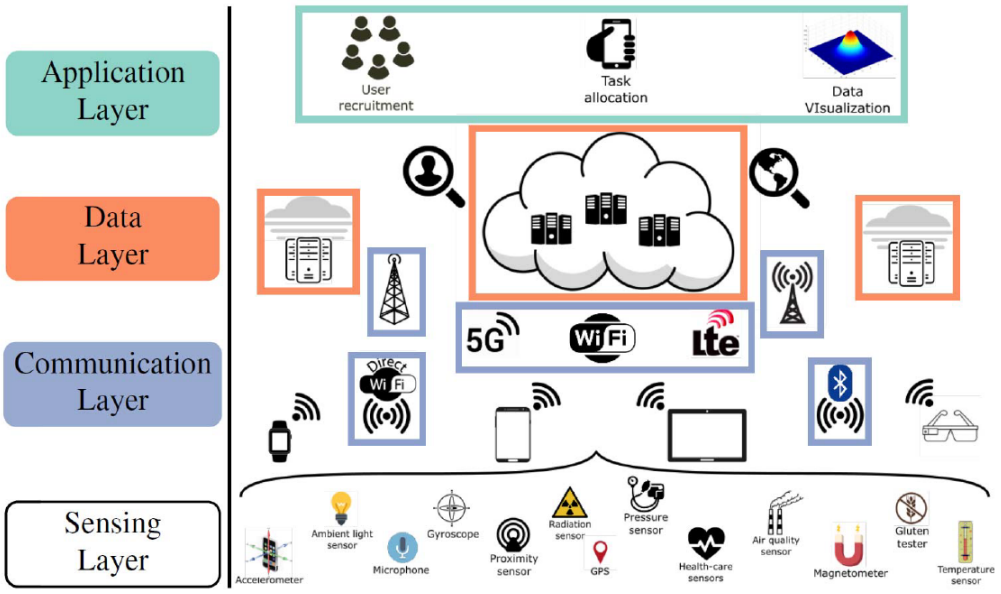

Figura 1.2: Architettura a strati di un sistema MCS

1.3 Piattaforme e Framework esistenti

Lo schema introdotto nella sezione precedente può variare e non poco in base al framework scelto, ovvero l'ambito vero e proprio che la campagna di crowdsourcing va a colpire. Tuttavia i vari framework possono variare per tipologia in base al tipo di interazione con il sistema oppure in base allo scopo finale della piattaforma stessa. Abbiamo già accennato le modalità di interazione partecipatoria e opportunistica degli utenti con i dispositivi di acquisizone, caratteristiche che vanno pesantemente ad influire sull'architettura software e che devono essere prese in considerazione sin dall'inizio.

L'articolo [4] presenta diverse idee e spunti che coinvolgono un ventaglio di sensori differenti, andando a definire task molto diversi tra di loro. In relazione all'obiettivo finale predicato dalla piattaforma MCS ed in base alla tipologia di dati raccolti, si va a definire un'architettura di tipo *domain-specific* qualora l'intento sia quello di monitorare un certo fenomeno, in modo tale da porre una soluzione a un problema; mentre si parla di *general-purpose* se la piattaforma è in grado di supportare più sviluppi allo stesso tempo.

Domain specific systems	
DOMAIN OF INTEREST	DESCRIPTION
Emergency prevention	Prevention of emergencies and post-disaster management
Environmental monitoring	Monitoring of resources and environmental conditions, such as air and noise pollution
E-commerce	Collection, sharing and live-comparison of prices of goods from real stores or specific places
Health care and wellbeing	Sharing mers of physical or mental conditions for remote feedback or exchange of informations about wellbeing
Indoor localization	Enabling indoor localization and navigation by means of MCS systems in GPS denied environments
Intelligent transportation	Monitoring of citizen mobility, public transport and services in cities, e.g., traffic and road condition, available parking spots
Mobile social networks	Establishment of social relations, meeting, sharing experiences and data (photo and video) of users with similar interests
Public safety	Citizens can check, share and evaluate the level of crimes for each areas in urban environments
Unmanned vehicles	Interaction between mobile users and driver-less vehicles
Urban planning	Improving experience-based decisions on urbanization issues, such as street networks design and infrastructure maintenance
Waste management	Citizens help to monitor and support waste-recycling operations, e.g., checking the amount of trash
WiFi characterization	Mapping of WiFi coverage with different MCS techniques, such as exploiting passive interference power, measuring spectrum and received power intensity

Tabella 1.1: lista di tutti i domini di interesse

Per quanto riguarda l'ambito del *domain-specific* il concetto di Mobile Crowdsensing è spesso celato dietro un'applicazione o un servizio web che utilizziamo tutti i giorni. Un chiaro esempio è l'applicazione *Prezzi Benzina* [19] attualmente esistente per smartphone: si tratta di un applicativo che permette il confronto del prezzo attuale della benzina (o anche gasolio, gpl e metano) tra tutte le stazioni esistenti in zona. In questo modo l'utente può scegliere in quale stazione recarsi senza girare tutta la città sprecando carburante e inquinando. Il dominio di appartenenza è quello dell'e-commerce e il vantaggio ottenuto dalla community è quello della sostenibilità economica ed ambientale. Un altro chiaro esempio è relativo alla diffusa app *Waze* [16], ovvero un navigatore per smartphone che però sfrutta le informazioni dagli altri utenti per fornire ulteriori informazioni di guida durante il viaggio; come

un'autovelox non segnalato oppure un incidente. Sono numerosi i domini che si possono coprire con un'applicazione MCS, come suggerito dalla Fig. 1.3.

Progettare un sistema *general-purpose* è sicuramente più complesso sotto ogni punto di vista. Innanzitutto supportare più task diversi significa, per forza di cose, includere una moltitudine di sensori all'interno dello stesso compito, per cui i dati raccolti sono tanti e molto diversi tra di loro. Un'esempio di studio riguardo il contesto *general-purpose* del Mobile Crowdsourcing è *ActiveCrowd*[8], un framework incentrato sulla selezione dei lavoratori per ambienti MCS multitask. Questo progetto si pone come principale obiettivo quello di selezionare il worker giusto per il task giusto, e lo fa secondo due situazioni: la selezione dei lavoratori basata sul movimento intenzionale per i task sensibili al tempo e il movimento involontario per i compiti tolleranti al ritardo. Per i compiti time-sensitive, i lavoratori sono tenuti a muoversi verso il luogo del task in maniera intenzionale e l'obiettivo è quello di minimizzare la lunghezza del percorso. Per i compiti tolleranti al ritardo, si selezionano i lavoratori il cui percorso abituale passa attraverso i luoghi dei task e l'obiettivo è quello di minimizzare il numero totale di workers. *ActiveCrowd* è un chiaro esempio di architettura *context-aware*, in cui la posizione del worker va a influire in modo pesante sul lavoro che egli sarà tenuto a svolgere, indipendentemente dal numero di sensori coinvolti.

Un altro studio molto interessante riguarda il framework TaskMe [14], con l'obiettivo di selezione dei partecipanti per ambienti MCS multi-task. In particolare, vengono studiate due tipiche situazioni di allocazione multi-task con l'obiettivo di ottimizzazione delle risorse.

- *Caso FPMT*: situazione in cui si hanno pochi workers in relazione alla mole dei compiti. Ad ogni partecipante è richiesto di completare più compiti e l'obiettivo di ottimizzazione è quello di massimizzare il numero totale di compiti completati minimizzando la distanza totale di movimento.

- *Caso MPFT*: si hanno molti partecipanti e pochi tasks a disposizione. In questo caso ogni partecipante è selezionato per eseguire un compito

basato su aree di lavoro pre-registrate (nel rispetto della privacy) e l'obiettivo di ottimizzazione è quello di minimizzare sia le ricompense totali dei tasks (lato admin) che la distanza totale percorsa.

Lo studio propone una soluzione attraverso lo studio di vari algoritmi che, di fatto, vanno ad orchestrare il workflow degli utenti della piattaforma; andando quindi a definire una base teorica per un'architettura MCS.

I framework presentati sono approfonditi dai relativi studi in via teorica, senza un approfondimento pratico mancando completamente di una piattaforma software vera e propria; per cui il limite è quello di una proposta che può solo aiutare durante la fase di incentivazione e di allocazione delle risorse sui tasks. Nella letteratura infatti è raro trovare un'approfondimento pratico per quanto riguarda le architetture *general-purpose* nel Mobile Crowdsensing, spesso piuttosto complesse anche nella loro definizione. L'unica vera eccezione, in tal senso, è rappresentata da *Amazon Mechanical Turk*, ovvero una piattaforma di sviluppo piuttosto recente [20] che permette ad un utente worker di iscriversi e aderire ad una campagna di crowdsourcing per poi, al completamento, ricevere una ricompensa in denaro. Ma questa non si tratta dell'unica possibilità offerta, in quanto ci si può iscrivere alla piattaforma anche come "requester", ovvero come utente che richiede l'esecuzione di un task che egli stesso va a definire. Ad esempio, il requester può richiedere dieci samples fotografici relativi alla struttura dei palazzi di Milano e, molto probabilmente, troverà un worker residente in città pronto ad inviare le fotografie ed a ricevere la propria ricompensa.

Si nota come il nome della piattaforma "mechanical turk" rimanda allo storico *turco meccanico*, ovvero il primo sistema che mise in atto il concetto di intelligenza artificiale, attraverso una sorta di robot in grado di battere chiunque nel gioco degli scacchi. Tuttavia egli non era un vero e proprio robot (anche perche si parla del 700') ma una macchina contenente al suo interno un essere umano che, attraverso una lente, riusciva a vedere dall'interno le mosse dell'avversario e quindi spostare le pedine di conseguenza.

Il concetto del turco meccanico di Amazon in effetti ricalca quello storico,

in quanto non esiste una vera e propria piattaforma di acquisizione dati in maniera automatica, ma bensì è il worker che li inserisce manualmente e il requester ne trae beneficio, rielaborando i dati per scopi di studio. Pertanto, anche quest'ultima non può definirsi una piattaforma vera e propria MCS ma bensì un sistema in grado di gestirne il contesto: micropagamenti, invio e acquisizione dei dati sono tutte procedure che avvengono in maniera "manuale", che prescindono dalla componente umana, proprio come il turco meccanico originale. Pertanto si pone un problema, ovvero della mancanza pratica di una vera e propria piattaforma specifica di Mobile Crowdsensing in grado di gestire tutto il flusso di un'applicazione *general-purpose* in maniera del tutto coerente con lo schema proposto nella Fig 1.2.

1.4 Motivazioni

Il problema esposto nell'ultima parte della sezione precedente pone anche un limite sul concetto di crowdsourcing, in quanto il termine non viene utilizzato in sviluppi pratici proprio perchè essi non esistono. Amazon Mechanical Turk è uno sviluppo piuttosto recente che, soprattutto grazie al marchio che ne fornisce il nome, sta iniziando a farsi conoscere nell'ambito MCS ma non va a rappresentare una piattaforma "consumer", ovvero facilmente gestibile anche dall'utente meno esperto. Proprio quest'ultimo poi rappresenta un concetto che spesso viene accantonato nelle varie teorie sull'incentivazione; in quanto proporre un progetto di crowdsourcing sarebbe interessante qualora riuscisse anche ad avvicinare dei workers non troppo avvezzi all'uso dello smartphone, se non per i compiti base. Altro problema non indifferente riguarda la mancanza di ogni controllo sulla qualità del singolo sample [9].

Il progetto *YouCrowd* è un progetto di Mobile Crowdsensing che, sulla base delle richieste specifiche dell'*end user*, pone al centro un'attività di raccolta dati che può essere diversificata. Pertanto *YouCrowd* può essere inteso come un progetto MCS general-purpose, in cui i sensori coinvolti e il tipo di azione che il worker deve compiere possono differire di parecchio rispetto

alla tipologia di task che egli deve portare a termine. Andiamo a descrivere con ordine le motivazioni che hanno portato avanti lo sviluppo di genere di implementazione.

La descrizione relativa al framework di questo sistema MCS è particolarmente incentrata sull'utilizzo di un'unica piattaforma software nella quale gestire ogni aspetto dell'attività di crowdsourcing. Nello specifico, attraverso una web app administrator si può inserire una nuova campagna di Mobile Crowdsensing, andando a specificare ogni suo aspetto, come il quantitativo di workers necessari e i sensori richiesti nel proprio dispositivo personale. Dall'altro lato, i worker si possono iscrivere alla piattaforma per vedere le campagne di crowdsourcing pubblicate, pertanto possono decidere se partecipare o meno in base ai propri gusti o alle proprie esigenze. Particolarità del sistema è il possibile utilizzo multipiattaforma della stessa: verrà approfondito come si è arrivati a poter implementare un sistema che sia compatibile con ogni device in grado di accedere ad internet tramite un browser web.

Lo sviluppo del sistema *YouCrowd* nasce dalla volontà di rendere intendere il Mobile Crowdsourcing come un ambiente integrato e user-friendly, ponendo una soluzione alle problematiche accennate. Rispetto alle teorie progettuali riguardo implementazioni MCS di tipo general-purpose, la proposta in questo caso è pratica e non va a cercare motivazioni in studi di fattibilità oppure in algoritmi in grado di selezionare i workers per sostenere l'incentivazione, ma piuttosto si "scende in campo" attraverso vari test della piattaforma. Lo studio propone quindi l'interpretazione relativa all'esperienza d'uso di questa piattaforma, ponendo l'accento sia su tutte le basi teoriche che ne hanno favorito lo sviluppo sia sulle opinioni dei workers.

Capitolo 2

Struttura del sistema

YouCrowd è un progetto di Mobile Crowdsensing, ambito ampiamente descritto e derivante dal concetto di crowdsourcing, come abbiamo visto. Nello specifico, la definizione di *YouCrowd* in realtà si allontana dal significato proprio di crowdsourcing, in quanto ciò che gli utenti vanno a condividere non è relativo a "codice" o dati grezzi senza alcun significato, ma bensì si parla delle rilevazioni dei sensori, in un contesto direttamente correlato all'interfaccia con l'utente. Ciò significa che l'utilizzo del sistema da parte di un worker permetterà ad egli di verificare con mano, prima dell'invio, cosa sta registrando il device attraverso la stessa app con la quale si verifica la propria posizione e l'invio dei samples. In questo capitolo approfondiremo tutti gli aspetti pratici, evidenziando tutti gli aspetti che caratterizzano il sistema sia ad alto che a basso livello.

2.1 Entità e relazioni

La descrizione approfondita di tutte le entità del sistema è fondamentale per capire, al di là dell'approfondimento tecnico che vedremo in seguito, la progettazione di *YouCrowd* e il suo funzionamento. Come appare dalla Fig 2.1 abbiamo quattro soggetti principali che interagiscono tra di loro.

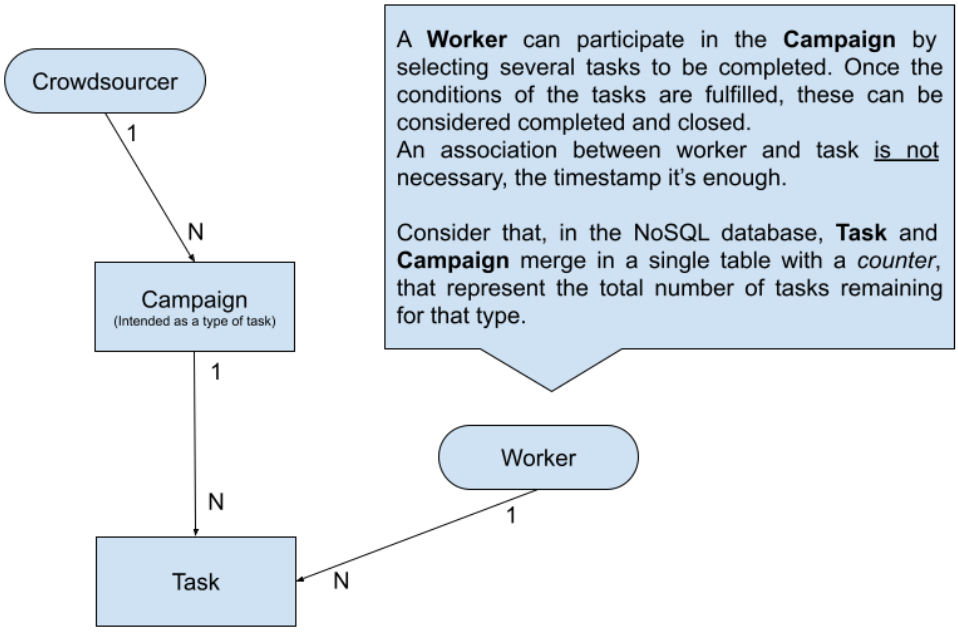

Figura 2.1: Schema relazionale del sistema

- ***Administrator***: Attore amministratore del sistema, l'unico che può accedere alla console di inserimento delle campagne. Può quindi pubblicare un numero illimitato di tasks divisi per tipologia e stabilire una data di scadenza. L'admin può quindi accedere ai dati inviati dai workers relativamente alle proprie campagne.

- ***Campagna***: Oggetto principale del sistema. Si intende una "tipologia di task", ovvero un insieme quantitativo di compiti che possono essere acquisiti dai worker fino a esaurimento. L'admin può inserirne più di una e specificare in esse quanti task (sempre dello stesso tipo) sono richiesti per il completamento. Nella campagna è anche specificata la ricompensa per la conclusione di un task. A livello logico la campagna rimane un'entità invisibile ai workers, in quanto solo gli amministratori potranno considerarla conclusa quando la disponibilità di tasks sarà terminata

- ***Task***: Elemento centrale dell'intero sistema. Un worker può acquisire un task una volta effettuato l'accesso, secondo disponibilità. Vi è un'associazione 1-N in quanto l'utente che seleziona un task da portare a termine non può eseguirlo nuovamente, oltre a non poter iscriversi due volte alla stessa campagna.

- ***Worker***: Attore crowdsourcer che utilizza il software per acqusire dati secondo le specifiche del task. Infatti il suo compito è quello di contribuire alla campagna relativa al task di iscrizione ma il task non è dell'utente. Ciò che veramente conta è la condivisione dei dati e ovviamente il reward corrispondente (al singolo compito).

Le entità descritte ovviamente non interagiscono direttamente tra loro, ma riescono ad eseguire tutte le loro azioni attraverso la web app che permette di eseguire qualsiasi operazione relativa al contesto. Anche il software stesso si divide in due parti, in quanto una è dedicata agli admin mentre l'altra ai workers veri e propri. *crowdsourcer-platform* è il nome in codice del servizio dedicato agli utenti amministratori e permette l'inserimento di una nuova campagna specificando ogni tipo di parametro. Il suo utilizzo non è strettamente correlato all'uso dello smartphone, anzi probabilmente l'uso più comune è quello di un normale sito desktop: come visibile dalla Fig 2.2 il layout per l'inserimento dati è stato pensato di conseguenza.

La pubblicazione di una campagna sul database del sistema fa si che quando un utente qualsiasi intende accedere egli riesca subito a vedere il task pubblicato, volendo poi può selezionarlo e iniziarlo. Il progetto *crowdplatform-service* riguarda invece la piattaforma workers e di fatto si tratta del core vero e proprio di tutto il sistema: attraverso questa web app l'utente può eseguire tutte le operazioni, tra cui iscriversi a un task, eseguirlo e metterlo in pausa, verificare la propria posizione in ogni schermata e così via. L'esecuzione del task è strettamente correlata alla propria posizione, quindi la valutazione di essa in relazione al *geofence* stabilito nella campagna. Infatti i dati da raccogliere verranno registrati e inviati se e solo se l'utente si trova all'interno dell'area definita dal datore di lavoro. Questo aspetto si lega al-

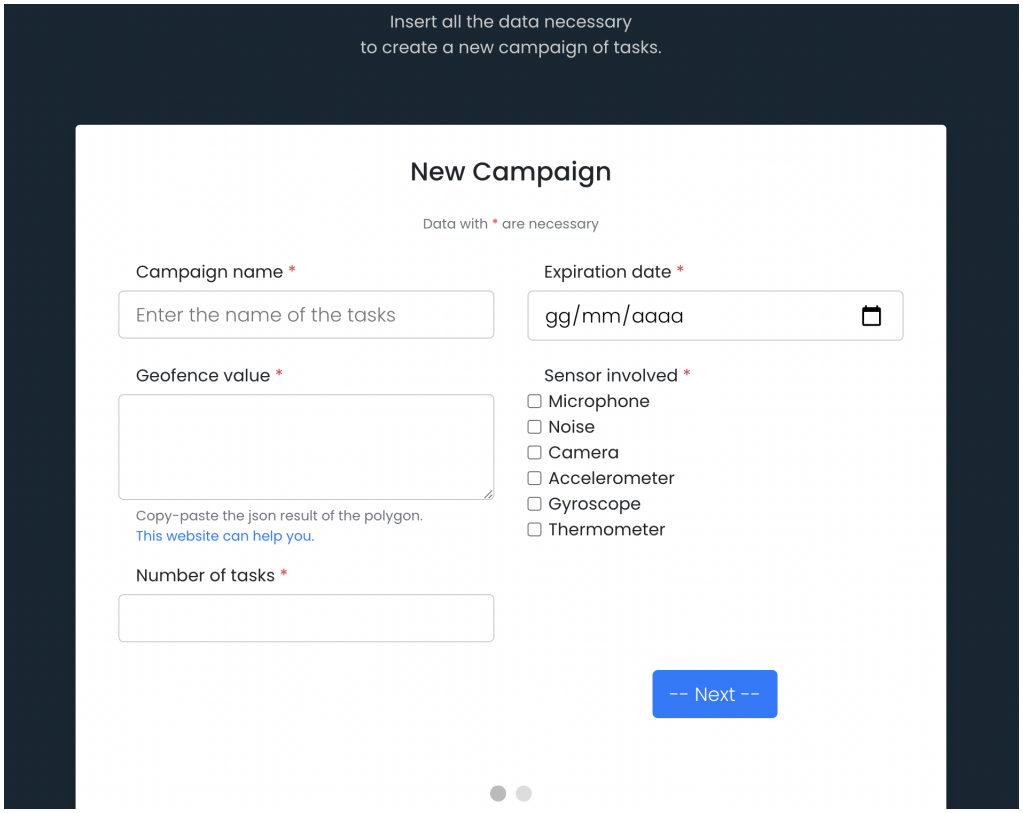

Figura 2.2: Screenshot del menu di inserimento di una nuova campagna

la tipologia di campagna e vedremo più avanti come influisce nell'uso della piattaforma. *crowdplatform-service* è poi progettata in maniera responsive in quanto si presume che i dispositivi principali sui quali verrà eseguita siano gli smartphone. Questa, ad esempio, è una delle tante scelte implementative che hanno una chiara motivazione tecnica.

Da un punto di vista logico il funzionamento dell'intero sistema è chiaro, ma per quanto riguarda le tecnologie utilizzate va definito un ulteriore approfondimento: la realizzazione di una web app piuttosto che un'applicazione vera e propria è stata frutto di utilizzo di vari framework software e di scelte implementative ben precise, improntate però a rendere l'utilizzo di tutto il sistema in maniera universale

2.2 Approfondimento tecnico

La necessità di spiegare certe scelte implementative deriva dall'esclusività della piattaforma nel contesto moderno, dove troppo spesso si tende a trasformare tutto in app, generando così ridondanza (nel caso di Android e iOS) e poca compatibilità con dispositivi di vario genere. Pertanto una web app "ultra compatibile" permette agli utenti di accedere al sistema da qualsiasi device, che sia esso un pc o uno smartphone, piuttosto che un tablet o una smart tv, ricalcando delle conseguenze molto importante. Si pensi anche allo scenario interattivo che coinvolge il crowdsourcer e l'admin: lavorare in un unico sistema è molto più conveniente da entrambi i lati, in quanto il datore di lavoro non deve creare un'applicazione apposita per quel task, il worker invece non deve necessariamente installare un'applicazione esterna. Il risultato è una semplificazione di *user experience* per gli utenti finali e non può fare altro che incrementare ancor di più l'incentivazione all'utilizzo. L'implementazione messa in campo permette inoltre all'interfaccia di adattarsi in base alla dimensione dello schermo, molto più di quanto accade in diverse applicazioni standalone.

Il contesto proposto conserva, tuttavia, alcune problematiche legate ovviamente all'utilizzo in un contesto web. Innanzitutto utilizzare dispositivi diversi significa sfruttare sensori diversi, pertanto le coordinate GPS percepite da un notebook, per esempio, potrebbero differire di parecchio rispetto ad uno smartphone che, solitamente, possiede un modulo GPS al suo interno: come vedremo, la geolocalizzazione è una componente fondamentale e non può essere interrotta durante il suo aggiornamento. Oltre a ciò, diventa necessario anche gestire la mancanza di alcuni sensori, perchè solitamente è piuttosto raro ritrovarsi un giorscopio in un pc portatile. In questi casi, qualora un utente esegua l'accesso ad un task da un dispositivo non totalmente compatibile, verrà riportato un warning, con il suggerimento di cambiare dispositivo per portare a termine il compito.

Fatta questa premessa sulla scelta relativa ad una web app responsive piuttosto che un'applicazione standalone, l'approfondimento tecnico riguar-

derà l'hardware coinvolto e tutte le componenti software che ne permettono il corretto funzionamento in un contesto web. Andiamo quindi a presentare le principali API che caratterizzano questo tipo di implementazione.

2.2.1 API relative ai sensori

Più che di smartphone (o dispositivi mobili) al giorno d'oggi si parla di veri e propri datacenter tascabili, in quanto anche durante l'utilizzo passivo del device esso invia in rete una moltitudine di dati derivanti dai propri sensori interni. L'utilizzo dello smartphone è strettamente connesso con lo sfruttamento della propria sensoristica: ruotando lo schermo per guardare un video significa utilizzare l'accelerometro, impostare una navigazione con Google Maps sottende l'uso del GPS, persino giocare ad un videogioco spesso coinvolge l'attività del giroscopio. Vi sono però alcuni contesti in cui diventa necessario estendere l'utilizzo dei sensori degli smartphone all'esterno delle app mobile, in quanto ad esempio un sito web potrebbe richiedere la geolocalizzazione anche da smartphone, oppure adattarsi in base all'utilizzo del device in verticale o in landscape.

Ma quali sensori la piattaforma è in grado di implementare? E soprattutto, come avviene l'integrazione di essi in un contesto web? La risposta è nelle API che sono state implementate nel sistema.

- ***Microfono***: necessario per andare a campionare dati vocali di vario genere. L'implementazione avviene attraverso le *MediaStream API*, ovvero delle componenti software Javascript che raccolgono l'intero supporto per l'acquisizione di contenuti multimediali.

- ***Fotocamera***: l'acquisizione di dati da questo sensore si limita, nel progetto, alle foto, in quanto il traffico dati generato dagli eventuali video sarebbe stato troppo elevato in fase sperimentale. Anche qui il supporto è garantito dalle *MediaStream API*.

- ***Accelerometro***: si tratta di un dispositivo che consente la misurazione e l'analisi dell'accelerazione lineare e angolare. Questa funzione è

essenziale in molti dispositivi e sistemi, utilizzati in quasi tutti i settori della nostra vita quotidiana. Le *Generic Sensor API* sono in grado di raccogliere tutti i dati provenienti dai sensori di movimento.

- ***Giroscopio***: tra i sensori più interessanti montati su smartphone, il giroscopio risulta essere essenziale per alcune funzionalità di quest'ultimo. Esso è in grado di rilevare i movimenti compiuti dallo smartphone nei tre assi X, Y e Z (spazio tridimensionale). Nello specifico il giroscopio permette allo smartphone di capire i movimenti che sta effettuando in maniera precisa e veloce. Anche questo sensore sfrutta le *Generic Sensor API*.

- ***GPS***: la geolocalizzazione GPS è invece possibile grazie alle *Geolocation API* che, attraverso due chiamate differenti, permettono di ottenere la posizione corrente oppure di aggiornare l'utente in base ad un intervallo definito. Il supporto viene garantito per tutti i browser principali, anche per le versioni iOS in questo caso.

Come citato poc'anzi, il supporto delle principali librerie è garantito su tutti i browser web, con un'eccezione però per gli smartphone Apple. In questo caso il funzionamento di tutto il sistema non è garantito in quanto le librerie riescono a riportare i valori corretti dei sensori solo su Android.

Il ruolo delle API è quello di fornire un'interfaccia tra il sensore fisico e il front-end umano, permettendo quindi di verificare in real time cosa il sensore sta facendo e, in alcuni casi, modificare anche il suo comportamento: anche con le API web riusciamo a modificare alcuni parametri, come ad esempio la frequenza di aggiornamento dell'accelerometro. Tutto ciò inizialmente era solo possibile sviluppando un'applicazione vera e propria, sfruttando le API di Android o iOS: prendendo in esame le prime, sono evidenti alcune differenze implementative, come spiega la tabella 2.1.

Sensor APIs comparision		
Sensor	**Mobile application**	**Web application**
Accelerometer	Adjustable frequency and precision	Only the frequency is adjustable
Gyroscope	Adjustable frequency and precision	Only the frequency is adjustable
Camera	All options available	Lower image quality
Microphone	All options available	Complete support but lower quality
GPS	Features battery saving options and interval settings	Three levels of accuracy are available

Tabella 2.1: Breve confronto tra le API mobile e web

2.2.2 NodeJS

Il linguaggio di programmazione principale di questo progetto è il *JavaScript*, in quanto si adatta alla perfezione in un contesto web dinamico come quello descritto finora. Tutti gli sviluppatori conoscono JavaScript in qualche misura, in quanto è il linguaggio più comune per realizzare alcuni task come le animazioni sulle pagine web o interazioni complesse tra di esse. La diffusione di Javascript è dovuta dalla necessità di dover rendere più dinamico possibile l'accoppiata HTML-CSS. Ogni pagina web costruita con puro HTML è inanimata, escludendo le animazioni CSS che però sono statiche. Per aggiungere effetti ed interazioni è necessario lanciare JavaScript nella mischia. Quindi come entra NodeJS in questo contesto?

Stiamo parlando di uno dei principali (se non il principale) framework Javascript in grado di implementare una logica client-server nel contesto di un linguaggio di programmazione principalmente orientato al front-end. Attraverso NodeJS possiamo quindi effettuare delle interazioni con un database, gestire le sessioni utente, implementare "codice complesso" attraverso i vari package disponibili e tanto altro ancora. Il vero senso di Node lo si capisce attraverso le librerie implementate; di seguito vi sono elencate.

ExpressJS : Una delle librerie maggiormente impiegate quando si crea un progetto in NodeJS. Si tratta di un framework flessibile e leggero che fornisce una serie di funzioni avanzate per le applicazioni web e per dispositivi mobili. Il framework Express consente di creare potenti API di routing e di impostare middleware per rispondere alle richieste HTTP, fornendo semplici meccanismi di debugging e una rapida integrazione con vari motori di templating. A sottolineare ancora la sua importanza, Express è anche uno dei quattro componenti basati su JavaScript del tech stack MEAN, insieme a MongoDB, Angular e Node.

MongoDB : Citato anche poc'anzi, si tratta della libreria di supporto al database Mongo. Questo framework è invece molto utile, in combinazione con il precedente, per instaurare una connessione al database e lanciare le query con estrema facilità e chiarezza, visto che anche queste vengono gestite in puro codice JSON. Diversi aspetti verranno poi chiariti nella sezione successiva.

Leaflet : Framework di riferimento per quanto riguarda la gestione della mappa e della geolocalizzazione. Attraverso Leaflet si riescono ad implementare le mappe dei vari fornitori principali; in questo caso sono state utilzzate le *OpenStreetMap*, facili da usare e soprattutto da modificare, in quanto si è fatto largo uso di marker e geofences. La cattura della latitudine e della longitudine può essere impostata dinamicamente, ad esempio in base al possibile mezzo di trasporto.

Bootstrap : Si tratta della libreria di interfaccia grafica più utilizzata al mondo. Bootstrap infatti permette, attraverso setup css già preconfigurati, di realizzare una UI apprezzabile e responsiva: tutti i componenti sono infatti predisposti per modificare il proprio comportamento sulla base della dimesione (e dell'orientamento) del display del device. In questo modo si ha un grosso risparmio di tempo, in quanto adattare tutte le regole css può risultare complesso, ed un risultato migliore. Nel caso del sistema presentato Bootstrap in realtà non viene invocato

direttamente dal motore NodeJS, ma è stato inserito come script nelle pagine html. Il risultato sarebbe stato comunque il medesimo.

Tutte le considerazioni relative a NodeJS valgono sia per il progetto *crowd-platform service* che per *crowdsourcer platform*, entrambi realizzati secondo lo stesso schema logico e con un'interfaccia simile. Infatti si nota come entrambe le piattaforme siano delle single-page application: questa ulteriore scelta implementativa deriva dalla volontà di avvicinare il più possibile la user experience di *YouCrowd* a quella di una normale applicazione per smartphone, in cui il contesto grafico segue una certa coerenza strutturale e di aspetto. Infatti a volte può risultare piuttosto "noioso" navigare all'interno di un sito web dal browser di uno smartphone.

Il vantaggio principale di NodeJS quindi risiede nella possibilità di creare in modo semplice e integrato delle web application, sfruttando le numerose librerie che ci vengono messe a disposizione. Ma le motivazioni dietro l'utilizzo di Node sono molteplici:

- Si adatta perfettamente a contesti real-time come chat, videogiochi e ovviamente progetti come quello che stiamo presentando. L'implementazione dei meccanismi asincroni, la gestione delle sessioni utente e il supporto a tantissimi framework grafici permettono tutto ciò.

- Si basa sul linguaggio Javascript, sicuramente più facile e leggero rispetto a Java e alla vera e propria controparte di Node, ovvero *Spring*. Quest'ultimo spesso viene scartato a priori proprio perchè il linguaggio su cui si basa è più complesso.

- NodeJS è una dei pochi framework a supportare pienamente lo streaming dei dati, quali file audio e video. Anche qui vengono in aiuto le varie librerie compatibili, ma lo streaming non sarebbe possibile se non vi fosse un overhead così basso.

2.2.3 MongoDB

All'aumentare dei dati in circolo, aumenta anche la persistenza degli stessi e pertanto la gestione in questo caso diventa fondamentale, soprattutto per un discorso di performance. MongoDB è un *DBMS* non relazionale, ovvero senza una struttura dei dati chiara e definita ma basata unicamente su una serie di documenti, apparentemente scorrelati tra di loro. Nel dettaglio, questo è un database di tipo *NoSQL* in quanto non è richiesto uno schema fisso (schemeless), evitando quindi le operazioni di giunzione (join) e scalando in modo orizzontale. Gli accademici e gli articoli si riferiscono a queste basi di dati come memorizzazione strutturata.

Il formato dei documenti è di tipo *JSON* e si adatta perfettamente ad un contesto NoSQL per svariati motivi. La semplicità di JSON ne ha decretato un rapido utilizzo specialmente nella programmazione in AJAX, infatti il suo utilizzo tramite JavaScript è particolarmente semplice in quanto il parsing, come vedremo successivamente, è molto rapito. Questo lo ha reso velocemente molto popolare a causa della diffusione della programmazione in JavaScript nel mondo del Web. MongoDB a sua volta ne sfrutta le potenzialità definendosi di fatto come un grosso container di contenuti JSON: in questo contesto è più appropiato parlare di documenti *BSON* che supportano un gran numero di dati diversi, come indicato di seguito.

- ***Boolean***: tipi di dato true e false.
- ***String***: stringhe di testo.
- ***Array***: insieme definito di dati di vario genere.
- ***Date***: relativo al formato delle date.
- ***Binary***: formato binario per il salvataggio di files.
- ***Double, Int, Decimal, Long***: tipi numerici.

Di seguito vi è uno snippet di un documento JSON presente nel db.

```
{
    "_id":{"$oid":"61c887d81e61f5aabd43fa94"},
    "campaign":"TASK1",
    "n_tasks":38,
    "expdate":{"$date":"2023-01-01T00:00:00.000Z"},
    "geofence":[{...}],
    "sensorlist":{
        "0":{
            "type":"accelerometer",
            "partopp":"opp",
            "temporal_samp":5000,
            "space_samp":5,
            "n_samples":10
        }
    },
    "tabtype":"campaign"
}
```


Queste righe di codice descrivono la campagna denominata *TASK1*, avente 38 tasks disponibili ed una data di scadenza ovviamente futura. Il geofence è racchiuso in un array contenente una lista di coordinate, ovvero una sequenza di coppie chiave-valore che indicano i singoli punti (latitudine e longitudine) che compongono l'area di rilevamento. La voce *sensorlist* contiene a sua volta altri documenti JSON che descrivono i parametri relativi ai sensori coinvolti nella campagna: le voci riguardano il tipo, la caratteristica partecipatoria o opportunistica, il sampling temporale, il sampling spaziale e il numero di samples richiesti. L'ultima voce (*tabtype*) ha un'utilità solo durante l'esecuzione di query e indica che il documento descrive una campagna.

La struttura di questo documento JSON va a definire una campagna, pertanto sarà compito di Node caricare tutti i documenti con la dicitura *tabtype* e mostrarli all'utente non appena egli avrà fatto il login. Continuiamo con la struttura degli altri due tipi di documenti JSON coinvolti.

```
{
    "_id":{"$oid":"61cd9eae59d6400f94c483f9"},
    "crowdsourcer":"Dome",
    "campaign":"TASK4",
    "date_assign":{"$date":"2021-12-30T11:57:34.185Z"},
    "closed":true,
    "microphone_remaining":0,
    "camera_remaining":0
}
```


Questo documento invece riguarda l'assegnazione ad un utente di un task, con tutte le informazioni necessarie relative allo stato di completamento. Sono infatti indicati il nome del crowdsourcer, il nome della campagna, la data di assegnazione e le informazioni sullo stato di completamento: le voci contenenti "_remaining" indicano, per ogni sensore coinvolto, quanti samples rimangono; mentre la voce *closed* definisce se un task è concluso o meno.


```
{
    "_id":{"$oid":"61cd9e1359d6400f94c483f5"},
    "camptask":"TASK3",
    "sample":"0.01439235371490576 8",
    "sender":"Dome",
    "position":{"lat":44.5146918,"lng":11.3166138}
}
```


L'ultimo JSON riguarda invece l'invio del sample al database: di fatto si può dire che questo documento contiene il dato utile all'admin. Infatti abbiamo il riferimento al nome della campagna, il sample (in questo caso un dato numerico), il riferimento all'utente e la posizione di invio. In particolare quest'ultimo dato può essere di fondamentale importanza per capire se il sample proviene o meno dalla zona di compatibilità.

MongoDB mette a disposizione anche una vasta suite software in grado di supportare diverse tipologie di applicazioni. In questo caso è stato sfruttato *MongoDB Compass*, ovvero un ambiente grafico con il quale è possibile interagire direttamente con i documenti BSON senza passare per l'interfaccia a riga di comando. Quest'ultima è sicuramente più performante per la gestione del database, ma meno semplice e immediata graficamente.

2.3 Componente umana

Nel contesto strutturale del sistema finora descritto la componente umana è di fondamentale importanza, in quanto seppur l'acqusizione di dati sia automatizzata essa richiede una certa consapevolezza da parte degli utenti. Lo studio già presentato relativo all'interfaccia grafica viene incontro anche a questa esigenza, in quanto sia un neofita che un utente esperto devono saper portare a termine tutte le operazioni previste, attraverso la semplificazione dell'utilizzo della piattaforma. Nei capitoli successivi si andrà anche a quantificare la capacità degli utenti che hanno testato la piattaforma, ma prima bisogna analizzare con precisione tutte le azioni possibili da parte degli amministratori e dei clienti.

2.3.1 Utente worker

Il crowdsourcer è quella tipologia di utente che esegue una serie di task a sua scelta al fine di generare un profitto economico per se stesso e un dato rilevante per l'amministratore. Il soggetto fisico del worker non è definito, in quanto non vi sono limiti all'accesso e all'utilizzo della piattaforma. Come visibile dalla Fig. 2.3 nello schema interattivo è presente al lato estremo, in quanto egli rappresenta l'entità che interagisce con la parte front-end ed è in grado di eseguire le seguenti operazioni.

- Accesso alla piattaforma.
- Iscrizione ad una campagna.

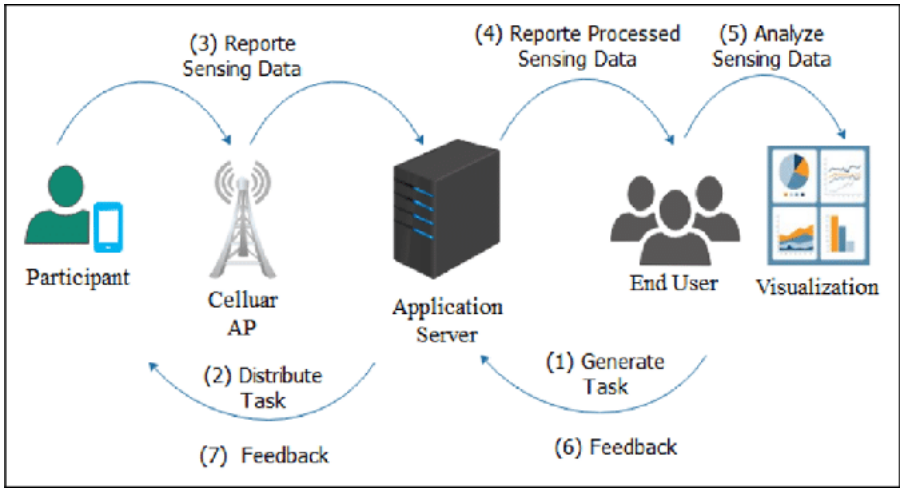

Figura 2.3: Schema di interazione tra le varie entità

- Esecuzione di un task.
- Sospensione di un task e selezione di un altra campagna.

Non vi sono requisiti tecnici per diventare un worker, infatti la piattaforma è accessibile a tutti in maniera libera e i tasks sono eseguibili fino alla loro data di scadenza. La remunerazione dei worker avviene direttamente da parte dell'administrator e solo nel caso in cui il task sia completato correttamente. Un worker può eseguire un quantitativo multiplo di tasks diversi tra di loro, infatti attraverso l'interfaccia è possibile eseguire lo switch tra i tasks ai quali l'utente è iscritto.

2.3.2 Utente administrator

Viene definito come admin l'utente che ha un particolare interesse per una certa campagna di studio: solitamente si tratta di un ricercatore, ma non è una condizione necessaria. Viene definito anche come *End user* in quanto di fatto si tratta dell'utente finale, ovvero dell'ultima entità alla quale i dati giungono e ne diventano proprietari. Può svolgere diverse operazioni:

- Pubblicazione di una campagna di tasks.

- Lettura e modifica dei dati nel DB.
- Ricezione dei samples.
- Approvazione dei pagamenti

Per modifica dei dati del db si intende, ad esempio, la revoca dell'assegnazione del task ad un worker: questo può accadere quando un task è scaduto, oppure se è stato assegnato ma l'utente non ha mai contribuito alla campagna. In quest'ultimo caso, come anche per quanto riguarda i pagamenti, non vi è nessuna automazione da parte di *YouCrowd*, pertanto sarà responsabilità dell'administrator procedere con l'operazione.

Nonostante non vi sia l'esigenza specifica di approfondire ulteriormente i ruoli della componente umana va rammentata l'importanza nell'intero sistema: oltre al profitto economico, uno degli obiettivi di *YouCrowd* è quello di generare conoscenza, attraverso lo studio di dati. Il supporto alla componente umana deriva dai tecnicismi analizzati fin'ora, i quali però necessitano di un ulteriore spiegazione tecnica attraverso il codice del software che caratterizza il progetto. Ciò che sarà spiegato nel prossimo capitolo riguarda un ulteriore "discesa di livello" nell'analisi dell'architettura software.

Capitolo 3

Implementazione del software

I framework presentati nel precedente capitolo espongono una serie di metodologie attraverso i quali risolvere problemi ben definiti: ad esempio *Leaflet* supporta il contesto della geolocalizzazione, oppure *MongoDB* permette il salvataggio dei dati in modo sparso. Ciò che *YouCrowd* vuole rappresentare è anche una precisa interpretazione di questo insieme di librerie, in modo tale da fornire una chiave di lettura software per il contesto del Mobile Crowdsensing e permettere anche un'ulteriore comprensione tecnica. Si procede quindi ad un'analisi, nelle due sezioni, di tutto il workflow necessario per realizzare l'intera piattaforma, chiaramente suddiviso nei due progetti già menzionati.

3.1 Crowdsourcer-platform

Il progetto con questo nome in codice è destinato agli amministratori di sistema, in modo da poter inserire facilmente tutti i dati relativi ad una nuova campagna di crowdsourcing. L'interfaccia (come mostrato nella Fig 2.2) è semplice, dato che si tratta di una single-page che non fa altro che l'inserimento di una grossa query direttamente al DB.

I dati da inserire in fase di configurazione sono molteplici, ma sicuramente il meno immediato da comprendere è il *Geofence value*, ovvero l'area di compatibilità del task. Per inserirla bisogna andare all'indirizzo al link

che rimanda al servizio *geojson.io* che, come suggerito dal nome, fornisce uno strumento per costruire il proprio file *GeoJSON*, ovvero un documento JSON utilizzato per archiviare una collezione di geometrie spaziali, come punti, linee e, appunto, geofences. Un GeoJSON generico relativo ad un poligono ha il seguente formato:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [ ... ]
        ]
      }
  ...
}
```

Per raccogliere tutti i dati è stato inserito un form dinamico HTML, il quale aggiunge o rimuove diverse voci relative ai sensori in base alle opzioni scelte. Il form raccoglie tutte le seguenti informazioni:

- **Campaign name**: Nome della campagna.
- **Expiration date**: Data di scadenza (ovviamente futura).
- **Geofence value**: Documento GeoJSON.
- **Number of tasks**: Quantitativo di tasks per la campagna.
- **Sensor involved**: Selezione dei sensori necessari.

Nella pagina successiva troviamo invece i dettagli relativi ai sensori e per ognuno di essi abbiamo:

- Selezione sulla caratteristica partecipatoria od opportunistica. Notare che nel caso fotocamera, microfono e rumore l'acquisizione non può essere opportunistica.

- *Temporal sampling*: specifica sui millisecondi per attivare il sampling temporale.

- *Space sampling*: numero espresso in metri per la validazione del sampling spaziale.

- *Total number of samples*: numero totale di samples richiesti.

L'acquisizione dei dati in relazione ai due valori di sampling verrà approfondita nella sezione successiva. Una volta inseriti tutti i dati si potrà procedere al salvataggio della campagna, quindi all'invio di una query al database. Attraverso degli snippet in Javascript vediamo come effettuale la connessione al database e quindi inviare delle query.

```
1 const {MongoClient} = require("mongodb");
2 const url = "mongodb://127.0.0.1:27017";
3 const client = new MongoClient(url, {
4   useNewUrlParser: true,
5   useUnifiedTopology: true,
6 });
7 ...
8 app.listen(PORT, () => console.log(`Server Running at port
      ${PORT}`));
9 client.connect();
```

In questo modo si richiede innanzitutto la libreria di supporto per *MongoDB*, successivamente si definisce l'URL (in questo caso localhost) e si effettua la connessione una sola volta per la porta specificata. Per *crowdsourcer-platform* abbiamo la porta 4000, mentre per *crowdplatform-service* si ha invece la 3000. La chiamata *app.listen()* è propria di ExpressJS ed è utile per mettere in ascolto l'host alla porta definita.

Express è fondamentale per le richieste al server, ovvero per effettuare le varie GET e POST che saranno via via necessarie per portare a termine tutte le operazioni. Nel caso del salvataggio della campagna abbiamo una POST denominata *sendCampaign*.

```
1  //send campaign to DB
2  app.post('/sendCampaign', async function(req, res) {
3    res.header("Access-Control-Allow-Origin", "*");
4    res.setHeader("Content-Type", "text/html");
5    console.log(`Inserting tasks campaign: ${
         req.body.campaign}`);
6    let ins = await newCampaign(client, req.body);
7  })
8  ...
9  //inserting a new campaign
10 async function newCampaign(clientP, fragment) {
11   try {
12     let datedb = new Date(fragment.expdate);
13     const database = clientP.db("test");
14     const collection = database.collection("inventory");
15     fragment.expdate = datedb;
16     fragment.tabtype = "campaign";
17     //query content
18     data = await collection.insertOne(fragment, function(
           err, res) {
19       if (err){
20         throw err;
21       }
22     });
23     return data;
24   } finally {
25     console.log("Inserted");
26   }
27 }
```

La POST in questo caso va a chiamare la funzione *newCampaign* che, come tutte le altre funzioni richiamate nelle varie GET e POST, è ovviamente

asincrona ed ha in parametro la costante relativa al client: questa è necessaria in quanto la connessione si effettua una sola volta, pertanto è utile per ottenere la collection di riferimento (in questo caso *inventory*) e quindi inviare la query. Quest'ultima, nel caso di un inserimento di nuovi dati, si effettua attraverso la chiamata *insertOne()*, all'interno della quale si passa in parametro il documento JSON che si intende inserire. In questo unico caso il JSON è passato direttamente dal client in quanto si tratta di un documento piuttosto lungo. Ma come fa il client a inserire nella richiesta tutto il JSON? La risposta è nello snippet sottostante.

```
1 fetch('/sendCampaign', {
2     method: 'POST',
3     body: JSON.stringify(jsonFragment),
4     headers: {'Content-Type': 'application/json'},
5 }).then(response => console.log(response));
```

Si va ad invocare la POST request attraverso la funzione Javascript *fetch()*, specificando appunto il metodo e il contenuto; in questo caso un'unica stringa JSON contenente tutto il fragment.

Il package *crowdsourcer-platform* ha il solo compito di inviare un unico documento JSON contenente tutte le specifiche relative ad una cammpagna di task, pertanto la complessità tecnica non è eccessiva. Discorso totalmente diverso invece per *crowdplatform-service*.

3.2 Crowdplatform-service

Il core principale di tutto il progetto si trova in questo package, in cui un utente worker può effettuare il login e, come mostrato nella Fig. 3.1, scegliere quale campagna eseguire in base alla propria posizione geografica, per poi eseguirne il task. Andiamo ad analizzare per gradi tutti gli aspetti.

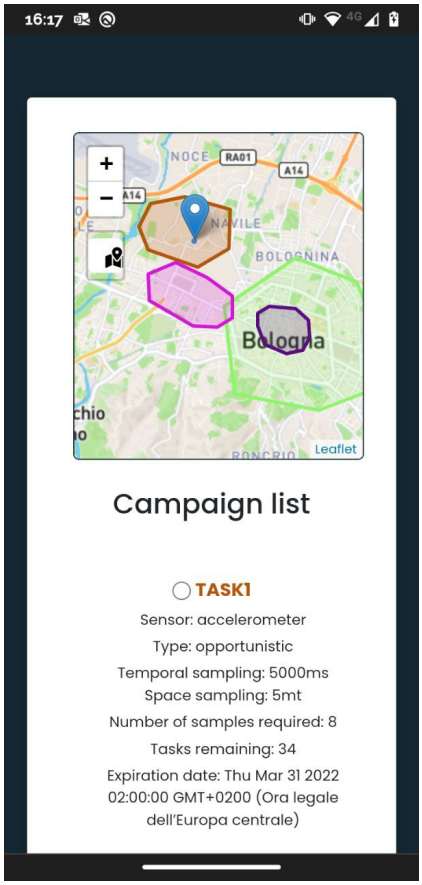

Figura 3.1: Screenshot della schermata di scelta del task. Notare le differenti aree di acquisizione dei vari task.

3.2.1 Scelta del task

Il login alla piattaforma avviene semplicemente inserendo il proprio nome utente e successivamente si andrà subito alla schermata di scelta del task. Uno degli elementi fondamentali di questo progetto riguarda proprio il salvataggio della sessione utente.

```
1 app.use(sessions({
2   secret: "thisismysoccsecrctekeysshhlol",
3   cookie: { maxAge: oneDay },
4   resave: false
5 }));
```

Per il mantenimento della sessione viene utilizzato il pacchetto *express-session* proprio di ExpressJS, ove si va a criptare la sessione con un codice e quindi mantenere un cookie nel browser con le proprie info di accesso, incluso il task attuale: quest'ultima in particolare è una variabile che andrà definita in fase di scelta del task. Con il salvataggio della sessione l'utente si ritroverà ad un bivio alla riapertura della web app.

```
1 app.get('/', (req, res) => {
2   session = req.session;
3   if (session.userid) {
4     res.sendFile('public/views/index.html', options);
5   } else {
6     res.sendFile('public/views/login.html', options);
7   }
8 });
```

Con questa GET si va ad intercettare le informazioni di sessioni, ovvero se esistono si va alla home del proprio task mentre se non esiste alcun *userid* allora si procede di nuovo al login.

Dopo il login ci si ritrova nella schermata della Fig. 3.1 ove sono listati tutti i task disponibili, quindi si può selezionarne uno dalla lista dettagliata e iniziarlo immediatamente. Qui è presente uno degli elementi fondamentali di tutto il sistema, ovvero la mappa di Leaflet.

```
1 var refTasks = L.map('maptasks').setView([44.4927,
    11.3083], 12);
2 ...
3 L.tileLayer(
4     'https://api.mapbox.com/...', {
5         maxZoom: 18,
6         id: 'mapbox/streets-v11',
7         tileSize: 512,
8         zoomOffset: -1,
9     }).addTo(refTasks);
```

La mappa viene dichiarata attraverso l'oggetto generico Leaflet "L" e si aggancia ad una tag con l'id indicato tra parentesi. I valori di *setView()* sono invece relativi alla posizione di default e allo zoom. Notare che Leaflet ragiona a tiles, infatti ogni livello deve essere dichiarato ed ha uno stile ben definito. Il primo livello, ovvero quello base, è ovviamente relativo alla mappa stessa ed è dichiarato attraverso la chiamata *tileLayer()*, in cui si va a definire alcuni parametri di visualizzazione, oltre che alla chiave per le api che sono state scelte per questa mappa, ovvero le più comuni di *Mapbox*. Su questa tile sono state anche disegnate le aree di rilevamento, ovvero i cosidetti *geofence*: in questo caso si sfrutta la chiamata *polygon()* e si impostano i parametri relativi alle coordinate e al colore (randomico) del poligono che sarà disegnato. Ad ogni campagna è stato assegnato un colore casuale che, nella mappa, risulterà essere lo stesso del poligono, in modo tale da facilitare l'individuazione dell'area da parte dell'utente. Nell'esempio l'intero poligono non è visibile in quanto più grande dell'area visualizzata, ma basterebbe sbloccare il blocco della visualizzazione ed effettuare un pinch-out per vedere la propria posizione rispetto al geofence.

La propria posizione è indicata dal marker nella mappa e la sua implementazione riguarda il codice che segue.

```
1  function onLoc(e) {
2      if (current_position) {
3          refTasks.removeLayer(current_position);
4          refTasks.removeLayer(current_accuracy);
5      }
6      var radius = e.accuracy / 2;
7      current_position = L.marker(e.latlng).addTo(refTasks).
           bindPopup("You are within " + radius + " meters
           from this point");
8      current_accuracy = L.circle(e.latlng, radius).addTo(
           refTasks);
9  }
10 function locateNow() {
11     refTasks.locate({
```

```
12          setView: lock,
13          maxZoom: 13
14      });
15  }
16  //location events
17  refTasks.on('locationfound', onLoc);
18  refTasks.on('locationerror', locError);
19  // call locate every 3 seconds... forever
20  setInterval(locateNow, 3000);
```

La funzione *onLoc()* va a piazzare un marker nel punto approssimativo di geolocalizzazione, indicato da un cerchio il cui raggio indica proprio l'ammontare in metri della precisione del GPS in quel momento. Entrambi gli elementi vengono dinamicamente rimossi e aggiunti ogni qual volta viene rilevata una nuova posizione: ciò è possibile in quanto *onLoc()* viene richiamata dall'evento *locationfound* di Leaflet, ovvero un'astrazione di alto livello per richiamare la geolocalizzazione. Il focus sulla mappa viene invece gestito da *locateNow()* che va a spostare ogni 3 secondi la posizione della mappa in maniera centrata rispetto al marker.

Ma cosa succede non appena un utente seleziona un task disponibile e conferma la propria scelta? Il form HTML è associato alla action *openTask*, ovviamente relativa alla richiesta lato server che segue

```
1  app.post("/openTask", async function (req, res) {
2    res.header("Access-Control-Allow-Origin", "*");
3    console.log("selected: " + req.body.campaign);
4    thistask = req.body.campaign;
5    await assignTask(client, session.userid, thistask);
6    //now let's open the task
7    res.sendFile('public/views/index.html', options);
8  });
```

La richiesta POST acquisisce dal client le informazioni circa l'id dell'utente e il titolo del task da assegnare, dopodichè queste informazioni vengono passate alla funzione asincrona *assignTask()*, che svolge diverse operazioni

direttamente sul database. Notare come nella prima parte della query che si va ad inserire viene assegnata la voce *closed* come valore booleano falso, ad indicare che il task logicamente deve ancora essere concluso.

```
async function assignTask(clientP, username, tTask) {
  try {
    ...
    //first find the task
    let info = await collection.findOne({
      campaign: tTask
    });
    var query = {
      ...
      closed: false
    };
    for (i = 0; i < Object.keys(info.sensorlist).length; i
        ++) {
      //insert samples remaining sensor per sensor
      let stype = info.sensorlist[i].type + '_remaining';
      let nsamples = info.sensorlist[i].n_samples;
      query[stype] = nsamples;
    }
    await collection.insertOne(query, function (err, res)
        {
      if (err) throw err;
    });
    //now decrease number of tasks in the campaign
    await collection.updateOne({
      campaign: tTask
    }, {
      $inc: {
        n_tasks: -1
      }
    });
  }
  ...
}
```

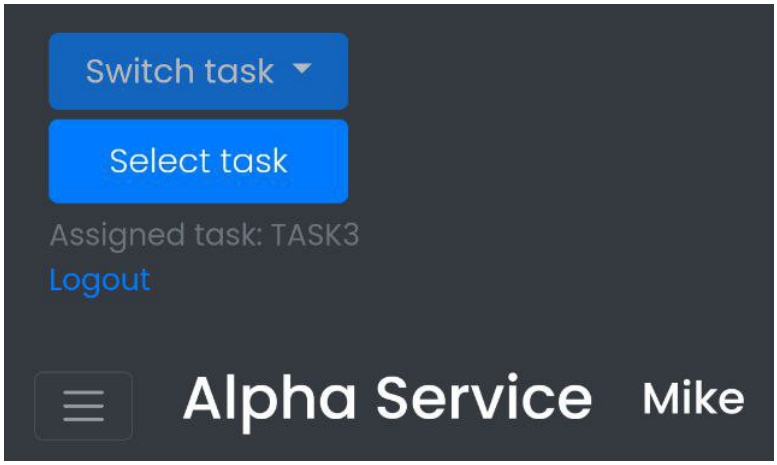

Figura 3.2: Dettaglio del menù del crowdsourcer

- Prima di tutto si ricerca il documento relativo al task che si intende assegnare e si costruisce una parte della query.
- In base ai sensori selezionati se ne inseriscono le voci nel JSON con il quantitativo di samples richiesti, che saranno poi aggiornati in rimanenti durante l'esecuzione del task.
- Una volta arricchito il JSON lo si inserisce nel DB e si decrementa il contatore di tasks disponibili per quella campagna attraverso *$inc* all'interno della query di aggiornamento *updateOne()*.

Una volta selezionato il task l'utente verrà reindirizzato alla home vera e propria del progetto e potrà iniziare le sue operazioni.

3.2.2 Visualizzazione del task

La schermata home varia in base al task richiesto ed è proprio questo il principale punto di forza dell'intera implementazione: si ha, in questo modo, la possibilità di sviluppare una miriade di applicazioni diverse a partire da un codice sorgente dinamico. Nel caso presentato dalla Fig 3.3 abbiamo la schermata home di un task che sfrutta la fotocamera e il giroscopio, con tutti gli elementi necessari predisposti per avviare l'attività. Si ha poi il

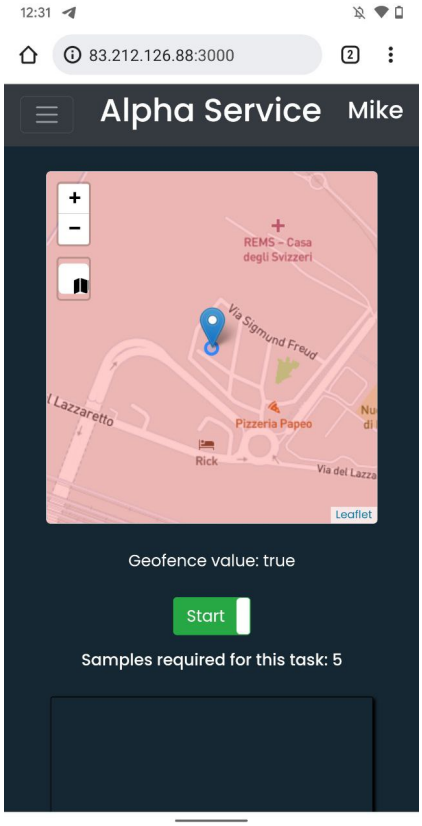

Figura 3.3: Screenshot della schermata home

pulsante di start, che di fatto va ad "accendere" i sensori, e la mappa, qui ingrandita in quanto non è inserita all'interno di un menù descrittivo, che ha un'implementazione identica a quella mostrata in precedenza.

Come visibile dalla Fig 3.2 (nella sezione precedente) l'utente può accedere anche ad un menù per switchare su un altro task, selezionare un secondo task oppure effettuare il logout. In questo caso il select di un task non fa altro che rimandare di nuovo alla pagina di selezione, permettendo una seconda assegnazione del task: in questo caso chiudendo la web app e rientrando in login viene caricato il task con scadenza inferiore. Si può comunque richiamare il secondo task attraverso il menù a tendina switch.

Nella home page entrano in gioco diversi script Javascript che, per struttura, sono esterni alla pagina html e separati in base al sensore. Pertanto

abbiamo gli script per: accelerometro, giroscopio, fotocamera, microfono e rumore. Oltre a questi abbiamo anche lo script generico *functions.js* che espone alcune funzioni di supporto per operazioni di vario genere. In generale negli script sono racchiuse tutte le implementazioni dei sensori che, in base al task richiesto, vengono caricate. La logica è molto simile in tutti i casi tranne per quanto riguarda l'implementazione di ogni sensore, in quanto derivano da API diverse: come visibile nel listing 3.1 abbiamo diverse similitudini tra microfono e fotocamera, a sua volta tra accelerometro e giroscopio.

```
1  //camera
2  navigator.mediaDevices.getUserMedia({
3      audio: false,
4      video: {
5          facingMode: {
6              exact: "environment"
7          },
8          width: width,
9          height: height
10     }
11 }).then(function (stream) {
12     video.srcObject = stream;
13     video.play();
14 })
15 ...
16 //microphone
17 const stream = await navigator.mediaDevices.getUserMedia({
18     audio: true,
19     video: false
20 });
21 const mimeType = 'audio/webm';
22 recorder = new MediaRecorder(stream, {
23     type: mimeType
24 });
25 ...
26 //accelerometer
27 const sensAcc = new Accelerometer({
```

```
28    frequency: 5
29  });
30  ...
31  //gyroscope
32  const sensor = new Gyroscope({
33    frequency: 5
34  });
35  ...
36  //noise
37  const meter = new DecibelMeter('unique-id');
```

Listing 3.1: provaprova

Al caricamento della pagina vengono chiaramente caricate tutte le informazioni relative al worker, tra queste vi sono anche i dati relativi al task e al geofence di validità.

```
1  postRequest('/getTask', {
2      taskname: ses
3  }).then(data => drawTask(data));
4  ...
5  function drawTask(model) {
6      informations = model;
7    console.log(new Date(model.expdate) + " expdate");
8    if ((new Date() <= new Date(model.expdate)) == true) {
9        console.log("Building the task");
10     //create the geofence
11     createGeofence(model.geofence);
12     //build sensor for the first time
13     buildSensor(model);
14   } else {
15     console.log("TASK EXPIRED");
16     //draw output for task expired
17   }
18 }
```

In questo snippet di codice è mostrata innanzitutto la POST *getTask* che

non fa altro che rigirare le infomazioni sulla tipologia di task da costruire, prendendo le informazioni dal relativo JSON: la funzione *postRequest* è una reinterpretazione della normale fetch ed è definita in *functions.js*. La chiamata *drawTask()* invece si prende in parametro il documento JSON, ne valuta la data di scadenza per poi effettuare due operazioni principali per il setup di tutto l'ambiente.

- **Creazione del geofence**: attraverso la funzione *createGeofence()*, avendo a disposizione le coordinate in parametro, si va a creare un poligono di colore rosso nella mappa.
- **Implementazione del sensore**: *buildSensor* estrapola la lista di sensori all'interno del JSON e va a "costruire" i sensori in base alla tipologia di essi. Infatti nei vari script relativi ai sensori vi è una chiamata *build* che va ad inserire dinamicamente gli elementi HTML fondamentali per l'implementazione

Come già specificato poc'anzi il bottone Start serve ad avviare l'acquisizione dei samples, prima di ciò i sensori sono comunque presenti ma in unno stato disattivo. Cosa accade? E come avviene l'acqusizione?

3.2.3 Esecuzione del task

Allo start tutti gli elementi si predispongono per cominciare ad acquisire dati. In particolare appaiono i vari bottoni di acquisizione per i sensori partecipatori mentre quelli opportunistici, secondo le regole di sampling, cominceranno subito ad acquisire dati. Ma andiamo con ordine.

```
1 ...
2 REF_POSITION = isMarkerInsidePolygon(current_position);
3 //if you're out of geofence, disable start
4 if (REF_POSITION) {
5   document.getElementById('bt_task').disabled = false;
6 } else {
7   document.getElementById('bt_task').disabled = true;
```

```
8 }
9 ...
```

Innanzitutto non è ammessa l'acquisizione di samples al di fuori del geofence di compatibilità, ciò significa che se ci si trova in questa condizione la pressione dello start non avrà alcun effetto. Il caso in cui si esce dal geofence durante l'acquisizione è modellato dallo snippet di codice.

La variabile *ref_position* si aggiorna ad ogni acquisizione della posizione, infatti questo estratto di codice si trova all'interno della funzione *onLocationFound()* che è del tutto simile alla *onLoc()* vista in precedenza. Il valore ricevuto da *isMarkerInsidePolygon()* sarà true solo se ci si trova all'interno del geofence, infatti solo nel caso contrario si andrà a disabilitare il bottone di start, quindi ad interrompere, se in esecuzione, l'acqusizione di samples. L'attivazione dei sensori comporta la visualizzazione dell'intera interfaccia: infatti, come visibile dalla Fig 3.4, la UI cambia e lo fa attraverso un ulteriore passaggio. Lo *startService()* comporta la chiamata delle varie funzioni *setSensor*, ovviamente in base alla tipologia di sensore in gioco. Pertanto diventa possibile iniziare la propria attività di crowdsourcing.

Prima di spiegare come vengono inviati i vari dati al database è doveroso specificare gli elementi del sampling temporale e spaziale, per vedere come entrano in gioco in questa fase.

- Il **sampling temporale** è un valore espresso in millisecondi che indica l'intervallo di tempo minimo necessario affinchè si possa procedere ad una nuova acquisizone.

- Il **sampling spaziale** invece si tratta di un valore in metri che esprime la distanza minima di acquisizione tra un sample e l'altro.

Questi due tipi di sampling sono sempre in abbinata, ciò significa che entrambe le condizioni devono essere rispettate per procedere ad una nuova misurazione. La verifica sul sampling temporale è molto semplice, come visibile dal listing 3.2.

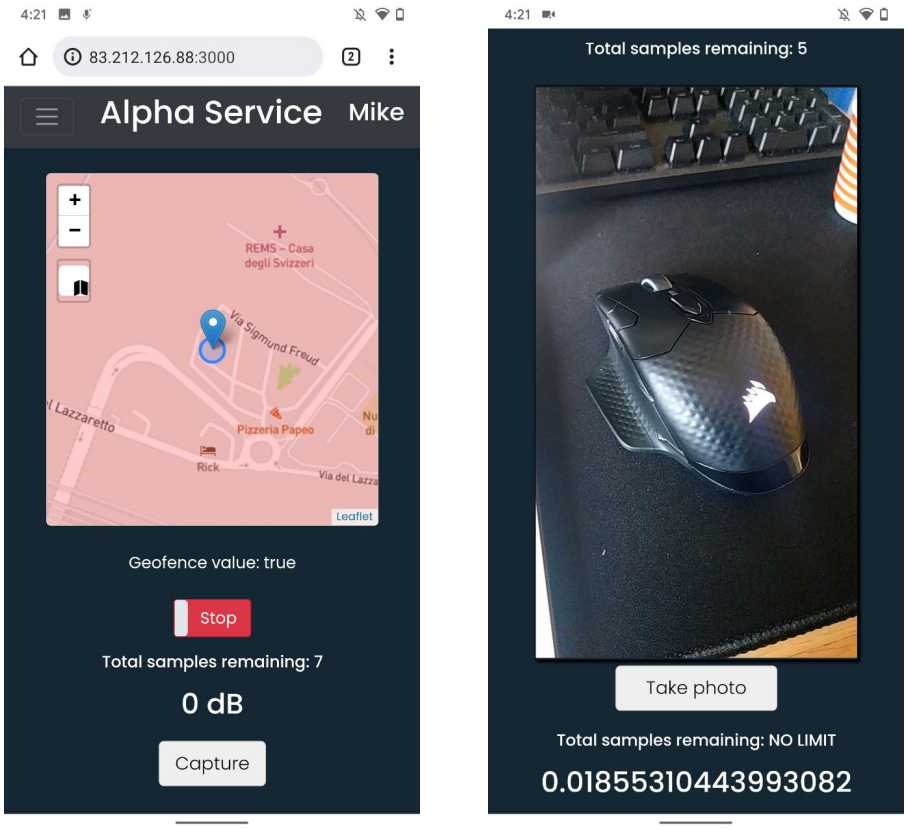

Figura 3.4: Screenshot di due tasks differenti in esecuzione

```
function tsRespected(temp_samp) {
  return new Promise((resolve, reject) => {
    setTimeout(function () {
      console.log("TS is ok.");
      resolve(true);
    }, temp_samp);
  })
}
```

Listing 3.2: Funzionamento del sampling temporale

la funzione *tsRespected* legge in parametro il valore di sampling ed implementa un semplice timeout al termine del quale viene restituito un valore true. L'implementazione avviene attraverso una *promise* permettendo quin-

di di rispettare il flusso asincrono di esecuzione, senza quindi che il timeout vada ad interrompere altri thread in esecuzione. Diverso è invece il discorso relativo all'implementazione del sampling spaziale.

```
1  function ssRespected(space, space_samp) {
2    return new Promise((resolve, reject) => {
3      //function loop every second that verify the position
4      function pp() {
5        let samplePosition = L.latLng(space.getLatLng().lat,
             space.getLatLng().lng);
6        //let respected = false;
7        setTimeout(function () {
8          let markerNow = exPos();
9          let newPosition = L.latLng(markerNow.getLatLng().
               lat, markerNow.getLatLng().lng);
10         if (space_samp < samplePosition.distanceTo(
               newPosition)) {
11           console.log("SS is ok.");
12           resolve(true);
13         } else {
14           console.log("Go away.");
15           pp();
16         }
17       }, 500);
18     };
19     pp();
20     //end of the manual loop
21   });
22 }
```

La funzione *ssRespected* acquisisce in parametro la posizione dell'invio del sample e il valore di sampling spaziale, in modo tale da confrontare ogni mezzo secondo la posizione attuale con quella del sample. In particolare, attraverso la funzione di Leaflet *distanceTo()*, si va a definire la distanza tra le due posizioni e quindi confrontarla con il valore di sampling spaziale. Se la condizione rispettata, si rilascia la promise con un true, altrimenti si ripete la

procedura. Si sottolinea ancora l'importanza della condizione asincrona delle due funzioni, condizione che permette alla web app di proseguire il proprio flusso senza che vi siano delle interruzioni.

Queste due funzioni vengono richiamate durante il processo di invio dei samples da parte dei sensori e se e solo se restituiscono due valori true allora si può procedere con il secondo invio. Rimane comunque anche in questo caso da fare una distinzione tra l'acquisizione del sensore partecipatoria o opportunistica. Analizziamo entrambi i casi prendendo spunto dall'implementazione dell'accelerometro.

Caso partecipatorio : Al bottone di acquisizione viene associato un listener che prevede innanzitutto il salvataggio del sample (in questo caso è un dato testuale), quindi la cattura della posizione e poi l'invio che avviene attraverso la POST *sendSample*. La funzione che richiama i due sampling (*accelerometerSamplings*) viene richiamata al termine di tutto ciò e va a bloccare il bottone finchè non siano rispettate entrambe le condizioni. Il codice di riferimento è il listing 3.2.

Caso opportunistico : Il dato viene catturato ed inviato solo nel caso in cui le condizioni di sampling siano entrambe *true* e questo controllo viene effettuato all'inizio. Ciò significa che, non essendoci alcun controllo sull'invio dei samples, ogni qual volta che vengano rispettate le condizioni l'utente può verificare l'invio automatico del dato, il quale viene inviato secondo le modalità già definite (vedi listing 3.3).

```
1 //set the button listener
2 capAcc.addEventListener('click', function () {
3     //send db sample
4     let magvalue = status1.textContent;
5     document.getElementById("accget").innerHTML =
          magvalue;
6     let posClick = exPos();
7     postRequest('/sendSample', {
8         sample: magvalue,
```

```
9           type: 'accelerometer',
10          coords: {
11              lat: posClick.getLatLng().lat,
12              lng: posClick.getLatLng().lng
13          }
14      }).then(data => {...});
15  }
```

Listing 3.3: Descrizione cattura partecipatoria

```
1  //if the two samplings are true, send data
2  if (temp_ris === true && space_ris === true) {
3      //send db sample
4      let magvalue = status1.textContent;
5      document.getElementById("accget").innerHTML =
           magvalue;
6      postRequest('/sendSample', {
7          sample: magvalue,
8          type: 'accelerometer',
9          coords: {
10             lat: posClick.getLatLng().lat,
11             lng: posClick.getLatLng().lng
12         }
13     }).then(data => {...});
14 }
```

Listing 3.4: Descrizione cattura opportunistica

Cosa succede quando un nuovo sample viene inviato al DB? La POST responsabile di ciò è *sendSample* e, come visibile dagli snippet precedenti, non invia il solo sample ma anche l'identificativo del sensore e le coordinate di acquisizione. Questa è una fase molto importante del workflow dell'interno progetto, soprattutto nel caso in cui nello stesso task siano presenti molteplici sensori: anche questo va a spiegare il motivo del largo uso delle chiamate di funzione di tipo *async*.

```
app.post('/sendSample', async function (req, res) {
  res.header("Access-Control-Allow-Origin", "*");
  res.setHeader('Content-Type', 'application/json');
  let risposta = await sendSample(client, req.body.sample,
      req.body.coords, req.body.type);
  res.send(JSON.stringify(risposta));
});
```

La POST va quindi a richiamare l'omonima funzione asincrona che va a costruire la query da inviare al DB. Questi dati possono essere di vario genere: nel database si va a salvare tutto come puro testo, seppur la natura dei samples sia differente. Per esempio nel caso dell'accelerometro e del giroscopio si va a salvare dei valori di *magnitude*, ovvero un valore numerico che quantifica i valori dati dagli assi XYZ catturati dai sensori. Nel caso della fotocamera e del microfono (eccezion fatta per il noise sensor) la questione è più complessa, in quanto il risultato dell'acquisizione è un file vero e proprio, il quale non può essere salvato come tale nel DB. La soluzione sta nella conversione del file come segue.

```
function takepicture() {
  var context = canvas.getContext('2d');
  if (width && height) {
    canvas.width = width;
    canvas.height = height;
    context.drawImage(video, 0, 0, width, height);
    var data = canvas.toDataURL('image/png');
    photo.setAttribute('src', data);
    dataSend = data.substr(data.indexOf(',') + 1);
    //start the samplings
    let actualPosition = exPos();
    cameraSamplings(actualPosition);
    //open the modal
    $("#modalSample").modal();
  } else {
    clearphoto();
  }
```

```
18 }
```

Nel listing è mostrato il processo riguardante la fotocamera che va dalla cattura della foto al suo invio. la funzione *takepicture()* viene richiamata ogni qualvolta si intende scattare una nuova foto, infatti al suo interno è presente la chiamata *drawImage()* che assume tutti i dati direttamente dal canvas riguardo l'altezza, la larghezza e, ovviamente, la cattura del video. A questo punto la foto è stata salvata attraverso un URL esadecimale che descrive un'immagine in formato PNG, ma il database non può accettare files del genere al suo interno: la conversione in binary è molto semplice in quanto basta considerare solo una sottostringa dell'URL (solo la parte esadecimale), ovvero quella contenente l'informazione vera e propria. Dopo la cattura dello scatto all'utente viene mostrata la foto in modo tale da poter decidere se inviarla o scartarla. Questa, come anche la modalità di salvataggio del dato binario, sono delle caratteristiche comuni anche al microfono, con la differenza che l'utente potrà ascoltare il proprio messaggio vocale in un player che dinamicamente apparirà nella pagina del task.

Si nota che l'invio di un sample come foto o audio richiede un tempo di invio più alto rispetto ad un semplice dato numerico, ciò potrebbe quindi comportare qualche rallentamento nell'interfaccia in caso di scarsa potenza di segnale di rete. Approfondiremo la questione nel capitolo successivo.

Uno degli snodi fondamentali per l'intero flow dell'esecuzione è il tracciamento dei samples rimanenti, in quanto si deve dinamicamente avvisare l'utente dello stato di avanzamento del proprio task. Pertanto si deve andare ad agire sul quantitativo di samples rimanenti.

```
1 //decrease number of samples required for that task
2 let senremaining = sen + '_remaining';
3 let verify = await collection.findOne({crowdsourcer:
      session.userid, campaign: thistask, closed: false});
4 if(verify[senremaining] > 0){
5     //do the decrease
6     await collection.updateOne({
```

```
7           crowdsourcer: session.userid,
8           campaign: thistask
9       }, {
10          $inc: {
11              [senremaining]: -1
12          }
13      });
14  }
```

Nel caso dell'accelerometro si va ad individuare il campo del JSON *accelerometer_remaining* dichiarando innanzitutto una variabile che ne contenesse il relativo testo, dopodichè si va a verificare se all'interno del documento che effettua l'associazione utente-task vi sia un campo con quel nome. Qualora si verifichi la condizione si va a decrementare il numero di samples rimanenti, ovviamente sempre in base al sensore che va ad eseguire l'invio.

Man mano che si prosegue nell'attività, sempre secondo i valori di sampling e di tutti i dati richiesti, si arriva alla conclusione del task.

3.2.4 Chiusura del task

Un task viene chiuso solo nella condizione in cui non si hanno più samples rimanenti da inviare: questa considerazione è necessaria in quanto non si può andare a chiudere un task semplicemente cliccando lo slider. In quel caso si avrebbe la sospensione del task, possibile qualora l'utente voglia proseguire la sua attività di crowdsourcing in un secondo momento. Si pone l'attenzione sulla possibilità che un utente durante l'esecuzione di un task, magari opportunistico, decida di mettere in stop il task: cosa succederebbe ai dati che egli nel frattempo sta trasmettendo al database?

```
1  function stopAccelerometerTask() {
2    //stop sensAcc
3    sensAcc.stop();
4    if (capAcc != null) {
5      capAcc.setAttribute("hidden", "true");
```

```
6    }
7  }
8  ...
9  //delete all elements under sen_type
10 senData.querySelectorAll('*').forEach(n => n.remove());
```

In tutte le implementazioni dei vari sensori è presente la chiamata di stop e in tutti i casi esegue lo stesso compito. Per quanto riguarda l'accelerometro *stopAccelerometerTask()* va a stoppare fisicamente il sensore e a nascondere un eventuale bottone di cattura. Successivamente, direttamente nell'index viene eseguita la riga di codice in basso nello snippet che va a prendere tutti gli elementi sottostanti a *sen_type*, ovvero l'id HTML che contiene tutti gli oggetti per il sensore, e li rimuove fisicamente. In questo modo, qualora l'utente volesse riprendere l'esecuzione di un task, alla nuova pressione dello slider gli elementi della campagna vengono ricreati e si può riprendere da dove si era rimasti. Da notare che, durante la condizione di stop, tutti i sample saranno inviati se e solo se (anche nel caso opportunistico) avranno sempre rispettato le proprie condizioni di sampling. La chiamata di stop viene sfruttata anche nel caso in cui invece il task termina, pertanto in questo caso bisogna effettuare diverse operazioni.

```
1 if (data.accelerometer_remaining <= 0) {
2     //set closed task
3     stopAccelerometerTask();
4     postRequest('/setClosedTask').then(
          document.getElementById("accfinished").
          removeAttribute("hidden"));
5 }
6 document.getElementById("accremaining").innerHTML = "Total
      samples remaining: " + data.accelerometer_remaining;
```

In ogni implentazione dei sensori è presente un controllo del genere che avviene nel momento in cui si invia un sample, ovvero nel *then* della *postRequest()*. In questo modo per ogni sensore, non appena raggiunge il proprio termine,

si va a bloccare ulteriori acquisizioni andando a mostrare un messaggio di termine e nascondendo (nel caso partecipatorio) il bottone di cattura. La POST *setClosedTask* viene chiamata sensore per sensore e va a controllare, sempre relativamente al documento che collega l'utente al task, se tutti i valori di samples rimanenti sono uguali a zero. Di seguito un estratto del codice interno alla chiamata di funzione asincrona relativa alla POST, una delle più onerose in termini computazionali.

```
1  //manually find sensors involved
2  var keys = Object.keys(control);
3  for (i = 0; i < keys.length; i++) {
4      if (keys[i].includes("_remaining")) {
5          names.push(keys[i]);
6      }
7  }
8  ...
9  //if they are all zeros send the query
10 let numbers = isallzero.every(item => item <= 0);
11 if (numbers) {
12     await collection.updateOne({
13         crowdsourcer: session.userid,
14         campaign: thistask
15     }, {
16         $set: {
17           closed: true
18         }
19     });
20 }
```

Anche in questo caso il nome della funzione asincrona è omonimo alla POST e, come visibile dall'estratto interno alla funzione, va in primis ad estrarre manualmente tutte le chiavi contenenti la sottostringa "_remaining", in modo tale da prendere in considerazione tutte le voci relative ai sensori. Dopo aver creato l'array di numeri *isallzero* contenente il quantitativo di samples rimanenti per tutti i sensori coinvolti si va a verificare che essi siano tutti

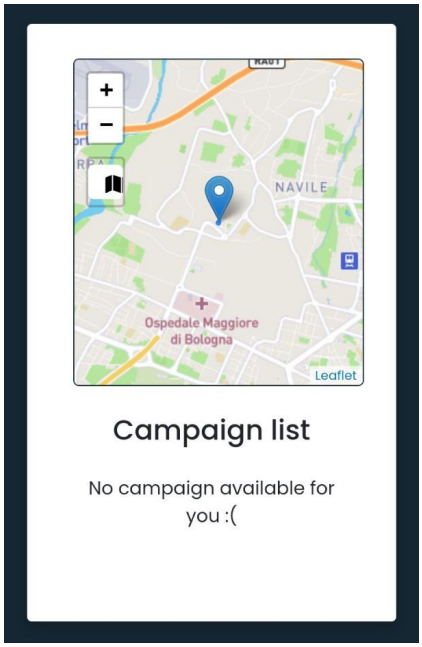

Figura 3.5: Schermata di selezione del task: l'utente ha completato tutte le campagne, per cui non può selezionarne altre.

equivalenti a zero. In caso positivo si va a modificare il documento JSON settando la voce *closed* a *true*.

Dal punto di vista progettuale questo è un passaggio fondamentale, in quanto in fase di accesso alla piattaforma si va sempre a controllare il nickname dell'utente inserito e, se esso appare in un documento il cui task è stato chiuso, la relativa campagna non apparirà nelle possibili scelte. Questo accade perchè ovviamente un utente non può ripetere la stessa campagna più di una volta, ne tantomeno si può accedere ad una campagna chiusa. La descrizione di tutto il flusso di esecuzione di *YouCrowd* può sembrare caotica, ma si è cercato quanto più possibile di sintetizzare i punti salienti dal punto di vista tecnico, andando a suddividere le varie fasi seguite, sia dal punto di vista della progettazione che dell'esecuzione. L'esposizione relativa a questo capitolo necessita comunque di un riscontro in campo reale, ovvero di un test che possa dare ancora più valore allo sviluppo tecnico.

Capitolo 4

Test della piattaforma

La descrizione dell'intera piattaforma ha evidenziato come è stato possibile terminare un'implementazione di un progetto con un grosso background teorico, senza però particolari evidenze di spunti o piattaforme del tutto simili. I tecnicismi utilizzati pertanto sono piuttosto tendenti al fallimento, in quanto non ci si è ispirati a nessuna soluzione pratica disponibile. Diventa quindi fondamentale imbastire un progetto vero e proprio di testing con il quale verificare se vengono rispettati tutti i requisiti sia teorici che pratici di una piattaforma di Mobile Crowdsensing, verificando il totale supporto sotto ogni punto di vista. Ma non è solo un puro riscontro sulle caratteritiche fisiche del progetto: questa fase è molto importante anche e soprattutto per valutare l'esperienza relativa all'utente finale, il quale dopo il test andrà a compilare un sondaggio riassuntivo della propria esperienza. I risultati ottenuti permetteranno una piena interpretazione di tutto il progetto, soprattutto per scopi migliorativi ed estensivi.

Si procede andando a descrivere l'ambiente di esecuzione del progetto, fornendo una spiegazione approfondita soprattutto sull'oggetto di test, ovvero l'esecuzione di quattro task diversi che coinvolgono tutti i sensori disponibili.

4.1 Ambiente di test

In fase di sviluppo sono stati eseguiti diversi test, atti a rilevare più che altro la presenza di bug, sulla macchina in locale che poggia l'esecuzione dell'intero progetto principalmente su due entità: il motore NodeJS e il database MongoDB. Per testare la piattaforma "dall'esterno" si necessita quindi di un servizio di hosting con macchina server. L'Università di Bologna ha messo a disposizione le macchine virtuali del servizio *Okeanos*, completamente configurabili in base a tutte le esigenze dell'utente. La macchina virtuale ha un IP pubblico pertanto può essere configurata per un servizio http oppure https. La configurazione utilizzata per la VM è la seguente:

- **RAM**: 4gb
- **ROM**: 40gb
- **Sistema Operativo**: Windows Server 2012 R2

Nella macchina virtuale sono stati installati tutti i software necessari per l'esecuzione dei due package previsti. In particolare è stata installata una versione GUI di MongoDB, ovvero *MongoDB Compass*, in modo tale da facilitare l'accesso e la modifica dei vari documenti JSON.

Ovviamente entrambi i progetti della piattaforma sono sempre in esecuzione e sono raggiungibili dall'IP pubblico alle porte 3000 e 4000. Dalle varie interfacce di terminale è possibile vedere tutti i log relativi all'utente attualmente attivo sui progetti.

Lato client il test della piattaforma avviene semplicemente con l'accesso all'indirizzo http://83.212.126.88:3000 (oppure 4000) da browser Chrome: questa specifica è richiesta in quanto diverse API non garantiscono il supporto per altri browser. Stesso discorso per quanto riguarda il sistema operativo mobile, infatti il test è orientato alla piattaforma Android in quanto anche su iOS non è certo garantire l'intero flusso di esecuzione.

Questo ambiente di test rappresenta quindi lo scenario ideale per l'applicazione web, realizzata tenendo conto anche della diffusione di contesti del

genere. Ma in cosa consiste realmente il test? L'utente selezionato andrà a concludere quattro tipi diversi di campagne di task, le quali coinvolgono tutti i sensori disponibili ma soprattutto espongono uno scenario reale.


```
{
    "_id":{"$oid":"61c887d81e61f5aabd43fa94"},
    "campaign":"TASK3",
    "n_tasks":38,
    "expdate":{"$date":"2023-01-01T00:00:00.000Z"},
    "geofence":[{...}],
    "sensorlist":{
        "0":{
            "type":"camera",
            "partopp":"part",
            "temporal_samp":7000,
            "space_samp":30,
            "n_samples":10
        },
        "1":{
            "type":"gyroscope",
            "partopp":"opp",
            "temporal_samp":5000,
            "space_samp":20,
            "n_samples":-1
        }
    },
    "tabtype":"campaign"
}
```


Listing 4.1: Documento JSON che descrive il Task 3

- ***Task 1***: il primo task è il più semplice di tutti, in quanto si tratta di un'acquisizione di dati relativi all'accelerometro in modo opportunisti-

co. Pertanto l'utente deve solo avviare il task all'interno del geofence di compatibilità e aspettare che lo smartphone invii automaticamente i dati. Lo scenario reale è quello della valutazione della varianza del movimento dello smartphone mentre l'utente lo usa durante una passeggiata.

- ***Task 2***: la campagna del secondo task è molto interessante in quanto consiste nella registrazione di samples di tipo noise, ovvero della cattura del livello di decibel nell'area urbana del centro di Bologna. In questo caso è l'utente a dover catturare il sample attraverso la pressione di un tasto. Questa campagna di task permette all'administrator di ottenere dati relativi ai decibel medi in un area da lui definita.

- ***Task 3***: questo task coinvolge più di un sensore, infatti il worker è chiamato a scattare delle foto nell'area urbana di Bologna mentre il giroscopio calcola il livello di oscillazione dello smartphone e lo invia in maniera autonoma. Lo scenario di questo task quindi comprende la valutazione da parte dell'admin del livello di precisione degli scatti fotografici degli utenti, in quanto più basso sarà il valore di magnitude migliore risulterà la qualità delle fotografie. Il documento JSON relativo a questo task riguarda il listing 4.1.

- ***Task 4***: l'ultima campagna raccoglie forse i dati più interessanti, in quanto consiste di scattare delle fotografie nell'area urbana di Bologna e di allegare ad essere una nota vocale, sfruttando quindi anche il microfono. Questa tipologia di task può essere molto utile a supportare i servizi web relativi alle guide turistiche, in quanto l'utente è chiamato a descrivere una foto che lui stesso scatta.

Lo scenario reale è coerente anche lato crowdsourcer, in quanto quest'ultimo può sfruttare i dati provenienti dai vari utenti per studi e ricerche molto utili ai fini statistici. D'altronde uno degli obiettivi di *YouCrowd* è anche quello di trovare un'ulteriore fonte alla ricerca scientifica.

4.2 Risultati dei test

I test relativi all'ambiente e ai task descritti in precedenza sono stati effettuati senza un particolare meccanismo di incentivazione, in quanto non vi era una particolare necessità. In questo caso, gli utenti testati sono stati "convinti" sulla base di un approccio volto alla curiosità piuttosto che alla possibilità di guadagno: si tratta di una condizione normale, in quanto come detto in precedenza non si ha un riscontro popolare di una piattaforma simile.

Le persone che hanno fornito un feedback sulla piattaforma sono 18, distribuite in maniera piuttosto equa per quanto riguarda il sesso e l'età. Tutti i soggetti coinvolti nel test hanno anche fornito un feedback sulla base di un form che è stato compilato alla fine della loro esperienza. Chiaramente i risultati del test sono direttamente correlati al feedback previsto, per cui si procederà in un'analisi in base alle domande previste.

4.2.1 Analisi del target

Prima di procedere all'analisi relativa a *YouCrowd* andiamo a descrivere accuratamente gli aspetti relativi ai vari utenti che hanno testato la piattaforma, secondo le caratteristiche relative ad essa.

Età e sesso: Il target di riferimento ha un età compresa tra i 17 e i 60 anni, distribuito in maniera uniforme e senza una particolare concentrazione in una certa fascia d'età. Relativamente al sesso abbiamo una percentuale del 64,7% di soggetti maschi e 35,3% invece di sesso femminile. Queste differenze relative alla partecipazione al test non hanno particolari evidenze significative.

Abilità con gli strumenti informatici: In una scala da 1 a 5 i partecipanti hanno indicato il loro livello di confidenza generale con gli strumenti informatici, quali ad esempio smartphone, tablet e computer. I risultati, visibili nella Fig. 4.1, mostrano un generale allineamento su un livello medio di capacità di utilizzo: il livello 3 può essere interpretato

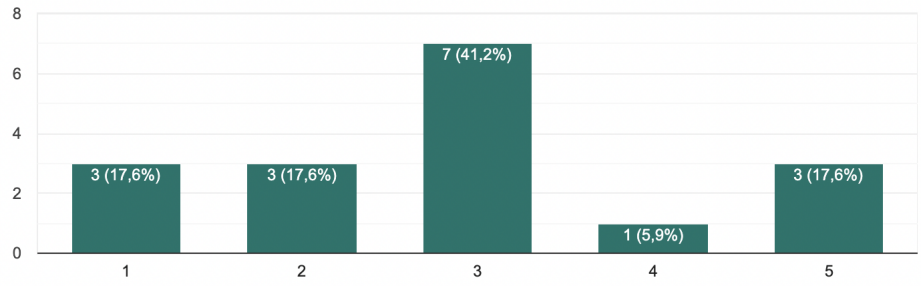

Figura 4.1: Istogramma che mostra il numero dei votanti per una certa scala di abilità con la tecnologia(da 1 a 5)

come "utente normale", l'esatta via di mezzo tra utente base e utente specializzato, e si intendo come un soggetto in grado di svolgere con autonomia i compiti quotidiani che richiedono la necessità dell'utilizzo degli strumenti tecnologici.

Gestione della privacy: Un importante quesito posto ai partecipanti al test è stato quello di esprimere, sempre in una scala da 1 a 5, il livello di "fastidio" nella cessione dei propri dati personali a utenti terzi. L'utilità di questa risposta sta nel capire il rapporto che l'utente ha con la privacy in generale, oltre al livello di confidenza che può riporre su questo sistema piuttosto che in altri. In generale si è notato un certo appiattimento delle risposte, con una concentrazione sui valori intermedi tra 2 e 4.

Nello specifico, il target di riferimento ha riguardato in particolar modo amici e parenti, per cui il livello di confidenza sul sistema (soprattutto per quanto riguarda l'ultimo aspetto) può essere anche influenzato da questo fattore.

4.2.2 Feedback su YouCrowd

Il sondaggio fornito agli utenti contiene una serie di domande su ogni task, in modo tale da capire l'intera esperienza di utilizzo. Tutti i quesiti sono mirati a capire sia la semplicità di utilizzo della web app secondo il

singolo task che il livello di reward che i singoli utenti si aspettano. La scala relativa al primo aspetto è più fine rispetto all'indagine sul target in quanto si prendono in considerazione valori da 1 a 10.

Semplicità login: Il primo quesito riguarda, secondo il flusso delle operazioni degli utenti, l'accesso alla piattaforma: la motivazione di questa domanda riguarda una questione di completezza piuttosto che di utilità pratica. Il sondaggio infatti ha riportato risultati concentrati quasi in totale sul valore massimo della scala, ovvero 10.

Task 1: Il task relativo alla prima campagna è dichiaratamente il più semplice in quanto, oltre a presentare un'acquisizione opportunistica, essa è anche piuttosto rapida visti i bassi valori di sampling temporale e spaziale (rispettivamente 5 secondi e 50 metri). Secondo gli utenti è, come da aspettativa, il task più semplice in quanto il 58,8% ha votato 9 mentre il 35,3% ha votato 10. Diverso il discorso per la remunerazione ove si evidenza un equilibrio per valori che vanno da 0.20 a 1 euro, come visibile nella Fig. 4.2.

Task 2: La situazione inizia a cambiare nel caso del secondo task, ove vi è comunque una concentrazione sul voto 9 (circa il 52%) che è bilanciata da altri utenti che hanno assegnato il punteggio 8 e 10. Stiamo parlando di un task semplice che però inizia a richiedere l'interazione attiva dell'utente, caratteristica che potrebbe già richiedere un certo livello di prestazioni di rete e del dispositivo. Il pagamento richiesto per la conclusione di questo task è di 1 euro per il 64,7% dei votanti, solo il 23,5% è arrivato a 1.50 euro mentre i restanti hanno espresso cifre intermedie.

Task 3: Questo task coinvolge più di un sensore (fotocamera e giroscopio) e presenta un'interfaccia differente rispetto ai due precedenti: questo è un fattore importante in quanto proprio la user experience spesso definisce il grado di difficoltà di esecuzione di una qualsiasi operazione sul sistema. I voti assegnati (Fig. 4.3) riportano un'elevata concentrazione

di utenti con voto 7 (52,9%) i quali, probabilmente, hanno incontrato difficoltà o rallentamenti del sistema durante l'acquisizione di foto. Questo potrebbe accadere a causa dell'avvicendamento dei due sensori nell'interfaccia web, che pertanto va a richiedere un numero più elevato di risorse computazionali in generale. Chiaramente la difficoltà di esecuzione causa un rialzo nella richiesta economica che in questo caso si attesta sui 2 euro nel 64% dei casi, percentuale giustificata anche dalla ristretta area di acquisizione che costringe l'utente a doversi spostare entro una certa area.

Task 4: Questo task è il più impegnativo di tutti in quanto, oltre a richiedere l'acquisizione da due sensori, abbiamo due catture di tipo partecipatorio; per cui si innalzano ancora di più le possibili difficoltà da parte di utenti con scarse connessione di rete e prestazioni a disposizione. I risultati sono visibili nella Fig. 4.4, ove si nota una certa distribuzione di voti pari a 5, 6 e 7, a sottolineare ancora quanto espresso poc'anzi. La retribuzione in gioco sembra quindi andare di pari passo con la difficoltà del task, visto che in questo caso il 47% dei partecipanti ha espresso un valore di 3 euro; seppur siano presenti alcuni outlier in questa analisi in quanto il 23,6% ha dichiarato di aspettarsi una remunerazione inferiore, pari a 2 e 1 euro.

La lista descrittiva appena presentata riassume il rapporto tra gli utenti e tutte le operazioni che hanno eseguito in *YouCrowd*, mentre la specifica sui numeri in gioco più rilevanti è mostrata nei grafici che seguono.

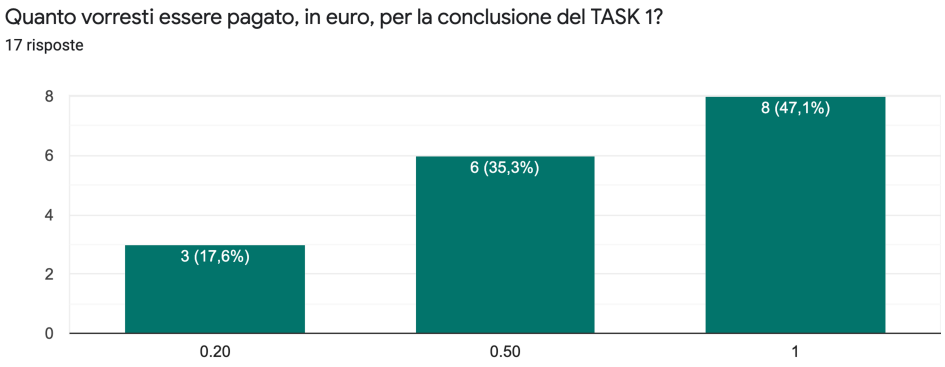

Figura 4.2: Ipotesi remunerazione per il task 1

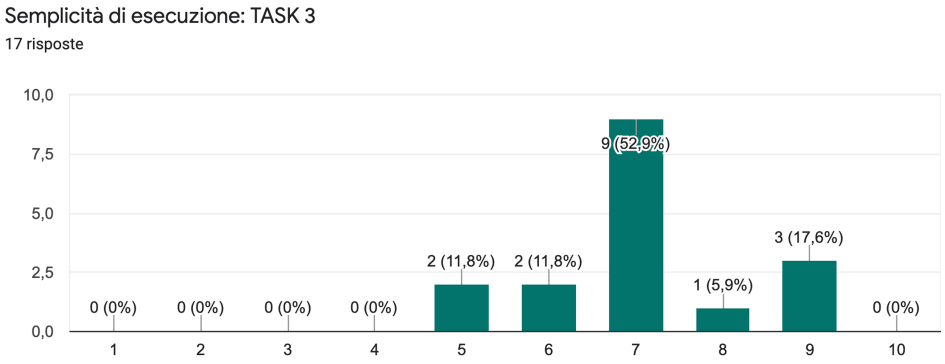

Figura 4.3: Numero di votanti per scala da 1 a 10 che rappresenta il livello di semplicità di esecuzione del task 3

I dati raccolti dalle interviste non potevano essere interpretati senza un'opinione aperta da parte degli utenti su *YouCrowd*, infatti nell'ultimo paragrafo del form gli è stato chiesto di segnalare eventuali problemi o difficoltà incontrati durante il test pratico della piattaforma. Sono emerse alcune informazioni molto utili, riportate di seguito.

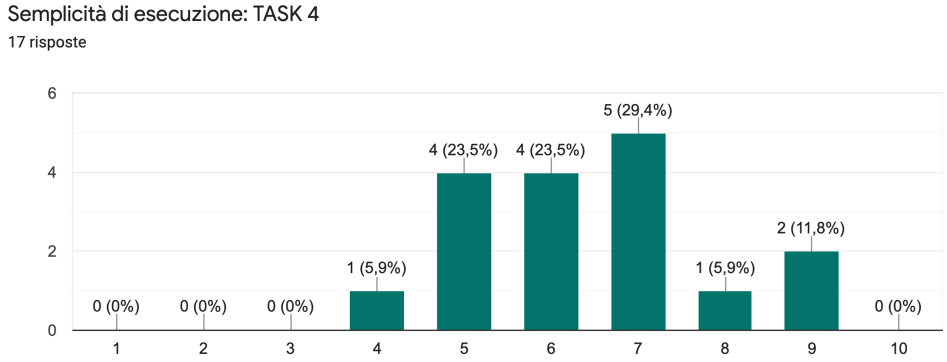

Figura 4.4: Numero di votanti per scala da 1 a 10 che rappresenta il livello di semplicità di esecuzione del task 4

- *"interfaccia lenta"*
- *"interfaccia fotocamera poco chiara"*
- *"Ci vorrebbe un'interfaccia più user friendly con maggiori indicazioni sui passaggi da eseguire, in modo da essere di facile utilizzo anche per utenti meno esperti"*
- *"interfaccia lenta su telefoni vecchi"*

L'interpretazione relativa a questi feedback diretti è, in parte, già stata fornita durante la descrizione relativa ai singoli task; pertanto tutto ciò non fa altro che confermare alcuni aspetti relativi alla prestazione della web app su dispositivi datati. Il framework NodeJS, la base sulla quale si appoggiano tutte le librerie necessarie, è molto estensibile e permette la progettazione di complesse web application, come in questo caso. Tuttavia la natura non *multi-threaded* del linguaggio Javascript richiede un alto livello di prestazioni di rete, ma non solo: l'implementazione di tutti i sensori, in particolar modo nel caso della fotocamera, richiede un particolare sforzo a livello computazionale da parte degli smartphone. Il render degli oggetti relativi alle API dei

sensori pesano perchè garantiscono una compatibilità elevata per una vasta gamma di smartphone aventi Android come sistema operativo e una versione del browser Chrome abbastanza recente. Questo discorso è stato poi anche sottolineato dall'utente che ha evidenziato una particolare lentezza nei device più datati, semplicemente perchè ha potuto effettuare un confronto.

Tra i feedback più significativi si evidenzia la segnalazione di un livello di esperienza utente non sufficientemente guidato per gli utenti meno esperti. Questo riscontro ha una forte motivazione legata alla natura modulare del progetto: attraverso *YouCrowd* si può progettare l'esecuzione di campagne di task molto variegate nei loro obiettivi. Qualsiasi sia la tipologia di task in esecuzione il motore di esso è il medesimo, ciò implica una certa generalizzazione estetica dal punto di vista dell'interfaccia; senza poi considerare la natura sperimentale della piattaforma. Proprio quest'ultimo punto ha costretto delle scelte implementative improntate sul minimalismo dell'estetica, in quanto un minor numero di "elementi di disturbo" dovrebbero chiarire le idee agli utenti su cosa andare a toccare per terminare la propria azione.

Conclusioni

La piattaforma *YouCrowd* ha richiesto un impegnativo studio relativamente alla possibilità di suscitare un particolare interesse a tutte le tipologie di utenti: le novità riguardanti lo sviluppo di una nuova applicazione o piattaforma in cui un soggetto qualsiasi può generare un profitto fanno sempre molto rumore. Ma la mera possibilità di guadagno economico non basta, in quanto bisogna cercare quanto più possibile di supportare la piattaforma sotto ogni punto di vista e questa è una caratteristica che è stata ereditata direttamente dal concetto base del Mobile Crowdsensing. Permettere agli utenti di svolgere ogni volta operazioni diverse dalla precedente aumenta sensibilmente l'incentivazione alla piattaforma, pertanto lo sviluppo di una piattaforma del genere in grado di supportare tante combinazioni di sensori espone questa possibilità.
L'implementazione di una piattaforma concettualmente nuova pone diverse difficoltà, non solo in fase di scrittura del codice ma anche durante la progettazione, in quanto realisticamente non esistono progetti fisici simili per cui non si ha la possibilità di effettuare un *benchmark*. Spesso le idee di successo hanno le proprie fondamenta su un progetto del tutto simile ma ingegnerizzato in una maniera più convincente. Un esempio su scala differente riguarda l'*iPhone*, un dispositivo concettualmente innovativo che pone la sua forza nell'SDK[15], ovvero nel software, ma che sostanzialmente rispecchia quanto già era in circolazione secondo gli altri brand di telefonia mobile.
La natura, già esposta, modulare di *YouCrowd* lo rende particolarmente estensibile e incline a sviluppi e miglioramenti futuri: riguardo questi punti

si può prendere spunto dai feedback rilasciati dagli utenti, sicuramente un indice evolutivo molto utile. Andiamo ad esporre innanzitutto due possibili miglioramenti sul progetto attuale.

- **Evoluzione interfaccia grafica**: sicuramente lo sviluppo pratico attuale vede una UI abbastanza essenziale, non tanto dal punto di vista prettamente estetico ma piuttosto da quello funzionale. Una delle possibili soluzioni in merito può essere la riorganizzazione della home page, attraverso la possibilità di ridimensionare o nascondere la mappa in modo tale da lasciare spazio alle componenti interattive del task. Queste ultime poi andrebbero "svecchiate" e poste al centro di uno studio in modo tale da aumentarne la coerenza grafica, per permettere anche all'utente meno esperto di capire cosa sta facendo.

- **Implementazione sistema di pagamento**: Il concetto di remunerazione è stato modellato solo nella teoria, in quanto un'eventuale integrazione nel sistema si rivelerebbe necessaria solo nel momento del rilascio di un vero e proprio servizio online. Ai fini di test non si rende necessaria tale feature.

Eventuali sviluppi futuri riguardano principalmente un'estensione della piattaforma rilasciata, sia per quanto riguarda il supporto ad altri device (come i dispositivi iOS) che nelle funzionalità previste. Nel primo caso non si può avere una diretta applicazione se non passando attraverso l'implementazione di API aggiornate, le quali però permetterebbero ad una grande fetta di utenti aggiuntiva di poter utilizzare *YouCrowd* senza alcun tipo di problema. L'ipotesi di estensione delle funzionalità della piattaforma è invece direttamente fattibile, in quanto non sono ancora stati integrati nella piattaforma i sensori relativi alla luce ambientale, prossimità, temperatura e altri ancora da librerie terze rispetto a *Sensor API*. L'integrazione di un maggior numero di sensori permette agli administrator di generare ancor più campagne di task, con la conseguenza di poter coinvolgere sempre più persone e aumentare i profitti sia per la ricerca che per i crowdsourcer stessi.

Bibliografia

[1] Neetima Agarwal, Sumedha Chauhan, Arpan Kumar Kar, and Sandeep Goyal. Role of human behaviour attributes in mobile crowd sensing: a systematic literature review. *Digital Policy, Regulation and Governance*, 2017.

[2] Abdulrahman Alreshidi and Aakash Ahmad. Architecting software for the internet of thing based systems. *Future Internet*, 11(7):153, 2019.

[3] Oscar Alvear, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. Crowdsensing in smart cities: Overview, platforms, and environment sensing issues. *Sensors*, 18(2):460, 2018.

[4] Andrea Capponi, Claudio Fiandrino, Burak Kantarci, Luca Foschini, Dzmitry Kliazovich, and Pascal Bouvry. A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities. *IEEE communications surveys & tutorials*, 21(3):2419–2465, 2019.

[5] Sunny Consolvo, David W McDonald, Tammy Toscos, Mike Y Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1797–1806, 2008.

[6] Gartner. Gartner says global smartphone sales stalled in the fourth quarter of 2018. https://www.gartner.com/en/newsroom/press-

releases/2019-02-21-gartner-says-global-smartphone-sales-stalled-in-the-fourth-quart.

[7] Wei Gong, Baoxian Zhang, and Cheng Li. Task assignment in mobile crowdsensing: Present and future directions. *IEEE network*, 32(4):100–107, 2018.

[8] Bin Guo, Yan Liu, Wenle Wu, Zhiwen Yu, and Qi Han. Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems. *IEEE Transactions on Human-Machine Systems*, 47(3):392–403, 2016.

[9] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67, 2010.

[10] Howe J. The rise of crowdsourcing. `https://www.wired.com/2006/06/crowds/`.

[11] Luis G Jaimes, Idalides Vergara-Laurens, and Miguel A Labrador. A location-based incentive mechanism for participatory sensing systems with budget constraints. In *2012 IEEE International Conference on Pervasive Computing and Communications*, pages 103–108. IEEE, 2012.

[12] Luis G Jaimes, Idalides J Vergara-Laurens, and Andrew Raij. A survey of incentive techniques for mobile crowd sensing. *IEEE Internet of Things Journal*, 2(5):370–380, 2015.

[13] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, 16(2):138–178, 2013.

[14] Yan Liu, Bin Guo, Yang Wang, Wenle Wu, Zhiwen Yu, and Daqing Zhang. Taskme: Multi-task allocation in mobile crowd sensing. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 403–414, 2016.

[15] Dave Mark, Jeff LaMarche, and Mark Dalrymple. *Beginning iPhone development: Exploring the iPhone SDK*, volume 5. Springer, 2009.

[16] Waze Mobile. Waze, that way looks awesome today. `https://www.waze.com/en/waze`.

[17] Mohamed Musthag, Andrew Raij, Deepak Ganesan, Santosh Kumar, and Saul Shiffman. Exploring micro-incentive strategies for participant compensation in high-burden studies. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 435–444, 2011.

[18] Sasank Reddy, Deborah Estrin, Mark Hansen, and Mani Srivastava. Examining micro-payments for participatory sensing data collections. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 33–36, 2010.

[19] Prezzi Benzina srl. Prezzi benzina, scopri dove risparmiare! `https://www.prezzibenzina.it/`.

[20] Justin C Strickland and William W Stoops. The use of crowdsourcing in addiction science research: Amazon mechanical turk. *Experimental and Clinical Psychopharmacology*, 27(1):1, 2019.

Ringraziamenti

A conclusione di questo elaborato, desidero menzionare tutte le persone, senza le quali questo lavoro di tesi non esisterebbe nemmeno.
Ringrazio il mio relatore Federico Montori che in questi mesi di lavoro ha saputo guidarmi, con suggerimenti pratici, alla realizzazione del progetto sperimentale e alla stesura dell'elaborato di tesi.
Grazie mamma e grazie papà. Grazie per avermi sempre sostenuto nei momenti di maggiore difficoltà durante gli studi, soprattutto nelle classiche situazioni di stress affrontate sempre con lucidità.
Grazie Sara, devo tanto a te della mia crescita personale. Grazie per tutto il tempo e la pazienza che mi stai dedicando, da tre anni a questa parte apprezzo la tua compagnia ogni giorno sempre di più.
Grazie nonna Anna, sempre pronta anche a 95 anni ad accogliere la nostra famiglia con pranzi e cene speciali durante le festività, in compagnia di tutti i nostri parenti. Grazie nonna sprint!
Infine un grande ringraziamento va a tutti i miei amici, per i quali non basterebbe un'ulteriore tesi per esserne sufficientemente grato! Senza di loro la carriera universitaria sarebbe stata senza dubbio più difficile, è stato bello (e spero lo sarà ancora) condividere tante serate e ore di studio, conoscere altre persone e divertirsi tutti insieme. Auguro a chiunque di poter vivere un'esperienza universitaria simile alla mia.