Report con stato di avanzamento dello sviluppo del software e rilascio di una versione dimostrativa del software con la funzione di tracciamento indoor.

Contenuto del documento

In questo documento viene presentato il processo di sviluppo eseguito per la realizzazione degli applicativi con annessi prototipi implementati per arrivare a tali prodotti finali. Nella prima parte vengono illustrati i prototipi realizzati per la comprensione dei vari aspetti su cui il progetto si fonda mostrando dunque le demo Android e demo Java. Nell'ultima parte invece viene descritto lo stato attuale di avanzamento dello sviluppo.

Implementazione Demo

L'applicativo si basa sull'utilizzo di dispositivi chiamati *beacon* i quali inviano segnali univoci bluetooth che possono essere rilevati tramite l'utilizzo di smartphone. Per questo motivo il primo passo per la realizzazione del progetto è stato quello di realizzare una piccola demo dimostrativa implementata in linguaggio nativo Android per apprendere come venissero ricevuti i segnali, tramite l'utilizzo di API (nel caso specifico si tratta di: android-beacon-library:2.16.1) per poi stamparli in una lista.

Una volta realizzata questa demo, sono stati eseguiti dei test volti a capire quale distanza fosse ottimale per far si che i vari segnali non si sovrapponessero al fine di evitare ambiguità. Capito questo poi, la demo è stata rimodellata per distinguere quale fosse il beacon più vicino all'utente che la utilizza. Per rendere chiaro graficamente questo è stata realizzata una matrice di pulsanti all'interno dei quali è stato inserito l'identificativo dei vari beacon in possesso e in base ai segnali

ricevuti, tramite un apposito algoritmo, veniva scoperto quale fosse l'identificativo più vicino all'utente per illuminare uno degli elementi della matrice.

L'algoritmo sfrutta il metodo getDistance() dell'API sopracitata.

Completata questa applicazione si è poi implementata un'ulteriore piccola demo per testare il protocollo MQTT tramite Android. Questa aveva lo scopo di capire come venissero inviati i messaggi ad un determinato broker dato un topic specificato. La demo garantiva due servizi: il primo era quello di visualizzare tutti i messaggi inviati su un determinato topic e il secondo consisteva nell'invio di un messaggio dato in input dall'utente sul topic in ascolto.

La demo mostra all'utente un'interfaccia grafica minimale tramite cui è possibile vedere nella parte superiore i messaggi in arrivo, mentre la parte inferiore è stata dedicata all'invio dei messaggi.

Una volta realizzata questa seconda demo, per testare l'effettivo funzionamento di tale messaggistica, questo applicativo è stato installato su due smartphone differenti per valutare quanto l'invio e la ricezione dei messaggi fossero effettivamente visualizzati in tempo reale.

Per rendere il test ancora più esaustivo è stata sviluppata un'applicazione web che offrisse la stessa funzionalità della demo Android. Questa web application è servita in seguito a capire come sviluppare l'interfaccia del capitano tramite cui gestire i messaggi che i vari utenti inviano.

Le due demo Android sono state poi integrate al fine di memorizzare lo storico dei beacon rilevati dall'utente su un determinato topic. In questa ulteriore demo dunque i messaggi inviati erano composti solo dall'identificativo del beacon più vicino con relativo timestamp del momento in cui veniva rilevato.

Completate le demo si è passati poi a ragionare sulla funzionalità principale del sistema che consiste nel guidare l'utente all'interno del piano verso un determinato punto di interesse.

La funzionalità è stata suddivisa in due micro-funzionalità:

1. Visualizzare il piano attuale in cui è situato l'utente.
2. Indirizzare l'utente verso un punto di interesse.

Per soddisfare il primo punto si è pensato di visualizzare la planimetria di un certo piano della nave in base a quale fosse il segnale del beacon più vicino che viene rilevato. Questo è possibile tramite

una tabella in cui viene memorizzato l'identificativo del beacon ed il relativo piano in cui è posizionato.

Per indirizzare l'utente verso una destinazione all'interno del piano tramite un percorso, si è pensato di effettuare un'astrazione che permettesse di rappresentare il piano sotto forma di grafo i cui cammini sono i percorsi ed i nodi i vari punti di interesse. Così facendo è stato possibile utilizzare vari algoritmi noti ed efficienti per l'instradamento degli utenti all'interno della mappa.

L'utilizzo del grafo come astrazione del piano è stato pensato nel seguente modo: i vertici sono coppie di coordinate che identificano la posizione dei beacon all'interno del piano (ad ogni svincolo è collocato un beacon) e dei relativi punti di interesse. Gli archi sono coppie di vertici A e B, per cui B rappresenta il vertice più vicino per cui esiste un collegamento percorribile dall'utente partendo da A.

Per l'editing di tale grafo si è sviluppato inizialmente un applicativo Java che lo realizzasse partendo dall'immagine della mappa. L'utente incaricato della realizzazione del grafo di un certo piano doveva, come primo passaggio, selezionare l'immagine della planimetria della mappa sulla quale estrarre il grafo descritto precedentemente. Cliccando su un certo punto, veniva inserito un vertice e, una volta immessi sull'immagine, poteva realizzare archi tra di loro come si vede nella figura in basso (Figura 1).

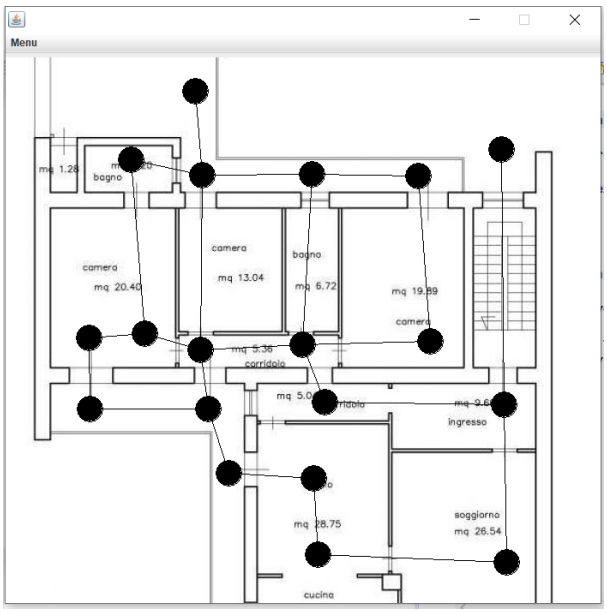

Figura 1: Applicazione Java

L’idea dell’applicativo Java è stata però successivamente superata tramite lo sviluppo di un applicazione web che integrasse al suo interno le varie funzioni di pannello di controllo per il capitano della nave oltre che le funzionalità di editing del grafo.

Stato attuale dello sviluppo

Grazie alla realizzazione di tutte le demo sopracitate si è giunti allo sviluppo di due applicativi:

- Una web application che funge da pannello di controllo attraverso il quale sarà possibile gestire la realizzazione del grafo dei percorsi, la messaggistica con gli utenti e segnalare situazioni di emergenza.
- Un’applicazione mobile (per ora disponibile solo in Android) da consegnare all’utente finale che permette di gestire il tracciamento e la navigazione indoor dello stesso.

Il pannello di controllo è realizzato, per quanto riguarda il lato client, mediante HTML5 e CSS3 più l’aggiunta di varie funzioni Javascript per garantirne il corretto utilizzo. La storicizzazione della mappa è affidata ad un database MySQL nel quale vengono inoltre salvati anche i vari utenti. Per la rappresentazione dei percorsi si è deciso di utilizzare i grafi come struttura dati per poter usufruire delle varie funzioni di ricerca e navigazione come antecedentemente accennato.

Ogni volta che l’utente aggiunge un nodo dovrà poi specificare l’identificativo del beacon ad esso associato.

Le operazioni di inserimento, modifica ed eliminazione sono gestite lato server da file PHP che risiedono sullo stesso server dell’applicativo. Per la gestione della messaggistica invece, viene utilizzato il protocollo MQTT che consente di avere interazioni rapide con un database MongoDB all’interno del quale vengono memorizzati tali messaggi al fine di avere uno storico dei movimenti degli utenti in forma anonima. Di seguito sono mostrati alcuni screenshot dell’applicativo (Figura 2, Figura 3).

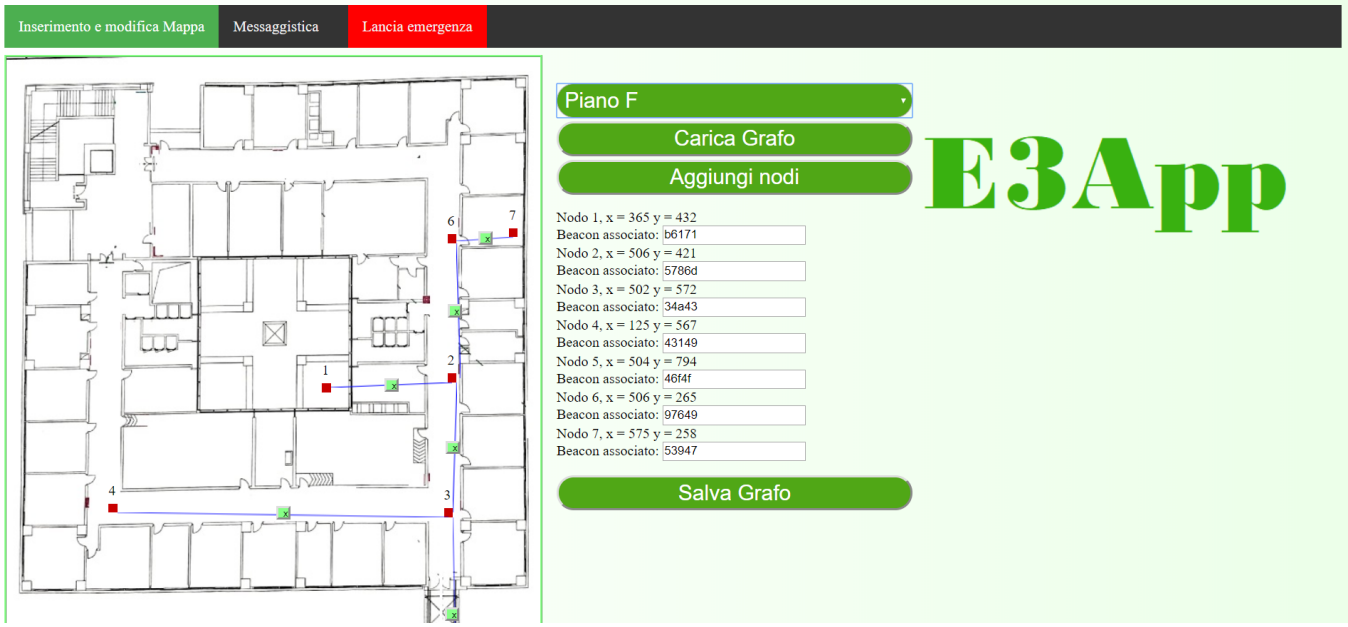

Figura 2: inserimento e modifica mappa.

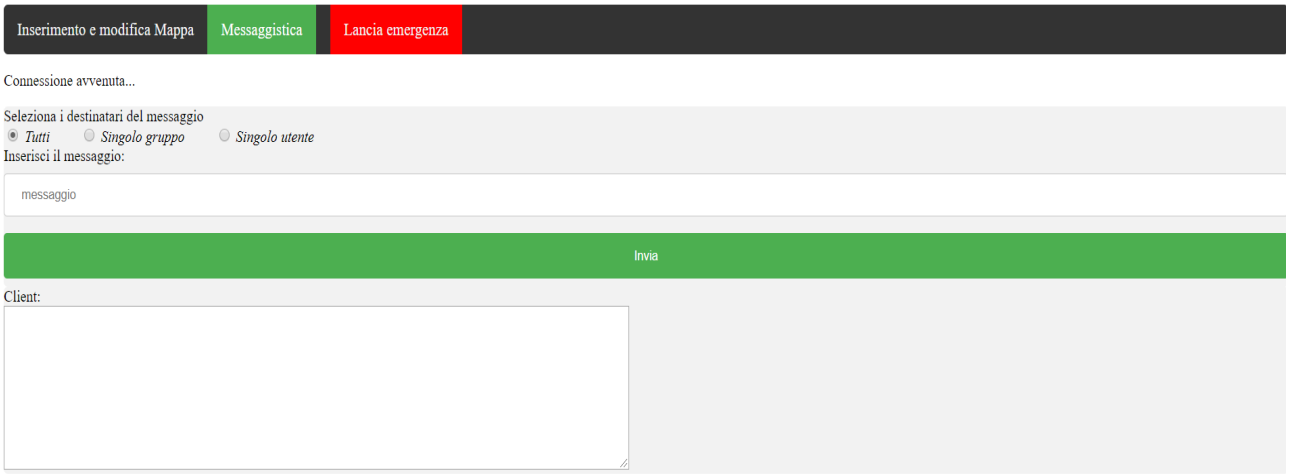

Figura 3: messaggistica.

Il fulcro del progetto è rappresentato dall'app Android. All'avvio preleva dal server il grafo dei percorsi percorribili per permettere all'app di funzionare anche in locale, senza bisogno di essere connessi alla rete wi-fi. L'app è in grado di individuare l'utente all'interno del piano e, inoltre, in base al punto di interesse che l'utente intende raggiungere, ne calcola il percorso minimo (partendo dalla posizione attuale) avvalendosi del grafo precedentemente caricato come mostrato in figura (Figura 4).

Figura 4: schermata applicazione Android.

La posizione corrente dell'utente viene rappresentata con il vertice più vicino rispetto alla posizione reale all'interno del piano. Tale vertice viene stabilito in base ai segnali (trasmessi dai beacon) ricevuti. La ricezione di questi segnali avviene tramite il protocollo EddyStone-UID, implementato attraverso delle librerie android. Ad ogni cambiamento della posizione (vertice sorgente), viene trasmesso un messaggio al CloudMQTT su un determinato topic, tramite protocollo MQTT, in modo da tener traccia della posizione attuale e di quelle passate dell'utente.

Per quanto riguarda il calcolo del cammino minimo dal vertice sorgente alla destinazione selezionata, viene ricavato in base all'algoritmo di Dijkstra applicato al grafo dei percorsi percorribili del piano in cui si trova l'utente.

Inoltre, per testare la possibilità di stabilire una destinazione, si è aggiunto, come si vede in figura (Figura 4), uno spazio dedicato alla selezione di uno dei vertici del grafo. Quando l'utente inserisce una nuova destinazione l'applicazione in automatico utilizzerà Dijkstra selezionando come vertice sorgente quello più vicino alla posizione attuale dell'utente (rilevata dal segnale del beacon più vicino) e come vertice pozzo quello che esso ha selezionato. Durante l'utilizzo dell'applicazione la sorgente viene più volte aggiornata seguendo gli spostamenti dell'utente grazie alla rilevazione costante e periodica del beacon attualmente più prossimo.

Per simulare il cambio del piano, soddisfando così la prima micro-funzionalità sopra descritta, si è deciso di aggiungere momentaneamente dei bottoni che cambiassero il piano attuale con relativo grafo.

In caso di emergenza, si sta progettando una nuova interfaccia minimale che farà visualizzare all'utente solo una freccia che gli indicherà la direzione da seguire per arrivare all'area di sicurezza più vicina. Per consentire a questa interfaccia di far puntare la freccia sempre nella direzione della destinazione si sta progettando un modello matematico che, basandosi sul coefficiente angolare degli archi, farà uso del giroscopio dello smartphone per indicare con esattezza il percorso di emergenza.