ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

CAMPUS DI CESENA

CORSO DI LAUREA MAGISTRALE
IN INGEGNERIA E SCIENZE INFORMATICHE

Smart Campus: elementi per il design di un sistema di crowdsourcing di segnalazioni indoor

---

Relazione finale in

Applicazioni e Servizi Web

Relatrice

Dott.ssa Silvia Mirri

Studente

Mattia Ricci

1° sessione

Anno Accademico 2018/2019

Non sono solito alle dediche:

ho percorso ogni passo fin qui da solo;

ma da solo mi sarei perso infinite volte.

A chi mi ha sopportato,

a chi mi ha supportato:

grazie.

- Mattia Ricci -

Indice

Introduzione

La 1° edizione dello "Smart City Expo World Congress" di Barcellona[1] ha registrato in solamente 3 giorni la partecipazione di più di 20 mila visitatori provenienti da oltre 700 città di tutto il mondo, con l'obiettivo di perseguire l'innovazione, trovare nuove opportunità commerciali, offrire servizi innovativi e comunicare con gli *smart citizens*, i "cittadini intelligenti"; tuttavia ancora fino a poco più di 10 anni fa la parola *smart city* risultava un vocabolo inusuale e quasi sconosciuto. Sebbene non si sia ancora giunti a una definizione precisa per tale termine[2] è possibile ciò nondimeno considerare validi almeno 4 fattori per considerare "intelligente" una città:

- si assiste ad un ampio utilizzo di tecnologie digitali ed elettroniche all'interno delle città e delle comunità che vi vivono (si pensi ad esempio alla sempre più ampia diffusione di dispositivi mobili e della connettività Wi-Fi);
- lo sfruttamento delle scienze informatiche finisce per trasformare sia la vita che gli ambienti in cui si vive (l'utente può ricevere tramite il proprio telefono varie informazioni in tempo reale sulla città e decidere di conseguenza come comportarsi);
- le informazioni generate da tali tecnologie vengono quindi incluse nei sistemi di governo (spesso allo scopo di ridurre costi di gestione e manutenzione o per migliorare l'amministrazione della città);
- a livello territoriale si sfruttano quindi delle pratiche che portano la gente e la tecnologia a incontrarsi in un rapporto di miglioramento reciproco (nel 2016 gli investimenti nel settore di tecnologie per smart city hanno superato il costo di 80 miliardi di dollari e si è previsto che salga oltre i 135 miliardi nel 2021[3]).

Le Tecnologie di Informazione e Comunicazione (ICT) vengono quindi incontro ai bisogni dei cittadini e al contempo ricevono una spinta evolutiva dalle informazioni così raccolte. E' importante soffermarsi sul fatto che lo sviluppo rimane incentrato sulle esigenze quotidiane dei cittadini: la smart city deve essere prima di tutto rivolta verso i bisogni e i desideri dei suoi abitanti e affinché questo avvenga è necessario che

[1] http://www.smartcityexpo.com/en/home
[2] https://smartcitiescouncil.com/smart-cities-information-center/definitions-and-overviews
[3] https://www.techrepublic.com/article/top-smart-city-predictions-for-2019/

amministratori informati e cittadini attivi condividano una visione comune sulla realtà in cui vivono e il modo in cui vorrebbero si trasformasse; le informazioni ottenute dai sistemi di reporting sono disponibili a chiunque e permettono al cittadino di comprendere l'ambiente in cui vive, apprendere nuove tecniche e metodi e quindi intervenire con soluzioni concrete.

È all'interno di questo contesto che nasce nei poli scientifici e didattici l'idea di uno *smart campus*: se l'uso delle tecnologie e delle informazioni è applicabile a livello di una città e per i suoi cittadini, lo stesso è infatti possibile all'interno di un ambiente didattico come un'università e verso la comunità di studenti (e non solo) che la costituiscono.

Come sottolinea Cisco[4], che offre proprio dei servizi per la digitalizzazione di campus, è possibile abbassare i costi operazionali della struttura scolastica, offrire una migliore esperienza agli studenti, effettuare analisi ottimali sullo sfruttamento dei servizi e degli edifici dell'ateneo, con l'obiettivo finale di gestire in maniera intelligente l'intero impianto: ovvero di guidare la produttività, anche solo a livello locale di una piccola area e non di intero campus, in base ai dati ottenuti.

Nel 2012 l'università di Trento insieme a TrentoRISE e alla Fondazione Bruno Kessler lanciò un progetto di ricerca e sperimentazione volto a creare una piattaforma collaborativa in grado di creare servizi tarati sulle esigenze degli studenti e denominata per l'appunto "SmartCampus"[5]. Il successo dell'iniziativa ha portato in seguito ad un'evoluzione del progetto in "Smart Community", ancora in attività in data odierna.

Prendendo spunto da tutte queste nozioni, nell'elaborato si vuole pertanto ideare come servizio usufruibile all'interno del campus universitario una piattaforma dove gli utenti possano fornire e ricevere informazioni in maniera attiva, contribuendo ad una migliore efficienza nella gestione della struttura in cui vivono e ricevendone così anche beneficio. In particolare l'utente potrà sfruttare il sistema per segnalare tutti gli eventuali disservizi che possono intercorrere all'interno del campus, di cui il personale addetto verrà prontamente informato, cercando così di semplificare e al contempo velocizzare la risoluzione degli stessi.

Si ponga ora attenzione a due parole nel titolo di questo documento.

[4] https://www.cisco.com/c/en/us/solutions/industries/education/connected-campus.html
[5] http://www.triwu.it/smart-campus-studente/

“Segnalazione” comprende tra i suoi sinonimi “indicazione, informazione, suggerimento”; in particolare tale vocabolo vuole rimandare al significato che qualcuno, in questo caso l’utente, esegua attivamente la segnalazione, in un’ottica di *crowdsourcing*: da qui emerge l’idea di una raccolta, in maniera strutturata, di tutte le segnalazioni, che vengono effettuate in maniera spontanea dalla collettività. L’analisi di tale base di dati potrà portare a predisporre strategie e azioni per il miglioramento della gestione della struttura.

“Indoor”, termine preso in prestito dalla lingua inglese, significa letteralmente “interno”: in questo caso si vuole invece porre l’accento sul fatto che il sistema operi all’interno di una struttura locale, al chiuso, in contrapposizione alle città dove spesso si pensa invece all’ambiente in senso generale, su grandi spazi, senza preoccuparsi che i luoghi siano all’aperto o chiusi. Come si descriverà in seguito, questo influenzerà fortemente anche le scelte tecnologiche e di progetto da effettuare.

Si riassume di seguito il contenuto generale di questo documento.

Nel primo capitolo verrà approfondito il concetto di Smart Campus, così da comprendere in che modo questa idea possa essere adattata all’ambiente universitario in cui viviamo e quali benefici potrebbero esserne ricavati; di seguito si offre quindi una panoramica legata all’utilizzo di varie modalità di crowdsourcing e in che modo possano essere applicate; sono inoltre esposti una serie di esempi che dimostrano possibili rischi legati all’uso del crowdsourcing e le relative contromisure che potrebbero essere adottate per far sì che aumentino le probabilità di successo del progetto.

Il secondo capitolo esplora invece l’attuale contesto tecnologico nell’ambito della localizzazione indoor, sia dal punto di visto degli hardware che degli algoritmi, cercando di individuare una soluzione valida da utilizzare per questo progetto: vengono quindi proposte due possibili sistemi applicabili.

Nel terzo capitolo viene definita l’intera pianificazione dalla quale è scaturito il design del sistema finale, proponendo una serie di tecniche proprie dell’ingegneria dei sistemi per la progettazione di un software e seguendo un approccio che partendo dalle linee generali vada a definire sempre più in profondità i dettagli della realizzazione dell’applicazione.

Il quarto capitolo presenta tutte le varie tecnologie utilizzate per realizzare il sistema, i motivi legati a tali scelte e le modalità implementative adottate per dare forma a quanto si era programmato nella sezione precedente.

Infine l'ultimo capitolo presenta le considerazioni finali al termine del lavoro svolto e presentato; per terminare la dissertazione vengono proposte alcune idee in merito ai possibili sviluppi futuri della piattaforma realizzata e di altri possibili sistemi correlati.

1. L'uso del crowdsourcing per uno smart campus

Nell'introduzione si è accennato a due termini che costituiscono il fulcro di questa dissertazione, ovvero "smart campus" e "crowdsourcing": qui di seguito se ne darà una spiegazione più approfondita che permetta anche di comprendere maggiormente il lavoro che verrà svolto nel resto di tale dissertazione.

1.1. Smart Campus: origini e obiettivi

Come sottolinea un articolo apparso su Dotmagazine Online, *"Smart Campus – Merging Smart City and Smart Home in Education for Digital Natives"* [DOT18], è necessario ripensare all'esperienza scolastica dei moderni studenti: nello scenario classico questi erano unità individuali che si presentavano in classe per apprendere la lezione dal professore e ripassare poi libri e appunti principalmente da soli o al più insieme a pochi altri; invece nell'Era digitale [WIKI-IA19] risulta che chiunque è collegato a qualcun altro (non solo gli studenti, ma anche professori e altro personale universitario) tramite la rete, i dispositivi mobili e le applicazioni.

Tuttavia anche l'infrastruttura stessa e tutte le sue estensioni (sensori, misuratori, porte, videocamere, *eccetera*) sono connesse alla stessa rete delle persone: si tratta dell'*Internet of Things*, l'Internet delle cose [BRO16]. Tutti questi oggetti high-tech producono e condividono dati in maniera continuativa per poter garantire al contempo un risparmio sui costi operazionali e la massima efficienza, ad esempio monitorando i consumi energetici, accendendo e spegnendo lampadine, condizionatori e riscaldamento al momento più opportuno, fornendo sorveglianza e sicurezza. Il passo successivo da compiere dovrebbe pertanto essere quello di sfruttare i dati forniti dalle persone e quelli forniti dai dispositivi, che già condividono la stessa infrastruttura tecnologica, e usarli per poter offrire nuove possibilità e aumentare ulteriormente l'efficienza.

Si immagini per esempio quanto possa risultare arricchita l'esperienza di tutti gli studenti che durante una lezione possano avere accesso ad una pletora di dati provenienti dal mondo reale e utilizzarli nei propri progetti; ancora, è possibile creare

progetti più complessi collegando il lavoro di studenti di corsi differenti che possono facilmente comunicare tra loro grazie ad una piattaforma di studio comune. I vantaggi non sono solo quelli diretti: un risparmio sul consumo energetico di una grande struttura permette di avere un maggiore portafoglio da investire in nuove iniziative e attività di ricerca; una corretta gestione degli accessi è sinonimo di maggiore sicurezza e al contempo fruibilità del campus; diminuire il tempo per trovare parcheggio nelle ore di punta ugualmente significa migliorare la qualità della vita degli utenti. Si possono trovare ulteriori esempi osservando il video pubblicato sulla piattaforma Youtube da Anixter, azienda che fornisce prodotti e servizi di sicurezza e comunicazione su scala mondiale, che cerca di rispondere alla domanda: “Cos’è uno Smart Campus?” [ANI17].

1.2. Definizione di crowdsourcing

Nella sua etimologia il vocabolo deriva dalle parole *crowd*, “folla”, e *outsourcing*, ovvero “appaltare, acquisire dall’esterno”. Venne utilizzato per la prima volta nel 2005 da Jeff Howe e Mark Robinson nell’articolo della rivista Wired “The rise of crowdsourcing” per spiegare come certe attività sfruttassero Internet per poter usufruire del lavoro svolto dalle folle [HOW06].

Nella realtà tale pratica può essere rivista in esempi ben antecedenti all’epoca di Internet: per esempio nel 1714 il governo britannico istituì il “Longitude Prize”, un concorso con premio monetario a chi avesse ottenuto il miglior risultato nella misurazione delle coordinate longitudinali di una barca. Nel 1848 Matthew Fontaine Maury, ufficiale della Marina degli Stati Uniti nonché cartografo ed oceanografo, iniziò la distribuzione gratuita del suo “Grafico delle correnti e dei venti del Nord Atlantico” ai marinai in cambio del loro diario di bordo redatto durante il viaggio: loro potevano condurre così viaggi più veloci e sicuri mentre lui otteneva nuove informazioni; alla fine del 1861 aveva distribuito in questo modo più di 200 mila copie.

Dopo aver studiato più di 40 differenti definizioni del termine nel 2012 Enrique Estellés-Arolas e Fernando González Ladrón-de-Guevara, ricercatori all’Università Tecnica di Valencia, ne pubblicarono una [EST12] che potesse collegare tutte le diverse sfumature di significato incontrate, di cui si pubblica di seguito uno stralcio:

“*Il crowdsourcing è una tipologia di attività online partecipativa nella quale una persona, istituzione, organizzazione non a scopo di lucro o azienda propone ad un*

gruppo di individui, mediante un annuncio aperto e flessibile, la realizzazione libera e volontaria di un compito specifico. […] L'utente otterrà, a cambio della sua partecipazione, il soddisfacimento di una concreta necessità, economica, di riconoscimento sociale, di autostima, o di sviluppo di capacità personali, il crowdsourcer d'altro canto, otterrà e utilizzerà a proprio beneficio il contributo offerto dall'utente, la cui forma dipenderà dal tipo di attività realizzata."

Tale descrizione collima in maniera ottima con gli obiettivi enunciati nel paragrafo precedente sullo smart campus: l'università rappresenta l'istituzione che propone un compito senza fini di lucro agli studenti, i quali costituiscono gli individui liberi e volontari di realizzarlo; l'annuncio aperto e flessibile può essere realizzato, come nel caso preso in esame, tramite un'applicazione accessibile a tutti; sia gli studenti che l'università ottengono poi benefici nel migliorare la gestione del campus.

Occorre qui effettuare una precisazione: la folla a cui è proposto il compito non è identificabile con un preciso gruppo (come quello degli studenti), ma ad uno più ampio ed eterogeneo: anche i visitatori occasionali da scuole e licei, il personale didattico, quello tecnico e quello amministrativo possono concorrere a formare questo insieme di individui; tale aspetto in particolare si contrappone al tradizionale outsourcing, dove invece l'entità che realizza il lavoro è specifica e riconoscibile. L'anonimato costituisce infatti anche un altro aspetto positivo: Andrea Grover, curatrice della mostra sul crowdsourcing "Phantom Captain: Art and Crowdsourcing" nel 2006, in un'altra intervista per Wired [DEV07] afferma che le persone tendono a sentirsi più sicure e quindi fornire maggiori informazioni per il fatto di non essere riconoscibili e non sentirsi giudicate.

1.3. Criticità

Verranno ora mostrate le principali criticità sollevate in merito all'uso del crowdsourcing [WIKI-CROen19].

Poiché la partecipazione è aperta a chiunque, spesso chi fornisce informazioni non ha particolari qualifiche o non dispone degli strumenti necessari a fornire un servizio di alta qualità; di solito questo implica che vengano investiti ulteriori fondi nel progetto per rendere i partecipanti qualificati oppure per scremare la grande quantità di dati ottenuti e migliorarne la qualità [BOR11]. Un esempio di questa pratica può essere il

LEGO Digital Designer [LEGO19], un software gratuito rilasciato dall’omonima casa di giochi per bambini che permette a chiunque di realizzare le proprie creazioni e idee con mattoncini virtuali e quindi condividerle sul loro sito. Nel caso preso in esame l’applicativo che verrà prodotto dovrà pertanto permettere a chiunque di fornire informazioni in maniera strutturata e al contempo gestirle.

Anche la minore spesa affrontata dagli imprenditori che vogliono realizzare un progetto in crowdsourcing può risultare un inconveniente: proprio il fatto di investire meno risorse può condurre infatti ad un comportamento meno responsabile e ad assumere con più leggerezza i rischi di fallimento [DUP13].

L’aumento esponenziale di idee e progetti da sovvenzionare può da una parte portare a scoprire proposte che difficilmente sarebbero emerse dalla massa e avrebbero quindi ottenuto finanziamenti; tuttavia un alto numero di progetti a basso costo e maggiore rischio può portare ad una catastrofe imprenditoriale e alla perdita di fiducia generale da parte degli investitori. Ad esempio nel 2014 la Elio Motors lanciò una campagna di crowdfunding per promuovere un innovativo motore elettrico a 3 ruote che raccolse più di 17 milioni di dollari: la compagnia tuttavia in pochi mesi bruciò l’intero capitale senza ottenere risultati né in seguito ulteriori finanziamenti [STE17].

L’assenza sia di un contratto sia di un guadagno diretto, per esempio quello monetario, porta inoltre i volontari a perdere interesse nel progetto e quindi ad abbandonarlo oppure a fornire liberamente uno scarso servizio, inferiore a quello previsto. La startup Quirky, che dal 2009 era riuscita ad ottenere un capitale di oltre 180 milioni di dollari, a commercializzare oltre 100 prodotti e a contare più di un milione di membri, è finita in bancarotta nel Settembre 2015: tra le cause del fallimento è che spesso era stato promesso ai membri, oltre che di coltivare le loro idee, di renderli partecipi ai processi decisionali aziendali, cosa non sempre fattibile per motivi tecnici o di mercato: la comunità, sentendosi tradita, finì per abbandonare il progetto [FIX16].

Un altro aspetto negativo del crowdsourcing è legato all’attendibilità delle informazioni percepite e delle decisioni che ne conseguono. Diversamente dal primo punto, dove i dati sono affetti dall’inesperienza o incapacità di chi li fornisce, in questo caso i dati risentono di altri fattori.

Poiché ogni individuo manca della visione d’insieme del progetto può essere portato a fornire dati inutili o fuorvianti: è quel che accadde in seguito agli attentati di Boston del

15 Aprile 2013, avvenuti durante la maratona annuale della città [WILL13] [CHI13]. In particolare su due popolari siti, Reddit e 4chan, si scatenò una sorta di investigazione in crowdsourcing, dove migliaia di utenti cercarono di fornire prove e indizi per trovare gli ideatori dell'attacco. Le informazioni, a volte anche contrastanti tra di loro, vennero poi fornite sia al Federal Boreau Investigation, che si occupava del caso, che alle testate giornalistiche, le quali a loro volta le ripubblicarono senza i dovuti controlli, alimentando ulteriormente il caos di notizie in circolazione.

Caso peggiore si presenta infine quando si ottengono dati volutamente sbagliati, magari col fine di ledere il progetto o l'azienda sostenitrice. Nel 2013 la Durex annunciò il lancio di una nuova app che avrebbe permesso agli utenti di ottenere i prodotti desiderati, ovunque essi si trovassero, ordinandoli sul momento; l'azienda chiese al pubblico di votare la città dove sarebbe dovuta iniziare la propaganda pubblicitaria, ma sfortunatamente per loro vinse, probabilmente per il nome condiviso con l'omonimo supereroe dei fumetti, Batman, in Turchia, una piccola cittadina dove la maggior parte della popolazione segue i principi della tradizionale famiglia mussulmana e quindi rappresenta un pessimo mercato per un'azienda produttrice di contraccettivi, la quale si risolse ad abbandonare presto tale campagna pubblicitaria [FAI16].

1.4. Metodi di prevenzione delle problematiche esposte

In merito ad alcuni dei possibili problemi sopra esposti si esplorano di seguito in maniera più approfondita taluni concetti e si mostrano delle possibili tecniche e/o metodi che possano aiutare a prevenire o risolvere tali situazioni.

1.4.1. Qualità di trustability

Durante la progettazione di un sistema quale quello preso in esame il problema dell'attendibilità delle informazioni percepite dagli utenti, con riferimento agli ultimi punti illustrati nel capitolo precedente, assume particolare rilevanza. Esempi banali ma molto esplicativi possono essere un utente che per scherzo o convenienza voglia indicare un guasto nell'aula di laboratorio nella quale si sta svolgendo un esame, o ancora la segnalazione di un problema che in realtà è già stato risolto, costringendo l'addetto alla risoluzione a perdere tempo per un ulteriore controllo.

Letteralmente per *trustability* si intende la capacità di ispirare fiducia o fede [EDU19] e può essere intesa in vari modi.

Nei modelli di cloud computing, che ricevono e servono richieste da grandi masse di utenti, il problema della sicurezza dei dati rappresenta una delle maggiori sfide possibili [KUY11]. Un’idea interessante valutata nel paper “*Trust evaluation model of cloud user based on behavior data*” [CHE18] è la creazione di un modello per la valutazione della fiducia da accordare ad un utente del cloud basata sui dati del suo comportamento: gli autori sostengono che un consumatore potrebbe comportarsi in maniera malevola per il sistema per propria natura oppure perché un altro utente con intenzioni ostili si stia fingendo lui, ad esempio dopo avergli rubato le credenziali; il valore della fiducia da associare ad un utente può essere calcolabile e viene riflesso dalle azioni che compie e dall’analisi di quelle compiute in precedenza dallo stesso utente e da quelle compiute da altri utenti. In particolare la *comprehensive trust*, la fiducia totale dell’utente, è il sommario ottenuto dalla *direct trust*, la fiducia accordata all’utente sulle operazioni che sta effettuando, la *recommendation trust*, quella ottenuta dagli altri utenti, e dalla *historical trust*, l’andamento storico della fiducia data a quell’utente.

Un altro metodo vantaggioso per giudicare l’utente e al contempo garantire sicurezza al sistema è quello di assegnare gli utenti ad una fascia, o ruolo, in base a cui gli sono permesse o meno certe azioni o privilegi; l’utente può poi venire riassegnato ad una fascia di livello maggiore o minore a seconda delle azioni che compie. Tale strategia viene adottata da Discourse [ATW18], il popolare software open-source per la gestione di forum e mailing-list su Internet, il cui fondatore Jeff Atwood asserisce che da una parte offre protezione contro gli utenti con intenzioni malevoli e software dannosi come gli spam-bot, dall’altra invoglia gli utenti meritevoli a migliorarsi e meglio comprendere la community in cui vivono.

Oltre a valutare gli utenti che effettuano le segnalazioni tramite l’applicazione da realizzare, si desidera ora comprendere cos’altro potrebbe essere fatto per rendere il software *trustworthy*, ovvero affidabile sia per chi lo utilizza che per chi lo gestisce.

Spostandosi nel mondo dell’intelligenza artificiale si può osservare che il concetto di fiducia rappresenta un concetto fondamentale e una chiave per il successo di tale campo scientifico, come si evince dall’articolo “*Breaking down AI’s trustability challenges*” [KHA14]. In questo caso l’autore, Bahador Khaleghi, ricercatore presso la ElementAI, azienda di livello mondiale per la produzione di software AI, fa riferimento proprio alla fiducia che gli utenti finali possono accordare ai risultati prodotti da un sistema e al modo in cui questi li ha creati. Tra le caratteristiche da accordare a tali programmi

devono figurare sia la robustezza contro eventuali attacchi malevoli, mirati a produrre volutamente risultati sbagliati, sia la responsabilità, ovvero la possibilità di mantenere la traccia di chi ha inferito dati sbagliati, così da poter prendere misure che puniscano o comunque inibiscano l'utente colpevole. Un esempio pratico dell'uso di un sistema di questo tipo può essere Facebook: il popolare social network ha infatti annunciato [KAS17] di aver messo in atto tecniche di machine learning per poter controllare il proliferare di *fake news* e *fake accounts* in giro nel proprio sistema.

Per poter valutare la fiducia occorre fare riferimento a un qualche modello, ovvero una rappresentazione degli aspetti fondamentali di cui si vuole tenere conto durante il calcolo. Al di là delle diverse definizioni che son state date nel corso del tempo al vocabolo, le principali caratteristiche emerse sono [YAN07]:

- la fiducia è diretta, tra chi la accorda (*trustor*) e chi la riceve (*trustee*);
- la fiducia è soggettiva, ovvero basata sulle considerazioni personali di chi la accorda;
- la fiducia dipende dal contesto, per cui un soggetto può essere degno di fiducia o meno a seconda di questo;
- la fiducia è misurabile e può essere espressa secondo diversi gradi;
- la fiducia dipende dalle esperienze passate, sia di chi l'accorda che di chi la riceve;
- la fiducia è dinamica, quindi il grado di fiducia di una persona è diverso a seconda del momento in cui viene calcolato;
- la fiducia è trasferibile, ovvero è possibile fidarsi di un soggetto A se un utente B, fidato, si fida a sua volta di A;
- la fiducia è una proprietà composta, quindi durante il calcolo occorre tenere conto di diversi altri attributi, quali lealtà del soggetto (*dependability*), affidabilità (*reliability*), onestà (*honesty*), sicurezza della situazione (*security*), veridicità delle informazioni (*truthfulness*) e competenza di chi valuta (*competence*).

1.4.2. Tecniche di gamification

Il termine *gamification* deriva dall'inglese *game*, gioco, e potrebbe essere tradotto in italiano come "ludicizzazione": si tratta di un neologismo introdotto per la prima volta nel 2010 da Jesse Schelle, durante la "Dice conference" di Las Vegas [VIO11a]. Con

tale vocabolo si intende la pratica di inserire meccaniche e regole tipici del mondo dei (video-)giochi all’interno di contesti e attività che non hanno direttamente a che fare con un gioco, per esempio nel mondo del lavoro aziendale o del business.

Definendo formalmente un gioco, si tratta di un’attività strutturata caratterizzata da un insieme preciso di regole e con obiettivi raggiungibili e ben definiti [PRA18]. A questo punto ci si chiede come sia possibile invece applicare tali dinamiche al di fuori di contesti videoludici. Sul finire del 1700 alcuni negozianti americani iniziarono a distribuire monetine di rame agli avventori che si recavano più spesso da loro, con i quali riscattare di seguito prodotti reali: un semplice meccanismo di guadagno di punti con il quale creare fidelizzazione. Più di recente nel 2010 The Huffington Post, il noto blog di notizie statunitense, ha introdotto un semplice sistema di badge con il quale è riuscito a incentivare i lettori a passare maggior tempo sul proprio portale, condividere di più le notizie sui social network ma anche a monitorare meglio la moltitudine di commenti postati giornalmente sul sito [VIO11b].

Gli obiettivi raggiungibili tramite la gamification di un sistema sono molteplici [WIKI-GAMen19] [GAM19a]:

- fidelizzazione dell’utenza, che è portata a usare più spesso il sistema per completare obiettivi e ottenere qualcosa, come nell’esempio precedente;
- stimolo dell’interesse e della partecipazione alle attività, che vengono rese più piacevoli e gratificanti in quanto l’utente non è più solo un fruitore passivo di informazioni ma deve far qualcosa per ottenerle;
- anche la possibilità di mostrare ad altre persone gli obiettivi ottenuti incentivano comportamenti sociali quali la competitività e lo stabilire relazioni con gli altri;
- veicolazione di messaggi in maniera più efficace, sia perché aumenta la concentrazione di chi ascolta sia perché è possibile associare l’informazione ricevuta ad un’esperienza diretta (per esempio è possibile lanciare un messaggio ecologico facendo compiere all’utente azioni quali pulire un parco dalle cartacce e buttare i rifiuti nella raccolta differenziata);
- partendo dal punto precedente, è possibile andare a modificare le abitudini e i comportamenti dell’utente (il quale, continuando l’esempio, è portato ad agire con più rispetto della natura in quanto sensibilizzato verso l’ambiente).

Sfruttando tecniche di gioco all'interno del sistema in progettazione sarebbe possibile dunque ottenere alcuni intenti degni di nota, quali mantenere attivo l'interesse del pubblico verso l'applicazione, motivarlo a fornire informazioni di valore, disincentivare comportamenti scorretti (per esempio assegnando badge pubblici con significati negativi a chi si è comportato male), ma anche trasmettere messaggi importanti come mantenere il campus in ordine e sicuro.

Per poter *gamificare* un'attività occorre far ricorso ad almeno tre principali ingredienti [GAM19b] [GAM19c] [GAM19d]:

- le dinamiche costituiscono i fattori più astratti di un gioco, quali i desideri e le necessità che gli utenti vorrebbero vedere realizzati (come far comprendere all'utente che la struttura è sicura e può concorrere a realizzare questo obiettivo);
- le meccaniche sono invece i concetti logici, e dunque le regole, i limiti e i vincoli (per esempio lo studente può solo segnalare e non risolvere un disservizio ma può ottenere qualche riconoscimento in base alle sue segnalazioni);
- le componenti sono invece gli strumenti pratici da utilizzare:
    - *achievement* (obiettivi ben precisi da compiere);
    - *avatar* (rappresentazioni grafiche del personaggio);
    - *badge* (rappresentazioni grafiche degli achievement ottenuti);
    - *boss fight* (sfide complesse da compiere, ad esempio un questionario al termine di un percorso formativo);
    - *collection* (insieme dei beni virtuali accumulati, premi, badge, eccetera);
    - *combat* (task semplici e di breve durata da compiere);
    - *content unlocking* (sbloccare solo in seguito a determinati obiettivi certe cose);
    - *gifting* (possibilità di regalare/condividere risorse accumulate);
    - *leaderbord* (visualizzazione grafica e pubblica di progressi, badge, achievement di tutti gli utenti nel sistema);
    - livello (indicazione precisa del progresso ottenuto);
    - punti (indicazione numerica del progresso ottenuto);
    - *quest* (sfide a cui sono associate ricompense);
    - *social graph* (rappresentazione grafica della rete di contatti di un utente);

- *team* (gruppi di utenti in grado di cooperare per ottenere obiettivi non usufruibili singolarmente);
- beni virtuali.

2. Tecniche per la localizzazione indoor

Come accennato a termine dell'introduzione, un'altra caratteristica principale del sistema in progettazione è la possibilità di operare all'interno di spazi chiusi e di dimensioni decisamente ridotte se rapportate agli spazi di una città, come spesso accade all'interno di un campus universitario, dove i locali sono nell'ordine di dimensione di pochissimi metri e sviluppati frequentemente su più piani sovrastanti. Infatti è necessario che gli utenti che utilizzino il servizio di segnalazione siano in grado di fornire l'ubicazione del disservizio in maniera quanto più precisa possibile e al contempo in modo semplificato, magari con un metodo addirittura automatizzato, così che gli operatori possano rapidamente e facilmente individuare e risolvere il problema.

Negli spazi esterni è ben noto l'utilizzo di sistemi di geo-radiolocalizzazione per poter rintracciare la posizione degli utenti, i quali si basano sullo sfruttamento di una rete di satelliti artificiali posizionati in orbita o sul piano terrestre e l'analisi dei segnali ricevuti e trasmessi tra questi e i dispositivi utente [WIKI-GNSS19]. Tra i più noti si possono certamente contare il celebre *Global Positioning System* (GPS) della statunitense NAVSTAR [STUR07], il russo GLONASS [WIKI-GLO19], l'europeo Galileo [WIKI-GAL19] e in Asia il cinese BeiDou [WIKI-BEI19], disponibili e spesso anche già integrati nelle più comuni apparecchiature elettroniche quali tablet, smartphone, smartwatch, laptop.

Tuttavia negli spazi interni l'utilizzo di queste tecnologie può risultare spesso una soluzione poco pratica o desiderabile per una serie di fattori:

- la compresenza di diversi strati architettonici quali muri, pavimenti, tetti può disturbare o rallentare la propagazione del segnale, rendendo meno precisa la rilevazione corretta della posizione;
- essendo pensati per rilevare movimenti in spazio aperto i rilievi hanno una granularità abbastanza ampia (anche superiore al metro e nell'ordine di secondi come tempistiche);

- a causa dei materiali di costruzione o della posizione certe strutture risultano completamente schermate (si pensi ad un parcheggio sotterraneo o a certe aule ospedaliere che contengono macchinari particolari);
- strutture private o ad accesso ristretto vorrebbero evitare la comunicazione di segnali con satelliti pubblici o di dominio comunque esterno.

Nel corso del tempo sono state sviluppate diverse metodologie per la localizzazione indoor, che variano a seconda dell'hardware sfruttato, dei costi implementativi, dei contesti di applicazione e infine del tipo e del metodo di analisi del segnale usato.

2.1. Differenze in base al mezzo di propagazione

La prima categorizzazione può essere effettuata sulla base del tipo di segnale usato.

Nelle tecniche a infrarossi i dispositivi mobili ricevono un segnale ottico da un emettitore fisso nell'ambiente e riescono, anche in base alla potenza del segnale, a calcolare la propria posizione: grosso limite di questa tecnologia è che occorre mantenere il contatto visivo tra i due apparecchi altrimenti il segnale di luce viene interrotto, come avviene quando si passa di fronte ad una fotocellula.

Un altro tipo di segnale utilizzato prende spunto in natura dai pipistrelli e sfrutta l'emissione di onde a ultrasuoni per la ricerca dei dispositivi [IJAZ13]: il costo di questo metodo è contenuto e la precisione è nell'ordine dei centimetri, ma anche in questo caso il segnale risente di disturbi ambientali dovuti alla riflessione delle onde acustiche sugli ostacoli o alla densità dell'ambiente fisico attraversato (che rallenta la propagazione del segnale).

Passando invece alle tecniche basate sui segnali radio (*Radio Frequency IDentification*, RFID) i problemi derivanti da ostacoli ambientali come muri e altri oggetti diminuiscono notevolmente. Rimangono invece, come negli altri casi, quelli legati alla *multipath propagation* [FS-MP19], ovvero la ricezione dello stesso segnale più volte a causa della riflessione su qualche tipo di superficie. I circuiti RFID possono essere passivi, con costi minimi e cortissimo raggio d'azione (nell'ordine dei centimetri), usati per esempio nelle etichette anti-taccheggio dei negozi o nei moderni metodi di pagamento "a contatto" con la tecnologia *Near Field Communication* (NFC) [VAN04]; i circuiti attivi al contrario hanno costi superiori, maggiore dispendio energetico e occupano uno spazio di qualche centimetro ma permettono il riconoscimento degli

obiettivi anche a un centinaio di metri. Un segnale radio estremamente diffuso è sicuramente quello del Wi-Fi, i cui ricettori sono in dotazione ormai in ogni tipo di dispositivo mobile esistente [WEB19].

2.2. Differenze in base all'algoritmo di analisi del segnale

Una volta scelto il mezzo di comunicazione fisico tra i due apparecchi, occorre analizzare il segnale ricevuto dai dispositivi mobili per calcolarne la posizione. Gli algoritmi si basano in genere sulle seguenti considerazioni:

- si vuole scoprire la posizione di un dispositivo mobile in trasmissione ma è nota la posizione del ricevitore, che è fisso nello spazio;
- occorre tenere conto della velocità di propagazione del segnale e del tempo in cui questo viene inviato e poi ricevuto;
- a causa del fenomeno di multipath propagation, spiegato poc'anzi, si potrebbero ricevere posizioni multiple nello spazio dello stesso dispositivo.

Di seguito si elencano i principali metodi di analisi.

La stima del tempo di arrivo (*Time of Arrival*, TOA) si basa sul presupposto che il segnale si propaghi ad una velocità fissa, dunque conoscendo l'orario di emissione dal dispositivo e quello di arrivo nel ricevitore è possibile calcolare la distanza dal quale è stato propagato. Usando soltanto un ricevitore si può ottenere solo un'intorno possibile entro il quale si trova l'emittente, pertanto è consigliabile sfruttare almeno 3 diversi riceventi e calcolare il punto d'intersezione tra le distanze (come in Figura 3.2.1). Un esempio di sistema che utilizzi tale metodo è il già citato GPS [WIKI-TOA19].

La stima sulle differenze dei tempi di arrivo (*Time Difference of Arrival*, TDOA) è al secondo posto per popolarità tra gli algoritmi di analisi; a differenza di ToA non richiede di conoscere l'orario di emissione del segnale, ma si basa solo sull'orario di arrivo: il segnale viene raccolto da almeno 2 ricevitori e si calcola la differenza tra questi due tempi di arrivo, ottenendo un'area ridotta delle possibili posizioni del dispositivo emittente. Aggiungendo anche in questo caso un 3 ricevitore è possibile calcolare con efficacia la posizione corretta, come si può osservare in figura 3.2.2; con 4 riceventi si può valutare anche la posizione tridimensionale dell'oggetto e dunque

ottenere l'altezza rispetto al piano di riferimento a cui si trova [OKE17]. Questo metodo, denominato anche multilaterazione, è utilizzato ad esempio per il tracking di oggetti avionici, come avviene negli aeroporti di Malpensa e Fiumicino [COR09].

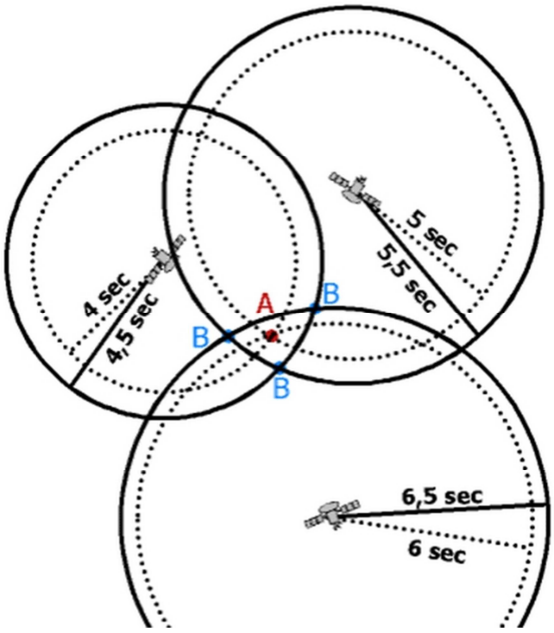


Figura 3.2.1: esempio di triangolazione satellitare del sistema GPS con risultato corretto (punto A) e con errori di misurazione (punti B)

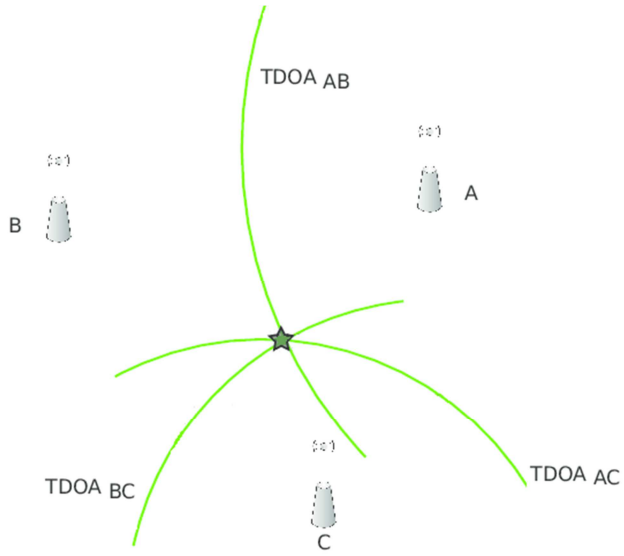


Figura 3.2.2: rappresentazione di multilaterazione con TDOA, le curve rappresentano i punti per cui rimane costante la differenza tra i ricettori

Il calcolo dell'angolo di arrivo (*Angle of Arrival*, AOA) presenta i vantaggi, rispetto alle precedenti tecniche, di non richiedere la conoscenza dei tempi di emissione né ricezione e la possibilità di ottenere la posizione del dispositivo mobile usando solo 2 ricevitori

nel caso di mappatura 2D (Figura 3.2.3) e 3 per quella 3D. Come dice il nome, si stima l'angolo di arrivo del segnale e intersecando le rette delle direzioni da cui si è propagato si risale al punto di emissione. Tuttavia per ovviare ai problemi di propagazione del segnale occorre utilizzare hardware di buona qualità, in grado di restituire informazioni precise, pertanto il costo è decisamente maggiore. Uso pratico di questo metodo si trova nella localizzazione dei telefoni cellulari o di stazioni radio [WIKI-AOA19] [MAO19].

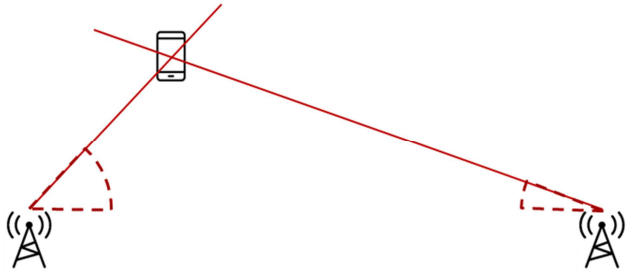


Figura 3.2.3: esempio di locazione di un dispositivo con AOA

Ultimo metodo qui revisionato è l'indicatore della potenza del segnale ricevuto (*Received Signal Strength Indicator*, RSSI) [RFID14]: basandosi sull'intensità del segnale ricevuto e conoscendo il fattore di riduzione della potenza dello stesso è possibile calcolare la distanza che il segnale ha percorso. Se da un lato questa tecnica è abbastanza semplice, dall'altro risente notevolmente delle interferenze ambientali e dello spostamento di persone e oggetti nell'ambiente, pertanto occorre effettuare un training abbastanza costoso per ottenere informazioni precise su come il segnale si propaghi nell'ambiente.

A seguito di questo breve studio sulle varie metodologie per poter effettuare un tracciamento automatico di un dispositivo mobile in ambiente chiuso si delineano di seguito due diverse possibilità che potrebbero essere implementate in un sistema come quello preso in esame in tale documento.

2.3. Utilizzare il *Wi-Fi fingerprinting*

Considerando la pervasione delle reti wireless (WLAN) in quasi ogni edificio moderno e la possibilità di utilizzare il protocollo di comunicazione 802.11 in tutti i dispositivi mobili, anche quelli ormai più datati, l'idea di sfruttare il Wi-Fi anche come meccanismo di localizzazione ha attirato fortemente i ricercatori. Già nei primi anni 2000, mentre ancora venivano ricercati e sviluppati hardware specifici per la

localizzazione indoor, la Microsoft aveva posto l'idea di sfruttare le reti wireless con lo stesso proposito nel suo progetto RADAR [ROS01].

In particolare la tecnica del Wi-Fi fingerprinting [HE16] è stata specialmente indagata per via dei seguenti aspetti:

- si tratta di una tecnica RSSI, che non richiede dunque calcoli molto complessi per l'hardware per poter effettuare la locazione;
- si basa sul Wi-Fi, una tecnologia considerabile uno standard-de-facto e che quindi non richiede ulteriori installazioni né di hardware né di software, spesso è anzi già presente sia nei dispositivi mobili che nell'edificio da monitorare;
- sempre in riferimento al punto precedente, il Wi-Fi è una tecnologia molto studiata, considerata stabile e su cui viene investito molto per effettuare miglioramenti [WIKI-WIFIen19];
- è utilizzabile anche all'interno di ambienti particolarmente complessi;
- richiede solo la memorizzazione di una semplice matrice dei valori di potenza del segnale per ogni Access Point.

Tale tecnica si svolge in due passaggi. Durante la prima fase, in modalità offline, si effettua il training del sistema, durante il quale innanzitutto si suddivide l'intera superficie da mappare in una griglia composta da *Reference Point* (RP); maggiore sarà il numero di RP, migliore sarà la precisione con cui in seguito localizzare la posizione dei dispositivi emittenti (si confronti con Figura 3.3.1). Per ogni RP poi si calcola la potenza del segnale rilevato da ogni Access Point della struttura e lo si memorizza nella tabella insieme al MAC address dell'AP. A questo punto il training è finito e si può passare alla modalità operativa del sistema, quella online, dove i dispositivi richiedono il calcolo della propria posizione: in base alla potenza del segnale ottenuto da almeno tre AP si riesce a stabilire in quale RP più verosimilmente si trovi il dispositivo richiedente.

I contro di questa tecnica sono quelli già precedentemente spiegati relativamente alle tecniche RSSI, ovvero il rischio di incorrere in errori di calcolo dovuti alle varie interferenze ambientali ed il conseguente aumento dei costi nella fase di training: per ogni RP occorre effettuare più di una scansione dei segnali, a intervalli cadenzati nel tempo, e decidere infine il valore più significativo da mantenere in tabella [ELM18].

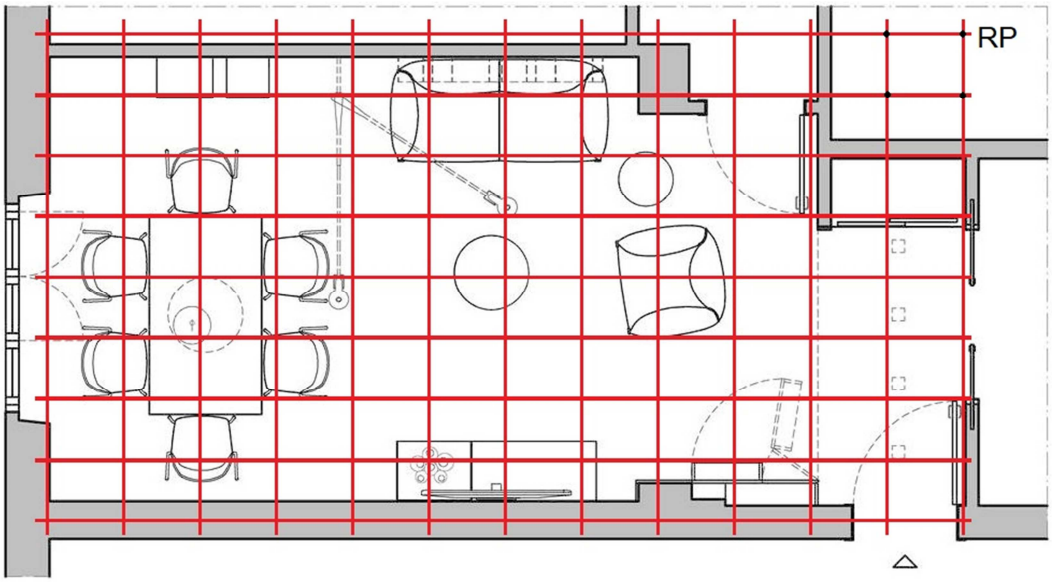


Figura 3.3.1: mapping di un ambiente domestico in Reference Point

Nonostante questo, la scarsa complessità computazionale richiesta e i costi quasi nulli di implementazione e non eccessivamente alti di manutenzione in seguito (una volta create, infatti, difficilmente le tabelle con la potenza dei segnali cambieranno, gli aggiornamenti principali possono riguardare l'installazione o lo spostamento degli Access Point) continuano a rendere questa tecnica una delle più desiderabili per ottenere la localizzazione automatizzata dei dispositivi mobili in ambiente indoor. Ulteriori studi sono stati effettuati per migliorare tale tecnica in ambiti particolarmente affollati ed estesi, per esempio cercando di diminuire i costi di training sfruttando i dati esposti direttamente dagli utenti, usando il crowdsourcing, o costruendo modelli basati sulle probabilità e sui comportamenti sociali [HE16].

2.4. Utilizzare l'aggancio di dispositivi Bluetooth

Il Bluetooth costituisce un'altra tecnologia basata su segnali radio ed estremamente in voga, con costi contenuti e un dispendio energetico limitato [WEBA19], sulla quale si fondano i Beacon, dispositivi hardware dalle dimensioni assai ridotte e molto adatti a ricevere e trasmettere piccoli messaggi su corte distanze [BEA19]: tant'è che tali strumenti si sono affermati prepotentemente nel mercato del marketing aziendale [NEA16], permettendo di effettuare pubblicità su misura ai consumatori direttamente sui loro dispositivi mobili, nel momento in cui questi passino accanto a un beacon. Altri modi interessanti di utilizzare i beacon e il bluetooth si sono visti nei musei, dove è possibile migliorare l'esperienza dei visitatori fornendo loro informazioni in tempo

reale sulle mostre in esposizione [BEA18], e nei punti vendita, nei quali i clienti possono essere riconosciuti, ricevere offerte in base ai propri consumi abituali, usare coupon, leggere dettagli sui prodotti [DIG16].

Similmente alla tecnica di localizzazione precedente è possibile installare una serie di beacon a un costo relativamente contenuto nella struttura e rilevare poi la posizione degli utenti in base ai beacon così agganciati.

Ciò nonostante tale tecnologia possiede, almeno per il momento, certi limiti [MCC19] che ne impediscono l'impiego pratico come metodo di localizzazione efficace: la portata del segnale raggiunta è ancora abbastanza limitata, costringendo a utilizzare un numero assai elevato di dispositivi e proporzionalmente alla superficie e al numero di piani dell'edificio da coprire; anche la frequenza di banda utilizzata, 2.4GHz, è la stessa di molti altri dispositivi wireless, per cui si presentano difficoltà in scenari dove sia presente un alto affollamento di persone e quindi un elevato numero di apparecchi elettronici che comunicano sulla stessa intensità di canale.

Tuttavia il 16 giugno del 2016 il Bluetooth Special Interest Group, l'associazione che si occupa dello sviluppo di questo standard di comunicazione, ha annunciato la versione 5.0 [HDB16], con la quale tali limiti dovrebbero finalmente venire superati, rivoluzionando il mondo dei dispositivi mobili e dell'Internet of Things: rispetto alla versione precedente la copertura del segnale dovrebbe aumentare fino a 4 volte, la velocità e la capacità di trasmissione migliorare ulteriormente e al contempo diventare più stabile a fronte di eventuali interferenze, garantendo così una maggiore affidabilità. Ma i prodotti montanti tale tecnologia sono stati rilasciati in commercio solo a circa metà del 2017 (ad esempio: Samsung Galaxy S8, S8 Plus e Note 8, Xiaomi Mi6 e Sony OnePlus 5 [WIZ19]), pertanto occorreranno anni prima che tale tecnologia si affermi e diffondendosi costituisca uno standard.

3. Progettazione del sistema: definizione, modellazione e design

Quanto descritto finora risulta fondamentale per poter avere una visione d'insieme sul contesto nel quale nasce il progetto, sulle ragioni che lo motivano e anche sullo stato attuale delle tecnologie con le quali è possibile produrlo. In questo capitolo si vogliono adesso definire le linee guide attraverso le quali tale progetto potrà diventare un'applicazione reale: per far questo vengono messe in campo alcune conoscenze, pratiche e metodologie di norma utilizzate e consigliate nella realizzazione di un progetto software. La fase di design rappresenta infatti la prima fase verso la trasformazione del problema in una soluzione, cioè un prodotto reale [THA19].

Quello che si ottiene è un *concept* del prodotto [CARL15], pertanto quanto viene presentato di seguito non sarà in maniera univoca la configurazione che assumerà l'applicazione finale, ma soltanto un insieme di linee guide per la fase di implementazione successiva.

In questa fase è possibile rifarsi ai *design pattern* [GAM02], ovvero una descrizione grafica o testuale di una soluzione comprovata ad un problema di progetto ricorrente in un determinato contesto: l'utilità di questi schemi di progettazione risulta nella possibilità di diminuire la complessità insita in un progetto, i tempi e i costi di sviluppo, fornire documentazione su attività e processi che andrebbero adottati e azioni che sono al contrario sconsigliate.

3.1. Comprensione degli obiettivi del progetto

La prima procedura che dovrebbe essere eseguita consiste nel limitare il campo d'azione del progetto, ovvero specificare da una parte tutti gli elementi che dovranno essere incorporati nel sistema, dall'altra comprendere invece quali non sono necessari. Per ottenere questo è necessario iniziare a definire in maniera più dettagliata le funzionalità e/o le proprietà che il sistema dovrà possedere.

È buona pratica partire dall'obiettivo principale del progetto che si era dichiarato, riassunto qui di seguito:

"*L'obiettivo che si vuole ottenere è la realizzazione di una piattaforma in grado di gestire le segnalazioni di disservizi effettuate dall'utenza interna ad un campus universitario e di avvisare rapidamente il personale tecnico affinché le risolvi. Sono richiesti i requisiti di semplicità nell'uso del sistema, la possibilità di localizzare in modo automatico o semi-automatico la posizione del disservizio ed elementi di* gamification*.*"

Allo scopo di evitare ambiguità e raffinare questi requisiti primitivi, tenendo inoltre conto degli studi preliminari condotti nei primi capitoli, sono stati condotti alcuni incontri di preambolo assieme alla relatrice, al termine dei quali sono state selezionate le seguenti specifiche di progetto:

- l'utente avrebbe avuto accesso al sistema grazie ad un'applicazione installata su dispositivo mobile, questo per via dell'enorme diffusione di tali apparecchi nonché per la possibilità di poter sfruttare i sensori, la capacità di accesso alla rete e la potenza di calcolo già insiti in tali dispositivi;
- per ovviare ai notevoli problemi tecnologici legati alla localizzazione indoor precedentemente spiegati senza dover incorrere in eccessivi costi economici per procurarsi l'hardware necessario o dover allungare eccessivamente i tempi di produzione del progetto, si è scelto di utilizzare una tecnologia già studiata e sviluppata precedentemente all'interno dell'università che fornisce all'utente la possibilità di sfruttare una mappa interattiva per selezionare una posizione della struttura e aver accesso a informazioni relative a quella immagazzinate all'interno di un database online; adattando questa tecnologia agli scopi del progetto, l'utente avrebbe potuto selezionare una posizione interna all'edificio scolastico e segnalarne un eventuale disservizio, al contempo avrebbe potuto ricevere la posizione di altri disservizi segnalati nel campus;
- vi era la necessità di raccogliere informazioni di vario genere sia sulle segnalazioni effettuate sia sugli utenti segnalanti, così da poterle in seguito analizzare, pertanto è stata prevista la costruzione di un database dove salvare questo storico di dati;
- gli utenti avrebbero ricevuto notifiche push in merito agli aggiornamenti compiuti da altri utenti sulle segnalazioni (per esempio un tecnico avrebbe

captato un messaggio di avviso quando fosse stata effettuata una nuova segnalazione e l'utente che l'aveva fatta avrebbe ricevuto notizia qualora fosse stata risolta);

- al fine di rendere più piacevole l'utilizzo di questa applicazione e al contempo di trasmettere un messaggio all'utente di un campus sicuro e da mantenere in ordine sarebbero stati ideati dei mini-giochi, i cui punteggi pro-capite potevano essere visualizzati in una classifica disponibile in una pagina dell'app; tuttavia tale requisito non è stato considerato necessario bensì facoltativo.

Studiando successivamente quale tipo di applicazione mobile creare, sono emerse le seguenti tre principali opzioni [HTML19]:

- applicazione nativa: si tratta di un'app specifica per un certo sistema operativo, il che la rende ottimizzata per la piattaforma sulla quale viene sviluppata, è veloce nell'operare, è in grado di funzionare anche offline e ha facile accesso a tutte le funzionalità del dispositivo mobile, ma proprio a causa di questa sua specificità ha costi di sviluppo maggiori, legati al dover implementare e aggiornare l'applicativo per ogni piattaforma su cui lo si vuole rendere disponibile;
- applicazione web: sono software che sfruttano un linguaggio indipendente dal tipo di piattaforma su cui operano, fondamentalmente si tratta di veri e propri siti web il cui funzionamento viene reso simile a quello di una normale app e come tali hanno quindi un impatto nullo sulla memoria o sulle capacità di calcolo del telefono, tuttavia necessitano della rete in modo costante per poter contattare il *web service* di appoggio; non risultano ottimizzate per nessuna particolare piattaforma e pertanto sono più lente rispetto al tipo di app precedente, inoltre possiedono un accesso nullo o solo parziale alle funzionalità del sistema operativo e non possono essere disponibili su nessuno store di applicazioni mobili, in compenso i costi di produzione sono alquanto contenuti;
- applicazione ibrida: come suggerisce il nome, si tratta di un'applicazione che prende caratteristiche da entrambe le precedenti tipologie, ovvero viene prodotto un sito web che sfrutta linguaggi cross-platform (in genere vengono utilizzati JavaScript, HTML5 e CSS) e che fa uso di un componente nativo (ad esempio una *WebView* in Android), grazie al quale l'esecuzione e la visualizzazione dei contenuti web ricorda all'utente quelli di una normale app; è possibile produrre

una versione unica per ogni piattaforma, per cui produzione, manutenzione e aggiornamento risultano poco costosi, inoltre l'accesso alla rete non è sempre necessario mentre è possibile sfruttare maggiormente le capacità dell'hardware e le funzionalità base del sistema operativo, anche se queste potrebbero variare a seconda della piattaforma.

Alla fine la scelta è ricaduta sulla tipologia ibrida, per la possibilità di avere accesso a certe funzionalità del dispositivo mobile (si è pensato per esempio che tramite la fotocamera l'utente avrebbe potuto scattare una foto a supporto della segnalazione che voleva effettuare, oppure GPS e Bluetooth sarebbero potuti risultare utili in futuro, volendo aggiornare il sistema di localizzazione del software), poiché non era necessario sviluppare diverse versioni del software specifiche per una singola piattaforma e diminuendo così i tempi di produzione e anche perché si è considerato che la app non avrebbe dovuto effettuare calcoli complessi e avrebbe ottenuto di conseguenza performance comunque accettabili.

3.2. Raccolta e analisi dei requisiti di sistema

Seguendo le linee guida dettate anche dal "*Project Management Body Of Knowledge*" [PMI14] è stata effettuata quindi la raccolta dei requisiti: formalmente un requisito [WIKI-REQ19] identifica una funzionalità chiave del sistema oppure un obiettivo che deve essere raggiunto tramite il suo ottenimento e rappresenta una proprietà del progetto, necessaria o desiderata, che si concretizza in seguito in una funzione, un attributo, una qualità o una caratteristica del software. I requisiti possono essere raccolti in uno schema concettuale gerarchico che aiuti i progettisti e gli *stakeholder* a mettersi d'accordo sulle componenti del progetto che verranno poi effettivamente implementate e che prende il nome di *Requirements Breakdown Structure* [WYS19]. Si spiega qui che gli stakeholder sono coloro che hanno un interesse nella realizzazione del progetto, solitamente si tratta di chi propone il prodotto oppure di chi lo andrà a utilizzare. Questo primo modello del sistema offre anche altre possibilità: per esempio aiuta a comprendere meglio il tipo di progetto che si vuole realizzare e di conseguenza permette di decidere fin dall'inizio il modo in cui dovrà essere gestito; consente inoltre di organizzare più facilmente i processi e le attività che dovranno essere compiuti per ottenere i requisiti dichiarati; in fase di implementazione lo schema delle componenti del sistema che dovranno essere realizzate può essere direttamente ottenuto dalla RBS.

I requisiti trovati saranno confrontati con gli obiettivi e le funzionalità precedentemente esposti: con questo metodo risulta più semplice e veloce comprendere se essi siano effettivamente proprietà o attributi necessari per il sistema.

La RBS realizzata può essere esaminata in Schema 4.2.1.

1. Sistema UniUtility

1.1. Gestione utenti

1.1.1. Gestione dati personali tramite interfaccia utente

1.1.1.1. Inserimento dati per il profilo

1.1.1.2. Modifica dati per il profilo

1.1.2. Visualizzazione dati tramite interfaccia utente

1.1.3. Gestione dati lato server

1.1.3.1. Calcolo dati d'interesse sul profilo

1.1.3.2. Invio di notifiche push per il profilo

1.1.4. Gestione mini-giochi

1.1.5. Gestione posizione utente

1.2. Gestione segnalazioni

1.2.1. Gestione dati tramite interfaccia utente

1.2.1.1. Inserimento dati per una segnalazione

1.2.1.2. Visualizzazione dati delle segnalazioni

1.2.1.3. Reperimento posizione della segnalazione

1.2.2. Visualizzazione dati tramite interfaccia utente

1.2.3. Gestione dati lato server

1.2.3.1. Calcolo dati d'interesse sulle segnalazioni

1.2.3.2. Invio di notifiche push al personale tecnico

1.3. Database

1.3.1. Analisi di dati d'interesse

1.3.2. Operazioni per le classifiche utenti

1.4. App ibrida

Schema 4.2.1: Requirements Breakdown Structure del sistema

Un'altra tecnica largamente utilizzata nella moderna ingegneria dei software è l'analisi dei casi d'uso [TAY03], il cui intento si può riassumere nell'ottenere una prospettiva più vicina al punto di vista dell'utente sull'uso e il comportamento del software, nonché di individuare i principali attori e componenti che sono parte del sistema e come questi interagiranno tra loro. Un caso d'uso [JAC14] rappresenta uno scenario nella quale si mostra come un requisito venga attuato o un obbiettivo venga ottenuto: viene definito da una serie di azioni o di passi che un certo ruolo compie per interagire col sistema così da ottenere un certo risultato. Solitamente vengono utilizzati template grafici come per esempio diagrammi UML e/o testuali per specificare un caso d'uso.

Di seguito si mostreranno i casi d'uso principali per l'applicativo sviluppato (Schema 4.2.2), seguendo questo modello:

Nome: <Nome identificativo del caso>
Precondizione: <Condizione che se verificata permette l'avverarsi del caso>
Flusso principale di eventi: <Lista ordinata degli eventi>

1. **Nome:** Interazione con la mappa
   Precondizione: Il dispositivo ha accesso alla rete.
   Flusso principale di eventi:
   1. L'utente accede alla pagina della mappa;
   2. L'utente seleziona il piano dell'edificio che vuole osservare
      a. L'utente può selezionare un punto nella mappa
      b. L'utente può cambiare piano osservato

2. **Nome:** Invio segnalazione
   Precondizione: Il dispositivo ha accesso alla rete; l'utente ha selezionato il piano dell'edificio in cui è avvenuto il disservizio.
   Flusso principale di eventi:
   1. L'utente seleziona il punto della mappa dove vuole segnalare il disservizio
   2. L'utente accede ad una form di segnalazione
   3. L'utente compila i dati della form
      a. L'utente può scattare una foto che verrà aggiunta ai dati della form
   4. L'utente effettua l'invio della segnalazione
   5. La segnalazione viene presa in carico da un server

3. **Nome:** Visualizzazione di una segnalazione
   Precondizione: Il dispositivo ha accesso alla rete; l'utente sta osservando un piano della mappa; sono presenti disservizi in quel piano.
   Flusso principale di eventi:
   1. L'utente seleziona il disservizio che vuole esaminare
   2. L'utente accede ad una scheda con tutti i dati inerenti il disservizio selezionato

4. **Nome:** Risoluzione di una segnalazione
   Precondizione: Il dispositivo ha accesso alla rete; un utente facente parte del personale tecnico della struttura ha ricevuto notifica di un disservizio; l'utente sta osservando un piano della mappa.
   Flusso principale di eventi:
   1. L'utente accede alle informazioni sulla segnalazione
   2. L'utente si reca alla posizione indicata della segnalazione
   3. L'utente accede nuovamente alle informazioni sulla segnalazione
   4. L'utente invia l'informazione che la segnalazione è stata risolta
   5. La nuova informazione viene gestita dal server

5. **Nome:** Registrazione account
   Precondizione: Il dispositivo ha accesso alla rete; l'utente possiede un indirizzo email.
   Flusso principale di eventi:
   1. L'utente accede alla pagina di gestione dell'account
   2. L'utente inserisce la propria mail e una password in una form
   3. L'utente si registra e invia le informazioni al server

6. **Nome:** Login
   Precondizione: Il dispositivo ha accesso alla rete; l'utente è già registrato al sistema.
   Flusso principale di eventi:
   1. L'utente accede alla pagina di gestione dell'account
   2. L'utente compila una form con la propria mail di registrazione e la password
   3. L'utente effettua la login
   4. L'utente accede alle proprie informazioni personali

7. **Nome:** Aggiornamento dati personali
   Precondizione: Il dispositivo ha accesso alla rete; l'utente ha già effettuato la login.
   Flusso principale di eventi:
   1. L'utente accede alla pagina di gestione dell'account
   2. L'utente seleziona l'informazione che vuole modificare
   3. L'utente invia le modifiche effettuate al server

8. **Nome:** Ricezione di una notifica push
   Precondizione: Il dispositivo ha accesso alla rete; l'utente ha dato il consenso alla ricezione delle notifiche push.
   Flusso principale di eventi:
   1. Il server elabora qualche nuova informazione
   2. Il server invia la notifica della nuova informazione a chi è necessario
   3. L'utente riceve la notifica sul proprio dispositivo
   4. L'utente può ricevere la notifica se la app è attiva
   5. L'utente può ricevere la notifica se la app è in background

Schema 4.2.2: Casi d'uso principali valutati per il sistema

3.3. Modelli concettuali del software

Una volta ottenuto l'insieme di tutti i requisiti del sistema che si vuole progettare è utile effettuare delle astrazioni ad alto livello in modo da ottenere dei modelli che descriveranno ad esempio quali entità interagiscano al suo interno, quali relazioni intercorrano tra di esse, in che modo si sposti il flusso dei dati, quali attività debbano essere compiute da quali processi, in che modo il sistema transiti da uno stato ad un altro [MAR04]. Sfruttando gli schemi costruiti nel capitolo precedente, ottenere questi nuovi modelli è risultato abbastanza semplice ed intuitivo. Un modello del dominio in ingegneria del software è un modello concettuale utile a mettere a fuoco le astrazioni principali del sistema complessivo, dando una rappresentazione formale della conoscenza del dominio tramite concetti, ruoli, regole [FOW03]. Si consideri che tali prototipi sono ancora ben lontani dall'effettivo design o implementazione che verranno

improntati in seguito: infatti un modello concettuale ha soltanto lo scopo di chiarire il significato dei termini in campo ed evitare ambiguità che possono verificarsi in merito ai concetti e alle loro relazioni tra chi analizza il problema, disegna il sistema, progetta il software o andrà ad effettuare una revisione della documentazione [WIKI-CM19].

La prima operazione utile è stata compilare un dizionario dei dati [MAR04] (Tabella 4.3.1): si tratta di una lista, strutturata secondo un qualche template, dove sono presenti i principali elementi del sistema insieme ad una loro definizione rigorosa. Tale schema è utile in primo luogo per evitare ambiguità in qualunque fase successiva di studio o implementazione del sistema, inoltre aiuta gli analisti a mettersi d'accordo con il committente del progetto sulle funzionalità che verranno inserite nel programma; infine, poiché è possibile che il progetto possa essere condiviso tra persone diverse o in futuro venga ripreso da qualcun altro, questo tipo di documentazione servirà loro a evitare fraintendimenti sui termini utilizzati.

Termine	**Sinonimo**	**Descrizione**
Utente	*Cliente, utente finale, utilizzatore*	Utente o utilizzatore finale del sistema implementato, possiede un account, utilizza tutte le funzionalità dell'applicazione.
Tecnico	*Operatore, personale specializzato*	Utente specializzato, fa parte del personale della struttura, è l'unico che può risolvere un disservizio e dichiarare risolta una segnalazione.
Applicazione	*Applicativo*	La parte software del sistema a cui il cliente ha accesso in modo locale sul proprio dispositivo, usata per interagire col sistema.
Dispositivo	/	L'apparecchio elettronico utilizzato da un utente sul quale è installata l'applicazione.
Server	/	Apparato software col quale gli utenti comunicano tramite l'applicazione, fornendo e ricevendo dati tramite la rete.
Rete	*Collegamento online, web, Internet*	Internet, rete ad accesso pubblico col quale l'applicazione può comunicare con il server.
Interfaccia utente	*GUI*	Lo strato di software visualizzabile graficamente dall'utente sul proprio dispositivo tramite il quale egli può sfruttare le funzionalità dell'applicativo.

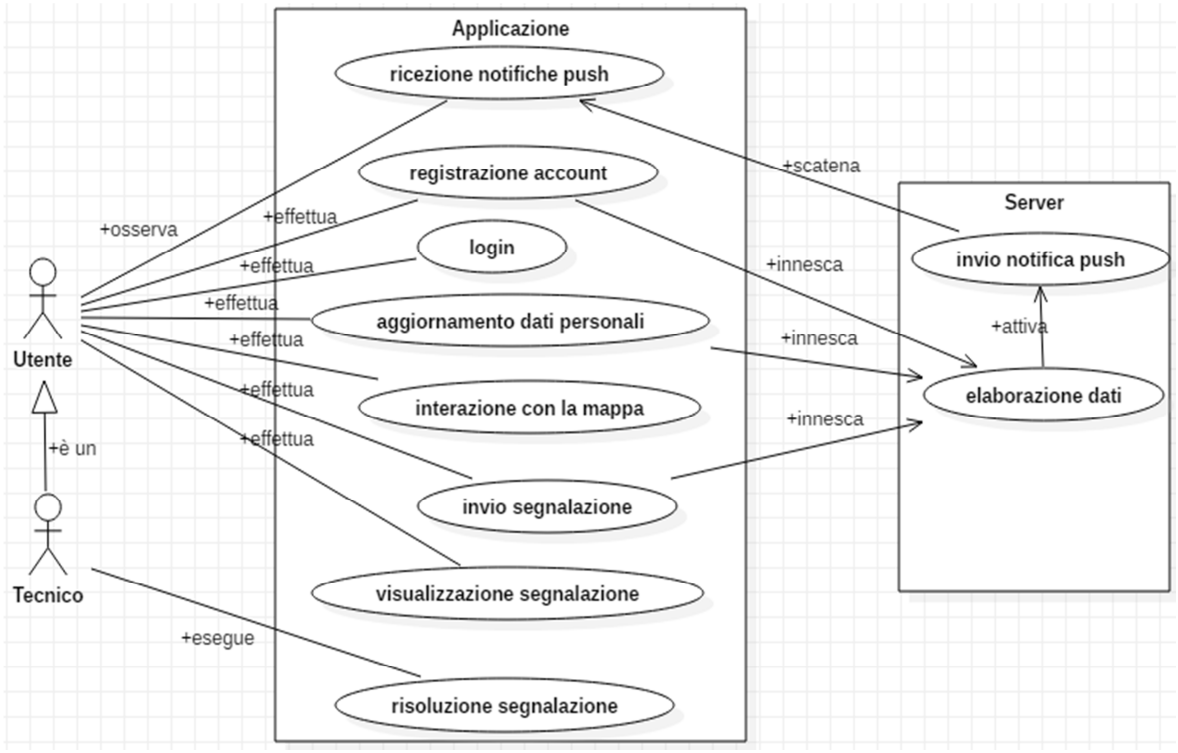
Account	/	Insieme di informazioni e permessi legati ad uno specifico utente.
Disservizio	*Malfunzionamento*	Un problema che si verifica durante il normale utilizzo di un qualche servizio interno alla struttura e che lo preclude.
Segnalazione	/	Comunicazione effettuata da un utente tramite l'applicazione a proposito di un disservizio, che viene reso noto a tutti gli utenti del sistema.
Posizione	*Punto, ubicazione, luogo*	Le coordinate che definiscono un punto unico e preciso nella mappa virtuale, corrispondente ad un luogo reale.
Notifica	*Avviso*	Messaggio inviato in maniera automatica dal server ai clienti in risposta al verificarsi di certi eventi.
Mini-gioco	/	Un elemento di gamification inserito come funzionalità nell'applicazione.

Tabella 4.3.1 : Dizionario dei dati

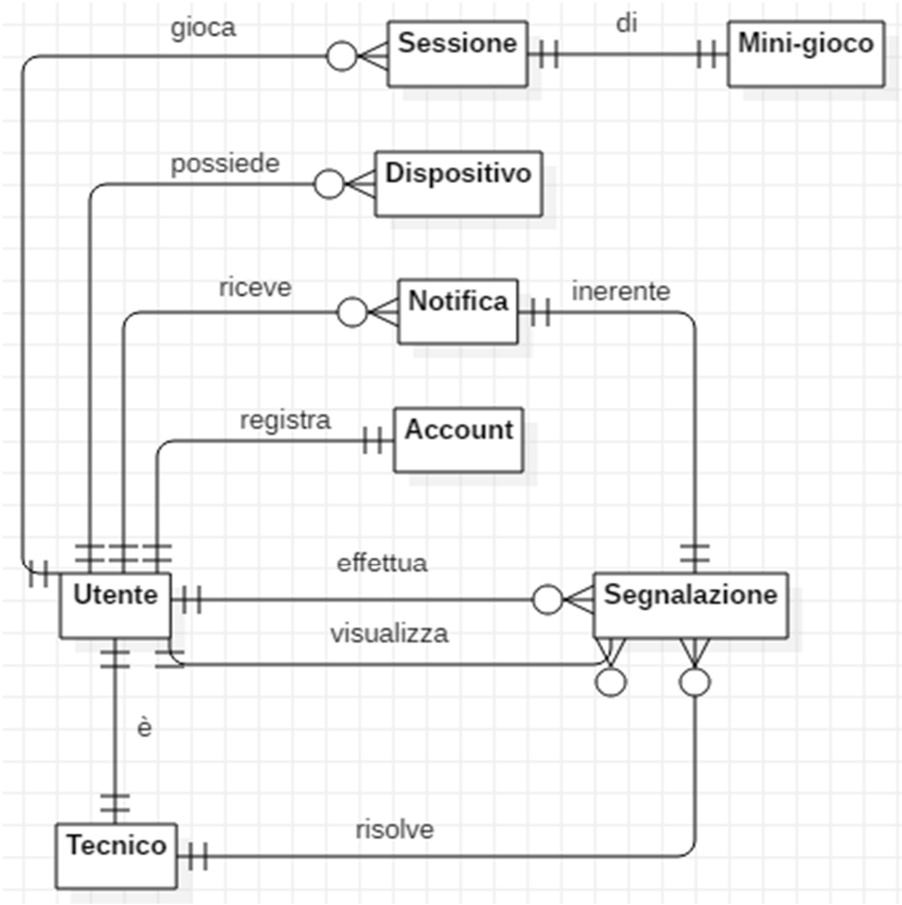
Partendo poi allo schema dei casi d'uso è possibile riassumere l'astrazione delle interazioni tra gli agenti esterni al sistema e il sistema stesso: nel caso preso in esame si è scelto di utilizzare un diagramma UML, strumento graficamente molto efficace per rappresentare questo tipo di modellazione (si veda Schema 4.3.1) e anche i grafici successivi.

Si può notare dal confronto con lo Schema 4.2.2 che sono stati aggiunti due casi d'uso relativi alla parte server ("Invio notifica push" e "Elaborazione dati") e che pertanto non interessano in modo diretto l'utente, motivo per il quale non erano stati conformati prima; tuttavia sono stati ritenuti a questo punto importanti dalla prospettiva dell'analista per avere un'idea migliore sul funzionamento generale del sistema.

Un altro modello concettuale molto vantaggioso è il diagramma entità-relazioni (ERD, dall'inglese *Entity-Relation Diagram*) [MAR04], il quale è in grado di descrivere le relazioni che intercorrono tra gli elementi del sistema, compresa la cardinalità delle relazioni (ovvero un oggetto può possedere una oppure multiple relazioni con un altro oggetto). Tale schema è osservabile in Schema 4.3.2.



Schema 4.3.1: Diagramma d'interazione del sistema



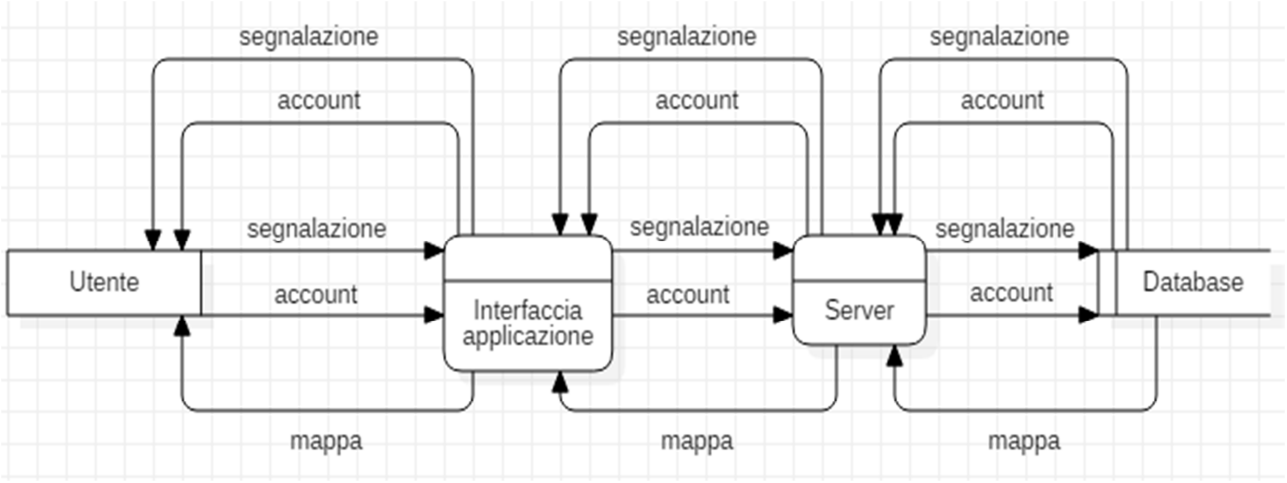
Schema 4.3.2: Diagramma ERD del sistema

Le relazioni sono bidirezionali, ovvero se per esempio un Utente possiede più di un Dispositivo, al contrario Più dispositivi sono posseduti da un Utente.

Spiegando di seguito lo schema:

- un Utente gioca a più Sessioni, una Sessione è di un solo Mini-gioco;
- un Utente riceve più Notifiche, inerenti una Segnalazione;
- un Utente registra un solo Account;
- un Utente effettua e visualizza più di una Segnalazione;
- un Tecnico è un Utente;
- un Tecnico risolve più di una Segnalazione.

L'ultimo grafico d'interesse realizzato è un diagramma di flusso dei dati (in inglese *DFD, Data-flow Diagram*) [MAR04], uno schema molto utile per presentare in che modo i dati fluiscano tramite processi, funzioni o componenti del sistema. Osservando il modello in Schema 4.3.3 si può notare la presenza di Database, elemento che non si era riusciti a specificare negli schemi precedenti, dove verranno salvate o recuperate le informazioni ritenute d'interesse; seguendo le direzioni indicate dalle frecce si nota che l'Utente è sia un elemento di input di dati che un consumatore di output; inoltre emerge il fatto che i dati inerenti la mappa sono già contenuti nel Database e vengono solamente prelevati per essere mostrati all'utente tramite l'Interfaccia della applicazione.



Schema 4.3.3: Diagramma DFD del sistema

3.4. Design dell'applicazione

Quanto descritto finora valeva a definire il dominio del problema, ovvero le proprietà strutturali e funzionali che un'applicazione debba possedere. A questo punto è possibile

arricchire il concept con altri elementi più dettagliati, inerenti la forma che il prodotto finale dovrà assumere.

Dovendo creare adesso un modello del software con cui l'utente finale andrà a interagire in modo diretto risulta utile rifarsi ai pattern per la progettazione dell'interazione tra uomo e macchina. I design pattern per l'interazione tra uomo e macchina, derivando dall'esperienza collettiva di altri progettisti, catturano le "*best practice*", ovvero quelle procedure che permettono di ottenere i risultati migliori; inoltre esprimono le invarianti che un buon progetto dovrebbe avere; aiutano il designer a pensare in maniera più ampia, senza i limiti imposti dalle tecnologie che conosce; risultano intuitivi anche per gli stakeholder; riescono infine a fornire soluzioni a problemi comuni appartenenti eppure a domini applicativi differenti [FOG19]. La norma è partire dall'idea iniziale dello spazio da progettare e applicare pattern, per parti e scendendo mano a mano verso dettagli più specifici, fino al livello di raffinamento finale: quanto si ottiene rimane tuttavia ancora un'astrazione della soluzione che suggerisce come il prodotto dovrebbe comportarsi di fronte all'utente; è un approccio diverso dai classici pattern di ingegneria del software, i quali al contrario tendono a combinare parti definite di strutture di codice che dovrebbero essere presenti nel prodotto finale.

Grazie ai pattern di interazione tra uomo e macchina è possibile giungere quindi alle mete di usabilità, ovvero:

- guidare l'utente verso il contenuto del mondo virtuale lasciandogli la possibilità di crearsi il proprio modello mentale del sistema;
- aiutare l'utente a svolgere un compito facendogli imparare l'azione adeguata durante l'interazione col sistema;
- permette all'utente di concentrarsi su cosa deve fare evidenziandogli i pattern spaziali e di navigazione che gli occorrono.

In questo modo è possibile ottenere la creazione di esperienze d'uso e spazi per sistemi d'informazione in cui l'utente si senta a suo agio [FOG19].

I pattern design utilizzati nel progetto sono stati trovati sul sito internet Ui Patterns [UIP19], che fornisce diversi schemi appositi per il Web Design, assieme a esempi, spiegazioni, casi d'uso e quando sono invece sconsigliati.

3.4.1. Struttura generale

Essendo l'applicativo pensato principalmente per l'utilizzo via dispositivi mobili, caratterizzati quindi in generale da schermi di piccole dimensioni, la prima necessità è stata trovare un modo di ottimizzare lo spazio di visualizzazione dei contenuti: seguendo il pattern "*Progressive Disclosure*", l'utente avrà la possibilità di concentrarsi su un solo compito alla volta ma al contempo avrà la possibilità di raggiungere tutte le funzionalità previste in maniera rapida; pertanto il contenuto è stato suddiviso in tre principali segmenti, corrispondenti ad altrettante pagine web, contenenti rispettivamente le funzioni inerenti la mappa e le segnalazioni, l'account e le impostazioni relative e infine alcuni elementi di gamification.

A questo punto si è scelto di applicare il pattern "*Module tabs*", utile quando il contenuto è suddiviso in più sezioni a cui accedere senza effettuare il ricaricamento delle pagine, di modo che l'utente non debba continuamente navigare avanti e indietro tramite vari link: la soluzione sarebbe stata inserire una barra orizzontale sul fondo della vista di ogni pagina contente tre pulsanti tramite cui scegliere quale pagina visualizzare al momento.

Per quanto riguarda le notifiche push invece esiste l'apposito pattern "*Notifications*" per informare su aggiornamenti e avvisi importanti:

- occorre attirare l'attenzione dell'utente con informazioni sensibili al fattore tempo, ovvero il cui interesse è ritenuto alto solo in un certo periodo di tempo (ad esempio un tecnico vorrebbe essere immediatamente informato di un nuovo disservizio e all'utente che l'ha segnalato interessa se viene risolto in un lasso relativamente breve di tempo);
- se l'utente sta già osservando l'informazione d'interesse è inutile notificargliela;
- se si tratta di un'operazione a cui l'utente non può partecipare né vi è coinvolto, non andrebbe avvisato (per esempio si è concluso che solo chi ha effettuato la segnalazione avrebbe ricevuto la relativa notifica di risoluzione da parte di un tecnico);
- occorre dare all'utente la possibilità di abilitare o disabilitare la ricezione delle notifiche;
- le notifiche devono risultare personalizzate per l'utente che le riceve: si è scelto di usare attributi quali il nome dato alla segnalazione, l'orario e il luogo.

Al riguardo dei mini-giochi invece ci si è risolti a seguire "*Periodic events*", una configurazione che prevede il ripetersi di certi eventi così da creare nell'utente un senso di interesse sostenuto nell'usare la app: ad esempio un mini-gioco può consistere nella presentazione periodica di cartacce in mezzo ai corridoi della mappa che vengano raccolte tramite un semplice click (come si era suggerito anche nel paragrafo 2.3.2).

3.4.2. Pagina "Home"

La pagina "Home" è quella contenente la vista sulla mappa interattiva e le segnalazioni da effettuare o effettuate, a cui si accede direttamente accendendo l'applicazione.

Come si era spiegato nel paragrafo 4.1, la mappa interattiva è una funzionalità già implementata e testata precedentemente al progetto preso in esame, pertanto non si è ritenuto di dover effettuare particolari modifiche dal punto di vista logico o funzionale: di fatti essa sarà il contenuto di sfondo principale in questa pagina, sopra la quale saranno applicati i nuovi elementi richiesti.

Per permettere all'utente di effettuare una segnalazione o visualizzare quelle esistenti si è stabilito l'uso del pattern "*Modal window*", in modo tale che egli visualizzi il nuovo contenuto in una finestra di dimensioni minori sovrapposta alla vista della mappa, che viene momentaneamente oscurata; l'utente può ritornare immediatamente al contenuto principale chiudendo la modale in due maniere, o terminando l'azione contenuta nella finestrella oppure cliccando sulla finestra principale sullo sfondo. Le modali sono state scelte per mostrare all'utente sia la possibilità di effettuare una nuova segnalazione, tramite un bottone inserito sul fondo della pagina, sia per visualizzare quelle già presenti, attraverso delle icone cliccabili lampeggianti sopra la mappa.

La funzionalità per segnalare un disservizio prevedeva che l'utente inserisse degli input in una form già strutturata, sia per caricare più velocemente le informazioni ritenute d'interesse per il sistema sia perché queste potessero essere poi facilmente processate dallo stesso. Si è deciso che l'utente avrebbe dovuto compilare alcuni campi fondamentali (il nome della segnalazione e la tipologia del problema); le coordinate del luogo sarebbero state recuperate in maniera semi-automatica dal sistema, ovvero in risposta al click dell'utente sul luogo indicato nella mappa; la foto e la descrizione del disservizio sarebbero rimaste invece scelte d'inserimento opzionali.

Secondo il pattern "*Input Feedback*" l'utilizzatore dovrebbe ricevere delle risposte immediate dal sistema in merito alle azioni d'inserimento e compilazione: pertanto prima di premere il pulsante di segnalazione l'utente avrebbe dovuto scegliere il luogo, se non l'avesse fatto sarebbe comparso un breve messaggio ad avvisarlo; prima di terminare la segnalazione l'utente avrebbe dovuto compilare i campi col nome e la tipologia altrimenti l'operazione non si sarebbe conclusa, con conseguente avviso; al termine dell'azione sarebbe infine stata presentata una nota di ringraziamento per il servizio svolto.

Un altro pattern utile per l'inserimento di input da parte dell'utente è "*Good Defaults*", quando vi è la presentazione di scelte numerose oppure complicate: il reperimento automatico delle coordinate del disservizio ricade in questo schema; ancora, sarebbe stato possibile che il sistema inserisse come scelta pre-impostata la tipologia del disservizio in base a certi parametri (per esempio all'interno di un bagno è probabile che il problema sia di tipo igienico, fatto meno plausibile all'interno di un'aula). Inoltre si è pensato che quando accade un disservizio in un luogo pubblico è più probabile che utenti diversi concorrano a segnalarlo: perciò si è scelto di mostrare agli utenti le segnalazioni effettuate precedentemente che potessero più presumibilmente corrispondere a quella che stessero cercando di effettuare, così da evitare loro di compilare l'intero modulo. Per questo si è ricorsi ad un altro modello, il "*Vote to Promote*", che consente agli utenti di scegliere in maniera democratica quale informazione debba avere maggiore valore: dunque gli utenti tramite il proprio voto avrebbero non solo effettuato una segnalazione, ma anche dato maggior peso a quello specifico problema, suggerendo al personale tecnico che si trattasse di una difficoltà di un maggiore rilievo per la comunità.

Per quanto riguarda la modale con le segnalazioni già effettuate, esiste la possibilità che all'interno di uno stesso ambiente siano presenti problematiche differenti; per non sovraccaricare di contenuti visivi la pagina ogni luogo avrebbe avuto una sola icona lampeggiante attiva, cliccando la quale si sarebbe aperta una piccola finestra con una lista dei nomi delle segnalazioni presenti: selezionando un titolo, si sarebbe aperta l'intera descrizione del disservizio. Questo è stato suggerito dal modello "*Accordion Menu*".

3.4.3. Pagina “Account”

Nella pagina “Account” sono contenute le funzioni principali inerenti l’utente, ovvero la registrazione di un account, il login dello stesso, la vista delle informazioni personali e la scelta delle impostazioni relative.

I suggerimenti dati dal pattern “*Account registration*” risultano qui cruciali: gli utenti sono poco propensi a registrarsi se non per via di eventuali benefici, che andrebbero quindi fatti presenti (ad esempio si è deciso che la ricezione di notifiche sarebbe avvenuta solo verso gli utenti registrati), o per la possibilità di effettuare certe azioni (quali sfruttare gli elementi di gamification dell’applicazione); non bisogna stressare l’utente con la richiesta di password eccessivamente complesse rispetto al contenuto offerto, in quanto può scoraggiare l’utente anche dall’utilizzo della app: si è scelto che ad esempio sarebbero stati aggiunti controlli automatizzati per verificare in particolare gli accessi del personale specializzato; una volta registrato l’account l’utente avrebbe automaticamente effettuato anche la fase di login, senza chiedergli di reinserire le credenziali.

Seguendo invece il modello di “*Lazy registration*”, si è deciso di inserire come secondo pulsante nella barra di navigazione principale il bottone per la pagina dell’account: se da una parte si vuole dare sufficiente importanza a questa vista, dall’altra è più probabile che l’utente preferisca prima utilizzare le funzionalità della pagina Home e in seguito decidere di registrarsi, accedendo così a ulteriori contenuti.

Registrandosi, l’utente avrebbe ottenuto oltre alla possibilità di ricevere notifiche push anche quella di sfruttare svariati elementi di gamification: un avatar personalizzato, un nickname da impostare, un livello di esperienza legato all’account, la lista delle pietre miliari completate riconoscibili da un badge, ovvero una piccola icona esemplificativa, la possibilità di effettuare i mini-giochi. Suggerimenti per queste opzioni sono stati presi dai pattern “*Competition*”, “*Levels*”, “*Achievements*”.

Sempre in questa pagina, come indicato dallo schema “*Settings*”, si sarebbe aggiunta la possibilità per il cliente di indicare alcune preferenze circa l’applicazione, quali l’attivazione dei mini-giochi e la scelta se ricevere le notifiche push.

3.4.4. Pagina “Classifiche”

La terza pagina dell’applicazione è utile per inserire l’ultimo elemento di gamification previsto, ovvero le classifiche degli utenti, questo per aumentare l’interesse degli utenti nell’usare l’applicazione favorendo la competizione, come suggerito anche dal pattern “*Leaderboard*”: nella scelta dei criteri di ordinazione è meglio evitare quelli legati alle attività degli utenti, perciò si è deciso di evitare ad esempio ordinamenti secondo il numero di disservizi segnalati o risolti; inoltre sono consigliate comparazioni basate su diverse categorie (livello dell’account, pietre miliari raccolte, punteggi nei mini-giochi) e lassi temporali differenti (punteggi migliori di sempre, settimanali, giornalieri). L’utente avrebbe potuto scegliere quale ordinamento applicare, visualizzando la conseguente classifica.

3.5. Verifica del modello

In seguito alla creazione dei modelli concettuali del software è utile applicare una fase di verifica degli stessi. Questo può essere ottenuto in vari modi, anche a seconda del modello che si vuole testare: per esempio mostrando al committente l’albero dei requisiti ottenuto (schema 4.2.1), gli schemi sulle funzionalità del sistema (schema 4.3.1 e 4.3.2) e un prototipo dell’applicazione, così da essere certi di aver colto tutti gli aspetti che dovrebbero essere inseriti nel progetto prima di effettuarne l’implementazione. Oltre a chi ha avuto l’idea, sarebbe bene soddisfare anche chi andrà a utilizzare in seguito il prodotto: per questo obiettivo è utile invece chiedere direttamente ai possibili clienti del sistema, tramite questionari online, riunioni pubbliche di presentazione del software, interviste a campione, raccogliendo in questo modo opinioni su quanto si vuole realizzare.

3.5.1. Creazione del prototipo

Parallelamente all’attività di design precedentemente descritta è stata svolta quella di creazione di mock-up esemplificativi dell’applicativo, in modo tale che il progettista potesse avere un riscontro immediato dell’aspetto che avrebbe assunto in seguito il software; il mock-up è infatti un prototipo utile a descrivere come sarà visivamente il prodotto finale [WIKI-MOC19]. Questo campione può essere utilizzato anche come campione di prova per il committente del progetto e da mostrare agli utenti e in seguito come spunto per i designer che implementeranno il software. Non avendo effettuato una

raccolta di opinioni antecedentemente infatti non si aveva la certezza sul giudizio del pubblico a riguardo del progetto e del prodotto da sviluppare. Occorre tuttavia fare attenzione a non illudere il committente e i clienti sul prodotto finale: può accadere che essi finiscano per desiderare fortemente che il prototipo stesso diventi il prodotto finale, evento non sempre possibile per via dei vari limiti insiti in un progetto (tempistiche, costi, mancanze tecnologiche); la norma è infatti che il prototipo venga eliminato [FAS19]. Tenendo ciò in considerazione il mock-up creato non contiene tutti gli elementi precedentemente considerati, ma soltanto quelli ritenuti indispensabili in questo progetto.

Ci si è avvalsi dunque di un tool predisposto alla creazione di mock-up di siti web e applicazioni mobile, Balsamiq [BAL19], disponibile sia online che in versione desktop: il software risulta semplice ed intuitivo da utilizzare e consiste in un'interfaccia grafica con editor WYSIWYG da cui selezionare le componenti desiderate da aggiungere al proprio prototipo (visibile in Immagine 4.5.1.1).

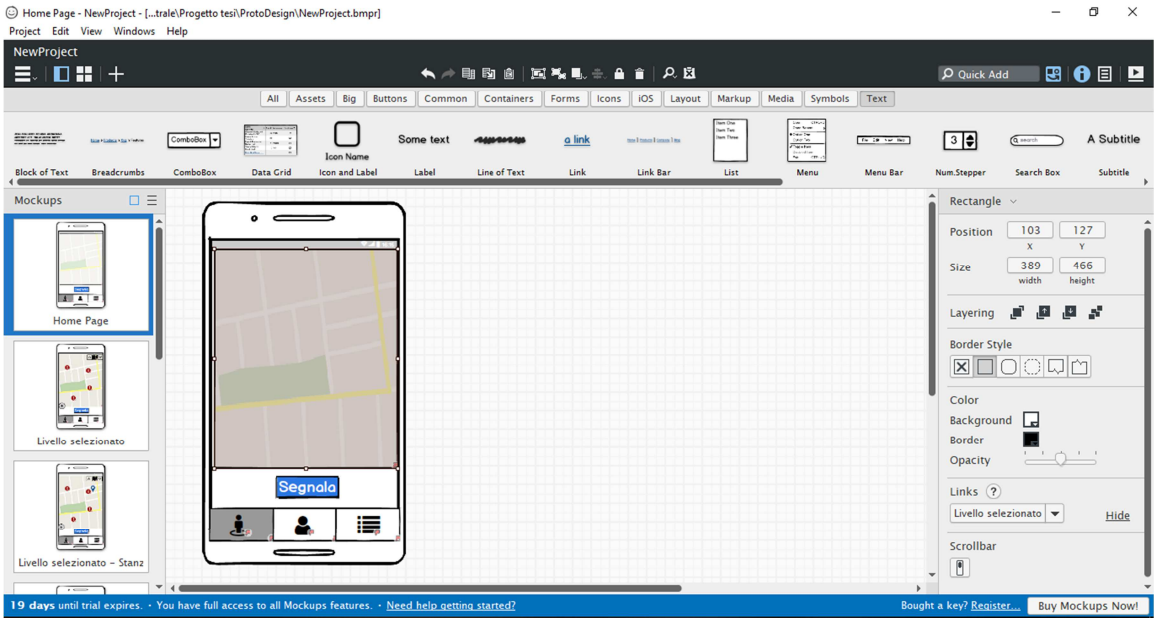


Immagine 4.5.1.1: L'editor di Balsamiq Mockups 3

In Immagine 4.5.1.2 è possibile osservare la schermata della pagina Home iniziale: sono presenti la mappa interattiva, il bottone per la segnalazione e la barra coi pulsanti per cambiare la pagina da visualizzare. Cliccando sulla mappa, verrà mostrato il livello della struttura selezionato e le eventuali segnalazioni presenti. I bottoni in alto a sinistra servono per cambiare livello della struttura (freccia su e giù) o per tornare alla vista precedente (tasto centrale).

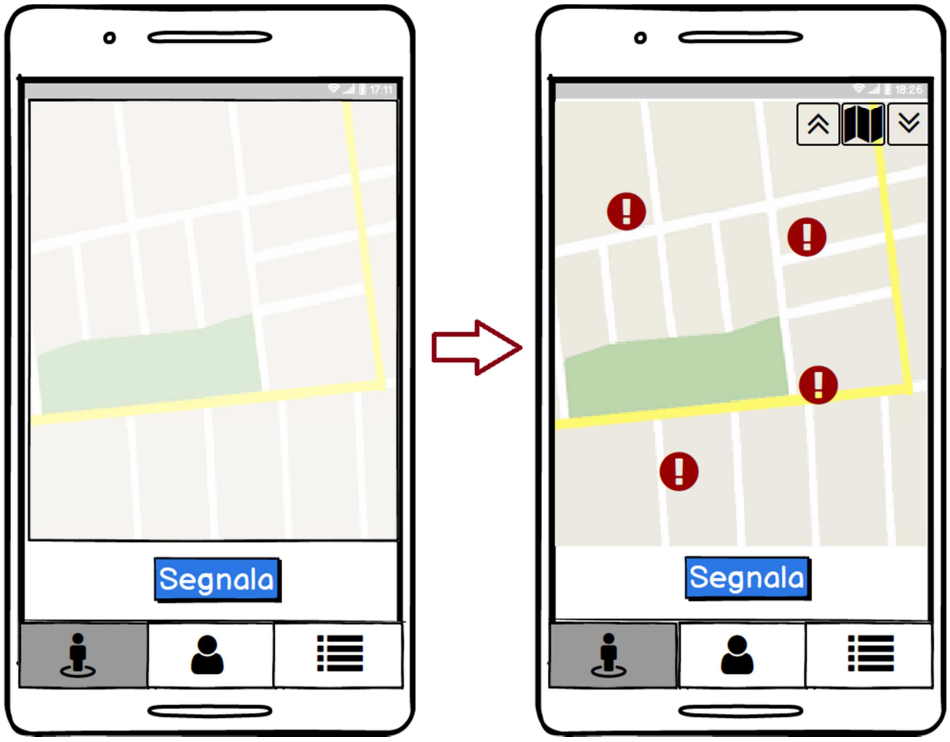


Immagine 4.5.1.2: Vista iniziale dell'applicazione e dopo aver cliccato sulla mappa

Selezionando un punto della mappa comparirà un marker in sua corrispondenza (Immagine 4.5.1.3): a questo punto l'utente può attivare il pulsante "Segnala", aprendo di conseguenza la modale di segnalazione con una form da compilare.

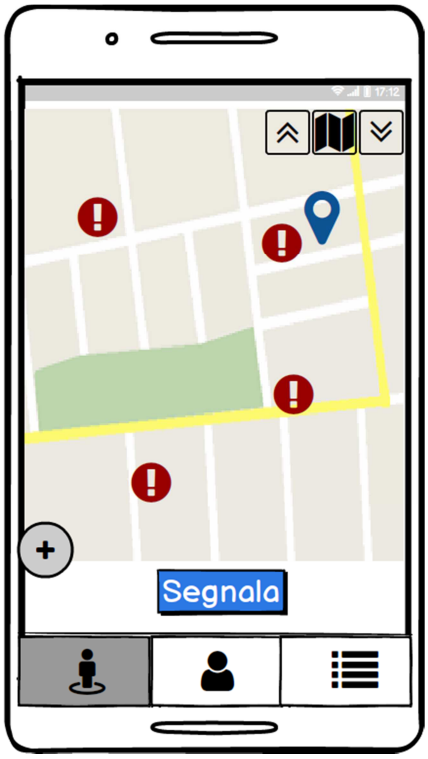


Immagine 4.5.1.3: Selezione di un luogo per cui effettuare la segnalazione

La finestra con il modulo di segnalazione è quella presentata in Immagine 4.5.1.4: le informazioni vicino all'etichetta "Dove:" sono state recuperate automaticamente dal sistema, l'utente deve selezionare la tipologia del disservizio incontrato da un menù a tendina accanto all'etichetta "Tipo:" e aggiungere un titolo chiaro ed esemplificativo alla segnalazione; le icone del fumetto e della fotocamera sono cliccabili e permettono rispettivamente di aggiungere un commento più lungo o una foto a supporto della segnalazione; sotto forma di bottoni sono aggiunti sul fondo suggerimenti su segnalazioni già effettuate.

Immagine 4.5.1.4: Finestra di segnalazione di un disservizio

Cliccando invece su una delle icone lampeggianti in rosso di Immagine 4.5.1.2 verrà mostrata la lista delle segnalazioni fatte per quelle coordinate; è possibile selezionare un titolo per espandere la corrispondente segnalazione e visualizzare tutte le informazioni relative (posizione, nome, tipo, foto e descrizione se presenti), come da Immagine 4.5.1.5.

Tramite il pulsante centrale della barra sul fondo è possibile raggiungere la pagina Account (Immagine 4.5.1.6): si possono notare la presenza di un avatar, un messaggio

di benvenuto, il livello raggiunto, la lista dei badge ottenuti dal raggiungimento di certi obiettivi, l'opzione per attivare la ricezione delle notifiche push.

Immagine 4.5.1.5: Vista di una delle segnalazioni effettuate

Immagine 4.5.1.6: Pagina Account

Infine si mostra l'aspetto della pagina Classifiche in Immagine 4.5.1.7: tramite la pulsantiera in alto si sceglie quale ordinamento applicare, nella lista si mostrano alcuni aspetti personali scelti dai vari utenti, quali account, nickname, badge preferiti e livello. La posizione dell'utente nella classifica risulta illuminata per aiutarlo a ritrovare la propria posizione in lista.

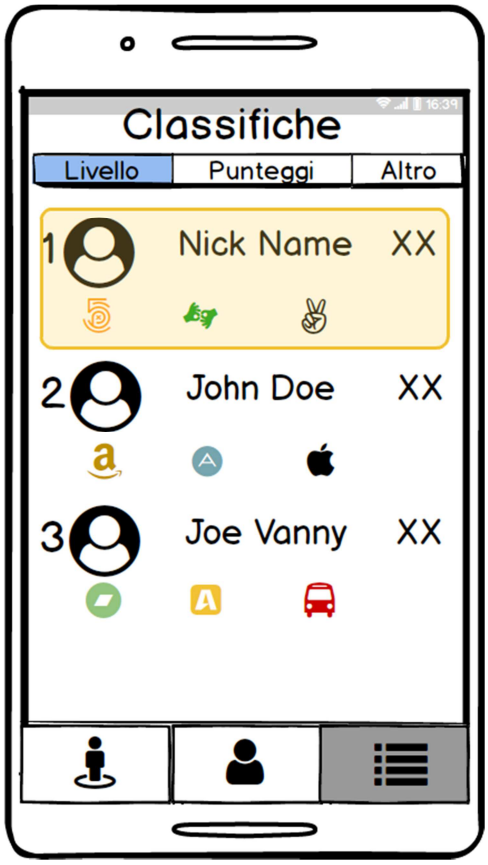


Immagine 4.5.1.7: Pagina Classifiche

3.5.2. Formazione di un Focus Group

Prima di effettuare l'implementazione del sistema, si è voluto cercare un primo riscontro dell'opinione del pubblico riguardo al prodotto da realizzare. Per far questo ci si è avvalsi di una tecnica per la rilevazione di idee e opinioni che nasce nel campo della ricerca sociale, molto utilizzata anche dalle aziende e in campo pubblicitario per valutare l'impatto di un servizio o di un progetto: il *Focus Group* (o Gruppo di Discussione, in Italiano) [WEBA11].

La modalità avviene coinvolgendo un piccolo gruppo di persone (in genere tra gli 8 e i 12 partecipanti) che sono invitati a discutere del tema d'interesse tramite attività o domande, guidati da un moderatore. I gruppi omogenei possono garantire un flusso

comunicativo maggiore dal momento che i partecipanti condividono stato sociale e cultura, tuttavia un gruppo eterogeneo apporta una ricchezza superiore dei punti di vista [MIG01]. Si è riusciti a raccogliere la presenza di almeno 13 individui, selezionati tra studenti di anni differenti del Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche: questo ha garantito un buon grado di partecipazione e intesa tra gli intervistati, sebbene siano venute a mancare le opinioni degli studenti degli altri Corsi di Laurea presenti all'interno del Campus universitario.

L'incontro è stato strutturato nel seguente modo: dopo una breve presentazione da parte del moderatore, ruolo interpretato dall'autore di questo documento, e la sincerazione che i partecipanti avessero ben compreso lo scopo della riunione, sono state fatte alcune domande generali sull'uso da parte degli utenti di dispositivi mobili nell'ambito quotidiano e della tipologia di applicazioni più utilizzate e se in particolare conoscessero o si servissero di alcune per l'ambito universitario. Si sono quindi rivolte domande in merito al tipo e frequenza di disservizi incontrati e alla modalità di risoluzione degli stessi, per cercare di comprendere la propensione che gli utenti avrebbero a usare un'applicazione per questo scopo: è emerso che il problema di segnalare disservizi è particolarmente sentito dagli utenti presi in considerazione, in quanto questi capitano abbastanza spesso: prendendo in considerazione gli esempi più rilevanti il malfunzionamento di uno schermo o di una macchinetta per la distribuzione delle vivande può accadere anche una o due volte alla settimana, il bisogno di cancelleria nuova in aula una volta o più al giorno, l'intervento di un tecnico per utilizzare il proiettore più volte alla settimana. Deve essere considerato che un disservizio è in generale l'interruzione dell'erogazione di un servizio, e può essere anche dovuto a cause banali, non necessariamente un guasto, quali il finire dei fogli di carta nella stampante o il dimenticarsi di rimettere a posto il telecomando del proiettore, pertanto è ragionevole che avvengano spesso. Un dato importante che è emerso di conseguenza è però il tempo impiegato per risolvere tali disfunzioni: se l'utente conosce quale tecnico si occupi di quella specifica problematica, allora è in grado di avvisarlo in modo diretto e risolvere subito (ad esempio per la cancelleria gli studenti sanno di potersi rivolgere alle guardiole installate nei corridoi di ogni piano); se gli utenti non sanno chi avvisare o in che modo farlo, l'unica soluzione è aspettare che qualcun altro lo faccia al posto loro (la macchinetta delle vivande potrebbe rimanere inusabile anche per giorni). Il gruppo si è quindi rivelato abbastanza entusiasta all'idea di un applicativo con cui avvisare in maniera semplice e veloce dei problemi riscontrati nella struttura. Il

passo successivo è stato spiegare loro l'idea generale del sistema, cosa avrebbe dovuto fare e come loro avrebbero potuto utilizzarlo. Per non influenzarli in nessun modo con quanto era già stato pensato si è chiesto direttamente a loro quali funzionalità avrebbe dovuto comprendere, che aspetto avrebbe assunto, come immaginavano di utilizzare il sistema, partendo dall'idea generale di "segnalazione" e attraverso la tecnica del *brainstorming*, permettendo alla quasi totalità dei partecipanti di esprimere il proprio pensiero. In Immagine 4.5.2.1 si può osservare la foto scattata alla lavagna al termine di questa sessione della riunione, dove sono riunite tutte le idee riferite. A conferma del lavoro di ideazione svolto precedentemente nei paragrafi 4.1 e 4.4, gli utenti hanno espresso la totalità delle funzioni che erano state pensate per questa applicazione.

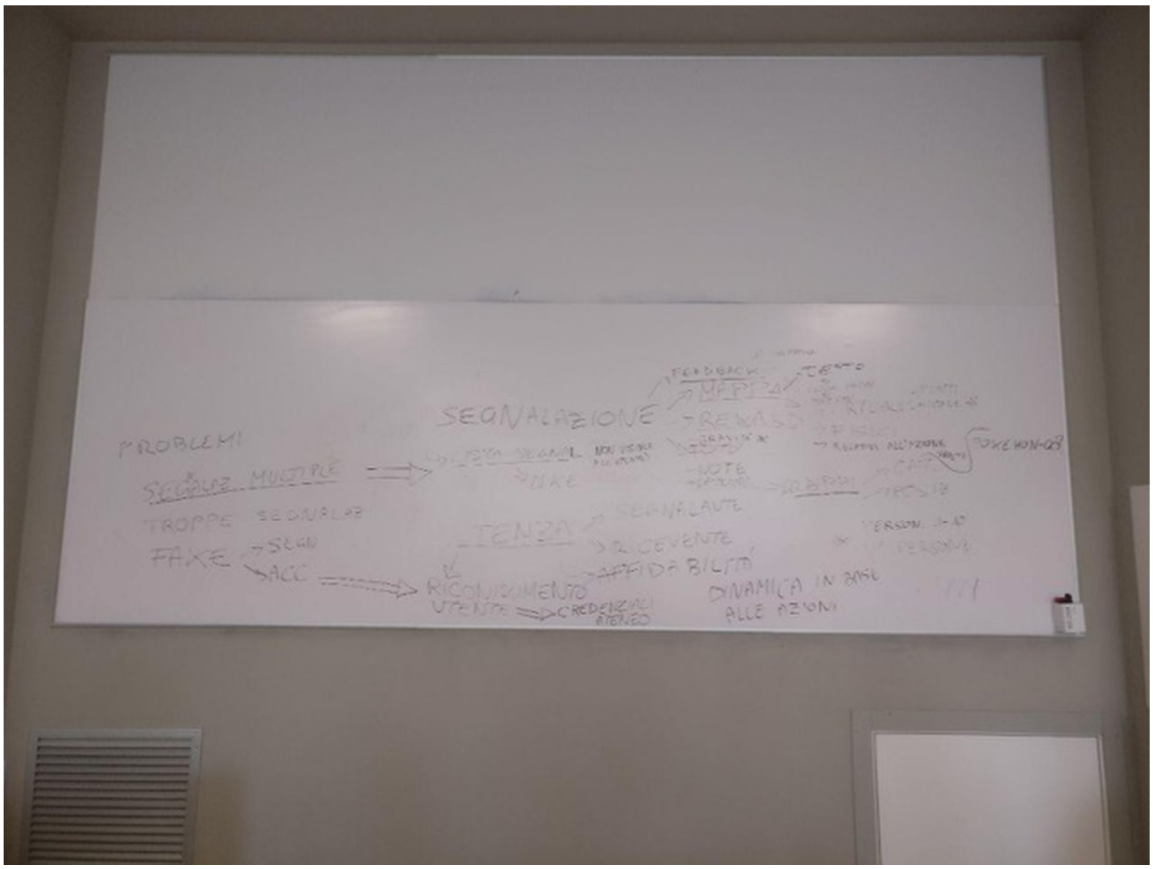


Immagine 4.5.2.1: Foto della lavagna su cui sono state raccolte le idee dei partecipanti

Dopo una breve pausa per permettere agli utenti di consolidare le proprie idee a seguito dell'attività di brainstorming è stata finalmente presentato il prototipo dell'applicazione che era stato prodotto (paragrafo 4.5.1): si è deciso di effettuare a questo punto un test di usabilità con la partecipazione di tutti i presenti, chiedendo loro un giudizio sull'aspetto grafico, se fossero in grado di riconoscere la funzionalità associata ai vari pulsanti e contenuti presentati, come avrebbero compiuto certe azioni (effettuare una

segnalazione, controllare lo stato di una segnalazione, osservare i dati relativi al proprio account); il software Balsamiq oltre alla produzione grafica dell'applicazione permette infatti anche di interagirvi, dal momento che è possibile associare link virtuali ai bottoni del prototipo. Gli utenti si sono rivelati soddisfatti della struttura generale del software e delle pagine "Home" e "Account", della quale hanno richiesto che le opzioni non fossero direttamente mostrate all'interno della pagina ma in una finestrella a comparsa; la pagina "Classifiche" è piaciuta, tuttavia il gruppo ha ritenuto che essendo presentata tramite uno dei bottoni della barra di navigazione principale risultasse fuorviante e che avrebbero preferito giungervi navigando tramite la pagina "Account". A questo punto è emersa una nuova idea ritenuta assai rilevante dalla platea, ovvero la possibilità di riassumere in una pagina tutte le segnalazioni effettuate dall'utente e il loro stato, così che l'utilizzatore non dovesse ricordare da sé quali aveva compiuto e andare a cercarle nella pagina "Home": continuando a chiedere opinioni è anche risultato che questa funzionalità era ritenuta importante e avrebbe potuto essere inserita nella barra di navigazione principale, al posto della precedente pagina "Classifiche".

A conclusione della riunione è stato infine distribuito un questionario cartaceo anonimo con un sistema a punteggio ed alcune domande sia in merito alla Focus Group stesso che all'applicazione presentata: le prime domande sono servite per capire i sentimenti dei partecipanti verso l'esperienza tenuta, la propensione a rispondere e con che affidabilità, mentre le seconde per raccogliere dati statistici sull'interesse verso l'applicativo e la propensione ad utilizzarlo. L'esito finale dalle risposte ottenute da 13 questionari viene mostrato di seguito. Per quanto riguarda l'incontro:

- il 53,8% si ritiene molto soddisfatto dell'incontro e il restante 46,2% soddisfatto;
- il 46,2% pensa che gli altri partecipanti siano rimasti soddisfatti, il 38,5% li ritiene molto soddisfatti e il 15,4% neutrali;
- il 69,2% ha trovato molto chiare le domande e il restante 30,8% chiare;
- il 46,2% ritiene di aver partecipato molto attivamente alle domande, il 7,7% di aver partecipato attivamente, il 15,4% di aver partecipato sufficientemente, il 23,1% di aver partecipato poco e il 7,7% di non aver partecipato a sufficienza;
- il 46,2% sostiene che le proprie idee saranno prese in considerazione, il 30,8% di aver fornito consigli preziosi, il 15,4% ritiene di essersi sentito ascoltato e il 7,7% di non sentirsi per niente preso in considerazione;

- l’84,6% vorrebbe partecipare nuovamente al Focus Group e il 15,4% lo desidera fortemente.

L’idea generale che emerge da questi dati confermano che gli studenti siano rimasti soddisfatti e abbiano partecipato volentieri e con molti interventi e che le loro idee influenzeranno il software da sviluppare, mentre qualcuno forse non è riuscito ad adeguarsi alla modalità eseguita per il Focus Group e pertanto ad esprimere le proprie opinioni; tuttavia il fatto che tutti desiderino partecipare nuovamente a questa attività lascia intendere che ad un’eventuale nuova sessione questa minoranza possa intervenire più attivamente.

In merito all’applicazione invece:

- il 100% ha capito molto chiaramente a cosa serva il sistema;
- il 53,8% trova che un sistema per il monitoraggio della struttura basata sul feedback degli utenti in generale sia utile e il 46,2% molto utile;
- il 61,5% trova che l’app che è stata loro presentata sia molto utile e il 38,5% l’ha trovata utile;
- il 38,5% si ritiene fortemente portato a scaricare tale app, il 30,8% lo scaricherebbe, il 30,8% è possibile che lo scarichi;
- il 61,5% userebbe questa app, il 15,4% la userebbe sicuramente, il 15,4% potrebbe utilizzarla in qualche occasione, il 7,7% non pensa la utilizzerebbe molto;
- il 61,5% ha capito molto bene le funzionalità della app, il 38,5% ha capito le funzionalità in generale;
- il 69,2% trova completa e funzionale questa applicazione, il 23,1% non aggiungerebbe altro, il 7,7% pensa che ci sia l’essenziale necessario.

Questi dati fanno pensare che gli utenti ritengano necessari sistemi di questo tipo all’interno dell’Università e che l’app che è stata pensata sarebbe utile per risolvere certi problemi ricorrenti; una buona parte degli utenti la scaricherebbe e la utilizzerebbe anche, inoltre l’interfaccia è stata ritenuta sufficientemente auto-esplicativa e il sistema pare contenere tutto quanto l’utente si aspetti di utilizzare a livello di funzionalità.

Pertanto si è ritenuto che il grado di completezza e usabilità del sistema fosse sufficiente e fosse possibile effettuare l’implementazione del software.

4. Implementazione del sistema

In questo capitolo ci si addentra adesso nella parte più tecnica dell'elaborato, presentando in che modo siano state effettivamente realizzate le scelte progettuali prese fino ad ora e seguendo un approccio di tipo manualistico, osservando le varie parti di codice realizzate.

Come stabilito in fase di analisi dei requisiti, l'applicazione che è stata creata è una *Progressive Web App* [IQUII19], ovvero un'applicazione web dall'aspetto di un regolare sito internet che può però implementare anche funzionalità tipiche di una normale applicazione, quale la possibilità di funzionare offline, ricevere notifiche push e l'accesso a proprietà hardware altrimenti precluse del dispositivo su cui funziona. Il termine "progressive" (in Italiano "graduale") si riferisce al fatto che dal punto di vista dell'utente, sebbene inizialmente percepite come normali siti web, si comportino in modo progressivo sempre più come una normale app. Si tratta inoltre di una applicazione multipiattaforma, disponibile sia su PC che dispositivi mobili, su sistemi operativi diversi e in grado di funzionare anche tramite browser differenti. Questo è permesso dal fatto che è possibile specificare all'applicazione in che modo comportarsi a seconda della piattaforma e del browser di lancio, definendo così un unico comportamento logico indipendente. Altri benefici per l'utente consistono in una migliore esperienza di navigazione, dal momento che il client possiede già i codici delle varie pagine e pertanto le transizioni da una all'altra risulteranno più fluide, e nel minore consumo di banda, dal momento che molte valutazioni vengono compiute a livello locale.

L'applicazione è anche una *Single Page Application*, ovvero un'applicazione che contiene di fatto un'unica pagina (nel caso in esame è stata chiamata "index.html") il cui contenuto cambia in maniera dinamica a seconda dell'interazione effettuata dall'utente: i vantaggi offerti da questa soluzione sono la transizione immediata dei contenuti (dal momento che il browser non deve caricare una pagina nuova ma solo modificare una parte di quella già costruita), una migliore gestione delle informazioni, le quali rimangono salvate in maniera locale e non devono essere recuperate ad ogni

successiva interazione, e di conseguenza anche un miglioramento delle performance e dell'esperienza dell'utente.

4.1. Scelta delle piattaforme di sviluppo

Prima di illustrare nei dettagli le varie componenti del sistema, si procederà con una descrizione dei framework software e dei linguaggi utilizzati per implementare il progetto, in quanto possono influenzare alcune scelte implementative.

4.1.1. Angular

Dovendo sviluppare un'applicazione di tipo Progressive Web e Single Page ci si è risolti a sfruttare il framework Angular 7.0 [ANG19], una piattaforma open-source per lo sviluppo di app client-side basata sul modello *Model-View-Controller (MVC)* [PIZ12]: questo permette allo sviluppatore di organizzare al meglio la pagina che andrà a codificare separando la logica dell'applicazione dai metodi per la sua resa grafica. Angular offre una propria interfaccia a linea di comando con alcune opzioni molto utili che automatizzano ad esempio la creazione di pagine, l'aggiunta di librerie, la configurazione dell'applicazione, la compilazione del programma. E' presente inoltre un sistema di moduli chiamati NgModules i quali descrivono blocchi di codice che suddividono il dominio dell'applicazione in segmenti logicamente coerenti. Ogni applicazione contiene almeno un modulo di base, detto AppModule, che serve ad effettuare l'avvio della stessa, in applicazioni ricche e complesse si può arrivare a ricorrere all'uso di svariati moduli (si veda Codice 5.1.1.1).

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from
'@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-
screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';

import { IonicStorageModule } from '@ionic/storage';
import { Camera } from '@ionic-native/camera/ngx';
```

```
import { AngularFireModule } from 'angularfire2';
import { AngularFirestoreModule, FirestoreSettingsToken }
from 'angularfire2/firestore';
import { AngularFireAuthModule } from 'angularfire2/auth';
import { AngularFireDatabaseModule } from
'angularfire2/database';
import { AngularFireStorageModule } from
'angularfire2/storage';
import { AngularFireMessagingModule } from
'@angular/fire/messaging';
import { AngularFireFunctionsModule } from
'@angular/fire/functions';
import { environment } from '../environments/environment';

import { AuthenticationService } from
'./services/authentication.service';
import { PersonalService } from
'./services/personal.service';
import { SignalService } from './services/signal.service';
import { FcmService } from './services/fcm.service';
import { ImageService } from './services/image.service';

import { SignalPageModule } from
'./pages/signal/signal.module';

import { RoomMalfunctionsComponent } from
'./components/room-malfunctions/room-
malfunctions.component';
import { SettingsComponent } from
'./components/settings/settings.component';

@NgModule({
  declarations: [AppComponent, RoomMalfunctionsComponent,
SettingsComponent],
  entryComponents: [RoomMalfunctionsComponent,
SettingsComponent],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    HttpModule,
    IonicModule.forRoot(),
    AppRoutingModule,
    IonicStorageModule.forRoot(),
    AngularFireModule.initializeApp(environment.fbconfig),
    AngularFirestoreModule,
    AngularFireAuthModule,
    AngularFireDatabaseModule,
    AngularFireStorageModule,
    AngularFireMessagingModule,
    AngularFireFunctionsModule,
    SignalPageModule
```

```
   ],
  providers: [
    AuthenticationService,
    PersonalService,
    SignalService,
    FcmService,
    Camera,
    ImageService,
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass:
IonicRouteStrategy },
    { provide: FirestoreSettingsToken, useValue: {} }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Codice 5.1.1.1: Struttura del file app.module.ts per il bootstrap dell'applicazione

Si possono notare alcune parole-chiave:

- *imports* specifica le classi importate da altri moduli;
- *declarations* indica alla app quali componenti si occupano del rendering grafico delle pagine;
- *providers* dichiara invece i servizi e li rende accessibili alle varie componenti della app.

Si possono osservare in particolare diversi moduli importati da angularfire2, la libreria ufficiale di Angular per l'interazione con la strumentazione offerta da Firebase, usati per l'autenticazione degli utenti, la comunicazione con il database di Firebase, il salvataggio e il recupero di dati dallo stesso, la creazione di token virtuali per la comunicazione asincrona di messaggi tra client e database; inoltre viene introdotto, tra gli altri, un modulo di Ionic (si veda prossimo sotto-capitolo) che contiene tutte le principali funzioni per usufruire della fotocamera di un dispositivo mobile.

Le Componenti in Angular sono le unità base per le interfacce grafiche, sono riconoscibili dal tag `@Component` e servono a gestire la logica di interazione con l'utente; all'interno di queste sono disponibili alcuni metodi utili per la loro gestione durante il ciclo di vita della app, per esempio `ngOnInit()` e `ngOnDestroy()`, che si attivano rispettivamente all'inizializzazione e alla distruzione della componente. Più avanti si vedranno nello specifico alcune delle componenti create per il software in esame.

I Servizi sono invece classi con un preciso compito per la gestione di dati o caratteristiche della app, dal momento che le componenti non dovrebbero occuparsi di altro che non sia gestire la logica dell'interazione per l'utente. Dichiarando i servizi a livello di bootstrap (Codice 5.1.1.1), tutte le componenti avranno accesso alla stessa istanza di servizio: questo è anche un metodo efficace per condividere i dati elaborati da quel servizio tra tutte le classi dell'applicazione.

Un'altra proprietà molto utile e spesso utilizzata all'interno dell'applicazione è lo sfruttamento delle direttive strutturali, ovvero delle componenti che estendono il codice HTML con delle etichette e alcuni attributi personalizzati, utilizzati per manipolare il DOM: per esempio `*ngIf` crea un elemento se si verifica la condizione che viene specificata, mentre `*ngFor` crea tanti elementi quanti quelli indicati in un certo insieme.

Il Template infine specifica in che modo debba essere renderizzata la componente e rappresenta dunque la Vista del modello MVC: si tratta di un meccanismo di resa grafica molto potente, soprattutto se associato ai metodi di *data binding* di Angular: questi comunicano al DOM i valori da mostrare che vengono elaborati dal Controllore.

4.1.2. Ionic

Un'altra piattaforma software open-source per lo sviluppo di applicazioni mobili ibride a lato client sia mobili che desktop alla quale si è fatto ricorso è Ionic 4.0 [ION19], che si basa su Angular, di cui ne estende le funzionalità, e su Cordova oppure Capacitator, i cui plugin permettono l'accesso a numerose funzionalità del sistema operativo ospitante l'applicativo.

Il compilatore di Ionic permette di selezionare per quale piattaforma sviluppare l'applicazione, rendendo più facile e veloce la distribuzione sui vari sistemi operativi come Android, iOS o Windows. L'interfaccia a linea di comando di questo framework consente di creare e aggiungere nuove pagine, servizi, componenti o librerie o effettuare test ed emulazioni sui vari sistemi in maniera semplice e immediata; inoltre sono a disposizioni diverse API specifiche per lo sviluppo mobile, quali moduli che si occupano della gestione della macchina fotografica, della geolocalizzazione del dispositivo o della componente 3D Touch. Sono garantite alte performance nel

rendering dinamico delle pagine, sia grazie al caricamento solo su necessità delle componenti richieste sia alla minimizzazione della manipolazione del DOM.

Un potente strumento che Ionic mette a disposizione dello sviluppatore sono le sue Componenti, un vasto insieme di elementi pre-impostati che costituiscono uno standard nelle applicazioni moderne, basati su codice HTML, CSS e JavaScript ma fortemente personalizzabili, con i quali creare il design della propria applicazione e garantiti per funzionare nei diversi sistemi operativi e su browser differenti. Tra queste componenti vi è anche una vasta gamma di icone, di cui si fa largo uso nelle applicazioni mobile ma anche di tipo desktop.

Un'altra caratteristica di Ionic è la possibilità di ricorrere a classi predefinite di CSS per la tematizzazione dell'applicazione, che rimangono consistenti su tutti i diversi dispositivi e che lasciano allo sviluppatore la possibilità di propagare velocemente le modifiche all'aspetto della app.

4.1.3. Firebase

Un'altra indispensabile necessità del sistema era la presenza di un database dove salvare lo storico dei dati, sia per dare la possibilità di effettuare analisi a posteriori sulla mole di informazioni che si sarebbe formata sia per mostrare agli utenti alcuni messaggi essenziali, quali i progressi personali o lo stato delle segnalazioni e quello dei disservizi internamente alla struttura. La scelta è ricaduta su Firebase [FIR19a] a seguito di una serie di considerazioni.

Innanzitutto Firebase è uno dei più potenti ma anche popolari servizi disponibili online in cloud che offrono soluzioni per la costruzione e l'amministrazione di database, mettendo a disposizione un ricco insieme di servizi per il back-end: questo ha permesso di sviluppare il lato server del sistema in maniera estremamente più veloce per concentrarsi quindi maggiormente sulla parte client, avendo comunque la garanzia di un sistema sicuro e completo: tra i servizi utilizzati all'interno dell'applicazione ad esempio vi sono *Firebase Cloud Messaging* per la gestione delle notifiche push, *Firebase Auth* per garantire l'accesso verificato agli utenti, *Firebase Cloud Firestore* per la costruzione del database e *Firebase Storage* per il salvataggio e il download di file, usato in particolare per mantenere le immagini delle segnalazioni e le grafiche dei badge. Inoltre la piattaforma è costruita sull'infrastruttura offerta da Google e perciò si

ridimensiona in maniera automatica anche su larga scala in base al contratto stipulato: per l'attuale uso dell'applicazione i limiti imposti dalla versione *Spark*, che è gratuita, sono più che sufficienti, ed hanno evitato di considerare i costi legati a macchine virtuali o fisiche. Un'altra caratteristica notevole è il fatto che il database abbia la capacità di salvare e sincronizzare i dati in real-time, molto importante per permettere agli utenti e ai tecnici di ricevere immediatamente informazioni sullo stato dei disservizi. In aggiunta a ciò Firebase permette di definire delle funzioni in cloud che si attivano in maniera automatica in risposta a certi eventi relativi al database, quali la creazione, l'aggiornamento o la cancellazione di file e documenti del database: in questo modo è possibile gestire tali elementi senza dover implementare un ulteriore server per tale compito, come verrà mostrato successivamente nel sotto-capitolo 5.3.5. Infine Ionic ha iniziato dalla versione 2.0 a garantire il supporto verso Firebase, fintanto che queste due piattaforme sono diventate una delle combinazioni più utilizzate nello sviluppo di app ibride e mobili, consentendo un avvio online dei sistemi molto veloce e semplificato.

E' da tenere da conto il fatto che Firebase sia un database NoSQL, ovvero non si fonda su tabelle collegate da schemi relazionali né vi sono record; la struttura dati alla base del framework è invece un albero indicizzato di documenti JSON-like con coppie del tipo *(nome, valore)*, per cui occorre fare attenzione a non creare alberi eccessivamente profondi per mantenere le performance alte; tuttavia la possibilità di scalare a fronte di un consistente numero di utenti e di segnalazioni e la possibilità di mantenere il controllo sulla forma dei dati, che può evolvere velocemente in progetti di questo tipo, rappresentano ulteriori punti a favore di questa scelta.

4.1.4. Express e Node.js

Come spiegato precedentemente, la tecnologia per l'implementazione della mappa interattiva era già stata studiata ed è stata adattata parzialmente per poter inserire questa funzionalità all'interno del progetto. Essa si fonda su un server che fornisce i dati da inserire nella pagina e gli script necessari per il rendering grafico della mappa al browser. Tale server è stato costruito utilizzando Express [EXP19], un software open-source gratuito che costituisce lo standard de facto per il supporto allo sviluppo di applicazioni web basate su Node.js.

Node.js [NOD19a] è un ambiente a run-time multipiattaforma e open-source per l'esecuzione di script JavaScript al di fuori del browser, utilizzato per scrivere codice da

eseguire lato server. Costruito sul motore JS V8 di Google Chrome, Node permette ad esempio la produzione del contenuto di pagine Web dinamiche prima che questa venga inviata al Browser dell'utente; è ottimo per applicazioni web di tipo Real-time e grazie alla sua architettura orientata agli eventi, che rende possibile lo scambio di input/output asincrono, punta ad ottimizzare le performance di comunicazione e la scalabilità del sistema. Inoltre Node contiene Node Package Manager [WIKI-NPM19], considerato "il più grande eco-sistema di librerie open source al mondo" [NOD19b].

4.2. Linguaggi di sviluppo

In questa sezione si vuole offrire una breve panoramica sui linguaggi utilizzati per codificare i vari elementi del sistema.

4.2.1. TypeScript

Per la scrittura del codice lato client e delle funzioni cloud di Firebase ci si è avvalsi di TypeScript [GIT19a]. Si tratta di un superset di JavaScript, che ne va ad estendere le funzionalità seguendo le specifiche ECMAScript 6 e 7 [WIKI-ECMA19] e viene poi compilato in puro JS per il suo utilizzo.

Tra i vantaggi vi è sicuramente la possibilità di usare il compilatore per individuare già a tempo di compilazione eventuali errori, inoltre uno script in JavaScript continua a essere accettato dal compilatore. Viene aggiunta la tipizzazione statica, ovvero è possibile definire il tipo di una variabile per effettuarne un controllo, anche se è possibile rinunciare a questa caratteristica; da questo è possibile definire anche il tipo dei parametri in ingresso e dei valori di ritorno di una funzione; si possono usare parametri opzionali o con valori di default ed effettuare l'*overloading* delle funzioni; viene esteso il supporto verso la programmazione ad oggetti potendo dichiarare interfacce e classi; sono supportati i costrutti di *namespace* e moduli, garantendo una migliore gestione del codice a fronte di applicazioni sempre più grandi e complesse; è previsto l'uso di funzioni asincrone, di funzioni di *callback* nonché l'uso del costrutto delle *Promise.*

4.2.2. HTML5

Per l'organizzazione del contenuto delle pagine web, Ionic si basa su HTML5 [W3C19], che rappresenta uno standard per i principali produttori di browser (Apple,

Google, Mozilla e Microsoft). Questo linguaggio di markup è il più idoneo alla scrittura di applicazioni e siti web complessi o di tipo mobile e multipiattaforma, dal momento che:

- introduce nuovi elementi di marcatura tipicamente utilizzati nei siti moderni, quali `<header>`, `<footer>`, `<svg>`, `<canvas>`, `<audio>`, `<video>`, nonché attributi di controllo per le form come `number`, `date` e `range` (inoltre rimuove elementi considerati ormai deprecati);
- supporta gli elementi SVG e MathML grazie ai tag `<svg>` e `<math>`;
- specifica nuove API che possono essere utilizzate con JavaScript, ad esempio Web Storage e Indexed Database per memorizzare dati localmente così che eventuali interruzioni di connessioni non affettino il funzionamento del software, GeoLocation per la condivisione della posizione del dispositivo, WebRTC per la comunicazione in RealTime;
- semplifica la sintassi precedente, a fronte di una maggiore flessibilità.

4.2.3. CSS4 & SASS

Ionic 4 utilizza sia SASS [SASS19] che CSS [MDN19] per definire lo stile di formattazione degli elementi grafici delle applicazioni o dei siti web.

SASS è un pre-processore di CSS, ovvero offre una forma dichiarativa più completa e potente ma al contempo più semplice, quindi genera un file CSS, l'unico formato interpretabile dal browser. SASS offre la possibilità di definire variabili tipizzate (coi tipi `Number`, `String`, `Color` e `Boolean`), nonché di nidificare blocchi di codice (semplificando così la stilizzazione di elementi interni ad altri); permette inoltre di iterare tramite variabili grazie ai costrutti `@for`, `@each` e `@while` e di passare il valore di variabili come argomento. Alla creazione di una nuova pagina o componente del programma, viene creato anche un relativo file .scss per la sua stilizzazione.

Le componenti di Ionic 4 comprendono già una propria stilizzazione base, tuttavia personalizzabile. Questa tematizzazione è raggiunta grazie all'uso delle variabili di CSS4: essendo CSS supportato in maniera nativa dai browser, tali variabili possono essere modificate a run-time, senza intervento di nessun compilatore, inoltre seguono la classica imposizione a cascata di un foglio CSS. Ionic offre anche un pacchetto di funzioni per CSS utili a modificare testi, riposizionare gli elementi o aggiustarne il

contenitore virtuale. Una tecnica molto utilizzata per definire stili differenti a seconda del tipo di dispositivo o di altri fattori (quali le dimensioni dello schermo) è l'uso della regola `@media`, che si rivela particolarmente valida dal momento che lo stesso applicativo dovrà essere visualizzato su differenti piattaforme o dispositivi.

4.3. Elementi principali del sistema

Si vuole offrire in questa parte una panoramica sulle maggiori componenti realizzate durante la costruzione del sistema, cercando di far emergere in che modo quanto era stato precedentemente analizzato e progettato è stato quindi implementato via codice. Si inizierà dal lato client (sezioni dalla 5.3.1 alla 5.3.3), quindi si procederà con il server per la mappa interattiva, le funzioni cloud di Firebase e infine la struttura dei dati sul database online. Per questioni di spazio e leggibilità non verranno presentati interamente i blocchi di codice, ma soltanto quanto ritenuto sufficiente ai fini di comprensibilità.

4.3.1. Struttura generale

Come accennato precedentemente (inizio del capitolo 5), trattandosi di una Single Page Application vi è un'unica pagina HTML denominata `index.html`, visibile in Codice 5.3.1.1. In essa vi è infatti contenuto il selettore `<app-root>`, che indica ad Angular a quale pagina associare il bootstrap dell'applicazione che era stato indicato nel file `app.module.ts`.

```
<!DOCTYPE html>
<html lang="en">

<head>

[...]

<meta name="keywords" content="mall map, 3d, css,
javascript,pin, levels, floor map" />
<meta name="author" content="Codrops" />

<script src="http://localhost:3000/js/modernizr-custom.js">
</script>
<script src="http://localhost:3000/socket.io/socket.io.js">
</script>
<script src="https://code.jquery.com/jquery-2.1.4.min.js">
</script>
```

```
<script
src='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.0.0-
alpha1/jquery.min.js'></script>
<script src="//code.jquery.com/ui/1.11.2/jquery-
ui.js"></script>

<script
src="https://code.highcharts.com/highcharts.js"></script>
<script
src="https://code.highcharts.com/modules/heatmap.js">
</script>

  <script>
    var socket =
io('http://localhost:3000/mySensorNamespace');
  </script>
  <script src="assets/js/socketClient.js"></script>

</head>

<body>
  <app-root></app-root>

  <script
src="http://localhost:3000/js/classie.js"></script>
  <script
src="http://localhost:3000/js/list.min.js"></script>

</body>
</html>
```

Codice 5.3.1.1: Contenuto del file index.html

In particolare si possono notare i diversi script che sono stati inseriti per poter utilizzare la mappa interattiva: in questa versione base essi vengono consegnati all'applicazione tramite un indirizzo di server locale che tuttavia segue lo stesso formato del server già attivo all'interno della struttura del Campus universitario di Cesena; cambiando soltanto tale indirizzo dunque l'applicazione sarebbe in grado di funzionare immediatamente all'interno dell'edificio. Oltre che per facilitare l'integrazione col sistema già installato e funzionante nel campus, questa scelta è stata fatta con l'obiettivo di snellire il codice: eventuali aggiornamenti nel server si rifletterebbero immediatamente sull'applicazione, senza dover ricompilare quei file sul client, inoltre il peso del programma è ovviamente minore. Un solo script è incluso nativamente nel software, ovvero il file `socketClient.js`: sebbene basato sull'originale, è stato necessario apportare alcune modifiche essenziali al funzionamento della app.

La prima cosa che al browser viene indicata è il caricamento della pagina “tabs”: essa contiene una componente di Ionic, `<ion-tabs>`, che si comporta come la direttiva Router di Angular, permettendo cioè la navigazione tra pagine diverse, accessibili tramite dei bottoni (si veda Codice 5.3.1.2): questi indicano le tre principali pagine che erano state decise alla fine della fase di progettazione, ovvero “Home” contenente la mappa interattiva, “Account” con le informazioni personali dell’utente e “Mie segnalazioni”, per accedere alla lista delle segnalazioni effettuate dall’utente.

```
<ion-tabs>
  <ion-tab-bar>

    <ion-tab-button tab="home">
      <ion-icon name="sunny"></ion-icon>
      <ion-label>Home</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="account">
      <ion-icon name="moon"></ion-icon>
      <ion-label>Account</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="my-signals">
      <ion-icon name="list-box"></ion-icon>
      <ion-label>Mie segnalazioni</ion-label>
    </ion-tab-button>

  </ion-tab-bar>
</ion-tabs>
```

Codice 5.3.1.2: Contenuto di tabs.page.html

Il tag `tab` fa riferimento al file `tabs.module.ts`, dove sono specificati a quali percorsi per le pagine da caricare fanno riferimento i valori `home`, `account` e `my-signals` (Codice 5.3.1.3). Cliccando quindi su un bottone della barra di navigazione si verrà re-indirizzati alla pagina relativa: essendoci comunque una sola pagina, quello che accade è che il browser carichi il contenuto della pagina selezionata all’interno del DOM e quindi mostri solo quella parte; cliccando su un altro pulsante dunque cambia la vista dei contenuti ma questi non vengono cancellati dal documento HTML, in questo modo la navigazione tra pagine evita ogni volta l’onere di effettuare caricamenti.

```
[...]
const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children : [
      {
        path: 'home',
        loadChildren: '../home/home.module#HomePageModule'
      },
      {
        path: 'account',

loadChildren:'../account/account.module#AccountPageModule'
      },
      {
        path: 'rank',
        loadChildren: '../rank/rank.module#RankPageModule'
      },
      {
        path: 'signal',
        loadChildren:
'../signal/signal.module#SignalPageModule'
      },
      {
        path: 'my-signals',
        loadChildren: '../my-signals/my-
signals.module#MySignalsPageModule'
      }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/home',
    pathMatch: 'full'
  }
];
[...]
export class TabsPageModule {}
```

Codice 5.3.1.3: dichiarazione delle rotte di navigazione della pagina Tabs

4.3.2. Servizi

E’ stato possibile individuare almeno 5 diversi tipi di servizi che sarebbero stati utili al fine sia di aver una migliore organizzazione del codice sia per non dover replicare blocchi identici tra pagine differenti; inoltre, come spiegato in 5.1.1, dichiarando i servizi in fase di bootstrap l’applicazione avrebbe creato un’unica istanza di tali elementi condivisa tra tutte le pagine, così da non dover implementare anche un

ulteriore metodo per comunicare i dati condivisi. Grazie all'uso della libreria angularfire2 [GIT19b], che contiene diversi moduli con funzioni per permettere la comunicazione tra le applicazioni in Angular e i database in Firebase, il lavoro svolto è risultato notevolmente semplificato.

Il primo servizio prodotto è il servizio di autenticazione dell'utente, contenuto nel file `authentication.service` (Codice 5.3.2.1), dove sono indicati tutti i metodi utili all'utente per la gestione del proprio account:

- `signup()` permette di creare un nuovo utente con una email e una password;
- `isAuthenticated()` risolve lo stato di autenticazione dell'utente;
- `getUser()` permette di accedere alle informazioni dell'utente, ritornando un oggetto di tipo User della libreria firebase;
- `login()` e `logout()` permettono rispettivamente di aprire e chiudere una sessione dove l'utente è autenticato tramite la mail e la password che aveva impostato precedentemente;
- `update()` consente di aggiornare i campi del nome utente e della foto associabili all'account.

```
export class AuthenticationService {

  private user: Observable<firebase.User>;
  private authenticated: BehaviorSubject<boolean>;

  constructor(private fbAuth: AngularFireAuth) {
    this.user = fbAuth.authState;
    this.authenticated = new
BehaviorSubject<boolean>(false);
  }

signup (email: string, password: string): Promise<any> {
    return new Promise ((resolve, reject) => {
            this.fbAuth.auth.
                  createUserWithEmailAndPassword(email,
password)
                  .then( value => {
                         console.log('Utente creato con
                                       successo!', value);
                         resolve({success: true});
                  })
                  .catch(error => { console.log('Qualcosa è
                        andato storto: ', error.message );
                                    reject(error);
```

```
                });
    });
  }

  isAuthenticated (): Observable<boolean> {
    return this.authenticated.asObservable();
  }

  getUser (): Observable<firebase.User> {
    return this.user;
  }

  logout () {
    this.fbAuth.auth.signOut();
    this.authenticated.next(false);
  }

  login (email: string, password: string) {
    this.fbAuth.auth
    .signInWithEmailAndPassword(email, password)
    .then ( () => {
      this.authenticated.next(true);
      console.log('Loggato!');
    })
    .catch (error => { console.log('Qualcosa è andato
                                  storto ', error.message);
    });
  }

  update (newName?: string, newPhoto?: string) {
    if (!this.authenticated) {
      console.log('Devi effettuare prima il login!');
    } else {
      this.fbAuth.auth.currentUser.updateProfile({
          displayName: newName,
          photoURL: newPhoto
      });
    }
  }
```

Codice 5.3.2.1: servizio di autenticazione dell'utente

I servizi SignalService e PersonalService servono per creare e recuperare sul database in cloud i documenti con le informazioni rispettivamente inerenti le segnalazioni effettuate e i dati personali degli utenti. Tali servizi presentano una logica estremamente simile quindi ne verrà mostrato soltanto uno a titolo esemplificativo. Il servizio prima di tutto recupera la collezione di documenti segnata sotto al percorso che gli viene specificato (`db.collection<any>(this.itemsPath`), quindi rimane in ascolto di una

risposta dal database: ogni qualvolta avviene una modifica a tale collezione, grazie al metodo `snapshotChanges()` viene recuperato il nuovo insieme e quindi tradotto in un formato dati locale. PersonalService si occupa sia dei documenti con le segnalazioni ancora attive (percorso "segnalazioni"), sia di quelle già risolte (percorso "risolte"). Viene utilizzato anche il servizio AuthenticationService per il recupero dell'id dell'utente che verrà associato alle segnalazioni che effettuerà. I metodi qui contenuti sono:

- `getSignals()` per il recupero di tutte le segnalazioni attive nell'edificio;
- `getSignal()` per il recupero di una specifica segnalazione, tramite il suo id;
- `getSolved()` per il recupero di una specifica segnalazione risolta;
- `updateSignal()` per aggiornare i campi del documento di una segnalazione attiva;
- `addSignal()` per aggiungere una nuova segnalazione;
- `removeSignal()` per eliminare una segnalazione attiva tramite l'id del documento.

```
export class SignalService {

private signalsCollection: AngularFirestoreCollection<any>;
private signals: Observable<any[]>;
private solvedCollection: AngularFirestoreCollection<any>;
private solved: Observable<any[]>;
private userId: string;
private itemsPath = 'segnalazioni';
private solvedPath = 'risolte';

constructor (public db: AngularFirestore,
            private authServ: AuthenticationService) {
   this.signalsCollection =
db.collection<any>(this.itemsPath);
   this.signals =
this.signalsCollection.snapshotChanges().pipe(
     map (actions => {
       return actions.map ( a => {
           const data = a.payload.doc.data();
           const id = a.payload.doc.id;
           return {id, ...data };
           }
         );
       }
     )
   );
```

```
   this.solvedCollection =
                      db.collection<any>(this.solvedPath);
   this.solved =
    this.solvedCollection.snapshotChanges().pipe(
    map (actions => {
      return actions.map ( a => {
          const data = a.payload.doc.data();
          const id = a.payload.doc.id;
          return {id, ...data };
          });
      })
  );
  this.authServ.getUser().subscribe( user => {
     if (user) { this.userId = user.uid; }
     }
   );
 }

getSignals () {
   return this.signals;
 }

getSignal(id) {
   return
this.signalsCollection.doc<any>(id).valueChanges();
 }
getSolved(id) {
  return this.solvedCollection.doc<any>(id).valueChanges();
}

updateSignal(item: any, id: string) {
   return this.signalsCollection.doc(id).update(item);
 }

addSignal(item: any) {
   return this.signalsCollection.add(item);
 }

removeSignal(id) {
   return this.signalsCollection.doc(id).delete();
 }
}
```

Codice 5.3.2.2: servizio per la gestione dei dati delle segnalazioni

Una notifica push è un messaggio che viene portato all'attenzione dell'utente senza che questo ne attivi in maniera diretta il suo invio e ne esistono di due tipi principali [WIKI-PT19]: le notifiche locali vengono programmate dal software stesso che rimane attivo in

background, mentre quelle remote vengono inviate all'applicazione da un server collegato tramite la rete. Il servizio FCMService si occupa di implementare le notifiche push di tipo remoto che vengono inviate dal database di Firebase in modo automatico in risposta a certi eventi (si vedrà come nel capitolo 5.3.5). Per ricevere una notifica occorre avere un token, associato al browser di uno specifico device: questo viene ottenuto tramite il metodo `getPermit()`. Un utente può ricevere o notifiche create appositamente per lui (ad esempio quando una sua segnalazione viene presa in carico o risolta) oppure generiche, se iscritto ad un topic (grazie al metodo `sub()`il personale tecnico può iscriversi al topic "Nuove segnalazioni" così da essere avvisato ogni qualvolta se ne presenta una, oppure cancellarsi tramite il metodo `unsub()`). Il servizio si occupa anche di ricevere il messaggio e quindi mostrarlo all'utente (rispettivamente tramite i metodi `showMessages()` e `makeAlert()`), pure se l'applicazione è in background. Anche in questo caso l'utilizzo dei moduli della libreria di firebase ha aiutato nello sviluppo del blocco.

```
export class FcmService {

  constructor (
              private afMessaging: AngularFireMessaging,
              private fnc: AngularFireFunctions,
              private alertCtrl: AlertController
  ) {}

  async makeAlert(title, message) {
      const alert = await this.alertCtrl.create({
            header: title,
            message,
            buttons: ['OK']
      });
      alert.present();
      setTimeout( () => alert.dismiss(), 8000);
  }

getPermit() {
        return this.afMessaging.getToken;
  }

  showMessages() {
      return this.afMessaging.messages.pipe(
          tap ( msg => {
              const body: any = (msg as
                                    any).notification.body;
              const title = (msg as
                                   any).notification.title;
```

```
                this.makeAlert(title, body);
            })
        );
    }

sub( signals, token: string) {
        for (const signal of signals) {
            this.fnc.httpsCallable('subscribe2Signal')
                        ({signal, token: token }).subscribe();
        }
        this.makeAlert('', 'Riceverai una notifica sulle tue
                                                segnalazioni!');
    }

unsub( signals, token: string ) {
        for (const signal of signals) {
            this.fnc.httpsCallable('unsubscribeFromSignal')
                ({signal, token: token }).pipe().subscribe();
        }
        this.makeAlert('', 'Le notifiche sono state
                                                    sospese');
    }
}
```

Codice 5.3.2.3: servizio per le notifiche push

L'ultimo servizio presentato è quello per le immagini dell'applicazione: esse possono essere gli avatar personali dell'utenti, le immagini dei disservizi da segnalare oppure le immagini dei badge collezionati da un account. Il servizio si basa su due moduli: quello di Ionic per la fotocamera, che permette di scattare una foto o caricarla dalla libreria del dispositivo, e quello di firebase per l'accesso allo *storage* in cloud, una porzione del database utile per caricare file statici. Le immagini catturate dal dispositivo verranno caricate e scaricate tramite quest'ultimo servizio. I metodi che ImageService (si confronti Codice 5.3.2.4) mette a disposizione sono:

- `takePhoto()`, permette di prendere una foto secondo una serie di opzioni e quindi la invia allo storage online, l'opzione di base è di scattare la foto;
- `loadPhoto()` serve per caricare la foto dalla memoria del dispositivo, si avvale del metodo precedente passandogli la nuova opzione;
- `downloadImage()` serve a recuperare la foto dal cloud.

```
export class ImageService {

  constructor( private camera: Camera,
               private storage: AngularFireStorage ) { }

  async takePhoto(refId: string, source?: number ) {
    let sourcePh;
    source ? sourcePh = source : sourcePh = 1;
    try {
      const options: CameraOptions = {
          quality: 50,
          destinationType:
                      this.camera.DestinationType.DATA_URL,
          encodingType: this.camera.EncodingType.JPEG,
          mediaType: this.camera.MediaType.PICTURE,
          sourceType: sourcePh
      };

      const result = await this.camera.getPicture(options);
      const image = '' + result;
      const picture = this.storage.ref(refId);
      picture.putString(image, 'data_url');
    } catch (e) {}
  }

  loadPhoto(userId: string) {
    this.takePhoto(userId, 0);
  }

  downloadImage( ref: string, image: string ) {
    return this.storage.ref(ref + image).getDownloadURL();
  }
}
```

Codice 5.3.2.4: servizio per la gestione delle immagini dell'applicazione

4.3.3. Pagine

Ogni qualvolta viene generata una nuova pagina tramite il comando `generate` della CLI di Ionic, la piattaforma in automatico crea cinque file: tre in TypeScript, rispettivamente per la dichiarazione dei moduli da usare nella componente (formato `.module.ts`), per la descrizione della gestione della logica applicativa di quella porzione di sito (`.page.ts`) e per effettuare unità di test (`.spec.ts`); uno è un file in SASS (`.page.scss`) per la stilizzazione della pagina; infine l'ultimo contiene il contenuto HTML che verrà inserito all'interno del DOM (`.page.html`).

Per tutte le funzionalità richieste dal sistema è stato necessario creare quattro pagine e tre componenti, oltre alla pagina “tabs” spiegata precedentemente in 5.3.1. In Angular esiste solo il concetto generale di “componente”, mentre Ionic, ricordando al lettore che viene costruito sulla base del framework precedente, aggiunge il termine “pagina”, differenziandolo leggermente: nei termini di Angular entrambi continuano a essere delle componenti; in Ionic invece una “pagina” è una componente con cui l’utente interagisce e costituisce una vista a cui vengono aggiunti dei metodi relativi al modo in cui questi può interagire con la pagina [STA18]; infine la “componente” in Ionic risulta invece una parte della pagina, la quale può essere costituita da una serie di componenti anche nidificate tra loro. Pertanto implementando l’applicativo si è deciso di costituire la pagina “tabs”, che rappresenta la vista generale con cui l’utente deve sempre poter interagire; le altre pagine rappresentano una vista locale su certe funzionalità; le componenti aggiungono dei template di codice aggiuntivo alla pagina sulla quale sono innestati. L’immagine 5.3.3 rappresenta tale struttura.

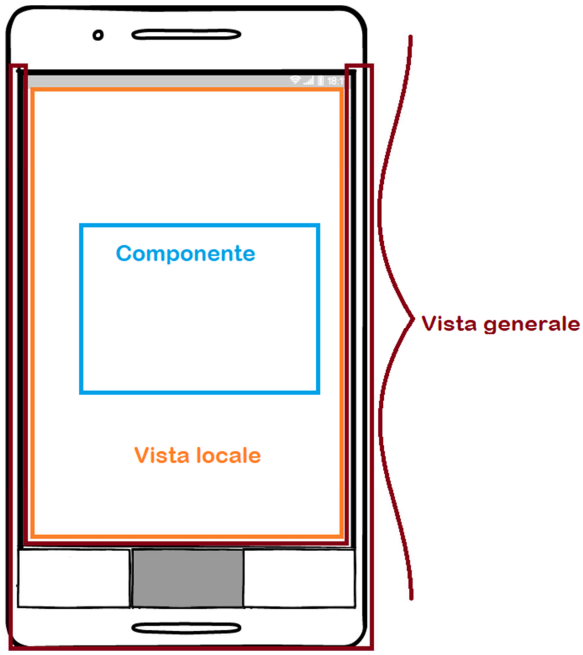


Immagine 5.3.3.1: struttura generica dell’interfaccia della app

La pagina “Home” contiene il codice per visualizzare e interagire con la mappa ed effettuare le segnalazioni. Il documento HTML contiene principalmente il codice in SVG per realizzare il disegno base dei vari piani dell’edificio (Immagine 5.3.3.2): tutti gli elementi aggiuntivi per disegnarvi sopra le varie stanze, effettuarne il rendering grafico e inserire le funzionalità per interagirvi sono incorporati a run-time attraverso i vari script di cui si era parlato in 5.3.1.

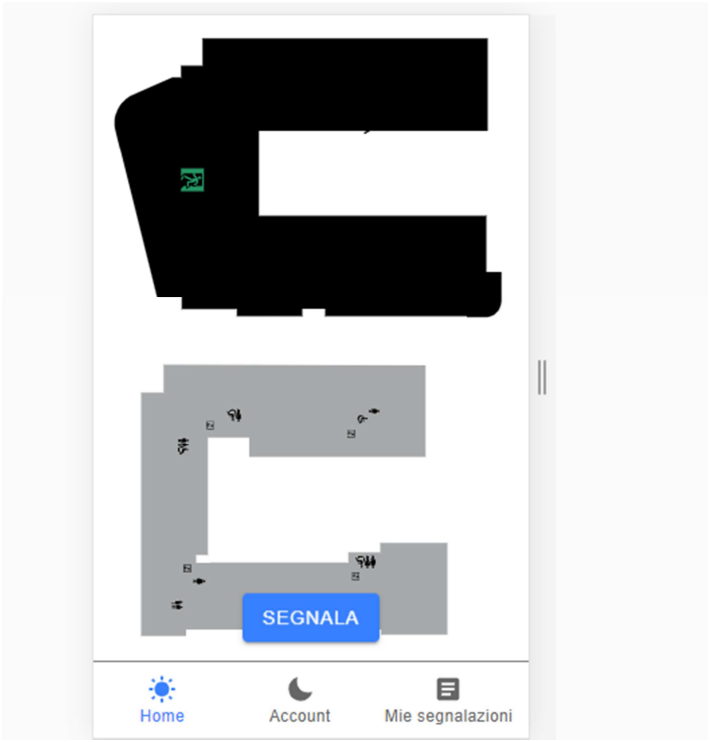


Immagine 5.3.1.2: mappatura base dell'edificio con server irraggiungibile

Alla creazione della pagina l'applicazione si iscrive a due metodi di tipo `Observable` (facente parte della libreria RxJS [RYL18]), ovvero `getUser()` del servizio AuthenticationService e `getSignals()` da SignalService (si confronti con Codice 5.3.3.3), i quali ritornano un valore in maniera asincrona ogni qualvolta viene risolta la loro funzione: in questo modo tutte le volte che l'utente effettuerà l'accesso o il logout oppure sarà presente una nuova segnalazione nel database in Cloud la pagina riceverà in aggiornamento un oggetto contenente le appropriate informazioni. In questa pagina dell'utente viene recuperato il nome associato all'account per poterlo visualizzare in alto; se l'utente non ha effettuato l'accesso sarà trattato come uno stato anonimo, dandone avviso e invitandolo a effettuare l'autenticazione per poter segnalare qualcosa. L'oggetto recuperato dal metodo `getSignals()` contiene l'insieme di tutte le segnalazioni ancora attive presenti sul database: per ognuna di esse viene creato un nuovo nodo HTML ben strutturato inserito nel DOM, in modo tale da recuperare da questo tutte le informazioni necessarie in seguito piuttosto che richiamare il server ogni volta e da funzionare anche in mancanza del segnale della rete.

```
ngOnInit() {
    this.subAuth = this.auth.getUser().subscribe ( user =>
{
      if ( user ) {
        this.userInfo = user.displayName;
      } else { this.userInfo = ''; }
    });
    this.subSign = this.signServ.getSignals().subscribe (
signals   => {
      for (let i = 1; i < this.levelNumber; i++) {
        document.getElementById('level__malfunctions_' +
i).innerHTML = '';
      }
      this.malfunctions = [];
      for (const malfunction of signals) {
        this.malfunctions.push(malfunction);
        // Nodo malfunction
        const malfunction_node =
                                document.createElement('a');
        malfunction_node.setAttribute('id',malfunction.id);
        malfunction_node.classList.add('malfunction');
        malfunction_node.setAttribute('nome',
                                        malfunction.name);
        malfunction_node.setAttribute('type',
                                        malfunction.type);
        malfunction_node.setAttribute('description',
                                    malfunction.description);
        malfunction_node.setAttribute('status',
                                       malfunction.status);
        malfunction_node.setAttribute('aria-
                                     label','MALFUNCTION');
[...]

}

[...]

async presentRoomPopover ( name: string) {
    const popover = await this.popCtrl.create({
      component: RoomMalfunctionsComponent,
      componentProps: {
        'malfunctions' : this.malfunctions,
        'roomName' : name
      }
    });
    return await popover.present();
}
```

```
async openSignalModal() {
    if (document.getElementsByClassName('pin—
                                    active').length > 0){
      this.getUserPosition();
      const modal = await this.modalCtrl.create({
        component: SignalPage,
        backdropDismiss: true,
        componentProps: {
          'userPosition' : this.userPosition,
          'catPosition' :  this.catPosition,
          'levPosition' : this.activeLevel,
          'stylePosition' : this.positionStyle,
          'signals' : this.malfunctions
        }
      });
      modal.present();
      modal.onDidDismiss().then(signaled => {
        if (signaled.data) {
            this.presentThanksToast();
        }
      });
      return;
    } else {
      this.presentWarningToast();
      return;
    }
  }
```

Codice 5.3.3.3: parte del codice implementato in `home.page.ts`

La funzione `presentRoomPopover()` si attiva quando l'utente clicca sul segnalino di una stanza ed è utile per poter aprire un *popover*, ovvero un riquadro presentato al di sopra della pagina, contenente tutte le informazioni delle segnalazioni presenti in quella specifica stanza. Tale popover viene realizzato dalla componente RoomMalfunctions, alla quale vengono passati la lista delle segnalazioni e la posizione e quindi mostra solo le informazioni inerenti la stanza o la segnalazione selezionate (si veda Immagine 5.3.3.4). Il nome della segnalazione è presentato in un riquadro verde se l'utente è l'autore di quella segnalazione oppure se non l'ha ancora supportata, altrimenti viene mostrato un bottone verde con una piccola icona di un pollice accanto al titolo.

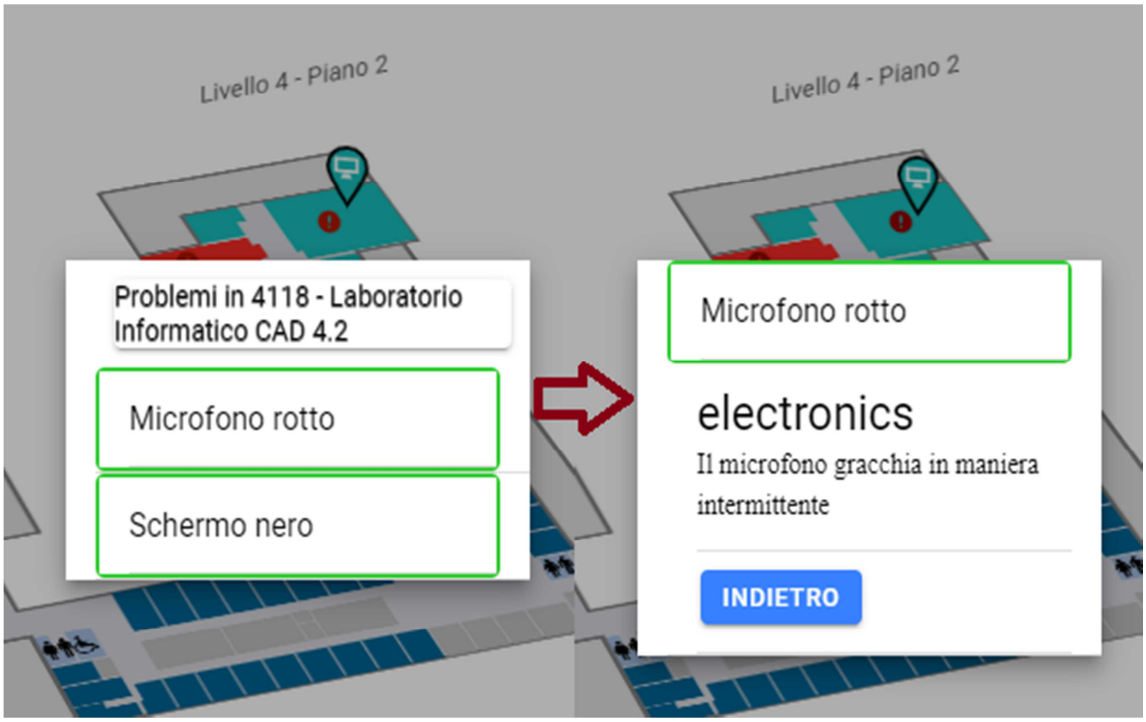


Immagine 5.3.3.4: lista delle segnalazioni di una stanza e vista di una specifica segnalazione

Qualora l'utente fosse un tecnico, egli vedrebbe visualizzati due ulteriori bottoni, rispettivamente per prendere in carico o risolvere la segnalazione e per darne una valutazione numerica (Immagine 5.3.3.5).

Immagine 5.3.3.5: vista per un tecnico di una segnalazione

Con il metodo `openSignalModal()` è possibile invece aprire una modale, una componente simile al precedente popover con la differenza che tuttavia interrompe le funzionalità della pagina sopra la quale viene aperta. Per potere attivare tale funzione l'utente deve prima aver selezionato una stanza della mappa e quindi cliccato sul bottone "Segnala"; in caso contrario compare un breve messaggio a spiegargli queste istruzioni. Questa modale inserisce qui la componente Signal, con la quale l'utente può effettuare la segnalazione (Immagine 5.3.3.6): la posizione è recuperata dal software in base a dove l'utente ha cliccato, il tipo è selezionabile da un menù a tendina e il nome viene inserito testualmente. Il bottone col simbolo della macchina fotografica permette di scattare una foto e inviarla allo storage online (grazie alle funzioni del servizio ImageService), mentre quello con l'icona del fumetto permette di aggiungere una descrizione testuale più lunga rispetto al semplice titolo. Cliccando su "Invia segnalazione" il metodo `addSignal()` del servizio SignalService si occupa di inviare al database tutti i dati per creare il nuovo documento e la modale si chiude, tornando alla vista della pagina Home.

Immagine 5.3.3.6: modale di segnalazione di un disservizio

Nella pagina Account (Immagine 5.3.3.7) l'utente può prima di tutto iscriversi al sistema oppure autenticarsi: se loggato, egli ha accesso alle proprie informazioni personali, quali un nickname (di base è la prima parte della email di iscrizione), il livello raggiunto e i progressi verso il successivo, il proprio avatar, i distintivi vinti; tutti questi dati sono associati ad un documento contenuto online al quale la pagina ha accesso grazie al servizio PersonalService, mentre il servizio ImageService si occupa di scaricare le immagini relative ai badge vinti e quella dell'avatar. Il bottone "Impostazioni" apre un nuovo popover che realizza la componente Settings, dove sono contenute poche funzionalità al momento ma che potrebbero essere arricchite in futuro: la ricezione della notifica e il ripristino dell'avatar di base.

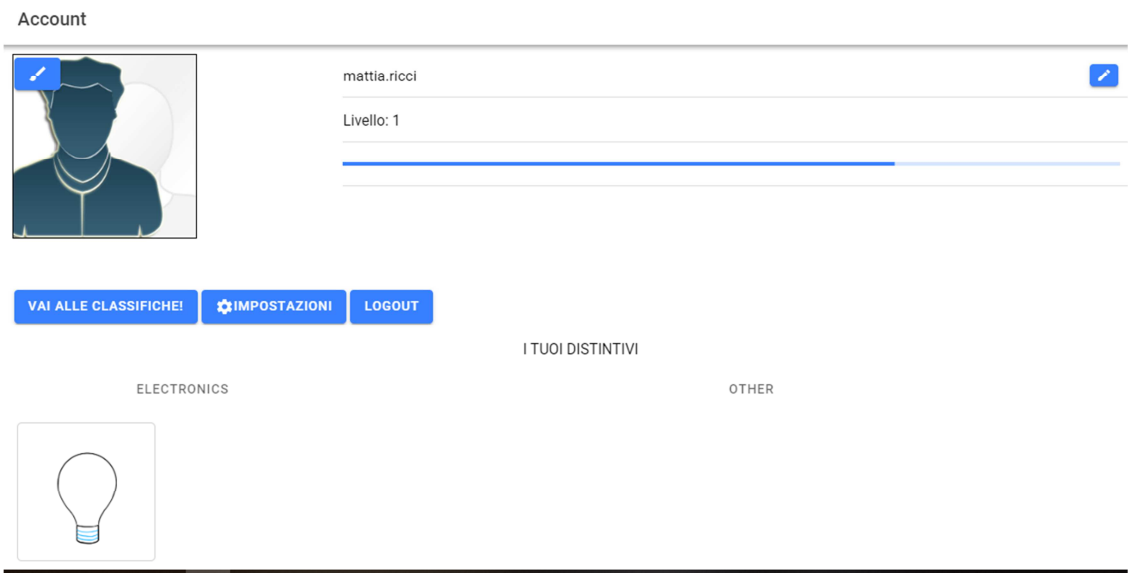


Immagine 5.3.3.7: vista della pagina account dopo aver effettuato l'accesso

Dal pulsante "Vai alle classifiche!" si accede alla pagina Rank, nella quale viene visualizzata la classifica degli utenti in base al livello e al numero di segnalazioni totali o per tipo. La riga dell'utente che sta visualizzando la classifica viene illuminata tramite un diverso colore di sfondo, inoltre è possibile cambiare l'ordinamento degli utenti premendo sul tipo di dato; l'ordinamento è di tipo decrescente. Per tornare alla pagina precedente è stato aggiunto un pulsante "Indietro".

L'ultima pagina implementata è MySignals (osservabile in Immagine 5.3.3.8), contenente quanto richiesto dai partecipanti al Focus Group, ovvero la lista delle segnalazioni effettuate dall'utente. Queste sono divise in due colonne principali tra quelle ancora attive e quelle già risolte e presentano altre varie informazioni, quali il

nome, la posizione, se sono state prese in carico o risolte, il tipo, la data della segnalazione e infine quante altre persone hanno condiviso quella stessa segnalazione; è possibile riordinare le segnalazioni cliccando sul titolo delle singole colonne. Questa funzione è stata semplificata dall'uso del servizio SignalService, che contiene già i metodi per il recupero dei dati sulle segnalazioni effettuate e risolte, mentre la pagina si occupa soltanto di presentare i dati in base all'utente che desidera visualizzarli.

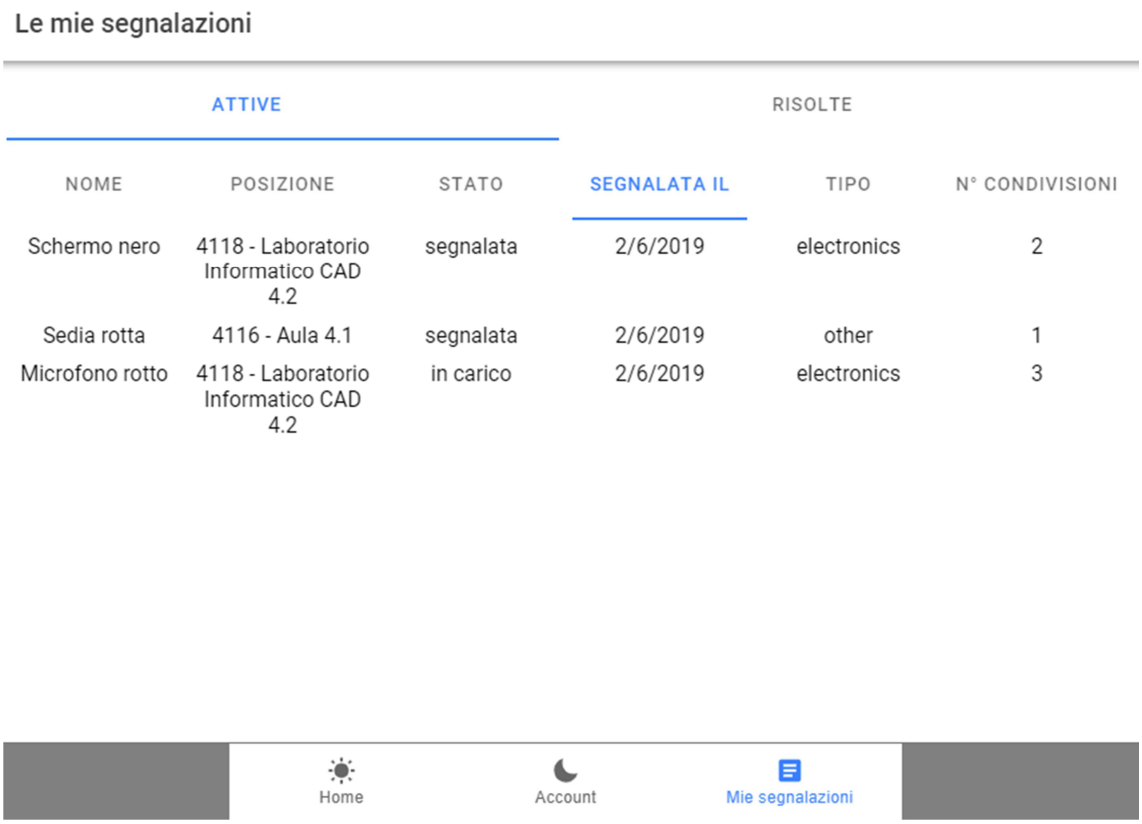


Immagine 5.3.3.8: visualizzazione delle proprie segnalazioni

4.3.4. Server per la mappa interattiva

Il server per il mapping interattivo è quello già funzionante all'interno della struttura del Campus universitario e che è stato ricreato in locale per effettuare test sul funzionamento dell'applicativo. Esso ha due principali funzioni: la prima è quella già accennata in 5.3.1 di consegnare il codice degli script richiesti dal client, contenuti in una cartella locale del server; la seconda è quella di inviare i dati per effettuare la costruzione della mappa, contenuti in un *dump* in formato `.sql` locale. Per la prima attività il server prende il percorso relativo specificato nelle richieste pervenute e lo risolve nella cartella locale dei file di scripting, dunque li invia al client; per la seconda attività invece si avvale della libreria Socket.io [SOC19], la quale fornisce i metodi per

realizzare una comunicazione bidirezionale e real-time: il server crea uno spazio dei nomi al quale il client deve iscriversi, poi procede ad effettuare una serie di query sul database locale e inviare i dati ben formattati come risposta.

Il principale cambiamento apportato al file `socketClient.js`, già accennato in 5.3.1, è stato l'inserimento di una funzione (Codice 5.3.4.1) per aggiungere al documento HTML le meta-informazioni (quali nome, descrizione, numero di posti e tipo di locale) sulle varie stanze contenute all'interno del dump.

```
for (i = 1; i < levelNumber; i++) {
        document.getElementById("level__infos_" +
                                        i).innerHTML = "";}
        valori = data.info.split(';');
        for (field in valori) {
            line = valori[field].split('***');

            var info_node = document.createElement("a");
            info_node.setAttribute("id","DS" + line[0]);
            info_node.setAttribute("data-category",
                                              line[1]);
            info_node.setAttribute("nome", line[2]);
            info_node.setAttribute("descrizione", line[3]);
            if(line[4] !== "null")
                info_node.setAttribute("posti", line[4]);

            info_node.setAttribute("href","#");
            info_node.setAttribute("aria-label", "INFO");
            info_node.style.display = "none";

            if (document.getElementById("level__infos_" +
                                        line[5])!= null) {
                document.getElementById("level__infos_" +
                          line[5]).appendChild(info_node);
            }
        }
```

Codice 5.3.4.1: funzione per aggiungere i meta-dati sulle stanze nel codice HTML del client

La funzione cerca il nodo `level__infos_` che è stato inserito nella pagina HTML della vista Home e quindi vi annida dentro i nuovi dati percepiti.

4.3.5. Funzioni cloud

Le funzioni automatiche di Firebase [FIR19b], come era stato introdotto in 5.1.3, hanno evitato la necessità di implementare un ulteriore server che si occupasse della gestione

dei dati in comunicazione tra il database in cloud e il client: in risposta a certi eventi relativi a funzioni di Firebase oppure richieste HTTP alla base di dati il codice specificato può essere attivato in background, dopo essere stato salvato insieme ai dati sul cloud. Allo sviluppatore spetta il compito di realizzare il codice della funzione, precisando il provider del cloud (servizio di autenticazione, servizio di storage, servizio di database in real-time, eccetera) e la condizione per cui attivare tale metodo; quando questa situazione si verifica il codice viene eseguito, e nel caso avvengano multiple invocazioni il server di Google si preoccupa di creare nuove istanze parallele della funzione per gestire più velocemente il lavoro.

Di seguito si vuole riassumere il lavoro svolto dalle funzioni sviluppate.

La prima funzione è `newUser` e si attiva quando un utente crea un nuovo account (Codice 5.3.5.1): dal momento che le funzioni vengono attivate in modo replicato per tutte le zone coperte dal servizio, la clausola `region` permette di specificare una regione di attivazione così da diminuire i costi di latenza e di propagazione, per trovare la propria località è sufficiente controllare le impostazioni del progetto; `auth` fa riferimento al servizio di autenticazione, mentre il metodo `onCreate()` è l'evento associato alla creazione di un nuovo utente. Si presuppone che nel database (o da qualche altra parte ma comunque in maniera ben formattata per permettere il controllo automatizzato) sia presente la lista di tutte le email del personale universitario, con specificato il ruolo del lavoratore: la funzione `writeUserData()` effettua prima un controllo della mail per poter riconoscere professori, tecnici o altri ruoli grazie ad un altro metodo, `getUserRole()`, quindi aggiunge un nuovo documento nel database sotto al percorso /utenti insieme ad altri dati.

```
export const newUser = functions
    .region('europe-west1')
    .auth.user().onCreate((user) => {
        return writeUserData (user.uid, user.email)
               .catch( error => console.log(error) );
});

async function writeUserData (userId: string, userEmail:
string | undefined) {
    let userRole;
    await getUserRole(userEmail).then( data => userRole =
data )
          .catch( error => console.log(error));
    let userTrustScore;
```

```
    if ( userRole === 'tecnico' || userRole ===
                                            'professore')
        userTrustScore = 1;
    else
        userTrustScore = 0.5;
    admin.firestore().doc('/utenti/' + userId).set({
        email: userEmail,
        role: userRole,
        trustScore: userTrustScore,
        nickname: userEmail!.split('@')[0]
    })
    .catch(error => console.log(error));
}
```

Codice 5.3.5.1: funzione attivata alla creazione di un nuovo utente

La seconda funzione è invece `newSignal` (Codice 5.3.5.2), la quale si attua quando un utente effettua una nuova segnalazione: in questo caso il segnale di attivazione è la creazione di un nuovo documento nella raccolta `/segnalazioni` sul database. La funzione aggiunge all'utente l'identificatore della segnalazione grazie al metodo `addSignal2User()`, in questo modo è facile tenere conto per ogni utente delle segnalazioni che ha effettuato; inoltre all'utente vengono aggiunti dei punti per il suo livello e in base alla segnalazione tramite il metodo `addPoints2User()`; viene calcolato il livello di affidabilità della segnalazione in base al valore di affidabilità dell'utente con il metodo `trustEvaluation()`, quindi si aggiungono questi nuovi dati al documento della segnalazione; infine viene diramata una notifica ai tecnici della presenza di una nuova segnalazione tramite il metodo `sendToTechnic()`.

```
export const newSignal = functions
        .region('europe-west1')
        .firestore.document('/segnalazioni/{signalId}')
        .onCreate(async (snapshot, context) => {
                let signalScores = [];
                const userScores = [];
                for (const user of snapshot.data()!.userId)
                {
                    userScores.push(trustEvaluate(user));
                    addSignal2User(user,
                                    context.params.signalId)
                    .catch( error => console.log(error));
                    addPoints2User(user,
                         snapshot.data()!.type, true)
                    .catch( error => console.log(error));
                }
                signalScores = await
                                    Promise.all(userScores);
                const postDate = Date().toString();
```

```
                let newScore = 0;
                for (const score of signalScores) {
                    newScore += (score * score);
                }
                newScore = Math.sqrt(newScore /
                                    signalScores.length);
                return snapshot.ref.set({
                    status: 'segnalata',
                    postDate: postDate,
                    score: newScore
                    },
                    { merge: true })
                    .then( () => sendToTechnic({
                                name:
                                 snapshot.data()!.name,
                                position:
                                 snapshot.data()!.position,
                                postDate: postDate
                                })
                    )
                    .catch( error => console.log('errore in
                                                "nuova": ' +
error) );
});
```

Codice 5.3.5.2: funzione cloud per le nuove segnalazioni

La funzione `updateSignal` risulta più articolata rispetto alle altre e per motivi di leggibilità si è scelto di non inserirne il codice all'interno del documento, ma si tenterà di darne una spiegazione quanto più chiara possibile. La complessità legata a questo blocco di codice è insita nel suo evento di attivazione, ovvero l'aggiornamento di un documento relativo ad una segnalazione: questo può avvenire in svariati momenti e per motivi differenti, pertanto occorre riconoscere l'evento che ha causato l'aggiornamento e differenziare di conseguenza il comportamento della funzione; non per ultimo, la funzione stessa va a modificare il documento della segnalazione, pertanto occorre prestare attenzione a non creare un ciclo infinito in cui il codice effettua l'aggiornamento e una nuova istanza del metodo viene creata come risposta a questo evento. Di seguito si spiega in che modo i dati di una segnalazione vengono manipolati e in risposta a quale tipo di eventi.

Alla creazione di una nuova segnalazione la funzione precedentemente descritta `newSignal` inserisce nuove informazioni nel documento: in questo caso non deve essere eseguita nessuna azione, dal momento che `updateSignal` è progettata per rispondere agli eventi causati da utenti.

Se un utente vuole aggiungere il proprio voto alla segnalazione devono avvenire le seguenti azioni: il suo identificatore viene aggiunto alla lista di utenti che hanno segnalato quel disservizio; all'utente devono essere aggiunti dei punti e l'id di questa segnalazione, riutilizzando i metodi già visti prima `addPoints2User()` e `addSignal2User()`; il livello di affidabilità della segnalazione deve essere ricalcolato in base al livello di affidabiltà dei nuovi utenti che hanno votato la segnalazione.

Se un tecnico sta prendendo in carico la segnalazione allora viene effettuato un controllo anche a lato server dei permessi di quell'utente tramite il metodo `checkPermit()`, nell'ipotesi che un utente sia riuscito a eludere i controlli a lato client per fingersi un membro del personale specializzato: in caso di riscontro positivo viene cambiato lo stato della segnalazione e una notifica push viene inviata a tutti gli utenti che hanno votato quella segnalazione tramite il metodo `sendOnCharge()` per avvisarli che la segnalazione è stata presa in carico.

Se invece un tecnico dichiara di aver risolto la segnalazione dopo il controllo sui suoi permessi viene creato un nuovo documento contenente tutti i dati necessari nella raccolta delle segnalazioni risolte nel database, quindi l'identificatore del documento passato viene cancellato dalla lista delle segnalazioni ancora attive di ogni utente, infine si elimina il documento stesso dalla relativa raccolta presente nel database;

Un tecnico potrebbe aver espresso una valutazione su una segnalazione, aggiornandone il relativo campo del documento: in questo caso viene invocato il metodo `trustOnRating()`, il quale causa un aggiornamento del livello di affidabilità degli utenti che hanno supportato quella segnalazione sulla base del giudizio espresso dal tecnico.

Infine se l'aggiornamento è stato appena causato dalla funzione `updateSignal` non deve essere eseguita nessuna azione, così da interrompere il ciclo di chiamate ricorsive.

Le funzioni `subscribe2Signal` e `unsubscribe2Signal` (Codice 5.3.5.3) servono per far si che un utente si iscriva o si cancelli dalle notifiche relative ad un certo topic grazie ai metodi `subscribeToTopic()` e `unsubscribeToTopic()` forniti dalla libreria Firebase Cloud Messaging, quindi inviano una notifica push all'utente per avvisare dell'esito avvenuto dell'operazione. Iscrivendosi ad un *topic*

l'utente riceverà una notifica automaticamente ogni qualvolta un nuovo documento verrà creato in quella specifica raccolta di documenti, metodo utilizzato per avvisare i tecnici di nuove segnalazioni, come era stato già spiegato in 5.3.2.

```
export const subscribe2Signal = functions
        .region('europe-west1')
        .https.onCall( async data => {
         await
admin.messaging().subscribeToTopic(data.token,
                                                data.signal);
         return 'subscribed to ${data.signal}';
        });

export const unsubscribeFromSignal = functions
        .region('europe-west1')
        .https.onCall( async data => {
         await
admin.messaging().unsubscribeFromTopic(data.token,
                                                data.signal);
         return 'unsubscribed from ${data.signal}';
        });
```

Codice 5.3.5.3: funzioni per iscriversi e cancellarsi da un topic

L'ultima funzione realizzata è stata `sendOnResolved` (Codice 5.3.5.4): essa viene attivata quando viene creato un nuovo documento inerente una segnalazione risolta, ovvero in seguito all'attivazione di una parte del codice precedentemente presentato per la funzione `updateSignal`; questa funzione si occupa di aggiungere a tutti gli utenti che avevano segnalato il disservizio il nuovo id relativo alla segnalazione risolta tramite il metodo `addSolved2User()`, dopo di che recupera dagli stessi utenti tramite il metodo `getToken()` la chiave d'accesso per l'invio della notifica: se presente, un avviso con l'avvenuta risoluzione viene notificato a tutti quanti gli utenti possessori del *token*.

```
export const sendOnResolved = functions
        .region('europe-west1')
        .firestore.document('risolte/{solvedId}')
        .onCreate( async (snapshot, context) => {
                const solved = snapshot.data();
                let userTokens = [];
                const tokens = [];
                for (const user of solved!.idUtenti) {
                    tokens.push(getToken(user));
                    addSolved2User(user,
                                context.params.solvedId)
```

```
                    .catch( error => console.log(error));
                }
                userTokens = await Promise.all(tokens);
                let validTokens: string[] = [];
                validTokens = userTokens.filter( elem =>
                                        elem !== undefined);
                if ( validTokens.length === 0 )
                                         { return null; }
                const date: string = solved!.postDate;
                const notification:
                          admin.messaging.Notification = {
                 title: 'La tua segnalazione "' +
                        solved!.name +'" è stata risolta',
                 body: 'Zona: ' + solved!.position +
                 '<br><br>Postata il: ' +
                            date.split('GMT+0000 UTC)')[0]
                };
                const payload:
                        admin.messaging.MessagingPayload =
                {
                  notification
                };
                return
                admin.messaging().sendToDevice(validTokens,
                                                payload);
        });
```

Codice 5.3.5.4: funzione per la notificazione delle segnalazioni risolte

4.3.6. Struttura dei dati

In questa sezione si vuole spiegare ora in che modo si è deciso di modellare la struttura della base di dati in cloud. Come si era spiegato alla fine di 5.1.3, il database è di tipo NoSQL, pertanto non vi sono restrizioni legate ad un preciso schema che i dati debbano seguire, di fatto aumentando quindi la flessibilità nel lavorare con queste informazioni. Sono state seguite alcune linee guida base, consigliabili da adottare nel creare un database di questa tipologia [VERG17], riassunte di seguito.

La maggior parte delle computazioni avverrà leggendo i dati piuttosto che scrivendoli, pertanto l'operazione di lettura dovrebbe risultare quanto più leggera possibile; nel caso in esame in particolare si può pensare al fatto che le informazioni di ogni segnalazione verranno propagate da un singolo utente a tutti gli altri utenti.

Denormalizzare è normale, ovvero è consigliabile duplicare le informazioni se questo permette di organizzare opportunamente i dati e migliorare le operazioni di lettura e

scrittura, ottenendo quindi performance superiori: questo può risultare infatti necessario quando troppi elementi sono raggruppati all'interno di un singolo documento o di una raccolta di documenti, che potrebbero invece essere suddivisi in entità diverse; ovviamente occorre fare attenzione dal momento che la duplicazione dei dati richiede di conseguenza il raddoppio delle operazioni di aggiornamento degli stessi. Questo metodo si è reso comunque necessario per poter distinguere le segnalazioni ancora attive da quelle ormai risolte: queste ultime mantengono infatti uno storico di dati destinato ad aumentare sempre di più nel tempo, mentre le prime risulteranno un insieme di cardinalità inferiore, di conseguenza fornire informazioni agli utenti sui soli disservizi ancora non risolti finirebbe per diventare un'operazione inutilmente pesante.

Infine l'albero dei dati non dovrebbe risultare eccessivamente profondo: da una parte tale logica rende più difficile allo sviluppatore tenere conto di tutti i dati contenuti all'interno di un singolo documento e lavorarvi sopra, dall'altro c'è il rischio di effettuare il recupero di una notevole mole di informazioni non necessarie per poter recuperare soltanto pochi dati utili, come nel precedente caso.

Il database è suddiviso in raccolte di documenti, ognuna con un nome specifico e rappresentante di un certo albero di dati pertinenti; ogni collezione comprende un insieme di documenti, i quali hanno un identificatore univoco e contengono un insieme di informazioni.

La prima raccolta è `personaleUnibo`: in essa sono contenute tutte le email relative al personale della struttura e il ruolo ricoperto dall'utente. Poiché la mail universitaria è univoca si è deciso di utilizzarla direttamente come identificatore del documento, così da trovare immediatamente il ruolo associato alla email durante la fase di autenticazione dell'utente, piuttosto che effettuare la ricerca in ogni documento della raccolta di un eventuale campo `email`.

Quando un utente si iscrive alla piattaforma, il suo documento viene aggiunto alla raccolta `utenti`: come identificatore viene utilizzato l'id creato dal servizio di autenticazione di Firebase, che quindi può essere facilmente recuperato e di conseguenza anche il documento dell'utente.

Questo contiene una serie di campi:

- `experience` mantiene il punteggio di esperienza maturato dall'utente;
- `livello` è il livello associato all'account;
- `nickname` lo pseudonimo scelto dall'utente;
- `role` è il suo ruolo all'interno della struttura, che può assumere i valori "studente", "professore" oppure "tecnico";
- `trustScore` rappresenta il livello di affidabilità dell'utente, fissato al massimo se questo è parte del personale del Campus;
- `signalList` è un vettore di identificatori delle segnalazioni attive effettuate dall'utente;
- `solvedList` è invece la lista degli identificatori delle segnalazioni effettuate o votate dall'utente e già risolte;
- `signalPoints` contiene i punteggi segnati all'utente a seconda del tipo di segnalazioni che ha effettuato o votato;
- `achievements` contiene la lista di badge vinti dall'utente.

La raccolta `segnalazioni` comprende invece i documenti delle segnalazioni attive all'interno della struttura:

- `name` è il nome della segnalazione;
- `position` il punto della mappa in cui è avvenuto il disservizio;
- `level` è il piano dell'edificio del disservizio;
- `postDate` l'orario in cui è stata fatta la segnalazione;
- `description` contiene l'eventuale testo descrittivo del disservizio aggiunto dall'utente;
- `photo` è l'identificatore della foto che è stata fatta al disservizio e salvata nello storage in cloud;
- `status` è lo stato della segnalazione, che può essere "segnalata", valore base, oppure "in carico" quando un tecnico dichiara che se ne sta occupando;
- `type` è il tipo della segnalazione;
- `score` è il livello di affidabilità della segnalazione;
- `userId` è il vettore con gli id degli utenti che hanno segnalato oppure votato quella segnalazione;

- `style` sono dei valori necessari per la resa grafica della segnalazione sulla mappa interattiva;
- `rating` comprende sia la valutazione che il tecnico ha dato alla segnalazione che l'identificatore di questo.

L'ultima raccolta definita è `risolte` e contiene le segnalazioni che sono state dichiarate risolte dai tecnici; i documenti racchiudono fondamentalmente un sotto-insieme dei campi già definiti per la raccolta precedente ovvero:

- `name`;
- `position`;
- `level`;
- `type`;
- `postDate`;
- `resolveDate`, che specifica l'orario di risoluzione della segnalazione;
- `risolutore`, ovvero l'identificatore del tecnico che ha risolto il disservizio;
- `description`;
- `photo`;
- `idUtenti`.

5. Conclusioni

Il lavoro svolto ha messo in evidenza come dal collegamento tra due ambiti di studio distinti, quali i metodi di crowdsourcing da un lato e le tecniche di localizzazione indoor dall'altro, possa nascere un nuovo paradigma alquanto efficace nella gestione delle infrastrutture.

La raccolta strutturata di informazioni fornite in modo spontaneo dagli utenti e in maniera amplificata grazie ai moderni dispositivi mobili configura l'accesso a nuove modalità di analisi dei sistemi e dei dati che si fondono all'interno degli edifici. Si è visto che sono presenti delle criticità che possono andare a inficiare questi metodi, tuttavia esistono anche delle soluzioni che vadano a risolvere o quanto meno a minimizzare i danni collaterali che possono emergere.

L'associazione poi di ulteriori dati, raccolti invece tramite metodi automatizzati grazie agli apparati sensoriali degli stessi dispositivi di cui si avvalgono gli utenti, aumenta il valore associabile a questi nuovi elementi: se l'utilizzo di solo una minima parte di tali sensori apre le porte a diverse nuove applicazioni di notevole rilevanza, cosa si potrebbe realizzare sfruttando pienamente la potenza tecnologica dei moderni smartphone, smartwatch e tablet? E aggiungendo anche i valori dei vari misuratori presenti all'interno degli edifici? In questo caso l'unico vero limite è quello rappresentato dal livello tecnologico stesso, che tuttavia si è visto non essersi ancora fermato.

Il risultato finale dalla congregazione di questi flussi di dati è un'interpolazione di nuove informazioni col quale è possibile risolvere nuove e vecchie necessità per le persone che circolano all'interno di questi ambienti.

5.1. Sviluppi futuri

Il sistema realizzato nella versione corrente contiene tutte le principali funzionalità che ci si era proposti fin dall'inizio di realizzare, tuttavia esistono differenti margini di miglioramento che potrebbero essere messi in pratica, emergenti sia dagli approfondimenti effettuati durante la prima fase di analisi del sistema, sia in base alle richieste degli utenti stessi, raccoglibili tramite nuovi questionari post-implementazione,

o a seguito di ulteriori studi. Nonostante l'approccio didattico perseguito nello svolgimento del lavoro, risulta verosimile la possibilità di estendere il software per utilizzi reali.

Un primo perfezionamento che potrebbe essere apportato al sistema riguarda sicuramente la modalità di recupero della posizione dell'utente all'interno dell'edificio. Nelle sezioni 3.3 e 3.4 si erano già proposte due possibili tecniche per effettuare questa operazione in maniera automatizzata, che tuttavia non è stato possibile costruire: il software implementato richiede ancora l'intervento dell'utente, seppure semplificato nei limiti del possibile tramite un semplice click o una pressione dello schermo del dispositivo in un punto della mappa. Gli esempi sopra riportati comunque necessitano di ulteriori approfondimenti per verificarne la fattibilità all'interno della struttura del campus, e in generale la ripetibilità di quelle tecniche all'interno di un qualunque edificio.

Un'altra miglioria notabile riguarda invece il calcolo dell'importanza di una segnalazione, un dato non di poco conto dal momento che dovrebbe suggerire al tecnico che sta visualizzando l'avviso di un disservizio con quale tempestività intervenire oppure da chi e da quante persone è richiesto il ripristino del normale servizio. Collegato a questo punto vi è inoltre il calcolo dell'affidabilità di un utente, importante per garantire la robustezza del sistema a fronte di eventuali malintenzionati e falsi avvisi: da una parte tali utenti andrebbero inibiti dall'utilizzo del sistema, dall'altra occorre evitare ai tecnici di svolgere del lavoro inutile. Essendo la versione base del sistema gli algoritmi dietro a tali computazioni seguono dei modelli di valutazione del comportamento e di interazione ancora semplici, pertanto andrebbero studiati e migliorati, approfondendo la ricerca iniziale.

Una volta creato e messo in funzione il sistema si avrebbe finalmente a disposizione una base di dati concreti: come si era proposto nel secondo capitolo tali informazioni andrebbero analizzate e da tale studio occorrerebbe quindi ottenere dei piani d'azione volti a migliorare la vita di chi usufruisce del Campus. E' possibile capire le cause di un disservizio, i metodi più efficaci per risolverlo, giungere a prevederne la comparsa e agire in maniera preventiva: per questo tipo di lavoro occorre anche l'intervento del personale tecnico-amministrativo della struttura, per poter meglio valutare i dettagli di cui andrebbe tenuto conto.

Svincolandosi invece dalle ricerche effettuate inizialmente e provando a immaginare alcuni tra i possibili miglioramenti futuri del sistema, viene in mente il fatto che i moderni dispositivi mobili sono dotati di attrezzatura tecnologica avanzata che potrebbe essere utilizzata per fornire una pletora di dati a costi minimizzati sullo stato interno della struttura. Si pensi per esempio se tali apparecchi potessero trasmettere informazioni sulla temperatura degli ambienti: un utente potrebbe non necessariamente segnalare una disfunzione del sistema termoregolatore, ma il sistema potrebbe comunque catturare l'anomalia nei dati e farla presente ad un tecnico per un controllo. L'utilizzo del microfono può essere utile per verificare la presenza di inquinamento acustico all'interno dell'edificio: una possibilità è che un gruppo di utenti non stia rispettando i limiti del decoro consentito in un'aula mentre non vi è personale nelle vicinanze a controllare la situazione. Queste rimangono soltanto alcune ipotesi, tuttavia non andrebbe sottovalutata la possibilità di ampliare il parco di rilevatori dati interni alla struttura, che oltremodo sono fissi, con un insieme di sensori mobili e gratuito.

Un altro modo per sfruttare al meglio i dispositivi potrebbe essere implementare alcune funzionalità di realtà aumentata nell'applicazione, sempre basandosi sui percettori elettronici di tali apparecchi: oltre al già citato recupero automatizzato della posizione dell'utente, si possono trovare metodi migliori sia per effettuare le segnalazioni che per visualizzarne le informazioni, allo stato attuale ancora basate principalmente sul giudizio soggettivo di ogni utente.

Infine eludendo dal contesto di questo specifico software si possono immaginare nuovi scenari di applicazione delle tecniche di crowdsourcing e di localizzazione indoor: ad esempio in ambito medico si può tenere da conto lo stato di salute di un paziente ricoverato all'interno di una struttura ospedaliera o di un ospizio, specialmente di coloro che non sono auto-sufficienti ma che necessitano di essere tenuti sotto controllo medico, come i malati del morbo di Alzheimer o di altri disturbi di tipo psicologico. All'interno di un'azienda sarebbe più facile effettuare il controllo su sale riunioni ancora libere da prenotare o la posizione di colleghi. Anche la gestione delle emergenze all'interno di un qualunque edificio ne gioverebbe, potendo determinare il migliore piano d'evacuazione conoscendo la posizione delle persone all'interno dell'edificio e il recupero di coloro che sono rimasti inabilitati a muoversi.

APPENDICE A

Questionario post Focus-group

DOMANDE SULL'INCONTRO

1. **Ti è piaciuto partecipare a questo Focus Group?**
   Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

2. **Pensi che agli altri sia piaciuto questo Focus Group?**
   Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

3. **Hai trovato chiare le domande che ti sono state esposte?**
   Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

4. **Hai partecipato rispondendo alle domande?**
   Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

5. **Pensi che le tue risposte saranno prese in considerazione?**
   Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

6. **Parteciperesti nuovamente al Focus Group?**
   Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

DOMANDE SULLA APP

7. **Hai capito cosa faccia questa app?**
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

8. **Trovi utile questo tipo di app*, in generale?**
* app per il monitoraggio di una struttura basata sul feedback degli utenti
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

9. **Pensi possa essere utile la app che ti è stata presentata?**
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

10. **Saresti portato a scaricare tale app?**
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

11. **Saresti portato a utilizzare tale app?**
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

12. **Hai capito come sfruttare le diverse funzionalità della app?**
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

13. **Trovi completa e funzionale questa app?**
Contrassegna solo un ovale.

	1	2	3	4	5	
Per niente	◯	◯	◯	◯	◯	Assolutamente

Bibliografia

[DOT18] dotmagazine, “Smart Campus – Merging Smart City and Smart Home in education for digital natives”, https://www.dotmagazine.online/new-work-and-digital-education/ICT4D/smart-campus-merging-smart-city-and-smart-home-in-education-for-digital-natives , 2018

[WIKI-IA19] Wikipedia, “Information Age”, https://en.wikipedia.org/wiki/Information_Age ,2019

[BRO16] E. Brown, “Who Needs the Internet of Things?”, https://en.wikipedia.org/wiki/Internet_of_things ,2016

[ANI17] Anixter (Youtube), ”What is a Smart Campus?”, https://www.youtube.com/watch?v=y0rbbvsV3nc ,2017

[HOW06] J. Howe, “The Rise of Crowdsourcing”, https://www.wired.com/2006/06/crowds/ ,2006

[EST12] E. Estellés-Arolas, F. González-Ladrón-de-Guevara, “Towards an integrated crowdsourcing definition”, http://www.crowdsourcing-blog.org/wp-content/uploads/2012/02/Towards-an-integrated-crowdsourcing-definition-Estell%C3%A9s-Gonz%C3%A1lez.pdf , 2012

[DEV07] L. DeVun, “(Q&A) Your assignment: Art”, https://web.archive.org/web/20121024130503/http://www.wired.com/techbiz/media/news/2007/07/crowd_captain?currentPage=all ,2007

[WIKI-CROen19] Wikipedia, “Crowdsourcing”, https://en.wikipedia.org/wiki/Crowdsourcing ,2019

[BOR11] I. Borst, “The case for and against crowdsourcing(Part 2)”, https://web.archive.org/web/20150912024759/http://www.crowdsourcing.org/editorial/the-case-for-and-against-crowdsourcing-part-2/2850 ,2011

[LEGO19] Lego, “Digital Designer”, https://www.lego.com/en-us/ldd ,2019

[DUP13] S. Dupree, “Crowdfounding 101: pros and cons”, http://market-found.com/crowdfunding-101-pros-cons/ ,2013

[STE17] I. G.Stein, “Crowdfunding Fraud –Lessons from Elio Motors”, http://laweconomicscapital.com/2017/01/crowdfunding-fraud-lessons-from-elio-motors/ ,2017

[FIX16] S. K. Fixson, “A Case Study of Crowdsourcing Gone Wrong”, https://hbr.org/2016/12/a-case-study-of-crowdsourcing-gone-wrong ,2016

[WILL13] M. Williams, “FBI urges media to 'exercise caution' after inaccurate arrest reports”, https://www.theguardian.com/world/2013/apr/17/fbi-media-exercise-caution-bombings ,2013

[CHI13] R. Chittum, “The *New York Post*’s disgrace”, https://archives.cjr.org/the_audit/the_new_york_posts_disgrace.php ,2013

[FAI16] Faisal (HBS Digital Initiative), “Crowdsourcing Fails (Durex, Mountain Dew, and NASA): Avoiding Internet Trolls”, https://digit.hbs.org/submission/crowdsourcing-fails-durex-mountain-dew-and-nasa-avoiding-internet-trolls/ ,2015

[EDU19] Educalingo, “Trustability”, https://educalingo.com/it/dic-en/trustability ,2019

[KUY11] S.O. Kuyoro, F. Ibikunle, O. Awodele, “Cloud Computing Security Issues and Challenges”, https://www.researchgate.net/publication/285011991_Cloud_Computing_Security_Issues_and_Challenges ,2011

[CHE18] Z. Chen, L. Tian, C. Lin, “Trust evaluation model of cloud user based on behavior data”, https://journals.sagepub.com/doi/pdf/10.1177/1550147718776924 ,2018

[ATW18] J. Atwood, “Understanding Discourse Trust Levels”, https://blog.discourse.org/2018/06/understanding-discourse-trust-levels/ ,2018

[KHA14] B. Khaleghi, “Breaking down AI’s trustability challenges”, https://www.elementai.com/news/2018/breaking-down-ais-trustability-challenges ,2014

[KAS17] R. Kaser, “Facebook enlists AI in war on fake news”, https://thenextweb.com/facebook/2017/08/03/facebook-enlists-ai-in-war-on-fake-news/ ,2017

[YAN07] Z. Yan, S. Holtmanns, “Trust modeling and Management: from Social Trust to Digital Trust”, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.2925&rep=rep1&type=pdf , 2007

[VIO11a] F. Viola, “Storia della Gamification”, http://www.gameifications.com/gamification/storia-della-gamification/ , 2011

[PRA18] C. Prandi, “Gamification as a tool to induce behavioral changes”, 2018

[VIO11b] F. Viola, “Gamification & Editoria Online”, https://blog.tagliaerbe.com/2011/06/gamification-editoria.html, 2011

[WIKI-GAMen19] Wikipedia, “”, https://en.wikipedia.org/wiki/Gamification , 2019

[GAM19a] Gamification.it, “Gamification e obiettivi principali”, http://www.gamification.it/gamification/gamification-e-obiettivi-principali/, 2019

[GAM19b] Gamification.it, “Oltre le Dinamiche e le Meccaniche gamificate: le Componenti”, http://www.gamification.it/gamification/oltre-le-dinamiche-e-le-meccaniche-gamificate-le-componenti/ ,2019

[GAM19c] Gamification.it, “Meccaniche e Dinamiche gamificate: cosa sono?-PARTE1-”, http://www.gamification.it/gamification/meccaniche-e-dinamiche-nella-gamification-esattamente-di-cosa-si-tratta/ ,2019

[GAM19d] Gamification.it, “Astrazioni e desideri: le Dinamiche nella Gamification”, http://www.gamification.it/gamification/astrazioni-e-desideri-le-dinamiche-nella-gamification/ ,2019

[WIKI-GNSS19] Wikipedia, “Sistema satellitare globale di navigazione”, https://it.wikipedia.org/wiki/Sistema_satellitare_globale_di_navigazione ,2019

[STUR07] R. W. Sturdevant, “NAVSTAR, the Global Positioning System: A Sampling of Its Military, Civil, and Commercial Impact”, https://history.nasa.gov/sp4801-chapter17.pdf, 2007

[WIKI-GLO19] Wikipedia, “GLONASS”, https://it.wikipedia.org/wiki/GLONASS ,2019

[WIKI-GAL19] Wikipedia, “Sistema di posizionamento Galileo”, https://it.wikipedia.org/wiki/Sistema_di_posizionamento_Galileo ,2019

[WIKI-BEI19] Wikipedia, “Sistema di posizionamento BeiDou”, https://it.wikipedia.org/wiki/Sistema_di_posizionamento_BeiDou ,2019

[IJAZ13] F. Ijaz, H. K. Yang, A. W. Ahmad, C. Lee, “Indoor Positioning: A Review of Indoor Ultrasonic Positioning systems”, http://icact.org/upload/2013/0409/20130409_finalpaper.pdf ,2013

[FS-MP19] Federal Standard 1037C, “Multipath propagation”, https://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm, 2019

[VAN04] J. Vanderkay, “Nokia, Philips And Sony Establish The Near Field Communication (NFC) Forum”, https://web.archive.org/web/20110628203003/http://www.nfc-forum.org/news/pr/view?item_key=d8968a33b4812e2509e5b74247d1366dc8ef91d8, 2004

[WEB19] Webopedia, “Wi-Fi”, https://web.archive.org/web/20120308123721/http://www.webopedia.com/term/w/wi_fi.html, 2019

[WIKI-TOA19] Wikipedia, “Time of Arrival”, https://en.wikipedia.org/wiki/Time_of_arrival ,2019

[OKE17] B. O’Keefe, “Finding Location with Time of Arrival and Time Difference of Arrival Techniques”, https://sites.tufts.edu/eeseniordesignhandbook/files/2017/05/FireBrick_OKeefe_F1.pdf ,2017

[COR09] CorCom, “A Malpensa e Fiumicino videosorveglianza super-tech”, https://www.corrierecomunicazioni.it/digital-economy/a-malpensa-e-fiumicino-videosorveglianza-super-tech/, 2009

[WIKI-AOA19] Wikipedia, “Angle of Arrival”, https://en.wikipedia.org/wiki/Angle_of_arrival ,2019

[MAO19] W. Mao, “Approaches for Angle of Arrival Estimation”, https://www.cs.utexas.edu/~swadhin/reading_group/slides/AoA.pdf ,2019

[RFID14] Rfid Global, “RSSI: significato e suo utilizzo nella misura di distanza con tag RFID”, http://www.rfidglobal.it/rssi-misura-distanza-tag-rfid/ ,2014

[ROS01] S. Ross, “Wherever You Go, There Is Connettivity”,

https://www.microsoft.com/en-us/research/project/radar/ ,2001

[HE16] S. He, S. H. Gary Chan, “Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons”, http://home.deib.polimi.it/redondi/WI/cst_fingerprint.pdf ,2016

[WIKI-WIFIen19] Wikipedia, “Wi-Fi”, https://en.wikipedia.org/wiki/Wi-Fi ,2019

[ELM18] Elmat, “WI-FI FINGERPRINTS: UNA SOLUZIONE INTELLIGENTE”, http://www.elmat.com/blog/wifi-fingerprints-la-soluzione-intelligente/ ,2018

[WEBA19] Web Archive, “What is Bluetooth”, https://web.archive.org/web/20170912002136/https://www.bluetooth.com/what-is-bluetooth-technology, 2019

[BEA19] Beaconitaly, “Cos’è un beacon?”, http://www.beaconitaly.it/cose-ibeacon/ ,2019

[NEA16] Nearit, “Beacon: tutti i segreti svelati da NearIt”, https://www.nearit.com/it/cosa-sono-i-beacon/ ,2016

[BEA18] Beaconitaly, “Ritratti di famiglia”, http://www.beaconitaly.it/ritratti-di-famiglia/ ,2018

[DIG16] Digital4, “Beacon in negozio, 5 consigli per innovare la shopping experience”, https://www.digital4.biz/marketing/mobile-e-social/beacon-nel-punto-vendita-cinque-consigli-per-costruire-una-shopping-experience-innovativa/ ,2016

[MCC19] S.McClain, “What Are the Limitations of Bluetooth”, https://www.techwalla.com/articles/what-are-the-limitations-of-bluetooth ,2019

[HDB16] HDBlog.it Mobile, “Bluetooth 5, rilasciato il nuovo standard che rivoluzionerà l'IoT”, https://mobile.hdblog.it/2016/12/08/Bluetooth-5-disponibile-standard-IoT/ ,2016

[WIZ19] Wizblog, “Bluetooth 5, caratteristiche del nuovo standard”, https://wizblog.it/bluetooth-5-caratteristiche-del-standard ,2019

[THA19] D. Thakur, “Principles of Software Design & Concepts in Software Engineering”, http://ecomputernotes.com/software-engineering/principles-of-software-design-and-concepts ,2019

[CARL15] S. Carletti, “Progettazione di sistemi multimediali Laboratorio UniMC al digitale”, http://docenti.unimc.it/simone.carletti/teaching/2014/400209787/files

/presentazione-concept-project-design/at_download/file, 2015

[GAM02] E. Gamma, R. Helm, R. Johnson, J. Vlissides, “Design Patterns - Elementi per il riuso di software ad oggetti”, Pearson Education Italia, 2002

[HTML19] HTML.it, “App mobile ibride, native o web: le differenze”, https://www.html.it/faq/app-mobile-ibride-native-o-web-le-differenze/ ,2019

[PMI14] Project Management Institute, “A Guide to the Project Management Body of Knowledge”, 5 edizione, Project Management Institute, 2014

[WIKI-REQ19] Wikipedia, “Requirement”, https://en.wikipedia.org/wiki/Requirement ,2019

[WYS19] R. Wysocki, “Lean Stage Planning In The Face Of An Incomplete Solution: Part 2 - The Requirements Breakdown Structure”, https://www.projecttimes.com/articles/lean-stage-planning-in-the-face-of-an-incomplete-solution-part-2-the-requirements-breakdown-structure.html ,2019

[TAY03] A. Taylor , “Analysis, Design, and Development Techniques with J2EE”, http://www.informit.com/articles/article.aspx?p=31942&seqNum=5, 2003

[JAC14] I. Jacobson, I. Spence, K. Bittner, “Use Case 2.0: The Guide to Succeeding with Use Cases”, Ivar Jacobson International, 2014.

[MAR04] M. Marzolla, “Modelli di sistemi”, https://www.moreno.marzolla.name/teaching/ingegneria-del-software/2004-2005/L07_ModelliSistema_4up.pdf ,2004

[FOW03] M. Fowler, “Patterns of Enterprise Application Architecture”, Addison Wesley, 2003, p. 116.

[WIKI-CM19] Wikipedia, “Conceptual model (computer science)”, https://en.wikipedia.org/wiki/Conceptual_model_(computer_science) ,2019

[FOG19] D. Fogli, “Pattern di Progettazione di Interazione Uomo-Macchina”, https://iol.unibo.it/pluginfile.php/364945/mod_resource/content/0/Pattern%20di%20progettazione%20di%20HCI.pdf ,2019

[UIP19] Ui Pattern, “Design Patterns”, http://ui-patterns.com/patterns ,2019

[WIKI-MOC19] Wikipedia, “Mockup”, https://it.wikipedia.org/wiki/Mockup ,2019

[FAS19] A. R. Fasolino, “Processi di sviluppo rapido del software”, http://www.federica.unina.it/ingegneria/ingegneria-software-ii/processi-sviluppo-rapido-software/ ,2019

[BAL19] Balsamiq, “balsamiq”, https://balsamiq.com/ ,2019

[IQUII19] Iquii, “PROGRESSIVE WEB APP (PWA): COSA SONO, PRO E CONTRO E I PRINCIPALI ESEMPI SUL MERCATO”, https://iquii.com/2019/03/04/progressive-web-app/, 2019

[WEBA11] Web Archive, “Focus group”, https://web.archive.org/web/20130127031820/http://www.urp.gov.it/Sezione.jsp?idSezione=52, 2011

[MIG01] L. Migliorini, N. Rania, “I FOCUS GROUP: UNO STRUMENTO PER LA RICERCA QUALITATIVA”, http://www.fqts.org/dati/doc/56/doc/114.pdf ,2001

[ANG19] Angular, “Angular”, https://angular.io/ ,2019

[PIZ12] M. Pizzonia, “Model-view-controller”, http://www.dia.uniroma3.it/~pizzonia/swe/slides/12_MVC.pdf, 2012

[ION19] Ionic Framework, “What is Ionic Framework”, https://ionicframework.com/docs/intro ,2019

[FIR19a] Firebase, “Firebase”, https://firebase.google.com/ ,2019

[EXP19] Express.js “Express: Framework web veloce, non categorico e minimalista per Node.js”, https://expressjs.com/it/ ,2019

[NOD19a] Node.js, “Node.js”, https://nodejs.org/it/ ,2019

[WIKI-NPM19] Wikipedia, “npm (software)”, https://en.wikipedia.org/wiki/Npm_(software) ,2019

[NOD19b] Node.js, “About the Node.js Foundation”, https://foundation.nodejs.org/about ,2019

[GIT19a] Github, “TypeScript”, https://github.com/microsoft/TypeScript ,2019

[WIKI-ECMA19] Wikipedia, “ECMAScript”, https://it.wikipedia.org/wiki/ECMAScript ,2019

[W3C19] W3C, “HTML5 Introduction”, https://www.w3schools.com/html/html5_intro.asp, 2019

[SASS19] Sass-lang, “Sass”, http://www.sass-lang.com/ ,2019

[MDN19] Mozilla Developer Network, “Learn to style HTML using CSS”, https://developer.mozilla.org/en-US/docs/Learn/CSS, 2019

[GIT19b] Github, “angularfire2”, https://github.com/angular/angularfire2 ,2019

[WIKI-PT19] Wikipedia, “Push tecnology”, https://en.wikipedia.org/wiki/Push_technology#Push_notification ,2019

[STA18] Stackoverflow, “Ionic 3 Componenet vs Page”, https://stackoverflow.com/questions/45279191/ionic-3-component-vs-page ,2018

[RYL18] C. Rylan, “JavaScript Promises Versus RxJS Observables”, https://coryrylan.com/blog/javascript-promises-versus-rxjs-observables ,2018

[SOC19] SocketIO, “Socket.io”, https://socket.io/ ,2019

[FIR19b] Firebase Docs, “Cloud Functions for Firebase”, https://firebase.google.com/docs/functions/ ,2019

[VERG17] J. Vergara, “Jumping from SQL to Firebase NoSQL Database”, https://gonehybrid.com/firebase-database-best-practices/ ,2017