Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Un'Architettura Ibrida Push-Pull per Mobile Crowdsensing usando il Web of Things

Relatore:
Prof.
FEDERICO MONTORI

Presentata da:
DAVIDE TINTI

Correlatore:
Dott.
LUCA SCIULLO

Sessione I
Anno Accademico 2022/2023

Astract

Il Mobile Crowdsensing (MCS) è un concetto in rapida crescita che sfrutta la diffusa presenza dei dispositivi mobili per raccogliere dati e analizzare l'ambiente circostante. Tuttavia, la scelta tra un'architettura basata su Push o Pull per un sistema di MCS può comportare limitazioni e una mancanza di flessibilità per i creatori delle campagne. Per superare queste sfide, questa tesi propone un'architettura ibrida Push-Pull per le campagne di raccolta dati basate sul MCS che sfrutta le potenzialità del Web of Things (WoT). L'architettura proposta si basa su tecnologie web consolidate per standardizzare le interfacce e le interazioni tra i dispositivi. Ciò consente ai creatori delle campagne (crowdsourcers) di supportare simultaneamente diversi tipi di campagne, bilanciando la quantità di dati raccolti con la loro qualità. Inoltre, la possibilità di geolocalizzare i lavoratori MCS consente di migliorare ulteriormente la qualità dei dati raccolti. Per mettere in pratica l'architettura proposta è stata progettata e implementata un'applicazione Android abilitata per il WoT che consente la partecipazione attiva dei dispositivi mobili nella raccolta dei dati. Attraverso l'adozione di questa architettura ibrida Push-Pull basata sul WoT, è possibile superare le limitazioni delle architetture tradizionali per il MCS e fornire una soluzione innovativa che offre una maggiore flessibilità e scalabilità nella raccolta e nell'analisi dei dati provenienti dai dispositivi mobili.

Indice

Elenco delle figure

Introduzione

Al giorno d'oggi l'utilizzo dei dispositivi mobili è ormai diventato parte integrante della vita quotidiana. Essi sono inoltre dotati di un'ampia platea di sensori, come telecamera, microfono, GPS, accelerometro, sensore d'illuminazione e bussola digitale e la loro presenza nella popolazione è destinata ad aumentare considerevolmente. Data questa premessa l'utilizzo del vasto numero di dispositivi mobili disponibili per creare un'infrastruttura di rilevazione dati geograficamente distribuita è un'opportunità reale.

Il Mobile Crowdsensing (MCS) è un paradigma che punta a realizzare quanto descritto, tramite campagne di raccolta dati organizzate dai crowdsourcer, entità interessate a raccogliere i dati, gli utenti possono iscriversi e partecipare raccogliendo dati dai sensori dei propri dispositivi rendendoli disponibili. Questo paradigme può essere utilizzato in una vasta gamma di applicazioni diverse tra cui il monitoraggio ambientale, l'assistenza sanitaria e la gestione del traffico. Tuttavia, il successo dell'MCS dipende molto dall'accettazione degli utenti e, pertanto, spesso vengono offerte ricompense per incentivare gli utenti a condividere i loro dati. Queste ricompense possono assumere la forma di sconti, vantaggi o crediti.

Considerando questi aspetti, negli ultimi anni sono state proposte diverse architetture per il MCS, che possono essere generalmente classificate come architetture Push-based o Pull-based. Nelle architetture Push-based, ai lavoratori viene esplicitamente inviata la richiesta di invio dei dati, o del completamento di un compito, il che consente un controllo più preciso su chi contribuisce alla campagna e garantisce una maggiore qualità dei dati

raccolti. D'altra parte, le architetture Pull-based richiedono invece ai lavoratori di contribuire senza alcuna richiesta specifica. Questo approccio è particolarmente adatto per fenomeni di interesse comune che richiedono più prospettive per essere descritti accuratamente e possono tollerare campioni di qualità potenzialmente inferiore data la maggiore quantità di partecipanti. Tuttavia, fare affidamento esclusivamente su un'architettura Push o Pull può comportare una perdita di flessibilità, non solo nell'architettura stessa, ma anche nelle tecnologie di comunicazione tra le entità coinvolte nell'architettura. Un'interazione Push-based richiede che il dispositivo mobile sia raggiungibile e in grado di ricevere richieste, mentre un'interazione Pull-based richiede al dispositivo mobile di inviare dati a un'entità esterna raggiungibile. Una volta che l'architettura è implementata, i crowdsourcer sono limitati a un singolo modo di interagire con i lavoratori, il che limita le loro possibilità di adattare dinamicamente le campagne a scenari diversi.

Questa tesi presenta una nuova architettura ibrida che combina approcci sia Push-based che Pull-based, aumentando così la flessibilità per i creatori di campagne. Per raggiungere questo obiettivo viene utilizzato il Web of Things, che mira a standardizzare le interfacce e le interazioni dei dispositivi utilizzando tecnologie Web consolidate. Consentendo ai dispositivi di esporre le loro interazioni disponibili attraverso diversi protocolli Web come HTTP e MQTT, l'architettura consente l'interazione sia con approcci Push-based che Pull-based. Più nel dettaglio:

- Viene proposta un'architettura ibrida sia per il MCS Push-based che Pull-based. La stessa architettura può supportare diverse campagne, garantendo piena flessibilità per i crowdsourcer. Inoltre, l'architettura consente la geolocalizzazione dei lavoratori, migliorando ulteriormente la qualità dei dati e riducendo la quantità.

- Viene presentato il design e l'implementazione di un'applicazione Android conforme al W3C WoT per la raccolta, l'esposizione e l'invio dei dati alle campagne. L'applicazione segue le specifiche del serviente W3C WoT,

La tesi è strutturata come segue: nel capitolo 1 viene descritto lo stato dell'arte delle tecnologie utilizzate: MCS, WoT e LA-MQTT; nel capitolo 2 viene presentata l'architettura proposta, trattando gli obiettivi principali e i componenti, illustrando ruoli, funzionalità ed interazioni tra essi; mentre nel capitolo 3 ne è trattata l'implementazione; all'interno del capitolo 4 vengono mostrati i risultati ottenuti tramite una demo dell'applicazione utilizzata dai partecipanti alle campagne; ed infine nel capitolo 5 sono portate le conclusioni ed i possibili sviluppi dell'architettura.

Capitolo 1

Stato dell'arte

In questo capitolo viene esplorato il concetto di Web of Things (WoT), il Mobile CrowdSensing (MCS) e l'estensione LA-MQTT. Partendo dal contesto dell'Internet of Things (IoT), che rappresenta la base fondamentale per lo sviluppo del WoT, verrà analizzato come il WoT standardizzi e semplifichi la comunicazione tra dispositivi connessi. Successivamente, sarà esaminata l'importanza del MCS come paradigma emergente per la raccolta di dati tramite dispositivi mobili e le considerazioni chiave da affrontare durante l'implementazione di un sistema di MCS. Infine, verrà introdotta l'estensione LA-MQTT, che si propone di supportare la consapevolezza della posizione nell'ambito delle comunicazioni publish/subscribe.

1.1 Internet of Things

Con il termine Internet of Things (IoT) si fa riferimento alla rivoluzione tecnologica che ha trasformato il concetto di dispositivi interconnessi. L'IoT rappresenta un paradigma in cui gli oggetti fisici del mondo reale, come sensori, attuatori e dispositivi intelligenti, sono abilitati a comunicare tra loro e a interagire con l'ambiente circostante attraverso l'infrastruttura di rete.
Durante gli anni '90, diverse aziende pionieristiche hanno intrapreso i primi esperimenti per collegare dispositivi e rendere possibile la condivisione di

informazioni tra di essi. L'ingegnere inglese Kevin Ashton, nel 1999, coniò il termine "Internet of Things" per descrivere oggetti che utilizzavano tecnologie come la radiofrequenza (RFID) per l'identificazione automatica [1]. Questi dispositivi interconnessi hanno aperto le porte a un mondo in cui gli oggetti del quotidiano possono trasmettere dati e acquisire informazioni in tempo reale.
Negli anni a seguire il concetto di dispositivi (o things) interconnessi che raccolgono e trasmettono dati in tempo reale ha rapidamente preso piede, soprattutto grazie all'ampio spettro di utilizzi commerciali che ha trovato, a partire dal settore dei trasporti, passando per quello della finanza e dell'intrattenimento, fino ad arrivare a quelli coinvolti più recentemente, la domotica e la salute personale. Il coinvolgimento dell'IoT in questi settori offre enormi potenzialità, tra cui la possibilità di migliorare la produttività, la qualità della vita e la sicurezza delle persone.

1.2 Web of Things

Il Web of Things (WoT) rappresenta un'estensione dell'Internet of Things (IoT) che mira a integrare i dispositivi connessi nella visione più ampia del Web. Il WoT si basa sui principi e i protocolli del Web per fornire un'interfaccia standardizzata per consentire l'interazione e la gestione semplificata dei dispositivi connessi.

1.2.1 Cos'è il Web of Things

Il concetto di WoT ha preso forma nel 2007, quando Kevin Ashton ha coniato il termine "Internet of Things". Successivamente, nel 2009, Dominique Guinard e Vlad Trifa hanno coniato il termine "Web of Things" e hanno iniziato a esplorare l'idea di estendere il Web per coprire anche il dominio degli oggetti connessi.

Il WoT si basa su principi e obiettivi chiave, tra cui l'interoperabilità, l'accessibilità, la scalabilità e l'estensibilità. L'interoperabilità è fondamen-

tale per consentire ai dispositivi di comunicare tra loro in modo affidabile e coerente, indipendentemente dalla loro tipologia o produttore. L'accessibilità si riferisce alla possibilità di accedere e controllare i dispositivi tramite l'interfaccia Web in modo semplice e intuitivo. La scalabilità si occupa della gestione efficiente di un grande numero di dispositivi connessi. L'estensibilità è importante per consentire l'aggiunta di nuovi dispositivi e funzionalità al sistema senza dover apportare modifiche significative all'architettura esistente.

Il World Wide Web Consortium (W3C) ha riconosciuto l'importanza del Web of Things e ha istituito il W3C Web of Things Interest Group nel 2014, seguito dalla creazione del W3C Web of Things Working Group nel 2017 al fine di sviluppare standard e linee guida per promuovere l'adozione e l'interoperabilità del WoT. Il W3C WoT Working Group ha prodotto diversi documenti chiave, tra cui:

- "Web of Things (WoT) Architecture": Questo documento definisce i principi e l'architettura generale del Web of Things, fornendo linee guida sulle interfacce e le interazioni tra i componenti del WoT.
- "Thing Description (TD)": Questo documento descrive la sintassi e la semantica dei Thing Description, fornendo uno standard per la descrizione dei dispositivi connessi nel Web of Things.
- "WoT Thing API": Questo documento definisce le interfacce e i metodi standardizzati per l'accesso e il controllo dei dispositivi connessi tramite il Web.
- "Discovery and Registration": Questo documento fornisce indicazioni sulle modalità di scoperta e registrazione dei dispositivi nel Web of Things, consentendo ai WoT Clients di individuare e accedere ai dispositivi disponibili.
- "Security and Privacy Considerations": Questo documento affronta le questioni di sicurezza e privacy nel contesto del Web of Things,

fornendo linee guida per proteggere i dati e le comunicazioni tra i dispositivi.

1.2.2 I costrutti del Web of Things

L'architettura del Web of Things (WoT) è basata su un insieme di componenti fondamentali che consentono l'interazione e la gestione dei dispositivi connessi. Al centro di questa architettura si trova il concetto di "Thing", che rappresenta un dispositivo fisico o virtuale all'interno del WoT. Una Thing può essere un sensore, un attuatore, un dispositivo smart, o qualsiasi altro oggetto connesso che si desidera rendere accessibile tramite il Web.

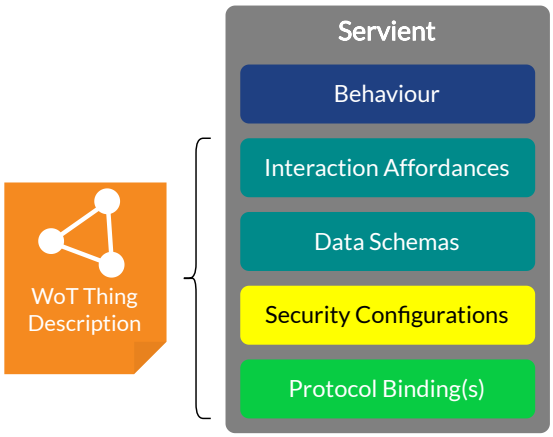

Figura 1.1: Architettura di una Web Thing

Come mostrato nella figura 4.12, una Thing nel WoT è caratterizzata da diversi aspetti chiave:

- Behaviour: rappresenta il comportamento di una Thing, definisce le azioni o le operazioni che vengono eseguite quando una Thing riceve una richiesta o un comando da parte di un'applicazione o di un'altra entità nel WoT.

- Interaction Affordances: rappresentano le azioni o le operazioni che un Thing può supportare.

- Data Schemas: rappresentano la struttura e il formato dei dati generati o consumati da una Thing, definiscono i tipi di dati, le proprietà e le relazioni che i dati possono avere.

- Security Configuration: definisce le misure di sicurezza e i meccanismi di autenticazione e autorizzazione per proteggere l'accesso alle Things nel WoT. Questo aspetto è fondamentale per garantire la privacy e la sicurezza dei dati scambiati tra i dispositivi e le applicazioni che li utilizzano.

- Protocol Bindings: specificano i protocolli di comunicazione supportati da una Thing per l'interazione con il WoT

Per descrivere un Thing nel WoT, viene utilizzata una Thing Description (TD), una componente fondamentale nel WoT che fornisce una descrizione strutturata e semantica di un Thing. La TD contiene informazioni dettagliate sulle caratteristiche, le proprietà, le interfacce, le Interaction Affordances, i protocolli di comunicazione e le configurazioni di sicurezza associate al Thing. La TD utilizza formati come JSON (JavaScript Object Notation) o JSON-LD (JSON for Linked Data) per consentire l'interoperabilità semantica tra i dispositivi e le applicazioni nel WoT. La TD permette alle applicazioni di scoprire, comprendere e interagire con i Things nel WoT in modo standardizzato e interoperabile.

L'interazione con una determinata Thing avviene attraverso i Consumer (soggetti capaci di leggere e processare le TD al fine di interagire con le Things) tramite diversi protocolli di rete, come descritto nei Protocol Bindings, mappature contenute nella Thing Description che definiscono la corrispondenza delle possibili interazioni del dispositivo ai diversi protocolli di rete supportati, come ad esempio HTPP, CoAP o MQTT.

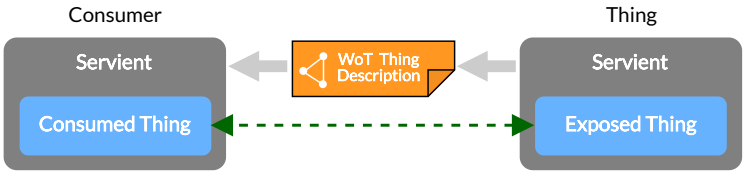

Figura 1.2: Schema di interazione tra due Servient

L'ambiente di esecuzione di una Thing viene chiamato Servient, il quale è un software che può interagire con Thing remote consumando le loro TD e può ospitarle ed esporle svolgendo contemporaneamente il ruolo di client e server. La più semplice delle interazioni, mostrata nell'immagine 1.2 si ha quando un Servient con il ruolo di server espone una Thing e la TD ad essa associata, ed un secondo Servient con ruolo di client consuma la TD esposta per interagire con la Thing.

L'esposizione e consumazione di una Thing sono due concetti alla base delle interazioni nel WoT. Il concetto di "consumazione" fa riferimento all'azione attraverso la quale un Servient interagisce con una Thing tramite il recupero e l'utilizzo della sua TD associata. Attraverso la consumazione di quest'ultima, il Servient ottiene una comprensione approfondita delle funzionalità offerte dalla Thing e delle modalità di interazione con essa. Questo consente al Servient di utilizzare la Thing in modo adeguato, ad esempio inviando richieste, ricevendo aggiornamenti o interagendo con le sue proprietà. Il concetto di "esposizione" invece si riferisce all'azione attraverso la quale un Servient ospita e rende accessibile una Thing per gli altri dispositivi o Servient all'interno del sistema. Un Servient che espone una Thing permette ad altre entità di accedere alla TD correlata e di interagire con essa utilizzando le modalità specificate nella TD.

Per interagire con una Thing, la sua Thing Description deve prima essere individuata, ottenuta ed elaborata. Il meccanismo che descrive le azioni appena citate e regola l'accesso alle Thing Descripions viene detto WoT Discovery ed è ampiamente descritto nella documentazione prodotta dal W3C

Working Group. L'ottenimento di una Thing Description può avvenire in diversi modi, a seconda dell'architettura del sistema, del comportamento della Thing associata e delle capacità del Consumer.

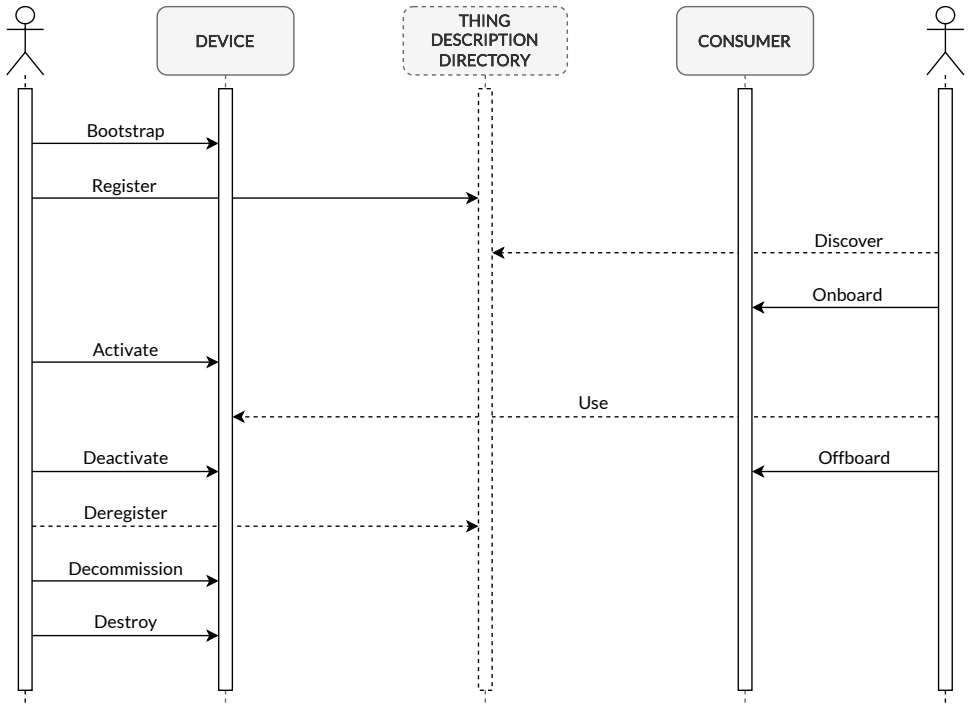

Figura 1.3: Schema delle interazioni per la registrazione presso una TDD

Uno di questi modi concerne nell'utilizzo di una Thing Description Directory (TDD), la quale non è altro che un registro che elenca e descrive Thing offrendo diversi vantaggi e funzionalità:

- Individuazione delle Things: la TDD consente agli utenti e agli altri dispositivi di individuare e accedere alle Things registrate, filtrandole sulla base dei loro metadata contenuti nei TDs;

- Pubblicazione e registrazione delle Things: le Things possono registrarsi presso la TDD per essere rese disponibili ad altri dispositivi o utenti;

- Gestione dinamica delle Things: la TDD permette un aggiornamento dinamico delle informazioni sulle Things, ovvero possono essere aggiunte, rimosse o modificate all'interno della TDD;

1.3 Mobile Crowdsensing

Il termine Mobile Crowdsensing (MCS) venne introdotto per la prima volta nel 2011, utilizzato per indicare un paradigma di rilevazione dati più generico e potente del Mobile Phone Sensing. Negli ultimi decenni, il Mobile Crowdsensing (MCS) ha guadagnato slancio grazie alla diffusa adozione di dispositivi mobili, come gli smartphone e i dispositivi IoT (Internet of Things). Questo paradigma di rilevazione dati ha aperto nuove possibilità, permettendo ai cittadini comuni di contribuire alla raccolta e all'invio dei dati generati dai loro dispositivi mobili. L'aggregazione e la fusione di questi dati consentono di estrarre informazioni utili e di offrire servizi incentrati sulla persona.

1.3.1 Cos'è il Mobile Crowdsensing

Il Mobile Crowdsensing è un paradigma di rilevazione dati che permette ai cittadini di contribuire alla raccolta di informazioni utilizzando i loro dispositivi mobili. In sostanza, ogni dispositivo diventa un "nodo" che raccoglie dati ambientali, spaziali o sociali e li invia a un server centrale per l'elaborazione e l'analisi. Questo paradigma si basa sulla partecipazione attiva degli utenti che, utilizzando le loro risorse hardware (sensori, connettività, capacità di calcolo), contribuiscono alla creazione di una rete distribuita di rilevatori.

La crescente diffusione di smartphone e dispositivi indossabili, dotati di un'ampia gamma di sensori, ha svolto un ruolo chiave nello sviluppo del Mobile Crowdsensing. Sensori come l'accelerometro, il giroscopio, il GPS, il microfono e la fotocamera consentono lo sviluppo di applicazioni MCS in vari ambiti, come la salute, l'ambiente, il monitoraggio del traffico e il controllo degli edifici.

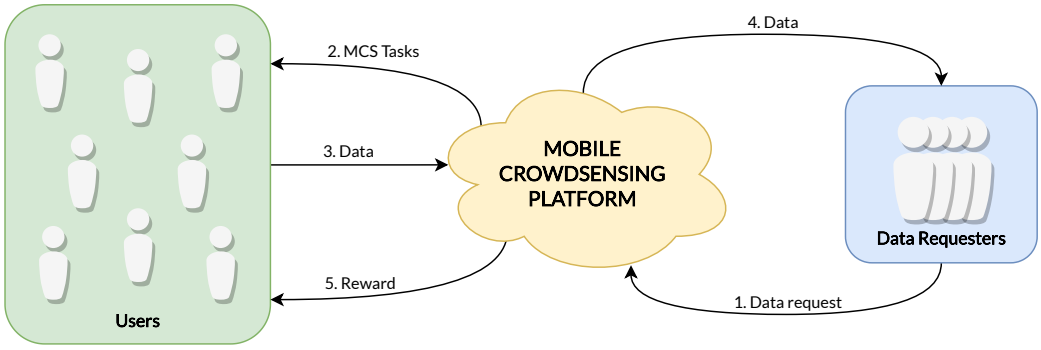

Figura 1.4: Interazioni tra le parti di una sistema di MCS

1.3.2 Tipologie di applicazioni di Mobile Crowdsensing

Le applicazioni di Mobile Crowdsensing possono essere suddivise in due categorie principali: Personal Sensing (rilevazione personale) e Community Sensing (rilevazione comunitaria).

Il Personal Sensing si concentra sull'analisi e il monitoraggio di eventi legati all'individuo. Ad esempio, queste applicazioni possono monitorare i pattern di movimento di una persona per determinare se sta correndo, camminando o facendo esercizio fisico. Allo stesso modo, possono rilevare le funzioni vitali come il battito cardiaco per scopi di monitoraggio della salute.

La Community Sensing, invece, si occupa del monitoraggio di fenomeni su larga scala che sarebbero impossibili da misurare per singoli individui. Ad esempio, queste applicazioni possono monitorare il traffico stradale o il livello di inquinamento dell'aria. La rilevazione comunitaria può avvenire in due modi: il partecipatory sensing coinvolge direttamente gli individui nella raccolta dei dati, mentre l'opportunistic sensing cerca di ridurre il coinvolgimento attivo degli utenti, ad esempio sfruttando il monitoraggio continuo della posizione dell'utente senza richiedere un'azione manuale.

1.3.3 Aspetti di interesse del Mobile Crowdsensing

Il Mobile Crowdsensing si focalizza su diversi aspetti chiave per garantire il suo successo e l'efficacia delle applicazioni che lo utilizzano. Uno di questi

aspetti è l'allocazione delle attività di raccolta dei dati ai partecipanti, noto come pianificazione delle attività. Questo processo consiste nell'assegnare compiti specifici ai lavoratori mobili, tenendo conto delle loro capacità, della loro posizione geografica e delle richieste delle attività stesse. Ci sono due strategie ben definite per la pianificazione, chiamate rispettivamente Push e Pull, note anche come proattive e reattive [2]. Nelle architetture basate sul Push, il lavoratore riceve la richiesta di eseguire un compito assegnato dal crowdsourcer. D'altra parte, nelle architetture basate sul Pull, i lavoratori decidono in modo proattivo quando e a quali compiti desiderano contribuire. Questa flessibilità nella pianificazione delle attività consente ai lavoratori di partecipare in base alle loro preferenze e capacità, consentendo una maggiore adattabilità e coinvolgimento nella raccolta dei dati.

Un altro aspetto fondamentale è la gestione dei pagamenti e delle questioni legate alla privacy. Poiché il Mobile Crowdsensing coinvolge la partecipazione degli utenti, è importante garantire meccanismi adeguati per gestire i pagamenti per i loro contributi. Allo stesso tempo, è fondamentale proteggere la privacy degli utenti, poiché la raccolta dei dati potrebbe coinvolgere informazioni personali sensibili. Pertanto, devono essere adottate misure per garantire la privacy e il trattamento sicuro dei dati raccolti.

La partecipazione degli utenti è un ulteriore aspetto cruciale per il successo del Mobile Crowdsensing. Per stimolare la partecipazione, è necessario adottare strategie che motivino gli utenti a contribuire attivamente alla raccolta dei dati. Una delle principali sfide è superare le possibili resistenze degli utenti nell'invio dei dati, specialmente quando si tratta di informazioni sensibili come la posizione geografica o altre informazioni personali. Per affrontare questa sfida, è comune ricorrere a meccanismi di incentivazione. Questi possono includere premi in denaro, sconti su servizi o ricompense non monetarie come badge, punti o riconoscimenti. Tali incentivi possono incoraggiare gli utenti a partecipare e condividere i loro dati in modo più aperto. È importante progettare sistemi di incentivazione equi, trasparenti e che rispettino la privacy degli utenti.

1.3.4 Il Mobile Crowdsensing per campagne di raccolta dati

Il Mobile Crowdsensing è diventato un prezioso strumento per condurre campagne di raccolta dati su larga scala. Grazie alla diffusione diffusa di dispositivi mobili tra la popolazione, è possibile coinvolgere un gran numero di partecipanti e raccogliere dati in modo efficiente e tempestivo. Questo approccio offre numerosi vantaggi rispetto ai tradizionali metodi di raccolta dati, aprendo nuove opportunità per la ricerca, la pianificazione urbana, la gestione ambientale e altre applicazioni.

Le campagne di raccolta dati basate sul Mobile Crowdsensing consentono di coprire aree geografiche estese che altrimenti sarebbero difficili da monitorare. Ad esempio, la mappatura del traffico in tempo reale può essere effettuata utilizzando i dati generati dai dispositivi mobili dei partecipanti, consentendo una visione più completa delle condizioni stradali e una migliore pianificazione dei percorsi. Allo stesso modo, la raccolta di dati ambientali, come la qualità dell'aria o il livello di inquinamento acustico, può essere estesa su vasti territori grazie alla partecipazione distribuita degli utenti.

Inoltre, il Mobile Crowdsensing offre una maggiore diversità di partecipanti rispetto ai tradizionali metodi di raccolta dati, consentendo di coinvolgere individui con background e prospettive diverse. Questo contribuisce a una maggiore rappresentatività dei dati raccolti e può portare a risultati più accurati e completi nelle analisi successive.

Un'altra caratteristica importante del Mobile Crowdsensing è la sua capacità di raccogliere dati in tempo reale. Grazie alla connettività dei dispositivi mobili, i dati possono essere inviati immediatamente ai server di raccolta e analizzati in tempo reale. Ciò consente di ottenere informazioni aggiornate e reattive, utili per la gestione di situazioni di emergenza, il monitoraggio di eventi temporanei o la valutazione di interventi pubblici.

Tuttavia, ci sono anche sfide da affrontare nell'utilizzo del Mobile Crowdsensing come strumento per le campagne di raccolta dati. Una delle principali sfide è garantire la qualità e l'affidabilità dei dati raccolti. Poiché i

partecipanti sono utenti privati e non professionisti, è importante sviluppare meccanismi per la validazione e la verifica dei dati raccolti al fine di evitare informazioni fuorvianti o errate.

1.4 LA-MQTT

MQTT (Message Queuing Telemetry Transport) è un protocollo di comunicazione leggero di tipo publish-subscribe ampiamente utilizzato negli scenari Internet of Things (IoT). Tuttavia, il protocollo MQTT standard presenta una limitazione significativa per le applicazioni di Mobile Crowdsensing (MCS) e altri contesti che richiedono servizi basati sulla posizione: non supporta la consapevolezza della posizione dei dispositivi.

Per superare questa limitazione, è stata sviluppata un'estensione chiamata LA-MQTT (Location-Aware MQTT) che introduce la consapevolezza della posizione nelle comunicazioni MQTT. LA-MQTT consente ai consumatori di ricevere solo i dati rilevanti in base alla loro posizione geografica e all'argomento di interesse. Ciò significa che i dispositivi MCS possono inviare e ricevere dati solo alle posizioni pertinenti e agli utenti interessati.

L'implementazione di LA-MQTT richiede l'aggiunta di un'applicazione backend collegata ai broker MQTT standard. Questa applicazione backend gestisce le informazioni sulla posizione e la relativa consapevolezza spaziale. I dispositivi mobili IoT, come gli smartphone, devono costantemente aggiornare il backend di LA-MQTT con le informazioni sulla loro posizione corrente. In questo modo, i broker MQTT possono notificare solo i consumatori interessati situati nelle aree pertinenti, riducendo così la quantità di dati trasmessi e migliorando l'efficienza complessiva delle comunicazioni.

L'utilizzo di LA-MQTT comporta alcuni scambi di messaggi aggiuntivi rispetto al protocollo MQTT standard, poiché richiede la gestione delle informazioni sulla posizione e l'aggiornamento continuo dei dati di posizionamento. Tuttavia, questo approccio consente un migliore bilanciamento tra la consapevolezza della posizione e le implicazioni sulla privacy. Gli uten-

ti possono beneficiare di servizi basati sulla posizione più mirati e rilevanti, mentre allo stesso tempo vengono preservate le loro informazioni personali.

L'implementazione di LA-MQTT apre nuove possibilità per le applicazioni di Mobile Crowdsensing. Ad esempio, nelle campagne di MCS che coinvolgono una vasta area geografica, come il monitoraggio ambientale o la rilevazione delle condizioni stradali, LA-MQTT consente di inviare dati solo alle posizioni pertinenti.

In sintesi, LA-MQTT rappresenta una soluzione innovativa per rendere MQTT consapevole della posizione, consentendo una comunicazione più efficiente e mirata in contesti di Mobile Crowdsensing e altre applicazioni IoT basate sulla posizione.

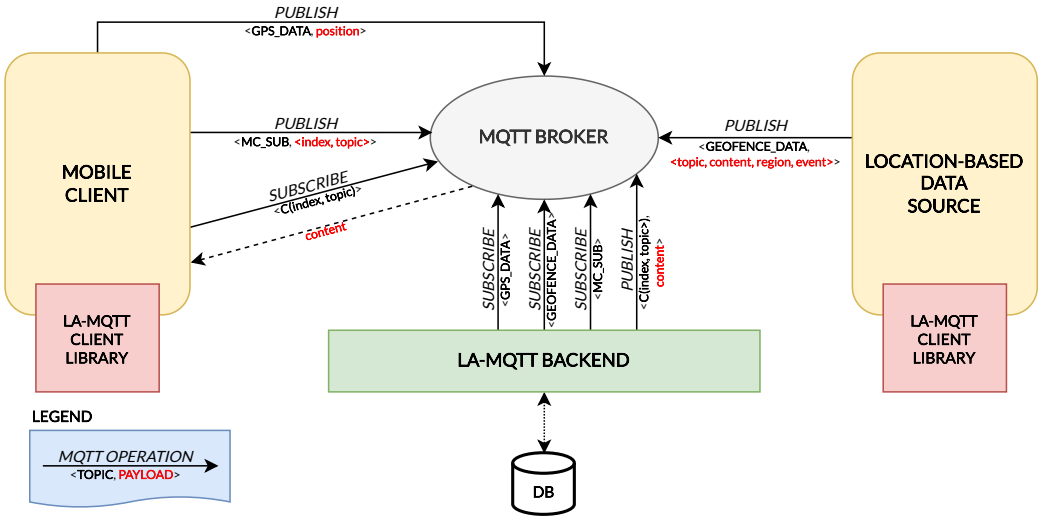

Figura 1.5: Schema di funzionamento del protocollo LA-MQTT

1.5 Motivazioni

Il Mobile Crowdsensing (MCS) ha l'obiettivo di raccogliere dati attraverso dispositivi mobili, sfruttando la vasta diffusione degli smartphone in diverse aree geografiche. Questo approccio offre la possibilità di ottenere dati con

una capillarità altrimenti impossibile da raggiungere con metodi tradizionali di raccolta dati.

Dall'altra parte, il Web of Things (WoT) si propone di standardizzare e semplificare la comunicazione tra dispositivi connessi, consentendo a una "Thing" di consumare la descrizione di un'altra "Thing" (Thing Description, TD) a condizione che implementi le specifiche descritte in essa. Ciò permette l'interazione e lo scambio di dati tra dispositivi eterogenei.

L'integrazione del Web of Things in un sistema strutturato per il Mobile Crowdsensing porta numerosi vantaggi, oltre a quelli offerti intrinsecamente dal framework WoT. In particolare:

- Diverse modalità di partecipazione alla raccolta dati: Grazie all'utilizzo del WoT, le parti coinvolte nel Mobile Crowdsensing possono comunicare attraverso un'ampia gamma di protocolli e metodi. Ciò aumenta le capacità di comunicazione rispetto a quanto sarebbe possibile utilizzando un unico protocollo.

- Trasmissione ibrida dei dati: L'utilizzo simultaneo di diversi protocolli consente una trasmissione ibrida dei dati, in cui è possibile scegliere il protocollo più adatto per ogni tipo di dato. Ciò garantisce una maggiore efficienza e flessibilità nella gestione dei dati raccolti.

- Accessibilità a una vasta gamma di dispositivi: L'uso di diversi protocolli per l'invio dei dati amplia la platea di dispositivi che possono partecipare al Mobile Crowdsensing. Questo favorisce la partecipazione di un numero maggiore di utenti e dispositivi, contribuendo a una raccolta dati più completa.

- Gestione semplificata delle interazioni tra dispositivi: L'architettura WoT si occupa di gestire le interazioni tra i vari dispositivi, riducendo il carico di lavoro per i programmatori. Ciò consente ai crowdsourcer di concentrarsi sulla definizione delle richieste di raccolta dati, semplificando lo sviluppo e la gestione dei sistemi di Mobile Crowdsensing.

Attualmente, non esistono soluzioni che sfruttino appieno il potenziale dell'unione tra Mobile Crowdsensing e Web of Things. Le architetture esistenti proposte per il Mobile Crowdsensing sono basate o sulla strategia Push o su quella Pull, con la conseguenza di una limitata flessibilità. Ciò impedisce la creazione di campagne che possono supportare entrambi gli approcci, limitando così le possibilità dei crowdsourcer di adattarsi a diversi tipi di scenari. La soluzione proposta in questo elaborato prevede uno scenario in cui siano presenti più campagne di MCS, ognuna con specifiche proprie, e consente ai device che prendono parte al sistema di decidere autonomamente quali tipologie di dati fornire, fornendo un incipit di privacy per l'utente, e fornendo all'utente la possibilità di decidere tra diverse modalità di invio dei dati, se inviarli autonomamente o se lasciare che sia la campagna stessa a leggerli dal dispositivo dell'utente quando ritiene che sia il momento migliore ai fini della campagna stessa.

Capitolo 2

Architettura

Nel presente capitolo, verrà presentata l'architettura proposta, progettata per consentire la partecipazione degli utenti alle campagne, i quali agiscono come sensori distribuiti per raccogliere dati e fornire informazioni utili. In questa sezione introduttiva, verranno presentati gli obiettivi principali dell'architettura e i componenti chiave coinvolti, illustrando ruoli, funzionalità ed interazioni tra essi.

2.1 Obiettivi

L'obiettivo principale della tesi è la realizzazione di un sistema per effettuare campagne di Mobile Crowdsensing utilizzando il paradigma Push-Pull e sfruttando le potenzialità offerte dal Web of Things. L'obiettivo è creare un sistema flessibile e scalabile che permetta di riconoscere le capacità sensoriali dei dispositivi mobili partecipanti e di offrire loro la possibilità di partecipare soltanto alle campagne per le quali sono idonei. Inoltre, si mira a sfruttare i diversi protocolli di comunicazione disponibili per creare diverse tipologie di campagne, sia localizzate che non.

Gli obiettivi specifici del sistema sono i seguenti:

- Implementazione del paradigma Push-Pull: Si intende sviluppare un sistema che consenta di utilizzare sia la modalità Push che Pull per

le campagne di Mobile Crowdsensing. La modalità Push permette di inviare richieste di raccolta dati ai dispositivi mobili in modo attivo, mentre la modalità Pull consente ai dispositivi di richiedere e rispondere a richieste di raccolta dati.

- Riconoscimento delle capacità sensoriali dei dispositivi: Il sistema deve essere in grado di riconoscere le capacità sensoriali dei dispositivi mobili partecipanti. Ciò consentirà di valutare quali dispositivi sono idonei per le diverse campagne di raccolta dati e di assegnare loro le attività adeguate in base alle loro caratteristiche hardware.

- Selezione delle campagne idonee per i dispositivi mobili: Il sistema dovrà valutare le caratteristiche dei dispositivi mobili partecipanti e offrire loro soltanto la possibilità di partecipare alle campagne di raccolta dati per le quali sono idonei.

- Utilizzo di molteplici protocolli di comunicazione: Sarà fondamentale sfruttare i diversi protocolli di comunicazione disponibili, come HTTP, MQTT o CoAP, per creare diverse tipologie di campagne di Mobile Crowdsensing. Si potranno così realizzare campagne localizzate, che richiedono dati specifici da una determinata area geografica, o campagne non localizzate che coinvolgono dispositivi mobili distribuiti in diverse località.

2.2 Attori

Nel sistema proposto, sono identificati due attori principali:

- Crowdsourcer: Il crowdsourcer rappresenta l'entità che lancia una campagna di Mobile Crowdsensing all'interno del sistema. Si tratta di un attore esterno che ha il compito di definire i dettagli della campagna, come il tipo di dati da raccogliere, il periodo di raccolta, le ricompense, gli intervalli tra ogni invio dei dati e le specifiche richieste per i

dispositivi partecipanti. Il crowdsourcer è responsabile di fornire le informazioni necessarie per la corretta configurazione della campagna e per valutare i risultati ottenuti.

- Utenti dell'app Android (Worker): Gli utenti dell'app Android rappresentano l'insieme dei dispositivi mobili che partecipano attivamente alle campagne di Mobile Crowdsensing. Essi svolgono il ruolo di worker nel sistema, contribuendo alla raccolta dei dati richiesti dalla campagna. Gli utenti dell'app Android devono installare l'applicazione sul proprio dispositivo e accettare di partecipare alle campagne proposte. Essi sono responsabili di raccogliere e inviare i dati richiesti secondo le specifiche della campagna.

2.3 Componenti

Il sistema è composto da 5 componenti che interagiscono tra loro come mostrato nella figura 2.1.

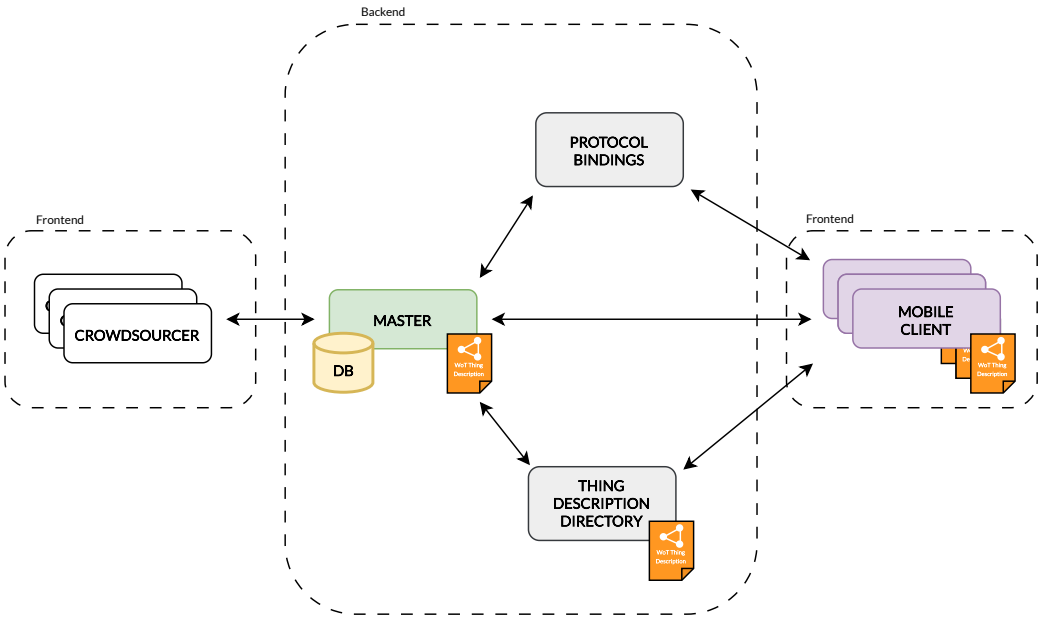

Figura 2.1: Architettura della comunicazione App-Server

Di seguito vengono elencate le funzionalità dei vari componenti:

- Mobile Client: Il Mobile Client rappresenta l'applicazione Android installata sui dispositivi degli utenti e consente loro di partecipare alle campagne. Ha il compito di gestire l'invio dei dati, raccogliendoli dai sensori del dispositivo ed inviandoli al server per l'elaborazione. Mostra agli utenti informazioni utili, come i dati che stanno inviando, l'avanzamento nel completamento delle campagne, le campagne disponibili per la partecipazione e alcune informazioni personali dell'utente. Implementa un Servient con ruolo di Server, per esporre la Thing Description del dispositivo presso la Thing Directory rendendo il dispositivo individuabile dal Server esponendo le sue capacità sensoriali, e di Client, per consumare la TD del Server ed interagire con esso.

- Master: Costituisce il componente principale, con il quale ognuno degli altri componenti si interfaccia. Gestisce la connessione e comunicazione con i client, inviando loro richieste di raccolta dati e ricevendo i dati raccolti. Implementa un Servient con ruolo sia di Client che di Server, rendendosi individuabile dai Client registrandosi presso la Thing Directory, venendo notificato da essa ad ogni connessione o disconnessione di un Client dal sistema. Elabora i dati ricevuti e fornisce i risultati ai crowdsourcer. Inoltre, il server gestisce l'autenticazione degli utenti e la gestione delle campagne attive.

- Thing Description Directory (TDD): La TDD ha il compito di raccogliere, mantenere e rendere disponibili tutte le Thing Description dei Mobile Client e del Master, i quali contengono, per i Mobile Client le informazioni relative alle loro capacità sensoriali, mentre per il Master le indicazioni per interagire con esso per autenticarsi, ottenere la lista delle campagne disponibili, partecipare ad esse e gestire l'invio dei dati.

- Database: Il database si occupata di memorizzare i dati raccolti durante le campagne e i dati degli utenti. Comunica con esso il Master,

il quale fornisce e legge i dati, tra cui le informazioni relative ad ogni utente, come i dati inviati per le campagne, le ricompense ottenute, le campagne completate e lo stato di avanzamento delle campagne non ancora terminate.

- Applicativi a supporto dei protocol binding: La comunicazione tra Mobile Client e Master avviene seguendo gli standard implementativi del Web of Things, e quindi tramite i Protocol Binding contenuti nelle Thing Description dei Mobile Client, e consumati dal Master e viceversa. A seconda del Protocol Binding utilizzato possono essere richiesti zero o più componenti esterni di supporto, come nel caso di utilizzo del protocollo MQTT che necessita di un broker per mediare le trasmissioni di messaggi tra le parti, o nel caso di LA-MQTT che fa utilizzo, in aggiunta ad un broker come per MQTT, di un applicativo backend che gestisce gli aggiornamenti delle posizioni dei dispositivi e si occupa di inviare i messaggi agli utenti.

2.4 Interazioni

Le interazioni tra i vari componenti del sistema sono fondamentali per il corretto svolgimento delle campagne di Mobile Crowdsensing. Ogni componente interagisce con uno o più componenti, seguendo le relazione descritte di seguito.

- Crowdsourcer e Master: Il Crowdsourcer interagisce con il Master durante la definizione dei dettagli di una nuova campagna di Mobile Crowdsensing. Il Crowdsourcer fornisce le informazioni necessarie al Master, come il tipo di dati da raccogliere e le specifiche richieste per i dispositivi partecipanti. Il Master elabora tali informazioni e crea la campagna corrispondente. Durante l'esecuzione della campagna il Crowdsourcer si interfaccerà poi con il Master per leggere ed analizzare i dati raccolti.

- Mobile Client e TDD: L'interazione tra il Mobile Client e la TDD avviene in maniera bidirezionale. Nel primo verso, dal Mobile Client alla TDD, il primo si registra presso la seconda inviando la propria Thing Description. Questa registrazione permette al Mobile Client di comunicare le informazioni sulle capacità sensoriali del dispositivo, inclusi i tipi di sensori disponibili e le relative caratteristiche. Nel verso opposto invece, la TDD fornisce al Mobile Client la TD del Master, che contiene una descrizione dettagliata delle funzionalità offerte, i servizi esposti e le sue proprietà (tra cui le campagne disponibili).

- Master e TDD: Anche l'interazione tra Master e TDD è bidirezionale, e consente al Master di interrogare la TDD per ottenere le TD dei dispositivi, consumarle per ottenere la consapevolezza dei dispositivi partecipanti al sistema e disponibili all'invio dei dati e le loro capacità sensoriali. Inoltre il Master espone la propria TD registrandosi presso la TDD così da essere individuabile dai dispositivi e rendere possibile l'interazione con essi.

- Mobile Client e Master: Il Mobile Client comunica con il Master per inviare i dati raccolti dai sensori del dispositivo e ricevere richieste di raccolta dati. Invia periodicamente i dati raccolti al Master, che li memorizza nel database per l'elaborazione successiva. Inoltre, può ricevere richieste di raccolta dati dal Master, che vengono visualizzate all'utente dell'app Android per l'esecuzione delle attività richieste.

2.5 Fasi d'esecuzione

Vengono ora elencate le principali fasi di esecuzione dell'architettura, descrivendo per ognuna i componenti coinvolti e come interagiscono tra essi.

Le prime fasi sono quelle relative all'avvio e inizializzazione dei componenti. All'avvio del sistema il Master deve infatti eseguire alcune azioni prima di diventare operativo, descritte di seguito e mostrate nelle immagini

2.2 e 2.3. La prima operazione da eseguire consiste nella registrazione del Master presso la TDD, così da renderlo disponibile a tutti i dispositivi che parteciperanno al sistema. Ciò avviene pubblicando sulla TDD un TD che rappresenta la Thing associata al Master, e che contiene tutte le operazioni che un Servient che la consuma può eseguire verso il Master.

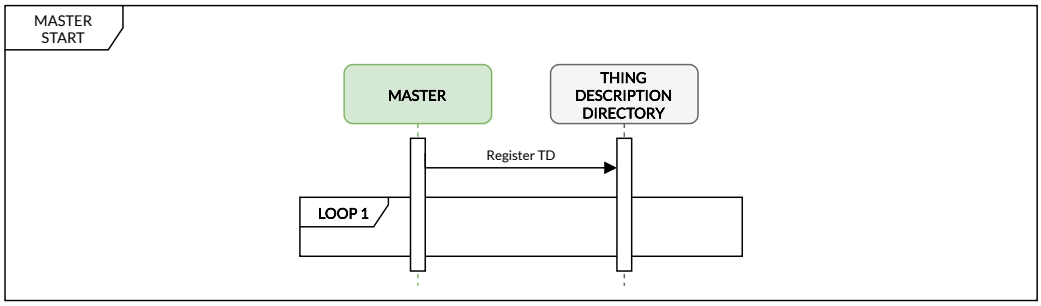

Figura 2.2: Interazioni d'avvio del Master

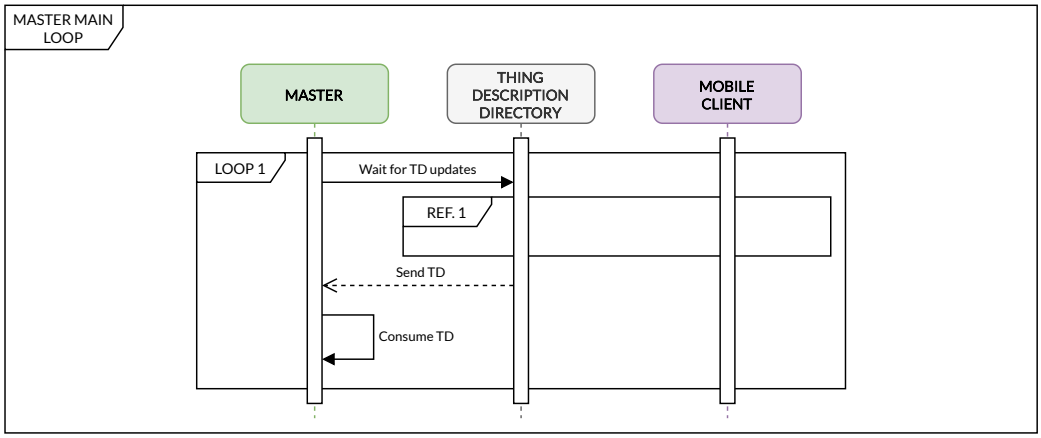

Figura 2.3: Interazioni del ciclo d'esecuzione del Master

Dopo aver pubblicato il proprio TD, il Master attende che altri dispositivi prendano parte al sistema, rimanendo in ascolto di aggiornamenti da parte della TDD. Come sarà mostrato successivamente, dopo la registrazione di una Mobile Client presso la TDD, un nuovo TD associato ad esso sarà reso disponibili ed il Master verrà notificato. Ciò consente al Master di ottenere

il nuovo TD e consumarlo, per ottenere l'accesso a tutte le operazioni e i dati messi a disposizione dal Mobile Client.

Per un corretto funzionamento dell'architettura, è necessario che siano presenti delle campagne di Mobile Crowdsensing alle quali i Mobile Client possano partecipare. L'azione di creazione di una nuova campagna, mostrata nella figura 2.4 ha origine dal Crowdsourcer, che ne definisce le proprietà, gli obiettivi e le specifiche. Dopo aver definito questo dati ed aver compilato un form che li rispecchia, idealmente tramite un'applicazione web, viene inviata al Master la richiesta di creazione di una nuova campagna, il quale, dopo aver effettuato dei controlli di verifica sui dati ricevuti, salva i dati della campagna sul Database.

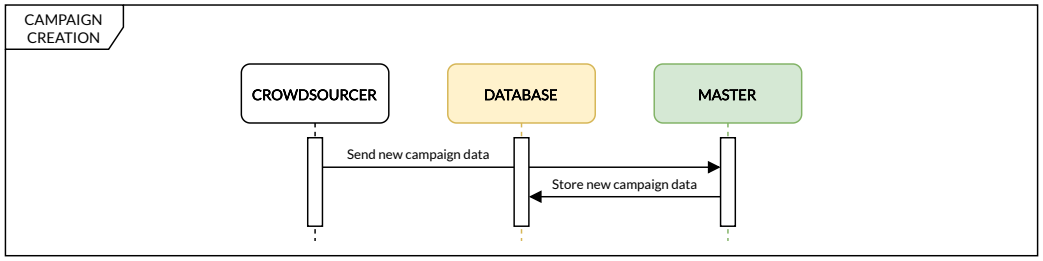

Figura 2.4: Interazioni per la creazione di una nuova campagna

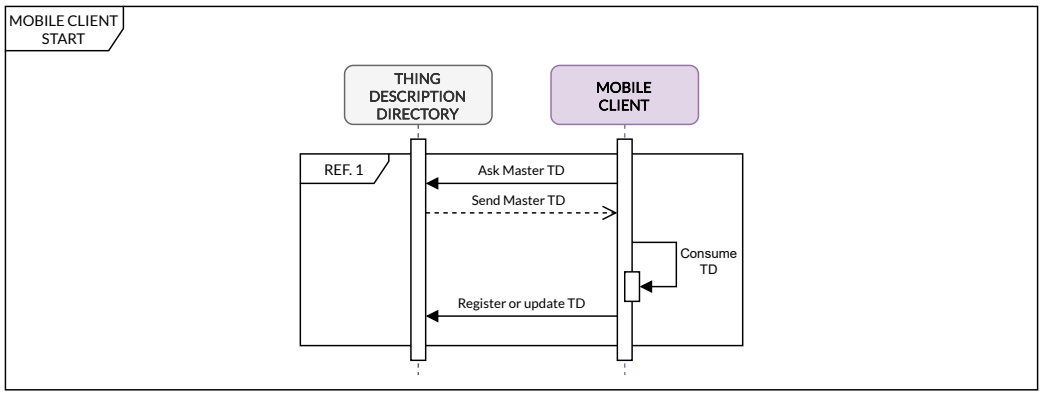

Figura 2.5: Interazioni d'avvio del Mobile Client

La figura 2.5 mostra la routine di avvio del Mobile Client, che inizia richiedendo il TD del Master, precedentemente registrato, alla TDD. Dopo

l'ottenimento di esso, il Mobile Client ottiene la visibilità sulle azioni possibili verso il Master. Una di queste, preliminare alle altre, è l'autenticazione, con cui l'utente effettua l'accesso al proprio profilo personale e consente al Mobile Client di richiamare azioni che necessitano di autenticazione, come la partecipazione ad una campagna o l'invio di dati per una di esse.

L'azione di richiesta e ottenimento della lista di campagne non è altro che una lettura di una proprietà esposta dal Master. Le campagne disponibili sono salvate sul Database, e quando il Master riceve la richiesta di lettura legge i dati da esso. La richiesta passa attraverso uno dei Protocol Binding disponibili sul Master, che gestisce la comunicazione tra esso e il Mobile Client.

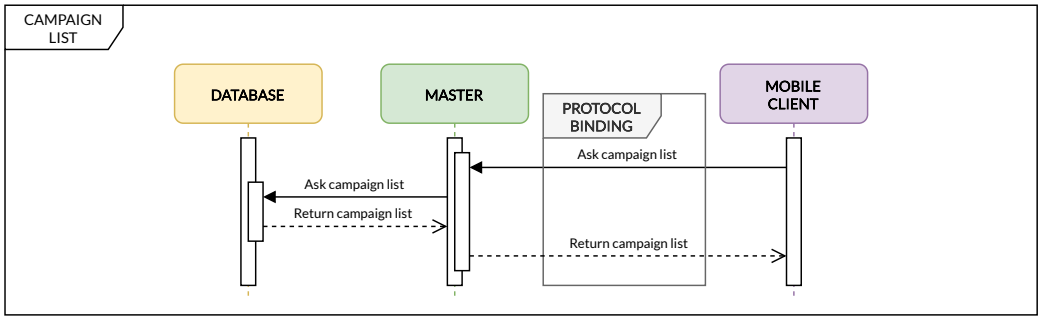

Figura 2.6: Interazioni per la lettura delle campagne disponibili

Una volta decisa una campagna dati a cui partecipare, l'utente deve effettuare la registrazione verso essa per confermare la sua volontà a contribuire all'invio dei dati, per verificare i permessi e le capacità sensoriali del dispositivo che sarà utilizzato e per comunicare al Master la modalità di invio dei dati scelta, tra Push o Pull, e alcune preferenze d'invio, come l'intervallo di tempo tra un invio e l'altro. L'azione di partecipazione ad una campagna ha quindi origine dal Mobile Client, il quale interagisce con il Master mediante un'azione messa a disposizione appositamente per la registrazione. L'attivazione di questa azione, assieme ad alcuni parametri passati in input, consente al Mobile Client di contrassegnare l'utente che ha effettuato l'autenticazione come partecipante alla campagna selezionata, salvando i risultati di questa

operazione sul Database. Una volta ricevuta la conferma di registrazione per la campagna il Mobile Client aggiorna il proprio TD presso la TDD, esponendo nuove proprietà che permettono al Master di leggere i dati richiesti dalla campagna. Il Master sarà quindi notificato dell'aggiornamento e leggerà dalla TDD il nuovo TD, consumandolo per avere accesso alle nuove proprietà messe a disposizione dal Mobile Client.

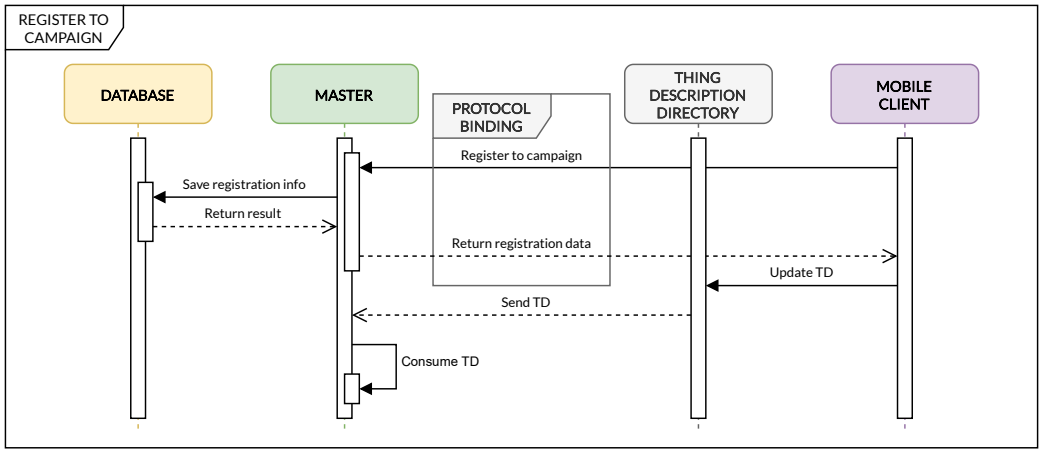

Figura 2.7: Interazioni per la registrazione ad una campagna

Ogni campagna può avere diverse modalità di invio dei dati disponibili, decise dal Crowdsourcer al momento della creazione, che si differenziano nelle modalità e nei dettagli riguardanti le specifiche che l'utente dovrà seguire durante l'invio dei dati. La differenza principale si ha nella strategia di invio dei dati che può variare tra Push e Pull e determina chi sarà il soggetto, tra Mobile Client e Master, incaricato a guidare l'invio dei dati. Di seguito gli schemi che ne illustrano le differenze e il funzionamento.

La strategia Pull consente all'utente di inviare dati in maniera indipendente, è infatti il Mobile Client ad originare la trasmissione dei dati verso il Master. Durante la registrazione ad una campagna se viene scelta la modalità Pull l'utente potrà decidere in autonomia quando ritiene che sia il momento più opportuno per l'invio dei dati. Le modalità di invio che utilizzano la modalità Pull non presentano infatti ulteriori importanti specifiche a cui l'utente dovrà attenersi, se non quelle legate alla tipologia di dati rac-

colti dalla campagna, ad esempio se la campagna è localizzata e richiede la presenza all'interno di una certa area geografica l'utente non potrà inviare i suoi dati a meno di trovarsi all'interno della zona adibita alla raccolta dati. L'unica specifica che può ricevere un utente che partecipa ad una campagna scegliendo la strategia Pull sarà quella riguardante l'intervallo raccomandato di invio dei dati, che sarà però a sua discrezione decidere se seguire o meno.

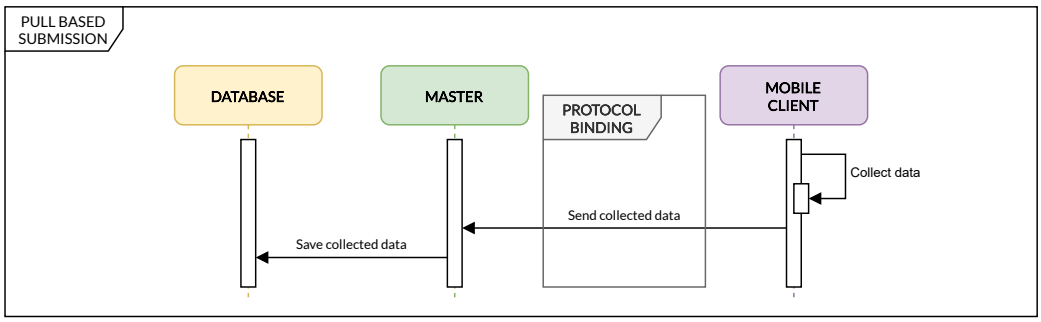

Figura 2.8: Interazioni per la strategia Pull

L'invio dei dati tramite la modalità Pull ha inizio dal Mobile Client, il quale raccoglie un campione dei dati necessari alla campagna, il che si può tradurre nella lettura di un sensore a disposizione del dispositivo, nello scatto di una fotografia, nella registrazione di un suono o qualsiasi altro dato che possa essere raccolto e inviato. Successivamente invia i dati al Master utilizzando un endpoint messo a disposizione da esso, specificato nel TD di quest'ultimo, utilizzando uno dei protocol bindings supportati dal Master e che implementa il Mobile Client. Una volta ricevuti i dati il Master effettua delle verifiche su essi per determinarne la qualità (quanto i dati sono puliti, leggibili e utili), la conformità (se rispecchiano la tipologia di dato richiesto) e la consistenza (se presenta errori, incongruenze o contraddizioni). Qualora i dati superino i controlli necessari vengono registrati nel database assieme alla conferma di avvenuto invio da parte dell'utente. Il vantaggio della partecipazione ad una campagna tramite la strategia Pull risiede infatti nella possibilità di decidere autonomamente il momento in cui inviare i dati e ciò garantisce all'utente un maggior controllo sui dati trasmessi.

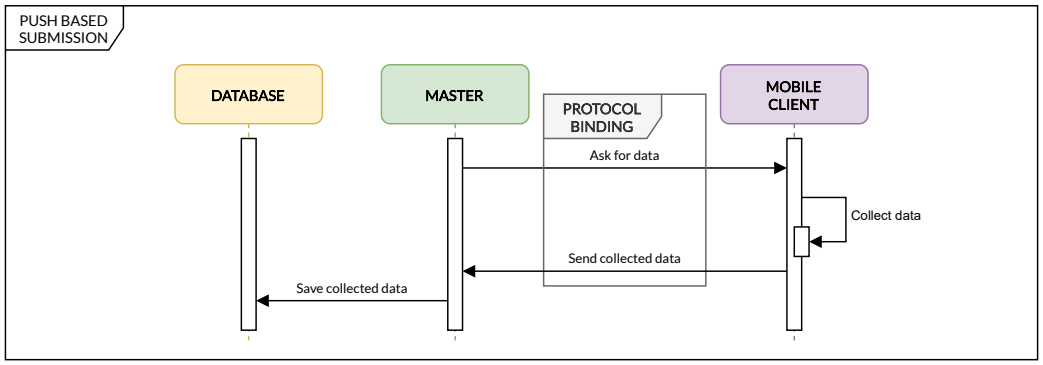

Figura 2.9: Interazioni per la strategia Push

Dall'altra parte, la strategia Push pone nel Master la responsabilità della gestione della raccolta dei dati. Vengono infatti utilizzate le proprietà esposte dal Mobile Client per comunicare con esso tramite il Protocol Binding per MQTT o LA-MQTT. Il Master consumando i TD dei diversi Mobile Client viene a conoscenza delle modalità per richiedere loro i dati, i quali saranno appositamente stati esposti dal device in concomitanza con la partecipazione ad un campagna. Ogni proprietà di un Mobile Client corrisponderà infatti ad una campagna a cui sta partecipando. Grazie a questa suddivisione il Master riesce ad individuare, a partire dalla richiesta, per quale campagna sta richiedendo e riceverà dei dati. Gli intervalli tra un invio e l'altro sono scanditi dal valore indicato nelle proprietà della campagna, o dalle preferenze che l'utente ha impostato durante la partecipazione ad essa. Una volta effettuata la richiesta, essa viene ricevuta dal Mobile Client il quale avrà un handler predisposto al fornire i dati richiesti. Viene quindi popolata la risposta con i dati e restituita al Master come valore di ritorno dell'interazione effettuata. Il Master riceve i dati ed analogamente alla strategia Pull effettua le verifiche su di essi ed in caso di esito positivo li salva sul database. L'utilizzo di un strategia Push porta al Master il vantaggio di aver accesso ai dati del Mobile Client a suo piacimento, i quali saranno messi a sua disposizione fintanto che il dispositivo rimarrà accessibile tramite una connessione Internet. In aggiunta, questa modalità permette la creazione di campagne di raccolta dati per le quali non si sa quando saranno richiesti i dati, consentendo agli utenti

che lo vogliono di fornire un più ampio controllo sui propri dati.

Capitolo 3

Implementazione

Dopo aver ampiamente descritto i componenti facenti parte dell'architettura, il capitolo corrente illustrerà il processo implementativo di essi trattando le tecnologie utilizzate, le scelte progettuali adottate e le funzionalità ottenute, ponendo maggior enfasi sugli aspetti tecnici che caratterizzano l'implementazione. La realizzazione dei diversi componenti ha seguito i pattern implementativi utilizzati nella libreria `node-wot` [7] realizzata in Typescript dal team Eclipse ThingWeb ed ormai riconosciuta in forma ufficiosa come implementazione standard delle specifiche del WoT del W3C.

3.1 Protocol Bindings

I Protocol Bindings comprendono tutti quei costrutti utilizzati per gestire le comunicazioni tra le diverse Things. Ogni Protocol Binding ha un protocollo associato e fornisce i servizi per il suo utilizzo lato Server o Client, rispettivamente, se il Servient che ne farà uso lo utilizzerà per esporre Things o per comunicare con Things consumate. Al momento il W3C non ha realizzato dei documenti normativi che ne descrivono l'implementazione, sono però presenti alcuni documenti che racchiudono le "best prectices" [5].

Nell'architettura presentata i componenti comunicano tra loro sfruttando diversi protocolli:

- HTTP e CoAP: protocolli di comunicazione client-server, vengono utilizzati per la comunicazione dal Mobile Client al Master.

- MQTT: protocollo di comunicazione publish-subscribe, viene utilizzato per le comunicazioni dal Master al Mobile Client in quanto si suppone che sia un dispositivo mobile di un utente e come tale non dispone di un indirizzo IP pubblico e statico che permette la comunicazione diretta con esso da altri soggetti.

- LA-MQTT: protocollo basato su MQTT, che ne eleva le potenzialità per l'utilizzo in ambienti "location aware". Rende possibile iscriversi a topic geograficamente delimitati ricevendo soltanto i messaggi interessanti, filtrati in base alla posizione.

3.1.1 Protocol Binding per MQTT

Nelle librerie utilizzate il Protocol Binding per MQTT presentava delle limitazioni, intrinseche al protocollo e alle raccomandazioni del W3C [6] seguite dagli autori delle librerie. Il protocollo MQTT mette a disposizione dei client solamente due azioni: `publish` e `subscribe`, e con un utilizzo semplice e lineare di esse non è possibile pubblicare un messaggio e leggere un valore di ritorno originato dalla ricezione di esso perché non previsto. Queste limitazioni comportano l'impossibilità di leggere le proprietà esposte da una Thing (a meno dell'utilizzo della flag `retain` durante la pubblicazione di un messaggio) o di invocare un'azione ed attendere un valore di ritorno. L'architettura realizzata necessitava di poter leggere dei valori di ritorno in seguito all'interazione con una Thing tramite il Protocol Binding per MQTT, ad esempio durante la partecipazione ad una campagna in modalità Push il Mobile Client deve esporre delle interazioni che verranno utilizzate dal Master per la lettura dei dati. Per gestire i valori di ritorno e la lettura delle proprietà esposte sono quindi state apportate delle modifiche al Protocol Binding.

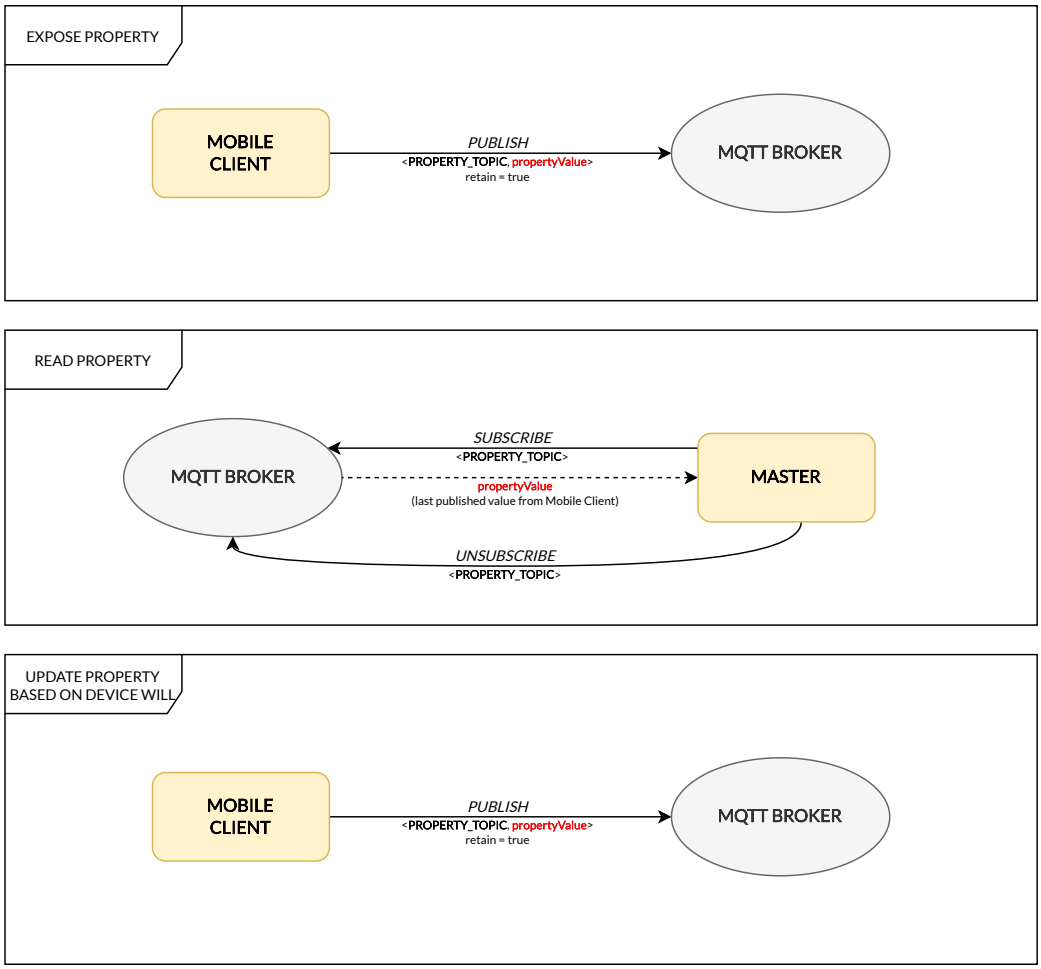

Figura 3.1: Protocol Binding per MQTT

La figura 3.1 mostra le operazioni eseguite dal Protocol Binding per l'interazione con una properità esposta prima che venissero apportate le modifiche. La figura 3.2 mostra invece come avviene l'esposizione di una Interaction Affordance e l'interazione con essa nel Protocol Binding modificato. Vengono usati due canali di comunicazione paralleli per ogni interazione esposta. Uno dedicato ai messaggi diretti alla Thing esposta (demonimato "ASK_TOPIC"), ed uno dedicato ai messaggi mandati in risposta dalla Thing esposta ai soggetti che interagiscono con essa (contrassegnato come "ANSWER_TOPIC").

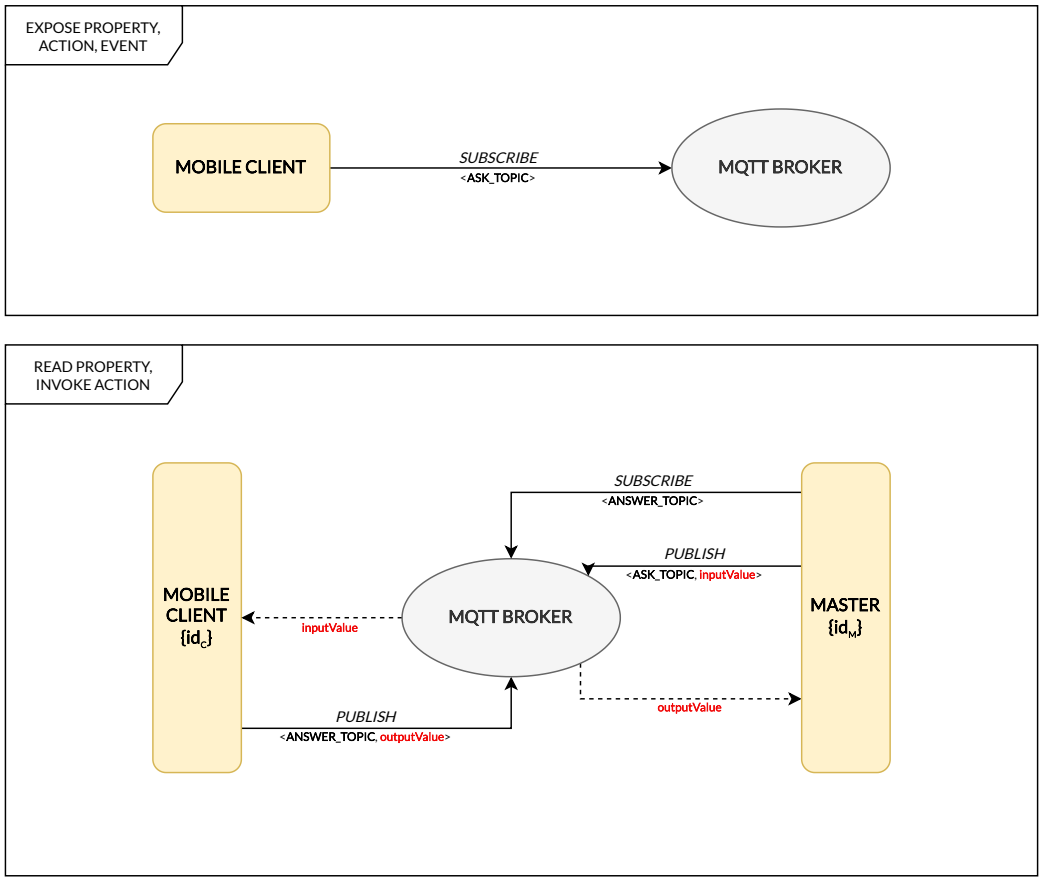

Figura 3.2: Protocol Binding per MQTT modificato

C = Servient che espone l'interazione

M = Servient che consuma C e utilizza l'interazione

i = indice dell'interazione i-esima di C

id_C = ID del Servient C

interaction_C^i = Interazione i disponibile per il Servient C

$$\text{ASK\_TOPIC}_C^i = ask(\text{id}_C, \text{interaction}_C^i) \tag{3.1}$$

$$\text{ANSWER\_TOPIC}_{C,M}^i = answer(\text{id}_C, \text{id}_M, \text{interaction}_C^i) \tag{3.2}$$

Al momento di esposizione di una Thing, ogni proprietà, azione ed evento messo a disposizione da essa origina una chiamata `subscribe` verso il broker al topic (3.1), generato a partire dall'ID della Thing e dall'ID dell'interazione, il quale verrà riportato anche nel TD che rappresenta la Thing appena esposta. Al momento dell'utilizzo di una interazione da parte di un Servient che avrà consumato la Thing, viene inviato un messaggio `subscribe` verso un topic (3.2) generato a partire dal topic (3.1) e dell'ID del Servient. Successivamente invia il messaggio verso il topic (3.1) aggiungendo nel contenuto del messaggio un riferimento al proprio ID. La Thing esposta riceve il messaggio, elabora la risposta ed una volta terminata l'elaborazione ne invia il risultato tramite un messaggio `publish` al topic (3.2). Il Servient riceve quindi il messaggio in risposta all'interazione e si disiscrive dal topic (3.2).

Per il suo utilizzo è necessario l'utilizzo di un broker, che ricopre il ruolo di mediatore tra i diversi client ed inoltra i messaggi. Viene utilizzato Eclispe Mosquitto, un broker open source che implementa il protocollo MQTT nelle versioni 5.0, 3.1.1 e 3.1. La scelta è ricaduta su questo broker data la sua semplicità di configurazione e la sua natura open source.

3.1.2 Protocol Binding per LA-MQTT

LA-MQTT viene utilizzato per la gestione delle campagne localizzate a livello di protocollo. Le librerie utilizzate non prevedevano nessun Protocol Binding per LA-MQTT, si è quindi proceduto alla sua implementazione.
Sono state utilizzate come base le classi del Protocol Binding per MQTT della libreria `node-wot`, in quanto condividono gran parte delle funzionalità. Le azioni messe a disposizione dal protocollo LA-MQTT sono mostrate in figura 3.3 e prendendo spunto dall'implementazione del Protocol Binding per MQTT sono state integrate nei flussi di esposizione e lettura/invocazione delle interazioni per rispecchiare il concetto "Voglio interagire con il dispositivo a patto che si trovi all'interno di una determinata area", oppure dall'altro verso "Voglio rendere i miei dati disponibili solo quando mi trovo all'interno di un'area specifica". La figura 3.4 ne mostra l'implementazione pratica.

API	Subject	MQTT OP	Topic	Payload
Position publish	MC	Publish	GPS_DATA	{position: P_i}
Topic subscription	MC	Subscribe	$C(i, t_s)$	*
	MC	Publish	MC_SUB	{ mc: i, topic: t_s }
Geofence publish	LDS	Publish	GEOFENCE_DATA	{topic: t_s, content: c_s, region: g_s, event: e_s}
Content publish	Backend	Publish	$C(i, t_s)$	{content: c_s}

Figura 3.3: Azioni di Publish-Subscribe di LA-MQTT

Le azioni trovano l'esecuzione in 3 momenti distinti della vita della Thing del Mobile Client. Il primo trova posto durante l'avvio del Servient, nel momento in cui vengono inizializzati i Protocol Bindings, e il Protocol Binding per LA-MQTT inizia ad inviare al broker aggiornamenti sulla posizione del dispositivo tramite l'operazione "Position publish".

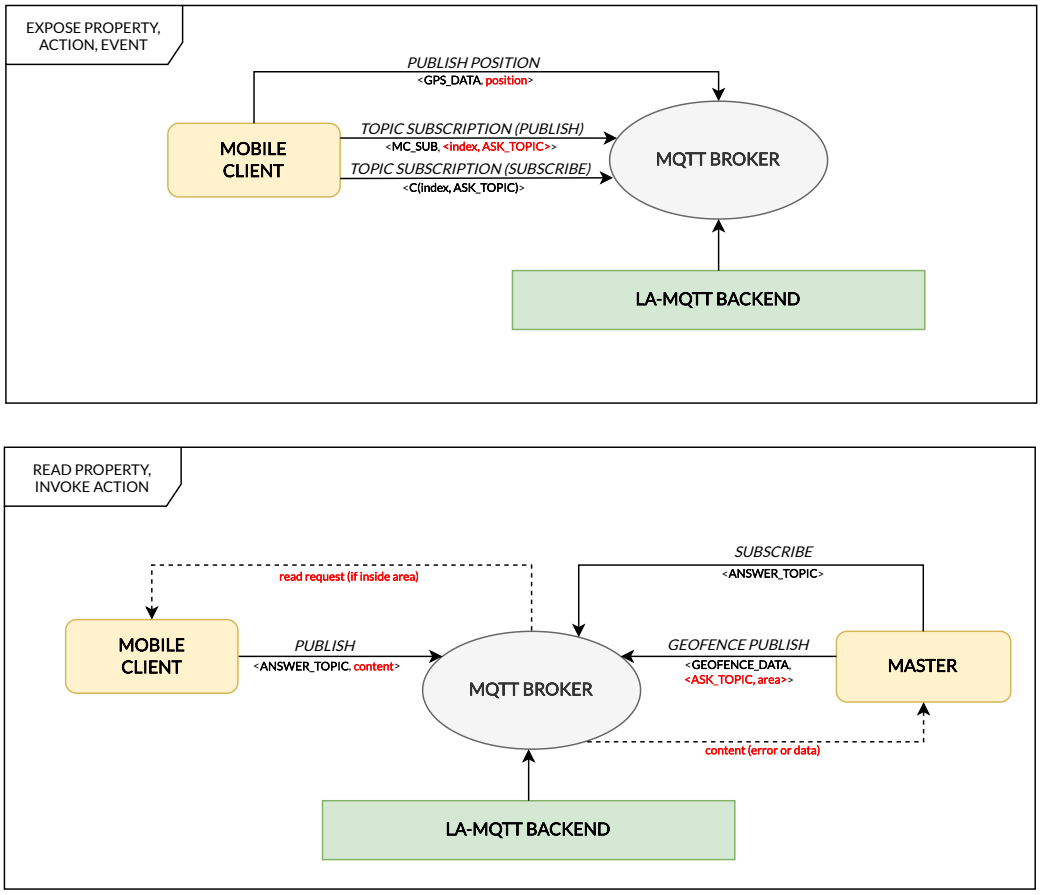

Figura 3.4: Scheda del Protocol Binding per LA-MQTT

Durante l'esposizione di una Interaction Affordance, il Mobile Client esegue l'operazione "Topic subscription" indicando come topic il suo "ASK_TOPIC" calcolato alla stessa maniera di (3.1).
Infine l'interazione con l'Interaction Affordance esposta inizia con la pubblicazione di un messaggio `subscribe` da parte del Master verso il topic di risposta ANSWER_TOPIC calcolato come in (3.2), per poi eseguire l'operazione "Geofence publish" verso il topic ASK_TOPIC indicando come regione l'area all'interno della quale si considera il dato rilevante. A questo punto il backend di LA-MQTT verifica la presenza o meno del Mobile Client all'interno dell'area ed in caso positivo inoltra la richiesta di lettura, per poi ricevere la risposta e inoltrarla al Master. In caso negativo invece, il Master si vedrà recapitare una risposta che indica la mancata presenza del target all'interno dell'area.

L'utilizzo del Protocol Binding per LA-MQTT necessita, in aggiunta al broker in maniera similare al Protocol Binding per MQTT, per il quale è stata utilizzata anche in questo caso l'implementazione di Eclipse Mosquitto, un'istanza della classe `SpatialMQTTBackEnd` attiva, in grado di ricevere, inviare e gestire i messaggi necessari al protocollo. L'istanza è attivata tramite il seguente frammento Typescript:

```
import { SpatialMQTTBackEnd } from "la-mqtt";

let backend = new SpatialMQTTBackEnd(
  "username",
  "password",
  "BROKER_URL",
  PORT
);

backend.start();
```

`username` e `password` sono campi nulli in quanto per i test i broker sono stati utilizzati con il parametro di configurazione `allow anonymus = true` che consentiva la connessione al broker bypassando l'autenticazione. `BROKER_URL`

è l'indirizzo del broker per MQTT al quale si appoggia LA-MQTT, ed è stato utilizzato lo stesso indirizzo del broker usato per il Protocol Binding per MQTT, che utilizza la porta 1883, default per il protocollo.

3.2 Mobile Client

Il Mobile Client non è altro che un'applicazione Android, che implementa le funzionalità di un Servient utilizzato sia come Client che come Server, racchiudendole all'interno di una dashboard che permette la gestione di campagne di MCS. L'applicazione consente all'utente di interagire con il Master, partecipare alle campagne e inviare i dati. Il codice sorgente dell'applicazione è open source e disponibile su GitHub[1].

3.2.1 Tecnologie utilizzate

Per la realizzazione dell'applicazione è stata ampiamente utilizzata la libreria `android-wot-servient` [3], che implementa i costrutti del Web of Things in Java, prendendo spunto dall'implementazione `SANE WoT Servient` [4], che a sua volta prende ispirazione dalla libreria `node-wot` [7].
`android-wot-servient` presenta diversi package, uno per ogni Protocol Binding implementato, più un package dedicato alle classi principali che implementano gli oggetti del WoT e che vengono utilizzate dall'applicazione. Ognuno dei package dei Protocol Binding contiene invece le classi che lo implementano, suddivise tra Server e Client da utilizzare a seconda del ruolo interpretato dal Servient mentre utilizza quello specifico protocollo. Dall'implementazione originale della libreria sono state apportate alcune modifiche per integrarla al meglio con le richieste dell'architettura da sviluppare.

- Prime tra tutte l'implementazione delle funzioni di registrazione e deregistrazione nella classe `Servient`, le quali prendono in input un oggetto di tipo `URI` rappresentante l'indirizzo della TDD presso cui effet-

[1] `https://github.com/Daveeeeed/crowdsensing-wot-android-client`

tuare l'azione e una `ExposedThing` oggetto dell'azione, ed effettuano una chiamata HTTP, rispettivamente PUT, passando un JSON che codifica la Thing, e DELETE, verso la TDD.

- La seconda modifica apportata alla libreria è stata la revisione del protocol binding MQTT come descritto nella sezione 3.1.1

- La terza e ultima variazione alla libreria è stata l'aggiunta del package contenente l'implementazione del Protocol Binding per LA-MQTT come anticipato nella sezione 3.1.2

Per la progettazione dell'interfaccia utente dell'applicazione è stata utilizzata la libreria **com.google.android.material:material**, che fornisce componenti di interfaccia utente predefiniti che rispettano le linee guida del Material Design di Google, garantendo un aspetto coerente e una miglior esperienza utente.

Per la gestione dei dati e delle risorse sono state utilizzate le librerie: **com.typesafe:config** per gestire le configurazioni del Servient nell'applicazione e **com.fasterxml.jackson.core:jackson-databind** per la gestione della serializzazione e de-serializzazione di oggetti in formato JSON, facilitando lo scambio di dati tra il Mobile Client e gli altri componenti dell'architettura.

3.2.2 Componenti e Funzionalità

L'applicazione è composta di un modulo ed una libreria. Il primo, spiegato più avanti nel dettaglio, comprende la logica dell'applicazione, i dettagli grafici e si occupa di gestire l'interazione tra l'utente e l'applicazione. La libreria invece implementa l'oggetto Servient secondo le specifiche del Web of Things. Il Servient è una classe con ruolo sia da Client che da Server e tramite esso avviene la comunicazione con altri dispositivi che implementano il framework WoT.

Il modulo, denominato `app`, comprende un package principale, il quale a sua volta include alcune classi Java di utilizzo comune e altri 3 sotto-package. Le classi contenute nel package principale sono:

- `ServientService`: Service che funge da wrapper per il servient e si occupa di ricevere le richieste di scambio dati dalle altre classi, per poi inoltrargli le risposte.
- `MainActivity`: entry point dell'applicazione, si occupa di gestire la schermata di splashscreen, implementata grazie all'utilizzo della libreria `core-splashscreen` di AndroidX.
- `MainViewModel`: view model contenente parametri globali utilizzati da tutti i componenti dell'applicazione, implementati mediante 3 LiveData:
  - un counter usato per tenere traccia delle operazioni asincrone che richiedono un'attesa all'interno dell'applicazione, se maggiore di zero mostra una schermata di attesa che impedisce l'esecuzione di ogni altra operazione.
  - una stringa rappresentante il token che l'utente ottiene dopo l'accesso, se diverso da `null` consente l'accesso alla dashboard, altrimenti rimanda al fragment per l'autenticazione.
  - un'eccezione, nulla di default, che quando assegnata mostra una notifica di errore a schermo per poi tornare ad essere nulla.
- `ForegroundServiceLauncher`: si occupa di avviare il ServientService in modalità foreground quando l'utente effettua l'accesso per consentire l'invio dei dati anche quando l'applicazione non è in primo piano.
- `CRWSNSApplication`: estende la classe Application e viene usata per l'inizializzazione del NetworkMonitoringUtil che necessita la sicurezza di essere inizializzato una e una sola volta durante il ciclo di vita dell'applicazione e prima che il codice della MainActivity sia eseguito.

3.2.3 ServientService

Il componente incaricato della gestione della Thing rappresentante il device è il `ServientService`. Questa classe segue il design pattern Singleton, e l'ottenimento della sua istanza avviene tramite il metodo `getInstance()` e la sua prima chiamata si ha all'avvio dell'applicazione e vengono eseguiti i seguenti passi:

- Viene controllata la presenza di una istanza già inizializzata, in caso positivo viene ritornata, altrimenti prosegue,
- Viene resettato il timer che si occupa di scandire il ping dello stato attivo del Mobile Client verso il Master,
- Viene avviato un Observer allo stato di connettività del dispositivo, che in caso di connessione non disponibile rimuove il Mobile Client dall'istanza WoT se precedentemente era stato esposto, interrompe l'azione di ping verso il Master e invalida l'istanza del Singleton, invece in caso di connessione disponibile richiede il TD del Master dalla TDD, il cui Url è noto, per consumarlo ed interagire con esso, e se il Mobile Client era precedentemente stato esposto, ne ripristina lo stato ri-esponendolo.

Il metodo `generateConfig()` si occupa di generare le configurazioni per il Servient, definendo i Protocol Binding per i quali deve essere abilitato lato Client e quelli per cui sarà abilitato lato Server. La configurazione è hardcodata mediante la creazione di un oggetto JSON che rispecchia a campi previsti dai diversi Protocol Bindings. Per l'implementazione realizzata viene generata la configurazione che presenta i Protocol Bindings per MQTT e LA-MQTT abilitati lato Server mentre CoAP e HPPT abilitati lato Client. Questo si traduce nella capacità del Servient di comunicare con Things consumate tramite i due protocolli abilitati la Client e di esporre Things raggiungibili mediante i due protocolli abilitati lato Server.
Le funzioni `register()`, `login()`, `fetchData()`, `fetchCampaigns()`, `applyToCampaign()`, `sendManualData()` e `leaveCampaign()` non sono altro

che chiamate che interagiscono con proprietà o azioni esposte dal Master.
Il metodo `esposeDevice()` invece genera il TD di default del Mobile Client, senza nessuna proprietà esposta, e ne registra la presenza presso il TDD, così da notificare il Master. Sempre in questo metodo viene avviato il timer che effettua il ping del device ad intervalli regolari verso il Master, che avviene tramite l'interazione con un'azione messa a disposizione di quest'ultimo e che prende come input l'ID del dispositivo. Un altro metodo incaricato della gestione del TD del Mobile Client è `exposeForCampaign()`, chiamato ogniqualvolta il dispositivo partecipa ad una nuova campagna o viene ripristinato lo stato del dispositivo in seguito ad una disconnessione o interruzione dell'applicazione, ed ha il compito di esporre una nuova proprietà per il Mobile Client, la quale rappresenta la campagna a cui si sta partecipando, e che ha un handler specifico in base alla tipologia di dato richiesto dalla campagna. Una volta aggiunta la nuova proprietà ad dispositivo esposto, viene aggiornato il TD presso la TDD tramite una chiamata PUT all'indirizzo del TD del dispositivo presso la Directory.

3.2.4 Auth

Il package `auth` comprende il fragment di autenticazione, implementato (come la maggior parte delle schermate dell'applicazione) mediante l'utilizzo del design pattern View-ViewModel.

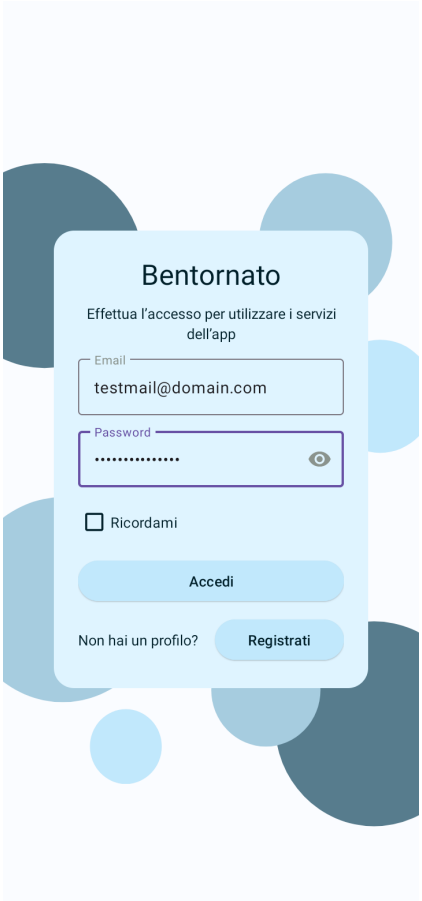

Figura 3.5: Schermata di accesso dell'applicazione

AuthFragment è la classe che implementa il fragment entry point dello schema di navigazione principale dell'app, il quale regola la navigazione tra dashboard e schermata di autenticazione. Ogniqualvolta l'applicazione viene avviata si è reindirizzati a questo fragment, e solamente quando l'utente sarà autenticato avverrà il redirect al fragment dashboard. La classe AuthView-

Model realizza alcune semplici funzioni per la verifica e aggiornamento dei TextInput di password e email quando i campi vengono compilati con errori.

Dopo aver compilato correttamente i campi, l'utente effettua l'autenticazione che è realizzata inviando una richiesta al server (tramite il framework WoT) mediante l'istanza static della classe ServientService e la chiamata di una sua funzione interna. Se la richiesta viene completata con successo, essa ritorna il token d'accesso, che viene salvato all'interno del MainViewModel.

Quando un utente si autentica spuntando la casella "Ricordami", i suoi dati d'accesso vengono salvati all'interno del preference file chiamato loginPreferences, in modo tale che ad ogni avvio successivo, quando viene verificato se l'utente ha effettuato l'accesso in passato spuntando l'opzione "Ricordami" controllando il preference file, la richiesta di autenticazione venga inviata in automatico così portando l'utente alla schermata della dashboard senza che siano richiesta azioni da parte sua.

3.2.5 Dashboard

Nel package `dashboard` sono contenute tutti i Fragment utilizzati nella dashboard, suddivisi in 3 sotto-package relativi alle 3 sezioni della dashboard. Questo package racchiude inoltre tre classi:

- `DashboardFragment`: fragment renderizzato all'interno del FragmentContainerView della MainActivity una volta che l'utente è autenticato. Implementa un layout che comprende una bottomNavigationBar della libreria MeterialUI, usata per muoversi all'interno delle 3 sezioni della dashboard, e una FragmentContainerView per renderizzare i layout delle sezioni.

- `DashboardViewModel`: probabilemente la classe più importante e condivisa tra le diverse View della dashboard, essa raccoglie tutti i dati usati per renderizzare il contenuto della dashboard, mediante l'utilizzo di diversi oggetti LiveData. Racchiude inoltre tutte le funzioni richiamate delle View della dashboard che invocano azioni da parte del Servien-

tService, e che spesso rappresentano azioni asincrone che richiedono di attendere per l'ottenimento dei dati, come la richiesta di informazioni su una campagna, la sottoscrizione ad essa o la richiesta di logout.

- `CampaignCompletedDialog`: dialog mostrato quando viene notificato il completamento di una campagna. Questo dialog è visualizzabile da ognuna delle sezioni della dashboard ed è per questo che si trova al livello root di questo pacchetto.

Parte fondamentale della dashboard è il grafo della navigazione tra schermate, contenuto all'interno della folder `res` sotto il pacchetto `navigation`. Esso permette le transizioni tra i vari fragment che compongono la dashboard e regola la visualizzazione dei dialog.

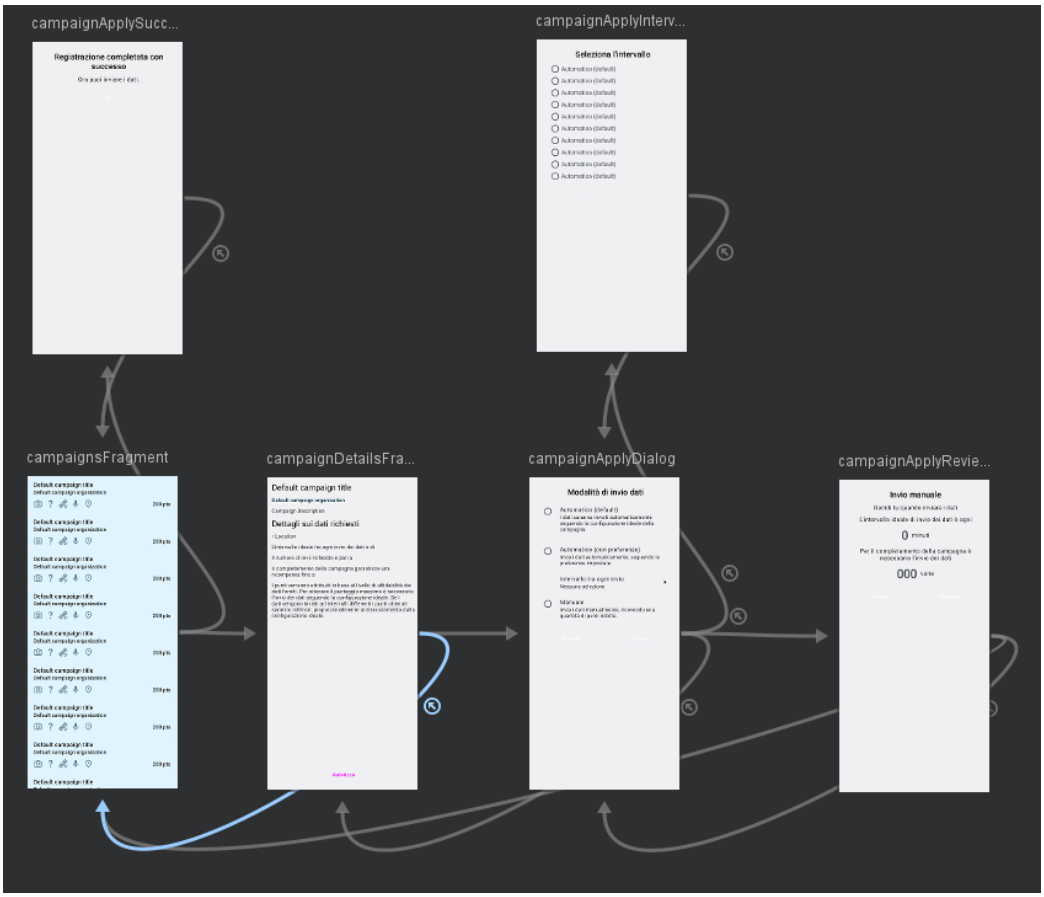

Figura 3.6: Grafo di navigazione della tab "Campagne"

Per ogni navigation action è stato fatto un uso intensivo della proprietà "popUpTo" e "popUpToInclusive" per visualizzare correttamente i dialog, i quali, se viene effettuata la navigazione tramite action da uno all'altro, rimangono sullo schermo, mentre grazie a questa proprietà è facile gestire la visualizzazione di un solo dialog alla volta rispettando anche l'ordine all'interno dello stack di navigazione.

Il grafo di navigazione è formato da 3 grafi indipendenti, uno per ognuna delle tab della sezione dashboard.

L'entry point della sezione "Campagne" della dashboard è la classe CampaignsFragment, che include una RecyclerView utilizzata per mostrare la lista delle campagne disponibili, ottenuta osservando la proprietà dedicata nella classe DashboardViewModel. Le classe CampaignsAdapter realizza l'adapter per l'appena citata RecyclerView, mostrando per ogni campagna il titolo, l'organizzazione per cui viene realizzata, il massimo numero di punti ottenibili per il completamento e una serie di icone indicanti la tipologia di dati richiesti per la campagna. Le tipologie attualmente categorizzate sono:

- `location`: ovvero i dati sulla posizione del dispositivo.
- `gps`: qualsiasi altro dato che richieda l'accesso alle componenti GPS del device, eccetto la posizione.
- `camera`: per le campagne che richiedono l'invio dei dati sotto forma di fotografia.
- `mic`: indica che la campagna richiede l'accesso al microfono del dispositivo.
- `other`: per ogni altro tipo di sensore non precedentemente indicato (non implementato)

La classe CampaignDetailsFragment si occupa di renderizzare il fragment contenente i dettagli della campagna, ovvero informazioni come la descrizione, la frequenza di invio dati consigliata e il numero di invii richiesti per il

completamento. Inoltre nella parte inferiore presenta un pulsante che consente all'utente di partecipare alla campagna qualora il dispositivo rilevi che sono stati concessi i permessi necessari per raccogliere con successo i dati della campagna. Nel caso in cui non siano stati concessi dei permessi necessari per accedere ai dati che richiede la campagna verrà mostrato un apposito pulsante che se toccato mostrerà il dialog di sistema per la richiesta delle autorizzazioni necessarie alla campagna. Solamente quando l'applicazione ottiene i permessi è possibile partecipare alla campagna. Nell'eventualità che il device non possegga sensori o funzionalità richieste per l'invio dei dati di una campagna allora il pulsante in questa schermata sarà disattivato e notificherà all'utente che il dispositivo non è idoneo per la partecipazione. Al momento queste specifiche sono state implementate per le campagne di tipo `location`.

La partecipazione ad una campagna avviene tramite una serie di dialog, che guidano l'utente durante la configurazione della modalità di invio dati preferita. Il primo dialog mostrato all'utente è quello rappresentato dalla classe JoinCampaignDialog, che mostra un elenco di radio button rappresentanti le tre tipologie di invio dati per una campagna:

- Modalità automatica: se viene scelta questa modalità il dispositivo non dovrà occuparsi dell'invio dei dati, bensì li renderà disponibili in rete, mediante il framework WoT, e verranno letti dal server quando lo necessita. Questa modalità consente all'utente di non dover inviare i dati manualmente e attenersi alla specifica richiesta della campagna, per ottenere il massimo del punteggio.

- Modalità automatica (con preferenze): con questa modalità l'utente potrà scegliere, mediante un dialog rappresentato dalla classe JoinCampaignIntervalDialog, un intervallo di tempo per l'invio dei dati, così da non doversi attenere alla richiesta della campagna. A differenza della modalità automatica, con questa modalità è il device che invia ad intervalli regolari, scanditi dall'intervallo di tempo scelto, i dati al server.

- Modalità manuale: nel caso in cui l'utente voglia avere il maggior controllo possibile sull'invio dei dati può scegliere questa modalità, che consente di inviare i dati solamente su input dell'utente tramite la schermata dei dettagli della campagna all'interno della tab "Le mie campagne"

Una campagna può non avere disponibili tutte e 3 le modalità di invio dei dati, infatti al momento di scelta della modalità nel dialog alcune di queste scelte possono essere disabilitate, a seconda di come la campagna è stata configurata dal creatore. L'utente può scegliere solo tra le opzioni abilitate per poi confermare la scelta e passare al dialog JoinCampaignReviewDialog in cui viene mostrato un riepilogo delle scelte effettuate e da le indicazioni all'utente su come verranno inviati i dati. Parte fondamentale della comunicazione tra i vari dialog è l'utilizzo della classe DashboardViewModel per mantenere la continuità dei dati, infatti ad ogni interazione i dati del ViewModel sono aggiornati e le schermate ne rappresentano lo stato corrente.

Nel dialog mostrato dalla classe JoinCampaignReviewDialog, al momento della pressione sul pulsante "Conferma" viene inoltrata alla classe DashboardViewModel la richiesta di partecipazione, la quale trasmette i dati che contiene al service ServientService che inoltre la richiesta al server. In caso di esito positivo:

- il campo applicationResult del ViewModel viene posto a true
- la classe CampaignFragment che osserva questo campo mostra a schermo un dialog che notifica l'utente dell'avvenuta partecipazione.
- la campagna viene salvata nel preference file con nome il token d'accesso dell'utente per poter essere mantenuta la continuità tra le campagne partecipate tra una sessione e l'altra. Questo perché sul server non vengono salvate le partecipazioni alle campagne da parte degli utenti ma soltanto le campagne attualmente completate dell'utente, e una partecipazione sarà rifiutata soltanto se l'utente ha già completato su un qualsiasi dispositivo quella campagna. Ciò rende possibile per un utente

partecipare ad una campagna non ancora completata su un dispositivo su di un nuovo dispositivo e riceverà i punti per essa soltanto dalla prima campagna completata su un qualsiasi dispositivo.

- in caso di campagna partecipata mediante la modalità automatica con preferenze sarà attivata la routine per l'invio dei dati

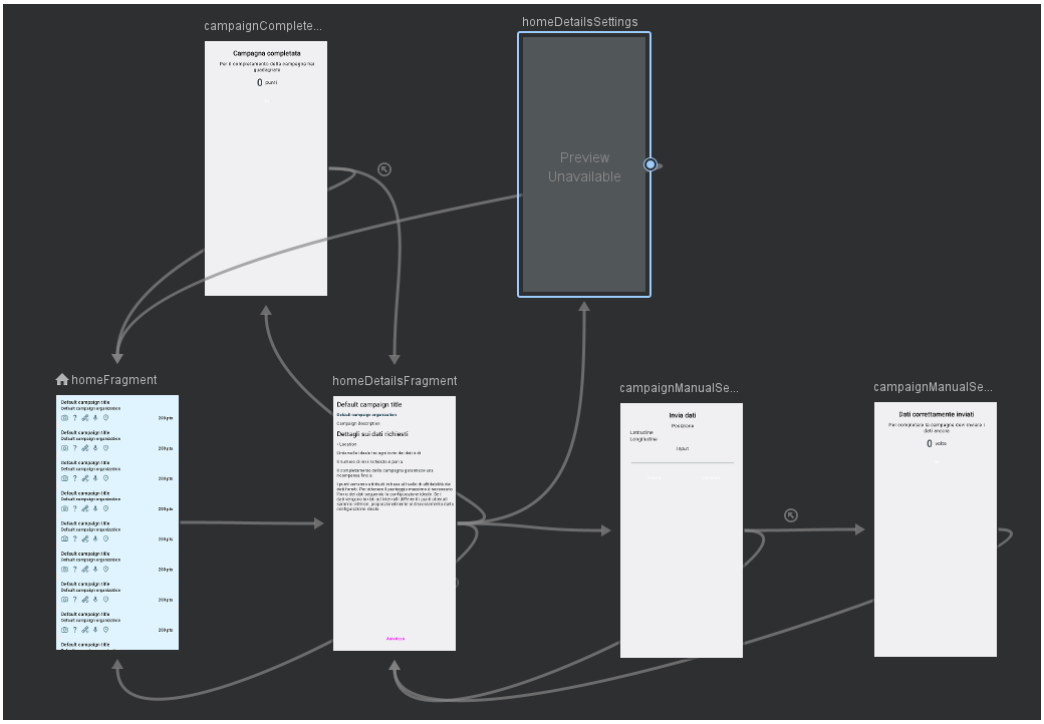

Figura 3.7: Grafo di navigazione della tab "Le mie campagne"

La sezione "Le mie campagne" pone il suo entry point nella classe HomeFragment che in modo similare alla sezione "Campagne" comprende una RecyclerView che mostra le campagne a cui l'utente sta partecipando. Una volta selezionata una campagna, mediante il componente di navigazione si è spostati all'interno della schermata di dettagli della campagna, che presenta un pulsante nella parte inferiore che varia la sua funzione a seconda della tipologia di modalità di invio dati scelta per la campagna.

Nel caso di campagne con invio dati automatico il pulsante consentirà di interrompere momentaneamente o riprendere l'invio dei dati, mentre per le campagne con invio dati manuale il pulsante consentirà di inviare i dati richiesti dalla campagna.

Una volta toccato verrà mostrato un dialog, rappresentato dalla classe CampaignManualSendDialog che riassume i dati richiesti, o ne richiede la compilazione se i dati non possono essere ottenuti automaticamente dal dispositivo (ad esempio se la campagna richiede l'invio di una foto) e un pulsante di conferma da toccare con completare l'invio dei dati. Se i dati vengono inviati correttamente il dialog CampaignManualSendSuccessDialog verrà mostrato per notificare l'invio effettuato correttamente.

La classe HomeDetails, che mostra i dettagli della campagna, aggiunge alla topBar un menu che rimanda alle impostazioni di essa dalle quali è possibile abbandonare la campagna.

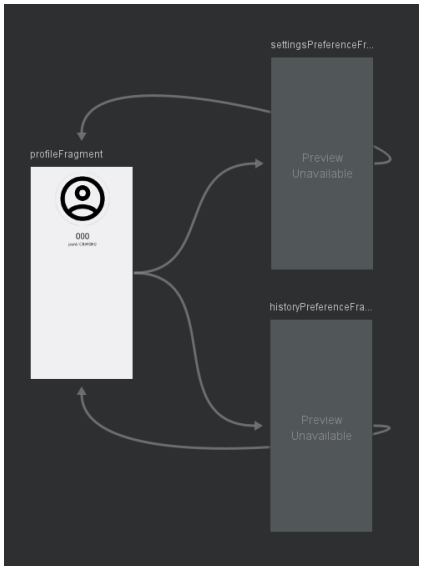

Figura 3.8: Grafo di navigazione della tab "Profilo"

3.2.6 Utilità

L'ultimo package contenuto nel package `dashboard` è il package `utils`, che racchiude alcune classi usate principalmente a supporto delle altre classi presenti nell'applicazione.

- `User`: implementa l'oggetto User, che racchiude i dati dell'utente raffigurati nella sezione Profilo della Dashboard.
- `DefaultItemDecorator`: classe che realizza un decoratore per gli item di una RecyclerView, usato per renderizzare le campagne della dashboard con dei piccoli spazi tra l'una e l'altra.
- `NetworkStateManager` e `NetworkMonitoringUtil`: classi che implementano una routine che osserva lo stato di connettività del device, ed espongono un oggetto LiveData che ne rappresenta lo stato e muta il suo valore quando la connettività del device cambia. Osservano i cambiamenti soltanto a livello di connettività CELLULAR e WIFI.
- package `campaign`: raccolta di classi usate per definire l'oggetto Campagna. Comprende una Enumeration delle diverse modalità di partecipazione alle campagne, una classe rappresentante una Campagna, ed uno rappresentante una Campagna a cui l'utente partecipa, che estende la prima ed implementa due proprietà che rappresentano l'intervallo di invio dei dati scelto, e la modalità di partecipazione selezionata. Sono inoltre presenti due package che racchiudono le diverse implementazioni delle aree di una campagna (`circle`, `rect` e `poly`) e i diversi tipi di dato che possono essere mandati (al momento implementato solo il tipo di dato Location).

3.2.7 Scelte progettuali

L'applicazione è stata sviluppata utilizzando intensivamente il design pattern View-ViewModel per la rappresentazione dei dati, in quanto consente una facile condivisone dei dati tra le View, separando UI layer e data layer

mantenendo una chiara separazione tra i vari ruoli delle classi. I ViewModel sono i principali incaricati all'aggiornamento e modifica dei dati.
Un'altra direzione presa durante lo sviluppo del lato grafico è stata quella di utilizzare la libreria di Material Design per la realizzazione dei layout, in quanto dona all'applicazione un buon colpo d'occhio e mantiene una certa linearità nei colori di tutti i componenti.
Per accedere ai componenti dei layout dalle classi è stata attivata la funzionalità di ViewBinding, all'interno del file build.gradle, che permette di accedere più facilmente agli oggetti grafici all'interno delle classi senza necessità di cast espliciti.
I dati persistenti nell'App vengono salvati tramite l'utilizzo delle SharedPreferences, una modalità leggera e semplice per gestire la memorizzazione e lettura di piccole quantità di dati. Gli unici valore salvati riguardano infatti le preferenze impostate dall'utente riguardo i sensori abilitati e le preferenze di invio, e le campagne attualmente in corso da parte dell'utente.

L'utilizzo dell'applicazione può mutare a seconda della tipologia delle campagne a cui si sta partecipando, per questo è stata ampiamente favorito un approccio modulare, che consente facilmente l'adattamento del sistema a nuove tipologie di dati da essere trasmessi, nuove proprietà delle campagne e nuove modalità di invio dei dati. Ipotizzando infatti di voler implementare una nuova tipologia di dato richiesto dalle campagne, le azioni necessarie lato applicativo Android sono:

- la creazione di un nuovo elemento nell'Enumeration `SubmissionType`,
- la realizzazioni di una nuova classe che estenda `DataSubmission` con il capo Type valorizzato con l'elemento appena aggiunto,
- l'eventuale adeguamento dei layout grafici

3.3 Master

Il Master ha la funzione di esporre la Thing con cui interagiscono i Mobile Client e gestire l'invio dei dati da e per gli utenti, la loro validazione e salvataggio. Il codice sorgente è open source e disponibile su GitHub[2].

3.3.1 Tecnologie

E' un'applicazione Node scritta in Typescript e per la sua realizzazione sono stati utilizzati i pacchetti:

- `@node-wot/core`: implementazione di ThingWeb del framework Web of Things per applicazioni Node, viene utilizzata la classe `Servient` come entry point per tutte le interazioni WoT.
- `@node-wot/binding-http`: binding per HTTP usato per esporre le azioni e le proprietà del Server, attraverso le quali i device interagiranno con esso.
- `@node-wot/binding-coap`: viene usato per le stesse funzioni del binding per HTTP, implementate via CoAP.

In aggiunta sono presenti due pacchetti nella cartella `lib`:

- `binding-mqtt`: è il binding per MQTT realizzato da ThingWeb al quale sono state apportate delle modifiche alla classe `MqttClient` come descritto in 3.1.1. Viene usato per interagire con i device e leggere i dati che espongono per le campagne.
- `binding-lamqtt`: binding per LA-MQTT realizzato seguendo la struttura del binding per MQTT utilizzando a supporto il pacchetto che implementa il protocollo LA-MQTT. Viene utilizzato in modo analogo al binding per MQTT, per leggere i dati esposti dai device qualora si trovino all'interno dell'area rilevante per la campagna.

[2] `https://github.com/Daveeeeed/crowdsensing-wot-proxy`

3.3.2 Componenti e funzionalità

Il Master si compone di due file soltanto: `utils.ts` e `index.ts`. Il primo contiene le dichiarazioni di alcuni tipi utilizzati per interagire con il database quali `Submission`, `User` e `Campaign`; e due classi, TimerTimeout e TimerInterval, che implementano la possibilità di verificare l'andamento, interrompere o riprendere istanze di un Timeout e di un Interval, utilizzate per la gestione dei dispositivi e delle campagne a cui partecipano.
Definisce inoltre il tipo DeviceMap, che segue la struttura

```
DeviceMap = {
  device: ConsumedThing;
  pinger: TimerTimeout;
  subscriptions: Map<string, TimerInterval | null>;
};
```

e rappresenta l'istanza di un Mobile Client connesso, con il riferimento alla Thing che espone, un Timeout che viene resettato periodicamente dal Mobile Client interagendo con il Master e che rappresenta il suo stato attivo, e una mappa delle campagne a cui sta partecipando, dove ha per ognuna l'ID di riferimento e un Interval che scandisce la lettura dei dati se la campagna è stata partecipata in modalità Push.
In aggiunta a quanto detto contiene alcune classi d'appoggio per le interazioni con il database, `getDatabaseClient` e `getItemWithId`, ed infine una costante valorizzata al TD del Master, che ne descrive le azioni, proprietà con i relativi valori di input e di ritorno.

```
const dashboardThingDescription: ExposedThingInit = {
  id: "crwsns:dashboard",
  title: "Dashboard",
  properties: {
    campaigns: {
      type: "array",
      observable: false,
      readOnly: true,
    },
  },
```

```
  actions: {
    registerNewUser: {
      input: {
        type: "object",
        properties: {
          email: { type: "string" },
          password: { type: "string" },
        },
      },
      output: {
        type: "object",
        properties: {
          authenticated: { type: "boolean" },
          bearer: { type: "string" },
          error: { type: "string" },
        },
      },
    },
    loginUser: {
      input: {
        type: "object",
        properties: {
          email: { type: "string" },
          password: { type: "string" },
        },
      },
      output: {
        type: "object",
        properties: {
          authenticated: { type: "boolean" },
          bearer: { type: "string" },
          error: { type: "string" },
        },
      },
    },
    retrieveUserData: {
      input: {
        type: "object",
```

```
49          properties: {
50            bearer: { type: "string" }, //In this stage is used
    the User Object ID
51          },
52        },
53        output: {
54          type: "object",
55          properties: {
56            email: { type: "string" },
57            points: { type: "number" },
58            campaigns: { type: "array" },
59          },
60        },
61      },
62      applyToCampaignPush: {
63        input: {
64          type: "object",
65          properties: {
66            bearer: { type: "string" },
67            campaignId: { type: "string" },
68            deviceId: { type: "string" },
69          },
70        },
71        output: {
72          type: "object",
73          properties: {
74            ok: { type: "boolean" },
75            error: { type: "string" },
76          },
77        },
78      },
79      applyToCampaignPull: {
80        input: {
81          type: "object",
82          properties: {
83            bearer: { type: "string" },
84            campaignId: { type: "string" },
85            deviceId: { type: "string" },
```

```
      },
    },
    output: {
      type: "object",
      properties: {
        ok: { type: "boolean" },
        error: { type: "string" },
      },
    },
  },
  sendPullCampaignData: {
    input: {
      type: "object",
      properties: {
        bearer: { type: "string" },
        campaignId: { type: "string" },
        deviceId: { type: "string" },
        data: { type: "object" },
      },
    },
    output: {
      type: "object",
      properties: {
        ok: { type: "boolean" },
        remaining: { type: "integer" },
        error: { type: "string" },
      },
    },
  },
  ping: {
    input: {
      type: "object",
      properties: {
        deviceId: { type: "string" },
      },
    },
  },
  leaveCampaign: {
```

```
124       input: {
125         type: "object",
126         properties: {
127           bearer: { type: "string" },
128           campaignId: { type: "string" },
129           deviceId: { type: "string" },
130         },
131       },
132       output: {
133         type: "object",
134         properties: {
135           ok: { type: "boolean" },
136           error: { type: "string" },
137         },
138       },
139     },
140   },
141 };
```

Listing 3.1: Struttura della Thing Description del Master

Il file `index.ts` contiene la logica del Master, che include le seguenti chiamate di inizializzazione:

- Il collegamento con il database.
- La produzione della Thing a partire dal TD e l'impostazione degli handler per tutte le Interaction Affordances descritte.
- L'esposizione della Thing creata. La Thing viene resa disponibile ai Mobile Client per interagire con essa.
- La registrazione presso la TDD. La Thing viene registrata presso la Thing Description Directory in modo che possa essere trovata e acceduta da altri dispositivi.
- La sottoscrizione agli eventi della TDD. Viene stabilita una connessione per ricevere gli eventi emessi dalla TDD. Tre metodi sono responsabili di gestire gli eventi `thing_created`, `thing_updated` e `thing_deleted`.

- Il metodo per l'evento `thing_created` riceve l'ID della Thing che ha generato l'evento. In caso di creazione di una nuova Thing, il metodo richiede il TD corrispondente per consumarlo e salvare la nuova Thing nella DeviceMap dedicata a essa. Inoltre, vengono ripristinate eventuali routine di gestione delle campagne associate alla Thing.
- Il metodo per l'evento `thing_updated` riceve l'ID della Thing che ha generato l'evento. In caso di aggiornamento di una Thing esistente, il metodo richiede il nuovo TD corrispondente per consumarlo e aggiornare la Thing corrispondente nella DeviceMap. Anche in questo caso vengono ripristinate eventuali routine di gestione delle campagne associate alla Thing.
- Il metodo per l'evento `thing_deleted` riceve l'ID della Thing che ha generato l'evento. In caso di eliminazione di una Thing, il metodo rimuove l'elemento corrispondente dalle DeviceMap, interrompendo tutti i processi in corso legati ad essa.

Gli handler sono responsabili di restituire i dati o eseguire le azioni associate alle Interaction Affordance definite nel TD (Thing Description). Uno degli handler più interessanti è quello relativo al metodo `applyToCampaignPush()`, che attiva una routine di lettura della proprietà esposta ad intervalli regolari. Questo viene realizzato tramite un TimerInterval, che è stato precedentemente salvato nella DeviceMap e associato alla campagna specifica. La routine di lettura viene eseguita automaticamente a intervalli specificati dal TimerInterval.

3.4 Database

Il sistema deve gestire i dati di accesso degli utenti, le informazioni sulle campagne e i dati inviati dagli utenti stessi. Per soddisfare questi requisiti è stato utilizzato MongoDB come DBMS, un database non relaziona-

le, orientato ai documenti, che organizza i dati in collezioni di documenti JSON-like.

L'implementazione del database con MongoDB garantisce una gestione efficiente e affidabile delle informazioni, consentendo di gestire una vasta gamma di dati adattandosi facilmente ad eventuali future modifiche.

L'implementazione prevede un unico database con tre collezioni principali, qui descritte assieme agli oggetti che contengono.

3.4.1 Campaigns

L'oggetto Campaign rappresenta una campagna a cui gli utenti possono partecipare ed ogni documento rispetta la forma mostrata in 3.2.

Il campo `ideal_submission_interval` rappresenta l'intervallo ideale di invio dei dati, scelto dall'organizzatore della campagna ed usato a fini valutativi degli invii dell'utente. Più la frequenza di invio si avvicina a questo valore, più alto sarà il punteggio attribuito all'utente al termine della campagna.

`points` indica il valore massimo di punti assegnabili ad un utente che completa la campagna, il quale subirà delle modifiche in base alla frequenza di invio e la qualità dei dati.

I tre booleani determinano quale tipologia di invio dei dati potrà essere utilizzata, scegliendo tra la modalità PUSH (il Master richiede i dati), PULL-AUTO (il Mobile Client invia i dati ad intervalli regolari) e PULL-INPUT (il Mobile Client invia i dati quando lo decide l'utente). Almeno una delle tre dovrà essere attiva.

`submission_required` rappresenta il numero di invii di dati dallo stesso utente necessari a completare la campagna ed ottenere i punti.

Il campo `submission_type` specifica la tipologia della campagna, determina la struttura e il tipo dei dati inviati. I valori che può assumere fanno riferimento a quelli presenti sono `location`, `camera`, `mic`, `gps` e `other` ed ognuno di essi per essere funzionante necessita di un handler specifico sul Master e sul Mobile Client, al momento l'unico implementato è quello per la tipologia `location`.

I campi contenuti all'interno di `area` sono riempiti solamente se la campagna è localizzata e specificano l'area di validità della campagna. Il campo `type` specifica la forma dell'area e può assumere i valori `CircleArea`, `RectArea` e `PolyArea`. Al momento l'implementazione è funzionante solamente per aree circolari.

```
Campaign= {
    _id: ObjectId,
    title: String,
    description: String,
    organization: String,
    ideal_submission_interval: Integer,
    points: Integer,
    pull_enabled: Boolean,
    push_auto_enabled: Boolean,
    push_input_enabled: Boolean,
    submission_required: Integer,
    submission_type: String,
    area: RectArea | CircleArea | null,
};

CircleArea = {
    type: String,
    center_lat: Double,
    center_lng: Double,
    radius: Double,
};

RectArea = {
    type: String,
    origin_x: Double,
    origin_y: Double,
    height: Double,
    lenght: Double,
};

PolyArea = {
    type: String,
```

```
    x: Double[],
    y: Double[],
};
```

Listing 3.2: Struttura di un oggetto della collezione campaign

3.4.2 Submissions

La collezione `submission` contiene tutti i dati inviati dagli utenti. Ogni documento rappresenta un singolo invio e segue la struttura:

```
Submission = {
    _id: ObjectId,
    bearer: String,
    campaign: ObjectId,
    device: String,
    data: Object,
    time: Date,
};

Location = {
    latitude: Double,
    longitude: Double
}
```

Listing 3.3: Struttura di un oggetto della collezione submission

I campi `bearer` e `device` contengono due stringhe identificative rispettivamente dell'utente e del dispositivo utilizzato.

`campaign` è invece un riferimento all'oggetto della collezione `Campaigns` di cui l'oggetto corrente è un invio.

Il campo destinato a contenere i dati inviati è `data` la cui struttura varia in base al tipo di dato fornito (il valore assunto nelle campagne che richiedono l'invio della posizione sarà del tipo `Location`).

3.4.3 Users

Contiene i dati relativi a ogni singolo utente rappresentati da documenti con schema:

```
User = {
    _id: ObjectId,
    email: String,
    salt: String,
    hash: String,
    points: Integer,
    completed_campaigns: Array<ObjectId>;
};
```

Listing 3.4: Struttura di un oggetto della collezione users

Il campo `completed_campaigns` racchiude tutti gli ID della campagne che l'utente ha già completato, mentre i campi `salt` e `hash` sono dedicati all'autenticazione, che avviene tramite il seguente frammento di codice nel Master.

```
AccessRequest = {
  email: string;
  password: string;
};

let db: Db;
let data: AccessRequest;

try {
    let users = db.collection("users");
    let user = await users.findOne({ email: data.email });
    if (user) {
        let hash = crypto.pbkdf2Sync(
            data.password,
            user.salt,
            1000,
            64,
            `sha512`).toString(`hex`);
        if (user.hash != hash) throw "Incorrect password";
```

```
20          else
21              return {
22                  authenticated: true,
23                  bearer: String(user._id),
24              };
25      } else throw "Incorrect email";
26  } catch (error) {
27      console.log(error);
28      return {
29          authenticated: false,
30          error: error,
31      };
32  }
```

3.5 Thing Description Directory

L'ultimo componente è la Thing Description Directory che contiene le Thing Description dei Mobile Client e del Master. È stata utilizzata l'implementazione in Go di TinyIoT[8], realizzata seguendo le specifiche del W3C. I Mobile Client e il Master interagiscono con essa mediante HTTP per le operazioni di registrazione, lettura e sottoscrizione ad aggiornamenti.

Capitolo 4

Risultati

In questo capitolo viene mostrato il funzionamento dell'applicazione Android per dimostrare il corretto funzionamento del sistema e la capacità di quest'ultimo di consentire la raccolta dati in modalità Push e Pull.

4.1 Demo dell'applicazione

L'applicazione realizzata implementa una dashboard con cui l'utilizzatore può partecipare a campagne di raccolta dati. Ogni utente partecipando ad una campagna mette i propri dati, o quelli che andrà a fornire, a disposizione dell'organizzatore della campagna.

L'utente per essere incentivato a fornire dati continuativi e qualitativamente accettabili verrà ricompensato con dei punti una volta portata a termine la campagna. In un'ipotetica implementazione reale i punti potranno successivamente essere scambiati con ricompense di vario genere.

Il cuore dell'applicazione è rappresentato dalla dashboard, la schermata principale dalla quale è possibile effettuare tutte le azioni di rilievo, come la sottoscrizione ad una campagna, l'invio dei dati e visualizzare lo stato del proprio profilo utente. Per ottenere l'accesso alla dashboard l'utente deve prima autenticarsi, procedura che avviene mediante una schermata di accesso e/o registrazione, come mostrato in 4.1 e 4.2.

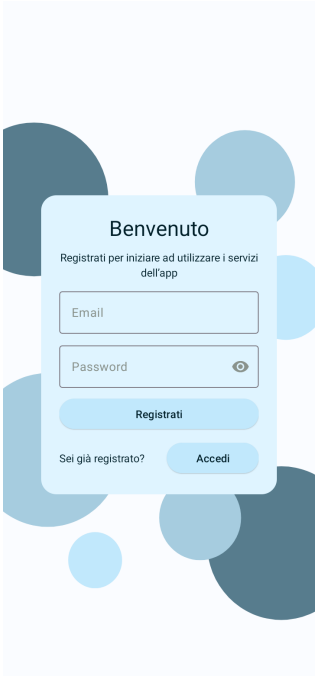

Figura 4.1: Schermata di registrazione

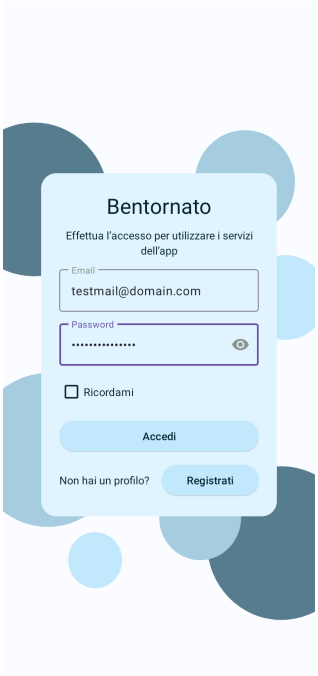

Figura 4.2: Schermata di login

Al primo avvio l'utente dovrà quindi autenticarsi compilando i campi di testo, una volta fatto ciò i dati dell'utente vendono salvati per permettergli di accedere più rapidamente in futuro.

Dopo aver effettuato l'accesso l'utente ottiene l'accesso alla dashboard, la quale è suddivisa in 3 sezioni.

La prima di queste, denominata "Campagne" raccoglie le campagne disponibili alla partecipazione, mediante un elenco che mostra per ognuna alcuni dati riassuntivi. Queste campagne sono ottenute in risposta ad una interazione con il Master. Da questa schermata l'utente può selezionare una campagna toccandola, per visualizzarne i dettagli e per interagirvi.

La seconda sezione raccoglie le campagne alla quali l'utente sta partecipando mostrando i dati in maniera analoga alla sezione precedente. Una volta toccata una campagna l'utente avrò la possibilità di interagire con essa per inviare i dati che richiede, abbandonarla o cambiare le modalità di invio

dei dati.

La terza e ultima sezione invece mostra i dettagli dell'utente quali la mail usata per l'accesso, il numero di punti accumulati dalla partecipazione alla campagne, lo storico delle campagne completate, le impostazioni per modificare le preferenze dell'utente e un pulsante per effettuare il logout.

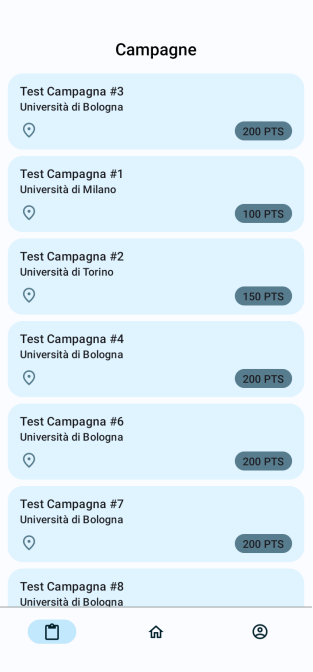

Figura 4.3: Sezione "Campagne"

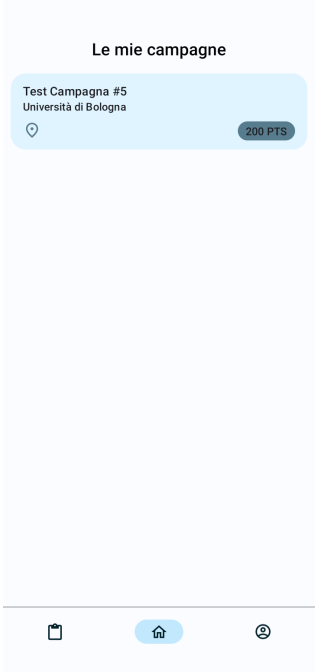

Figura 4.4: Sezione "Le mie campagne"

In seguito all'autenticazione l'utente è notificato dell'attività dell'App tramite una notifica persistente (4.5) che rappresenta l'esecuzione del ForegroundService incaricato di gestire le interazioni del Servient per l'invio dei dati. L'attività del Service sarà continuata fino al logout dell'utente.

La partecipazione ad una campagna ha inizio dalla selezione di essa dalla tab dedicata. Una volta all'interno della schermata di dettaglio della campagna vengono mostrate le informazioni della campagna e un avviso che informa gli utenti delle modalità di retribuzione dei punti (4.6).

Il bottone posto nella parte inferiore della schermata si presenterà di colore rosso qualora l'applicazione rilevi di non avere tutti i permessi necessari all'esecuzione delle campagna e la sua pressione richiederà i permessi all'utente. Sarà invece disabilitato nel caso in cui l'applicazione rilevi che il dispositivo non soddisfa tutte le capacità sensoriali richieste dalla campagna. In seguito alla verifica dei permessi, in caso di esito positivo l'utente potrà avviare il processo di partecipazione alla campagna tramite la compilazione di una serie di dialog che gli pongono davanti alcune scelte relative alla modalità di partecipazione.

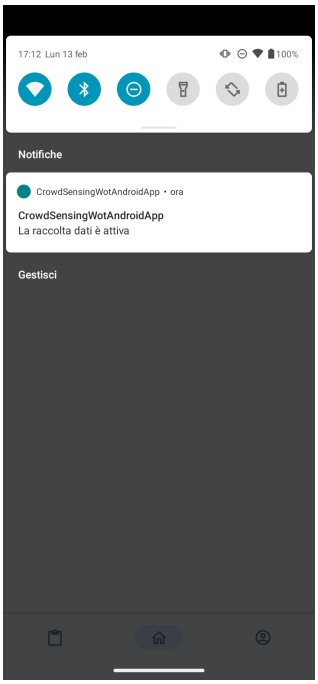

Figura 4.5: Notifica del foreground service attivo

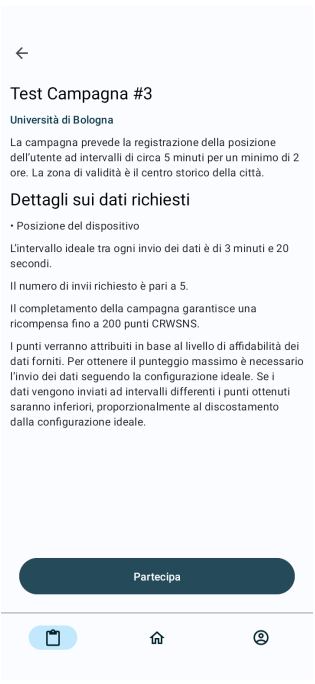

Figura 4.6: Schermata dei dettagli di una campagna

In caso di partecipazione alla campagna in modalità "automatica", ovvero utilizzando la strategia Push, in seguito alla conferma di avvenuta partecipazione sarà attivato nel Master il Timer relativo al polling dei dati dal Mobile Client. Allo scadere del Timer il Master effettua la richiesta dei dati e il

Mobile Client ne registra l'avvenuta lettura, consultabile dalle impostazioni della campagna.

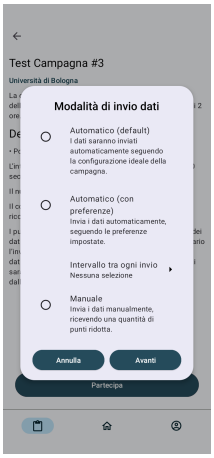

Figura 4.7: Scelta della modalità di invio dei dati

Figura 4.8: Conferma partecipazione invio auto

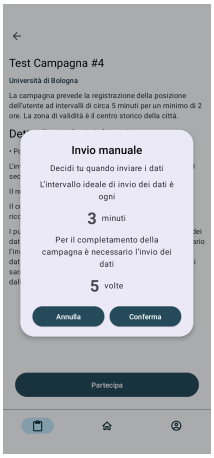

Figura 4.9: Conferma partecipazione invio manuale

Figura 4.10: Conferma partecipazione conclusa

Nella tab "Le mie campagne" in maniera del tutto similare a quanto visto nella tab "Campagne", toccando una campagna si potrà accedere alla schermata di dettaglio che varia solo nelle funzioni del bottone posto in fondo ad essa e alle azioni del menu. Le nuove azioni consentono infatti all'utente di modificare durante la partecipazione ad una campagna la strategia di invio scelta, oppure di abbandonare la campagna e rinunciare all'invio di altri dati. Il pulsante posto in fondo alla schermata invece cambia il suo utilizzo in base alla modalità di invio scelta.

Invece, qualora sia stata selezionata la modalità "automatica con preferenze" o "manuale", l'invio dei dati ha origine dal dispositivo. Nella modalità "automatica con preferenze" ciò avviene in background grazie al Service legato al ForegroundService, mentre in modalità "manuale" l'azione di invio è originata dall'utente nella schermata di dettagli della campagna

```
PUT /things/crwsns:dashboard HTTP/1.1 204 1.1018ms
GET /things/crwsns:dashboard HTTP/1.1 200 1.3478ms
PUT /things/smartphone_d020f780-bcc4-41a7-a9ed-72b57cc67b21 HTTP/1.1 204 455.8µs
GET /things/smartphone_d020f780-bcc4-41a7-a9ed-72b57cc67b21 HTTP/1.1 200 607.1µs
```

Figura 4.11: Log della TDD che mostra la registrazione del Master e di un Mobile Client

```
Thing smartphone_d020f780-bcc4-41a7-a9ed-72b57cc67b21 created
Resetted timeout
Device resetted
Thing smartphone_d020f780-bcc4-41a7-a9ed-72b57cc67b21 updated
Resetted timeout
mqtt read resource
Resetted timeout
Successfully collected data!
mqtt read resource
Resetted timeout
Successfully collected data!
mqtt read resource
Resetted timeout
Successfully collected data!
mqtt read resource
Resetted timeout
```

Figura 4.12: Log del Master durante la partecipazione di un Mobile Client ad una campagna in modalità Push

Capitolo 5

Conclusioni

In questa tesi inizialmente è stata presentata una panoramica attuale dello stato dell'arte nelle principali tematiche trattate, per poi evidenziare uno scenario non ancora studiato e realizzato, ovvero un'architettura che consenta di realizzare e gestire campagne di Mobile Crowdsensing sia in modalità Push che Pull tramite l'utilizzo del Web of Things. Questo elaborano ne tratta l'implementazione, illustrando prima di tutto l'architettura nei componenti che la compongono e come interagiscono tra loro. Successivamente viene discussa l'implementazione realizzata. L'architettura presentata permette di raccogliere qualsiasi tipo di dato ottenibile dai soggetti partecipanti al sistema, senza limiti di scalabilità o di interfacce. Ogni dispositivo capace di interagire con il Web of Things, se correttamente configurato può prendere parte al sistema e contribuire alla raccolta dati. Il risultato ottenuto è un'architettura altamente modulare e predisposta alla configurazione secondo le esigenze degli utenti, che consente di raccogliere dati sfruttando sia la strategia Push che la strategia Pull.

5.1 Possibili sviluppi

L'ampia modularità dell'architettura lascia spazio a diverse implementazioni e migliorie dei componenti già presenti. Di seguito ne vengono elencati

alcuni:

- Web App per i CrowdSourcer: al momento la creazione delle campagne avviene manualmente tramite l'inserimento di un oggetto nel database. Il processo può essere migliorato tramite la realizzazione di una Web App che comunica con il Master e che in seguito ad una autenticazione sia eventualmente possibile, oltre che alla creazione di nuove campagna, la gestione e la modifica di esse, e la visualizzazione e analisi in tempo reale dei dati raccolti,

- Master distribuiti: il carico di dati gestiti dal Master può crescere vertiginosamente con l'aumentare dei Mobile Client e delle campagne disponibili, si ritiene quindi ottimale la suddivisione della mole di lavoro tra un pool di Master, che eseguono lo stesso codice ma sono distribuiti in un'area geografica più o meno ampia. In questo modo il Mobile Client si interfaccerà con il Master più vicino con il carico minore.

- Maggio privacy: la gestione della privacy degli utenti può essere notevolmente incrementata, per consentire che i dati inviati non siano trasmessi in chiaro, che le Interction Affordances esposte siano protette da letture indesiderate e che i dati salvati dal Master non contengano informazioni che rimandino all'utente che li ha originati, ma che questa informazione sia accessibile solamente all'utente stesso.

- Più campagne: le campagne attualmente gestite sono soltanto quelle che richiedono unicamente la posizione, ma è possibile creare richieste dati più complesse, che richiedono una gestione apposita, ma che aumentano le possibilità a disposizione dei Crowdsourcer.

- Riconoscimento sensori avanzato: i sensori attualmente disponibili sono prestabiliti e fissi, ma possono esistere alcuni dispositivi che non ne implementano alcuni o ne posseggono alcuni non gestiti. Si richiede quindi una gestione maggiormente automatizzata e parametrica, che

individui i sensori disponibili e plasmi le capacità del device a partire dai dati ottenuti.

- Più protocolli: l'utilizzo di un bacino più ampio di protocolli, sia lato Mobile Client che Master, consente di ampliare i dispositivi che possono interagire con il sistema e quindi aumentare le possibilità di raccolta dati.

Bibliografia

[1] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[2] 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.

[3] Silvano Carradori. *Studio e implementazione di un Servient Android per W3C Web of Things*. PhD thesis, Università di Bologna, 2021.

[4] Smart Networks for Urban Citizen Participation. Sane web of things servient. `https://github.com/sane-city/wot-servient`, 2022.

[5] Web of Things Working Group. Web of things (wot) binding templates. `https://w3c.github.io/wot-binding-templates/`, 2023.

[6] Web of Things Working Group. Web of things (wot) mqtt binding template. `https://w3c.github.io/wot-binding-templates/bindings/protocols/mqtt/`, 2023.

[7] Eclipse Thingweb. Eclipse thingweb w3c web of things implementation on nodejs. `https://github.com/eclipse-thingweb/node-wot`, 2023.

[8] TinyIoT. Tinyiot thing direcotry. an implementation of the w3c web of things (wot) thing description directory (tdd), a registry of thing descriptions. `https://github.com/TinyIoT/thing-directory`, 2021.