Università degli Studi di Milano Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Un'app a supporto della navigazione indoor nel campus UniMiB

Relatore: Micucci Daniela
Co-relatore: Ginelli Davide

Relazione della prova finale di:
Elio Gargiulo
Matricola 869184

Anno Accademico 2022-2023

"You can't change what happened. But you can still change what will happen."
Sebastian Vettel

A Hermes e Antonella, che mi hanno dedicato la loro vita.
Alla mia famiglia per essere sempre presenti, anche nei momenti difficili.
Ai professori Micucci e Ginelli per la fiducia riposta in me.
Ai miei amici con cui ho potuto condividere questo splendido viaggio.
A me stesso per non essersi mai arreso.

Abstract

Muoversi tra un'aula ed un'altra di un campus universitario, come quello dell'Università Bicocca, può risultare dispersivo e impegnativo, soprattutto per i nuovi studenti che, ogni anno, l'Università accoglie.
Questo lavoro di stage analizza e progetta un'applicazione per la navigazione indoor all'interno del campus UniMiB, consentendo all'utente di variare percorso in base alle proprie preferenze.
La tesi inizia con un'analisi approfondita delle diverse API disponibili per la navigazione indoor. Sono state esaminate diverse opzioni, valutando le loro caratteristiche, funzionalità e adattabilità alle esigenze specifiche di un campus universitario. Questo ha portato alla scelta dell'API di Navigine come soluzione più adatta.
L'elaborato procede con la progettazione dell'applicazione stessa, l'implementazione dell'API, includendo l'architettura generale, i requisiti di sistema e le funzionalità chiave. Viene posta particolare attenzione sull'uso fatto di Navigine e delle sfide riscontrate nello sviluppo e test dell'applicazione. Si completa il tutto discutendo sui risultati ottenuti e sui possibili sviluppi futuri anche con l'ausilio di servizi universitari per migliorare ulteriormente l'esperienza dell'utente nel campus UniMiB.

Indice

Introduzione

La navigazione attraverso dispositivi mobili è diventata ormai da diversi anni uno strumento fondamentale ed efficace per ogni individuo.
La possibilità di poter conoscere precisamente la propria posizione e dove dirigersi in tempo reale, utilizzando solamente il proprio dispositivo, ha evidenziato una vera e propria evoluzione del concetto di orientamento e spostamento.
L'utilizzo dei segnali emessi dal sistema GPS (Global Positioning System) permette un'implementazione semplice di innumerevoli applicazioni, gratuitamente scaricabili su molteplici dispositivi, al fine di fornire indicazioni chiare ed immediate quando si è al di fuori di un edificio.
Sorge però una problematica da non sottovalutare: l'utilizzo della localizzazione tramite il sistema GPS è ristretto e non sempre utilizzabile. In particolare, se consideriamo l'interno di un edificio di grandi dimensioni o centri come università ed aeroporti, dove il segnale non è in grado di essere ricevuto per via dell'attenuazione, le applicazioni di navigazione risultano inutilizzabili.
Sarà quindi necessario dover analizzare delle alternative valide e concettualmente semplici per poter garantire la possibilità di navigazione nelle zone in cui il sistema GPS non risulta essere la soluzione adeguata.
Questo è il compito delle applicazioni di navigazione "indoor", le quali forniscono gli stessi servizi sopraelencati ma utilizzando approcci e tecnologie differenti.

L'attenzione della successiva analisi, e quindi di tutto l'elaborato, si concentrerà sull'analisi e progettazione di un'applicazione per la navigazione all'interno del campus UniMiB, partendo da un'analisi approfondita dei sistemi alternativi al GPS, fino alla realizzazione vera e propria dell'applicazione utilizzando il servizio di Navigine, ovvero la soluzione trovata alla problematica evidenziata.

Capitolo 1

Contesto e Motivazioni

1.1 Panoramica del Contesto

1.1.1 Necessità e Obiettivi dell'App per la Navigazione Indoor

Nel campus UniMiB è necessaria una soluzione affidabile per aiutare studenti, personale e visitatori a navigare negli edifici e a trovare informazioni importanti. La navigazione interna può essere difficile, soprattutto in strutture grandi e complesse come un campus universitario. Di conseguenza, si è deciso di creare un'applicazione di navigazione interna che fornisca indicazioni precise e personalizzate. Questa applicazione mira a semplificare il processo di localizzazione di aule, uffici, servizi e altre destinazioni all'interno del campus.

1.1.2 Possibile Scenario di Utilizzo

L'applicazione è stata pensata per soddisfare al meglio le esigenze di un utente, il quale vorrebbe navigare comodamente all'interno del campus UniMiB. Pertanto l'utilizzo dell'applicazione permette un orientamento efficace in ogni edificio, attraverso l'utilizzo di punti di interesse e percorsi dinamici, con supporto alla navigazione su più piani e routes adattabili in base alla posizione dell'utente.
I punti di interesse (POI) sono stati concepiti per essere il più differenziati possibile, nei limiti imposti dalla soluzione utilizzata.
Vi è dunque una vera e propria classificazione in base al tipo di punto, che assumerà un aspetto differente a livello grafico e potrà essere filtrato.
L'applicazione mira ad essere utilizzata, da studenti o personale, come un vero e proprio navigatore, con lo scopo di permettere il raggiungimento con semplicità e precisione di aule, scale, bagni e ulteriori POI.

1.2 Panoramica delle Motivazioni

1.2.1 Indagine e Analisi delle Principali Difficoltà

Al fine di realizzare un'applicazione di navigazione è necessaria un'indagine preliminare, che permetta, dopo un'analisi dei risultati ottenuti, di identificare le esigenze e difficoltà riscontrate dagli utenti.
Soprattutto per coloro che sono appena entrati nell'ambiente del campus UniMiB, è risultato che la navigazione tra i vari edifici universitari può risultare confusionaria e tediosa, in quanto il campus è notevolmente vasto e distribuito.
La problematica è presente notevolmente anche all'interno di ogni edificio, i quali presentano spesso molteplici piani e spazi di grandi dimensioni. Identificare il percorso per raggiungere l'aula di un professore oppure semplicemente una lezione può risultare complicato, specialmente se si considerano persone affette da disabilità motorie, dove la necessità di avere chiarezza, efficienza ma soprattutto dinamicità nell'orientamento è ancora più pronunciata.
L'aiuto fornito da un'applicazione di navigazione mira pertanto a mitigare le difficoltà evidenziate.

1.2.2 Vantaggi e Miglioramenti

I vantaggi e miglioramenti che comporterebbe l'utilizzo dell'applicazione non sono da sottovalutare. La navigazione, fornita attraverso il semplice utilizzo di uno smartphone, permette di essere accessibile da chiunque ne abbia necessità in ogni momento. L'utente può accedere all'utilizzo dei servizi offerti senza doversi affidare a strumenti o dispositivi specifici.
La riduzione dello stress e dell'ansia generata dal costante bisogno di capire dove dirigersi ed il tempo risparmiato dalle ricerche dei punti di interesse sono punti cruciali. L'app fornisce indicazioni precise e dettagliate con tempi di percorrenza e percorsi ottimizzati utilizzando i punti di interesse, consentendo agli utenti di risparmiare tempo durante i loro spostamenti nel campus. Questo risulta in un vantaggio considerevole.
La presenza di un'app di navigazione semplice, efficace ed estendibile può sicuramente essere un grande miglioramento fornito al campus UniMiB, con lo scopo di migliorare l'esperienza complessiva di navigazione e la creazione di un ambiente più accogliente e user-friendly.

1.3 Obiettivo

L'obiettivo di questa tesi è quello di analizzare, progettare e sviluppare un'applicazione che possa assolvere al ruolo di navigatore indoor per il campus UniMiB. Si sono posti i seguenti obiettivi:

- Analisi e confronto delle diverse API disponibili per la navigazione indoor (Capitolo 2)
- Selezione dell'API più adatta a soddisfare le esigenze specifiche del progetto (Capitolo 3)
- Analisi e progettazione dell'applicazione (Capitolo 3)
- Implementazione dell'API e valutazione delle prestazioni dell'applicazione (Capitolo 4)

Nel Capitolo 2 verranno presentate alcune delle API per la navigazione indoor. Saranno analizzate le caratteristiche e le funzionalità delle diverse API prese in considerazione, evidenziandone vantaggi e svantaggi. Successivamente, nel Capitolo 3, verrà descritta la progettazione dell'applicazione per la navigazione indoor, inclusa l'architettura generale, i requisiti di sistema, l'interfaccia utente e le funzionalità chiave. Infine il Capitolo 4 si vedrà la realizzazione della soluzione e una prima considerazione sui risultati ottenuti.

Capitolo 2

Analisi e Confronto di API

2.1 Fine Applicativo

Prima di procedere con lo sviluppo dell'applicazione è stato fondamentale effettuare una ricerca approfondita sulle diverse API disponibili. L'obiettivo è identificare l'API che soddisfi i requisiti specifici dell'applicazione, inclusa la precisione delle indicazioni di navigazione, la compatibilità con le piattaforme mobile ed eventuali le funzionalità aggiuntive.
Nella ricerca dell'API abbiamo preso in considerazione i seguenti fattori:

- Supporto dei linguaggi di programmazione
- Piattaforme supportate (Android, iOS o multi-piattaforma)
- Tecnologie di navigazione
- Facilità di integrazione (Documentazione, Esempi di Codice)
- Funzionalità
- Licenza e Costi

Sulla base di questa ricerca, sono state identificate diverse API candidate, tra cui MapsPeople, Navigine, Steerpath, MazeMap, Situm e IndoorAtlas. Nel capitolo verranno presentate in dettaglio le caratteristiche di ciascuna API, inclusi i linguaggi supportati, le piattaforme compatibili, le tecnologie di navigazione e altre informazioni rilevanti.

2.2 Introduzione alle API

Le API (Application Programming Interface) semplificano lo sviluppo e l'innovazione del software consentendo alle applicazioni di scambiare dati e funzionalità in modo semplice e sicuro. Consentono alle aziende di rendere disponibili i dati e le funzionalità delle loro applicazioni a sviluppatori esterni, partner commerciali e dipartimenti interni. Questa interfaccia documentata consente ai servizi e ai prodotti di comunicare tra loro sfruttando dati e funzionalità di ciascuno. Gli sviluppatori possono utilizzare le API senza conoscerne l'implementazione, semplicemente utilizzando l'interfaccia per comunicare con altri prodotti e servizi. Questo risulta in un grosso vantaggio per i programmatori che possono ricorrere a queste per evitare di risparmiare tempo sulla stesura del codice e lavorare su un livello di programmazione più alto. Le API sono diventate essenziali nelle applicazioni web di oggi e molte di quelle più diffuse si basano su di esse per le loro funzionalità [9].

Esistono tre tipologie di API [2]:

- API Pubbliche: sono accessibili pubblicamente, come ad esempio quelle utilizzate da social come Facebook e Google. Questo tipo di API sono disponibili per l'utilizzo da parte di sviluppatori di terze parti, i quali ottengono l'accesso quasi istantaneamente, a seconda del fornitore. Tale aspetto, infatti, può dipendere dallo scopo previsto e dal volume di attività.

- API Private: sono disponibili solo all'interno di un'organizzazione specifica per chi ne fa parte. Gli sviluppatori che lavorano per l'azienda possono utilizzare queste API a differenza degli sviluppatori di terze parti. Poiché queste API non sono documentate in kit di sviluppo software accessibili pubblicamente, sono spesso completamente sconosciute al pubblico.

- API Partner: sono condivise esternamente con i partner commerciali dell'organizzazione. Generalmente, vengono utilizzati per creare integrazioni profonde, condividere informazioni sensibili o realizzare cose che ciascuna organizzazione non potrebbe individualmente. Vengono quindi usate come strumento di comunicazione reciproco.

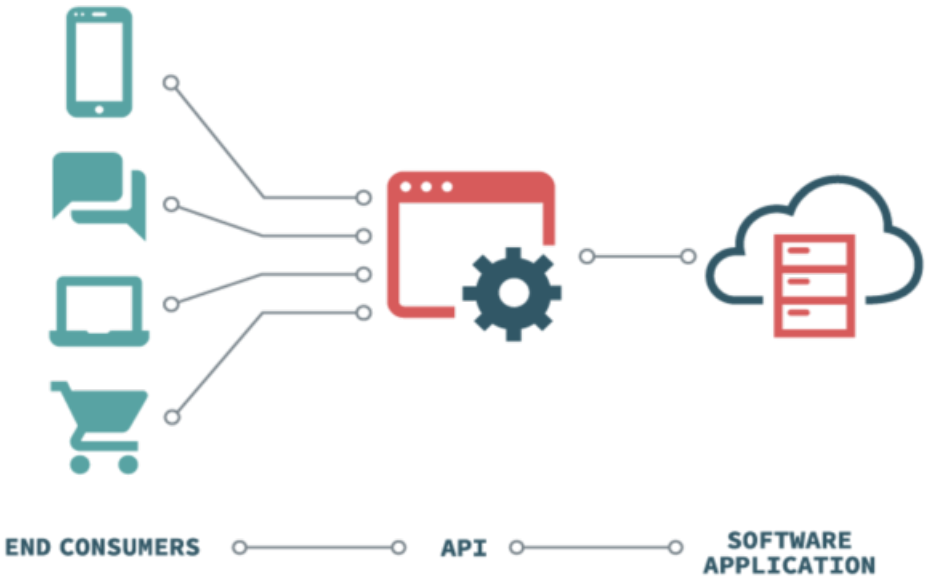

Figura 2.1: Diagramma di funzionamento delle API

L'immagine 2.1 mostra il ruolo vitale dell'API nel collegare il client al database. Gli sviluppatori sono responsabili di stabilire una connessione solida tra il client e l'API, mentre il fornitore dell'API si occupa di gestire il trasferimento dei dati tra la sua API e il server.

In questo capitolo analizzeremo aspetti positivi e negativi di alcune delle varie API disponibili sul mercato per sviluppare una soluzione utile e funzionale con il nostro campus universitario. Infine procederemo con un confronto tra le varie soluzioni.

2.3 MapsIndoors

MapsIndoors è una realtà sviluppata da MapsPeople sulla base di applicazioni di navigazione esterne come Google Maps o Mapbox, ma estende le loro capacità sulla navigazione indoor [14].
Consente di sviluppare sia per iOS, sia per Android, usando come linguaggi di programmazione: Java, Kotlin e Swift, supportando di fatto solo lo sviluppo in linguaggio nativo.

2.3.1 Vantaggi

MapsIndoors pone l'attenzione sull'importanza di due fattori principali:

- La possibilità di visualizzare i percorsi e le mappe in 3D (oltre che in 2D) usando i vettori per rappresentare gli elementi della mappa. In questo

modo la visualizzazione risulta sempre definita e dettagliata anche a ingrandimenti molto alti.

- La capacità di vedere gli aggiornamenti delle mappe in tempo reale permettendo di tenere traccia dei percorsi delle persone, degli spostamenti e degli oggetti. Questo permette di organizzare meglio spazi e tempi

2.3.2 Svantaggi

Nonostante i punti su cui fa leva, MapsIndoors presenta anche dei punti a sfavore che fanno optare per soluzioni diverse per i nostri fini.
Il primo di questi riguarda i costi che non sono disponibili sul sito, ma esplicitano che sia l'API che l'SDK (Software Development Kit) risultano siano a pagamento e per conoscere eventuali prezzi è obbligatorio prenotare una demo.
Inoltre non consentono la possibilità di prenotare demo a scopi didattici, ma solo a fini commerciali, precludendo la possibilità di scoprire ulteriori informazioni.

2.4 Steerpath

Steerpath è una azienda che fornisce diversi servizi per uffici, campus universitari e ospedali.
L'azienda fornisce vari SDK per qualsiasi piattaforma: iOS, Android, Web e supporta lo sviluppo sia nei linguaggi nativi che in React Native.

2.4.1 Vantaggi

Interfaccia chiara e pulita, unita ad una documentazione per ogni ambiente di sviluppo, sono sicuramente due dei punti a favore di Steerpath. Oltre a questo l'azienda invia, insieme all'acquisto della chiave API, i beacon necessari per la struttura designata.
Fornisce anche degli SDK differenti per il posizionamento e l'instradamento, utilizzabili nella stessa soluzione. Ne esistono versioni differenti per i diversi linguaggi di programmazione che supportano (Java, Swift e React Native).
Steerpath separa la mappa globale dalle differenti mappe indoor che consentono di inserire infinite ambientazioni interne basate sulla planimetria esterna. L'inserimento delle varie ambientazioni è facilmente implementabile da un'interfaccia web, basata a sua volta sulle interfacce REST.
Ultimo vantaggio è la possibile integrazione con altre REST API esterne che

permettono di modificare dinamicamente le mappe (come ad esempio una API per le prenotazioni degli alloggi universitari) [23].

2.4.2 Svantaggi

Il primo punto a sfavore è sicuramente il costo, assolutamente elevato di 690 €/mese per il planning base per un campus universitario, giustificato per i beacon inviati dalla compagnia il primo mese.

2.5 Mazemap

Mazemap differenzia le sue soluzioni dipendentemente dall'utilizzo a cui deve essere sottoposto. Per aggiungere Mazemap alla propria soluzione (Mobile o Web) è necessario inserire, nel proprio progetto, il codice seguente:

```
<link rel="stylesheet" href="https://api.mazemap.com/js/v2.0.100/
    mazemap.min.css">
<script type='text/javascript' src='https://api.mazemap.com/js/v2
    .0.100/mazemap.min.js'></script>

<!-- Da inserire nel <body> -->
<div id='mazemap-container' style="width: 100%; height: 100%;"></div>
```

```
var myMap = new Mazemap.Map({container: 'mazemap-container'});
myMap.on('click', function(e){
    myMap.flyTo({center:e.lngLat, zoom: 18});
});
```

2.5.1 Vantaggi

Mazemap offre, oltre alla navigazione indoor, anche un sistema interno per prenotare spazi messi a disposizione dell'università, come posti in aula studio o in biblioteca, direttamente dalla mappa in 3D aggiornata in tempo reale [15].
Inoltre, fornisce una funzione **heatmap**, che consente di vedere il traffico in tempo reale, molto utile per organizzare gli spazi e i percorsi che attraversano zone molto affollate. Grazie a questa funzione è possibile impostare la temperatura della stanza dinamicamente in base all'affollamento della stessa o dell'area, utilizzando una tecnologia di terze parti [16].

2.5.2 Svantaggi

I punti critici di Mazemap sono sostanzialmente due:

- L'API è pensata per un utilizzo Web e quindi non supporta i linguaggi nativi di Android e iOS, ma consente solo lo sviluppo in React Native.
- Per creare le mappe degli interni degli edifici è necessario pagare, rendendo, di fatto, questa soluzione a pagamento.

2.6 Situm

Situm, a differenza delle precedenti soluzioni, si concentra solo su mappatura, navigazione e tracking indoor, offrendo, agli sviluppatori esterni, una vasta varietà di linguaggi tra cui scegliere: Android, iOS, Cordova, React Native e Capacitor. Infine, forniscono anche delle REST API per integrare cartografia e geodati in applicazioni, siti web e sistemi di terze parti.
Situm offre anche una dashboard di gestione per gli amministratori, in cui possono controllare gli spostamenti in tempo reale delle persone connesse all'applicazione, i loro spostamenti e i percorsi più affollati [22]

2.6.1 Vantaggi

La società offre agli sviluppatori esterni il codice sorgente degli SDK su GitHub diviso per ambiente di sviluppo, separati a loro volta da una sezione Getting Started, per chi approccia per la prima volta a Situm, e una sezione completa con tutte le funzionalità. La documentazione fornisce anche il codice sorgente di un'applicazione per Android e una per iOS pronta per essere modificata dagli sviluppatori esterni.
Oltre ad un'ottima documentazione, viene data la possibilità di una prova gratuita di 30 giorni in cui è possibile usufruire del pacchetto completo di Situm.

2.6.2 Svantaggi

Dopo la prova di 30 giorni è necessario inviare una mail per chiedere un preventivo alla società, in cui si esplicitano le specifiche del progetto. Essendo l'unica sede fisica in Spagna, può risultare gravoso in termini temporali una richiesta mail in nazioni con fusi orari molto diversi (ex. USA, Cina o Giappone).

2.7 IndoorAtlas

Questa piattaforma, intuitiva per gli sviluppatori, fornisce un flusso di lavoro completo e un set di strumenti per il posizionamento indoor. Consente di creare servizi basati sul posizionamento interno come ricerca, wayfinding e marketing di prossimità per una maggiore soddisfazione e coinvolgimento degli utenti [10].
Oltre al servizio base di navigazione indoor, IndoorAtlas fornisce anche la possibilità di uno strumento di navigazione in AR (Augmented Reality) che, attraverso la fotocamera dello smartphone, evidenzia il tragitto da percorrere su ciò che la videocamera fa apparire sullo schermo. Oltre al percorso da seguire, l'AR visualizza anche i POI (Point Of Interest) quando questi vengono inquadrati.
Per il posizionamento dell'utente, vengono utilizzati contemporaneamente sei livelli di sensori:

- Il GPS viene utilizzato per sapere la città o la posizione approssimativa dell'utente rispetto al mondo.
- Wi-Fi e BLE (Bluetooth Low Energy - Beacon) sono le soluzioni a prestazioni migliori per il passaggio da navigazione indoor ad outdoor e viceversa.
- La bussola viene utilizzata in rapporto al campo magnetico terrestre per sapere l'orientamento dello smartphone.
- Il giroscopio consente di conoscere tutti i movimenti fatti dallo smartphone, come rotolamento o volo.
- Il loro algoritmo VIO, che connette lo smartphone alla realtà aumentata, consente alla geolocalizzazione tramite AR e ad alte prestazioni.
- Il barometro, infine, consente di percepire la modifica della pressione atmosferica utile a capire quando l'utente cambia piano.

IndoorAtlas consente lo sviluppo in iOS, Android, Cordova, React Native, Unity e Xamarin permettendo agli sviluppatori esterni un'ampia scelta per sviluppare la propria applicazione.

2.7.1 Vantaggi

I punti a favore di IndoorAtlas sono davvero molteplici. Sicuramente il primo tra tutti è la realtà aumentata che nessuno degli altri competitor visti precedentemente offre. L'utilizzo di sei sensori per il posizionamento, di cui

fa parte un algoritmo interno di posizionamento per l'AR, fornice un sistema di localizzazione molto accurata e precisa.
Altro punto a favore risulta in una documentazione dettagliata e aggiornata costantemente nel tempo, in cui si evidenziano cambiamenti e bug fixing separatamente.
Per concludere la società offre un servizio di prova di 60 giorni in cui è possibile creare e provare una propria applicazione utilizzando i propri SDK.

2.7.2 Svantaggi

Nonostante i molti punti a favore, IndoorAtlas ha anche delle criticità.
La prima di queste consta nel prezzo (3500 €/mese) indipendentemente dalla grandezza dell'area mappata o da altri fattori. Questo porta ad una spesa gravosa per piccole aziende, mentre in una spesa irrisoria per realtà molto più grandi.
Il secondo punto critico risulta nell'assenza di un source code di una soluzione demo per testare il funzionamento, ma tuttavia è possibile richiederne una versione precompilata via mail. Infine, il punto più importante risulta essere un altissimo utilizzo della batteria, necessario per poter utilizzare diversi sensori contemporaneamente. Questo può essere poco rilevante per brevi utilizzi (qualche ora), ma risulta gravoso, nel nostro caso, dovendo ipotizzare studenti e personale che potrebbero necessitare dell'applicazione per tutta la giornata.

2.8 Navigine

Navigine è risultata essere la soluzione più semplice e conveniente da implementare, oltre che open source. Consente di sviluppare sia in linguaggio nativo per Android e iOS sia in Flutter, un framework di sviluppo multipiattaforma. Ciò significa che è possibile utilizzare Navigine per creare applicazioni di navigazione indoor per dispositivi Android e iOS, nonché per sviluppare app cross-platform utilizzando Flutter.

Il suo punto di forza, pubblicato dagli stessi sviluppatori, è l'Open Source. Essi riportano cinque punti su cui fa forza l'Open Source [18]:

- Community: Molte persone tendono a collaborare ed unirsi in una community per un software open source.

- Sicurezza: Chiunque può modificare il codice ora per correggere un errore, ora per aggiungere una funzionalità omessa dagli stessi sviluppatori.
- Controllo: Gli sviluppatori esterni possono esaminare il codice per controllare che esso faccia esattamente quello per cui è stato programmato e modificare delle parti che non servono o piacciono.
- Stabilità: Essendo open source, gli utilizzatori si affidano al codice sorgente pubblico per avere la certezza che questo non sia dismesso dagli stessi sviluppatori.
- Allenamento: Un codice sorgente pubblico consente ai programmatori di imparare e migliorarsi leggendo e studiando il codice del software

2.8.1 Vantaggi

Navigine offre per i propri sviluppatori diversi strumenti per diverse esigenze che dipendono dal grado di precisione che si vuole avere, dal budget e dalle attività che si devono svolgere. Questo, comporta nella vasta gamma di scelte di posizionamento indipendenti tra loro, che navigine utilizza (Wi-Fi, Beacon, Wi-Fi RTT (Round-trip time), Ultrasuoni, AOA (Angle of Arriving) for Direction Finding with Bluetooth, Wireless Mesh).
Gli sviluppatori integrano, nel codice, degli algoritmi di ottimizzazione che consentono di utilizzare meno risorse hardware e meno batteria, ottimo per usi giornalieri.
Il passaggio da navigazione indoor ad outdoor, utile per quando bisogna cambiare edificio, non richiede accorgimenti da parte degli sviluppatori, ma viene fatto in modo automatico dagli algoritmi di instradamento utilizzati.
Infine, uno strumento utilissimo per gli sviluppatori è la presenza di un'app completa per iOS e una per Android e i rispettivi codici sorgenti. Questo permette di testare le mappe inserite senza dover prima completare lo sviluppo della propria applicazione o di una demo della stessa.

2.8.2 Svantaggi

Navigine trasforma in png le mappe caricate, anche quelle nei formati dinamici (ex. SVG). Questo comporta a un calo di qualità quando si effettuano zoom molto grandi. Tale problema è ovviabile caricando un file di dimensioni molto alte. Rispetto ad altre soluzioni analizzate precedentemente, la grafica delle mappe è quella che lo sviluppatore carica e non esiste un tool per uno loro sviluppo.

Ad ora non vi è uno strumento di visualizzazione 3D delle mappe che in molte delle altre soluzioni abbiamo riscontrato.

Capitolo 3

Analisi e Progettazione dell'applicazione

3.1 Analisi e Obiettivi dell'Applicazione

3.1.1 Obiettivi dell'Applicazione

Gli obiettivi che si vogliono raggiungere con questa soluzione sono i seguenti:

- Fornire indicazioni di navigazione precise e affidabili all'interno degli edifici, guidando gli utenti lungo i percorsi ottimali per raggiungere le destinazioni desiderate.
- Individuare e visualizzare la posizione corrente dell'utente all'interno degli edifici del campus, consentendo loro di orientarsi facilmente.
- Offrire informazioni aggiuntive sugli edifici, come orari di apertura, descrizioni delle aule o uffici, servizi disponibili e altre informazioni utili per gli utenti.
- Migliorare l'esperienza degli utenti nel campus UniMiB semplificando la navigazione e riducendo il tempo impiegato per trovare le destinazioni desiderate.

La progettazione ha lo scopo di fornire un'esperienza intuitiva e precisa per i suoi utenti. Per la navigazione e gli algoritmi indoor di navigazione si è optato per la scelta dell'API di Navigine.
Il resto del capitolo approfondirà l'architettura generale dell'applicazione, i requisiti di sistema, l'interfaccia utente e le caratteristiche più importanti. Spiegherà, inoltre, come l'applicazione utilizzi l'API per fornire indicazioni

precise, localizzare le posizioni all'interno degli edifici e fornire agli utenti altre informazioni preziose.

3.2 Architettura Generale

Per l'architettura dell'applicazione si è scelta l'architettura MVVM (Model-View-ViewModel) [17].

- Model: Contiene i dati dell'applicazione. Non può comunicare con la View direttamente, ma solo con la ViewModel, solitamente utilizzando delle classi dette Observables[I].
- View: rappresenta l'interfaccia utente dell'applicazione priva di qualsiasi logica dell'applicazione. Osserva il ViewModel da cui ottiene i dati e li mostra in corretto all'utente.
- ViewModel: è il collegamento tra il Model e la View. Si occupa dell'elaborazione dei dati ottenuti dal Model, mantenendoli tramite LiveData[II]. Utilizza le callback per aggiornare la visualizzazione dei dati chiedendoli al modello.

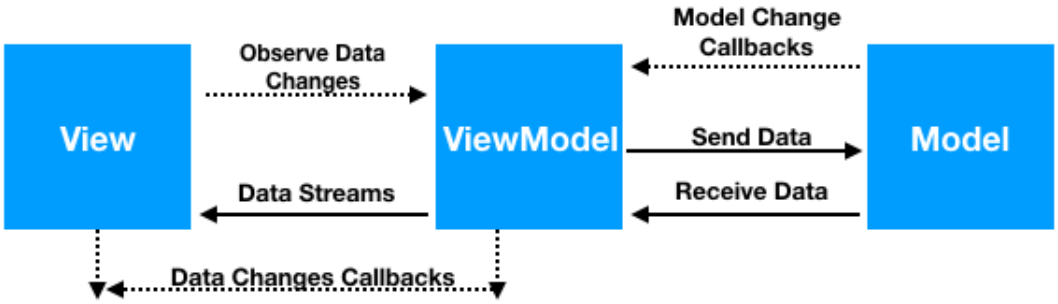

Figura 3.1: Schema del funzionamento del Pattern MVVM

Su questa archiettura si è concluso che fosse più efficiente e facile da mantenere una struttura con poche Activity e più Fragment che si scambiano tra loro.
Ci sono due Activity, la Main Activity, cuore pulsante dell'applicazione, e la Login Activity che si occupa dell'accesso degli utenti.

[I] oggetto che può essere "osservato" da altri componenti dell'applicazione. Quando lo stato di un Observable cambia, notifica automaticamente i suoi osservatori, consentendo loro di reagire di conseguenza.

[II] consentono alla View di osservare i dati forniti dal ViewModel e di essere automaticamente aggiornata quando i essi cambiano.

Nella Main Activity si alternano principalmente tre Fragment più uno di debug:

- Locations: In questo Fragment è possibile selezionare l'edificio in cui si vuole cercare la destinazione o i POI (Point Of Interest).
- Navigation: Qui vi è lo strumento di navigazione. È possibile cercare POI, digitare la destinazione, sapere la propria posizione rispetto alla planimetria e vedere la strada da seguire.
- Profile: Qui si possono vedere i propri dati (come User Hash o nome utente) e impostare le proprie preferenze per i percorsi (come evita scale o evita strade affollate).

La Login Activity consta di nessun Fragment a causa della sua semplicità e della sua durata vitale breve. Mantenere la Login Activity come una semplice Activity invece di utilizzare Fragment può semplificare l'interazione, evitando sovraccarichi di interfaccia utente e semplificando la gestione del flusso dell'applicazione in questo specifico contesto.

3.2.1 Motivazione dell'Architettura Scelta

L'architettura dell'applicazione è stata selezionata in questo modo per diversi motivi. Innanzitutto, questa approccio consente di ottenere una maggiore modularità dell'interfaccia utente, facilitando la gestione e la manutenzione della soluzione, che comporta ad una più semplice estensibilità delle funzionalità. I Fragment possono essere sviluppati e mantenuti separatamente, consentendo una migliore separazione delle responsabilità e la riutilizzabilità del codice.
Inoltre, l'utilizzo dei Fragment consente di avere una struttura scalabile dell'applicazione, in cui nuove funzionalità o schermate possono essere facilmente aggiunte come nuovi Fragment senza dover modificare l'architettura di base dell'applicazione.
L'architettura scelta favorisce anche una migliore gestione dello stato dell'applicazione. I Fragment possono gestire autonomamente il proprio stato e comunicare con l'Activity principale o con altri Fragment tramite interfacce o eventi. Ciò permette una gestione efficiente dello stato e una migliore separazione delle responsabilità.

3.3 Requisiti di Sistema

L'implementazione dell'applicazione richiede l'attenzione di diversi requisiti di sistema, sia a livello hardware che software. Questi requisiti sono necessari per garantire il corretto funzionamento dell'applicazione. Lo sviluppo dell'applicazione è stata compiuta solo in Android, perciò di sequito verranno inseriti solo i requisiti per questo OS (Operating System).

3.3.1 Requisiti Hardware

L'applicazione richiede un dispositivo smart con all'interno diversi sensori.
Il principale sensore è il GPS, fondamentale per la localizzazione primaria, in cui viene localizzato l'utente rispetto il pianeta. Questo viene utilizzato per la navigazione outdoor tra due edifici, qualora l'utente dovesse impostare questo percorso, e per una prima localizzazione grossolana.
I secondi sensori necessari sono:

- Accelerometro: Ha la funzione di percepire il movimento attraverso l'accelerazione dell'utente.
- Giroscopio o Magnetometro: Questi hanno la funzione di percepire la velocità angolare e quindi la rotazione dello smartphone. È necessario almeno uno di questi.
- Barometro: Questo strumento serve per capire a quale piano ci si trova, confrontando la pressione del dispositivo e confrontandola con quella di altri dispositivi (beacon).

3.3.2 Requisiti Software

Lo sviluppo della soluzione è stata fatta in ambiente nativo Android. La versione di sviluppo durante lo stage richiedeva come requisito minimo Android 5.0 e raccomandato Android 12. Con l'aggiornamento di Giugno 2023, il requisito minimo diventa la versione 6.0. Secondo le stime di Android Studio, questo diminuisce di poco la percentuale di dispositivi supportati (da 97.7% a 92.4%), in cambio dell'aggiunta di alcune feature, tra cui la possibilità di misurare la distanza tra due punti sulla planimetria e, per gli sviluppatori, di modificare POI.

3.4 Interfaccia Utente

L'interfaccia utente è progettata per essere user-friendly e coerente con le convenzioni di design moderne. L'obiettivo è quello di permettere ad un pubblico più ampio possibile di utilizzare l'applicazione in modo facile e veloce. Si è deciso di mantenere come canone da seguire la pagina web di Navigine per il design grafico, ma cambiare la tonalità in base a quello dell'università degli Studi Bicocca. Di seguito sono descritti alcuni elementi chiave dell'interfaccia utente dell'app:

- Mappa interattiva: La mappa del campus costituisce il cuore dell'interfaccia utente. L'utente può visualizzare la mappa in diversi livelli di zoom e navigare all'interno degli edifici. La mappa visualizza anche i POI, come aule, uffici e servizi, in modo da consentire all'utente di selezionare la destinazione desiderata.

- Campo di ricerca: Un campo di ricerca, posto in alto, consente agli utenti di cercare destinazioni specifiche all'interno del campus. È possibile cercare specifici punti, come l'aula di un professore o un laboratorio, oppure fare una ricerca per categoria, come aule studio o toilette.

- Indicazioni di navigazione: Dopo aver selezionato una destinazione, l'app fornisce indicazioni di navigazione passo-passo per guidare l'utente lungo il percorso ottimale. Le indicazioni possono essere visualizzate come istruzioni grafiche sulla mappa.

- Informazioni sui POI: L'interfaccia utente offre anche informazioni dettagliate sui punti di interesse, come aule, uffici o servizi. Gli utenti possono accedere a orari di apertura, descrizioni, foto e altre informazioni utili relative ai singoli punti di interesse.

3.5 Funzionalità chiave dell'app

La applicazione fornisce diverse funzionalità per la navigazione indoor, che migliorano l'esperienza di navigazione e forniscono informazioni utili agli utenti. L'obiettivo è quello di unire funzionalità avanzate ad un'esperienza intuitiva e fluida.

3.5.1 Posizionamento interno

Una delle funzionalità chiave dell'app per la navigazione indoor nel campus UniMiB è il posizionamento interno. Utilizzando l'API di navigazione selezionata e i sensori presenti nei dispositivi mobili, l'app è in grado di individuare e visualizzare la posizione corrente dell'utente all'interno degli edifici del campus. Questo offre agli utenti un feedback in tempo reale sulla propria posizione mentre si spostano all'interno del campus. La precisione del posizionamento interno dipende dalla qualità dell'API di navigazione utilizzata e dalla disponibilità dei sensori nel dispositivo. Grazie a questa funzionalità, gli utenti possono avere una maggiore consapevolezza della propria posizione e seguire le indicazioni di navigazione con maggiore precisione.

3.5.2 Informazioni degli Edifici

Un'altra funzionalità essenziale dell'app per la navigazione indoor è la fornitura di informazioni dettagliate sugli edifici presenti nel campus UniMiB. Gli utenti possono accedere a una vasta gamma di informazioni, come orari di apertura, piani degli edifici, informazioni di contatto e altre dettagliate informazioni relative agli edifici. Questa funzionalità aiuta gli utenti a comprendere meglio l'ambiente circostante e a identificare specifiche destinazioni all'interno degli edifici. Offrendo informazioni aggiuntive sugli edifici, l'app contribuisce a migliorare l'esperienza degli utenti e a facilitare la loro navigazione nel campus.

3.5.3 Navigazione indoor

La navigazione indoor è la funzionalità principale dell'app per la navigazione indoor nel campus UniMiB. Utilizzando l'API di navigazione selezionata, l'app fornisce indicazioni di navigazione precise per guidare gli utenti lungo i percorsi ottimali all'interno degli edifici. Gli utenti possono selezionare una destinazione desiderata tramite la mappa interattiva o la funzione di ricerca, e l'app genererà un percorso dettagliato che indicherà i passi da seguire. Le indicazioni di navigazione possono essere presentate sia sotto forma di testo, fornendo istruzioni passo-passo, che attraverso istruzioni grafiche sulla mappa. Questa funzionalità permette agli utenti di muoversi facilmente all'interno del campus UniMiB, riducendo il rischio di smarrirsi e ottimizzando il tempo impiegato per raggiungere le destinazioni desiderate.
Queste funzionalità chiave, come il posizionamento interno, le informazioni sugli edifici e la navigazione indoor, lavorano sinergicamente per offrire agli

utenti un'esperienza di navigazione completa ed efficiente all'interno del campus UniMiB. Sfruttando le potenzialità dell'API di Navigine e integrando informazioni dettagliate sugli edifici, l'applicazione si propone di semplificare la navigazione indoor e migliorare la fruibilità dell'ambiente universitario.

3.6 Utilizzo dell'API Navigine

L'API di Navigine è il core di tutta l'applicazione. Essa è stata utilizzata per diversi scopi che sono riportati di seguito:

- Mappatura indoor
- Geolocalizzazione dell'utente
- Navigazione indoor

3.6.1 Mappatura Indoor

Nella demo dell'applicazione è stato utilizzato l'edificio 14[III] dell'università Bicocca come campo di test.

Creazione e Caricamento delle Planimetrie

Le planimetrie dell'edificio sono state realizzate utilizzando Planner5D [21], web site utile per mantenere le proporzioni tra le varie stanze.
Una volta create le planimetrie, i file saranno caricati sul sito web dell'API in uno dei formati disponibili: JPG, PNG o SVG (in questo caso PNG). Per ogni piano, bisogna posizionare il file sulla mappa satellitare dell'edificio e successivamente dargli un nome. Qui si può selezionare se il file caricato fa parte di un edificio già esistente o di un nuovo edificio.[IV]
Qui si riscontrano delle difficoltà dovuto al sistema di posizionamento su mappa troppo minimal: non consente di streachare la planimetria per combaciare con degli angoli dell'edificio, costringendo diverse volte a fare lo screen della mappa satellitare, per poi utilizzare un software esterno (ex. Photoshop o SketchBook) per ridimensionare le mappe correttamente.

[III] chiamato U14 successivamente

[IV] La scelta di dividere le planimetrie in edifici è puramente semantica. È possibile tenere tutti gli edifici in una sola location, ma questo rende difficile il mantenimento.

Dashboard per l'Elaborazione della Planimetria

Dalla dashboard per gli sviluppatori, nella sezione *Location*, è necessario mappare i muri, le stanze e le aree dell'edificio, per poi procedere con POI, WiFi Transmitter e/o Beacon e i differenti routes. Infine per collegare i piani l'uno all'altro, c'è una sezione separata chiamata *Elevation Mode*.

Inserimento degli Edifici nell'Applicazione

Nella sezione Applicazione della Dashboard è possibile creare una o più applicazioni. Ognuna di questa avrà una differente API-Key: necessaria per distinguere eventualmente diverse applicazioni, condividendo lo sviluppatore.
Ogni applicazione così creata consentirà le seguenti funzionalità:

- Team: Funzionalità che consente di creare un proprio gruppo di lavoro e condividere il progetto.
- Location: Questa sezione permette di selezionare le Location da aggiungere all'applicazione che utilizzerà quell'API-Key
- Opzioni: Qui è possibile impostare delle funzionalità senza dover mettere mano al codice, ma direttamente dalla dashboard.

3.6.2 Geolocalizzazione dell'Utente

La RSSI (Received Signal Strenght Indicator) è una misurazione della potenza di un segnale radio, in questo caso tra client e Access Point (Router Wi-Fi o Beacon). La potenza di segnale viene utilizzata per stimare la posizione della sorgente rispetto alla destinazione: maggiore è la potenza del segnale minore sarà la distanza tra i due punti.

Measurement Preprocessor

Le informazioni sui transmitter sono salvate nel file `/logs/transmitters.txt` nel formato `tid, coordinates, type`[V]:

```
1 (38945,2275,F7826DA6-4FA2-4E98-8024-BC5B71E0893E) 238.52 21.3 BEACON
```

[V] tid rappresenta il codice univoco del trasmettitore

Utilizzando queste informazioni, si calcola la distanza stimata dell'utente e si salva nel log `/logs/measurements.log` nel formato `timestamp, tid, rssi, type`

```
1 1541003875242 (56813,7748,F7826DA6-4FA2-4E98-8024-BC5B71E0893E) -82
    BEACON
```

Tutte le misure sono divise in pacchetti. Il preprocessore controlla se le misurazioni nel pacchetto sono valide:

- Se il tipo di segnale è supportato
- Se l'RSSI si trova nei bordi limite

Successivamente le ultime misurazioni vengono aggiunte nel buffer delle misurazioni. La dimensione del pacchetto measuremets è mantenuta limitata.

Algoritmo $\alpha\beta$-Filter

Per aumentare l'accuratezza del posizionamento, viene utilizzato l'algoritmo $\alpha\beta$-**Filter** in post-elaborazione.
Questo tipo di approssimazione utilizza l'integrale della velocità nel tempo e assumendo che la velocità di movimento v resti costante per piccoli intervalli di tempo ΔT[VI]. Viene così previsto lo spostamento x dell'oggetto utilizzando l'equazione:

$$\hat{x_k} = \hat{x_{k-1}} + \Delta T \hat{v_{k-1}}$$

Siccome la velocità v si assume costante, la velocità all'istante successivo sarà uguale a quella precedente:

$$\hat{v_k} \leftarrow \hat{v_{k-1}}$$

Viene così calcolato l'errore tra la previsione e il valore effettivo della posizione:

$$\hat{r_k} \leftarrow x_k - \hat{x_k}$$

Si scelgono così α e β, positivi e piccoli, per correggere la stima della posizione, rispettando le seguenti condizioni:

$$0 < \alpha < 1$$

$$0 < \beta \leq 2$$

[VI] Intervallo tra una misurazione della posizione ed un'altra

$$4 - 2\alpha - \beta > 0$$

La correzione viene effettuata come segue:

$$\hat{x_k} \leftarrow \hat{x_k} + \alpha\hat{r_k}$$
$$\hat{v_k} \leftarrow \hat{v_k} + \frac{\beta}{\Delta T}\hat{r_k}$$

Per ottenere la distanza sorgente-destinazione dalla potenza si utilizza la seguente equazione:

$$P(d) = A - B \log d^2$$

da cui si ricava la distanza:

$$d = \sqrt{\exp\left(\frac{A - P(d)}{B}\right)}$$

Di seguito viene riportato il suo codice in linguaggio C[25]:

```c
int main() {

  float dt = 0.5;
  float xk_1 = 0, vk_1 = 0, a = 0.85, b = 0.005;

  float xk, vk, rk;
  float xm;

  while (1) {

    xm = rand() % 100; // input signal

    xk = xk_1 + (vk_1 * dt);
    vk = vk_1;

    rk = xm - xk;

    xk += a * rk;
    vk += (b * rk) / dt;

    xk_1 = xk;
    vk_1 = vk;

  }
}
```

3.6.3 Navigazione Indoor

Per trovare il miglior percorso si eseguono in ordine questi passaggi:

1. Costruzione di un grafo in coordinate cartesiane.
2. Algoritmo di Kosaraju
3. Algoritmo di Dijkstra

Algoritmo di Kosaraju

L'algoritmo di Kosaraju serve per determinare se un grafo è fortemente connesso.

Definizione 1 *Sia G un grafo orientato e V l'insieme dei suoi vertici. G è fortemente connesso $\iff \forall u, v \in V$ esiste un cammino da u a v.*

Per verificare che il grafo sia fortemente connesso l'algoritmo esegue due DFS Visit del grafo, la prima viene eseguita normalmente, mentre per eseguire la seconda è necessario trasporre il grafo e poi eseguire la DFS dagli ultimi nodi visitati nella prima scansione [24]. Il tempo per l'eseguizione di questo algoritmo è $\Theta(V + E)$.

DFS Visit

Di seguito viene riportato lo pseudocodice dell'algoritmo DFS Visit.

```
DFS(G)
    FOR EACH u ∈ V
        col[u] = White
        π[u] = NIL
    time = 0
    FOR EACH u ∈ V
        IF col[u] == White
            DFS-VISIT(u)

DFS-VISIT(u)
    time++
    d[u] = time
    col[u] = Gray

    FOR EACH v ∈ Adj[u]
        IF col[v] == White
            π[v] = u
            DFS-VISIT(v)

    col[u] = Black
    time++
    f[u] = time
```

Algoritmo di Dijkstra

Verificato che il grafo sia fortemente connesso, si applica l'algoritmo di Dijkstra. Questo viene utilizzato per cercare il cammino minimo tra due nodi di un grafo. Ogni arco ha un peso che varia in base a:

- Lunghezza del percorso tra i due nodi
- Preferenze dell'utente

- Inagibilità o sovraffollamento del percorso

Il tempo di esecuzione dell'algoritmo è $O(|\mathrm{V}|^2 + |\mathrm{E}|)$.

```
DIJKSTRA(G, w, s)
     INIZIALISE-SINGLE-SOURCE(G, s)
     V' = ∅
     Q = V //Q = V - V'
     WHILE Q ≠ ∅
        u = EXTRACT-MIN(Q)
        V' = V' ∪ {u}
        FOR EACH v ∈ Adj[u]
            RELAX(u, v, w)

INIZIALISE-SINGLE-SOURCE(G, s)
    FOR EACH v ∈ V
        v.d = ∞
        v.π = NIL
    s.d = 0

RELAX(u, v, w)
    IF v.d > u.d + w(u, v)
        v.d = u.d + w(u, v)
        v.π = u
```

Capitolo 4

Implementazione e Risultati Ottenuti

4.1 Ambiente e Implementazione

4.1.1 L'Ambiente di Sviluppo

L'ambiente di sviluppo è una componente fondamentale al fine della realizzazione di un'applicazione. La scelta di un IDE (Integrated Development Environment) ottimale fornisce un aiuto fondamentale nella scrittura del codice e nella gestione delle risorse. Come indicato nel precedente capitolo si è scelto di realizzare l'applicazione per un ambiente Android in linguaggio Java[11], pertanto l'IDE selezionato è stato Android Studio [1] Electric Eel 2022.1.1 Patch 2. L'IDE è stato creato da Google per offrire ad ogni sviluppatore uno strumento efficace per la creazione di applicazioni Android, fornendo nello specifico le seguenti funzionalità:

- Editor di Testo: Android Studio, come ogni altro IDE, fornisce un ottimo editor di testo con controlli precisi sulla sintassi, completamento automatico e refractoring. I linguaggi supportati sono Java e il più recente Kotlin [12].
- Debugging: Un aspetto importante nell'ambito della programmazione è la fase di debug. Gli strumenti offerti da Android Studio, ovvero servizi per l'analisi e la correzione degli errori nelle applicazioni, sono fondamentali per ogni sviluppatore.
- Emulatore: Al fine di poter testare le proprie applicazioni, senza la necessità di possedere uno smartphone Android, viene fornito un emulatore personalizzabile in base alla versione e al tipo di dispositivo.

- Designer: Nel contesto delle applicazioni è fondamentale, oltre alla logica del codice, l'aspetto grafico. Android Studio permette di progettare in modo semplice la propria interfaccia grafica, utilizzando un layout editor.
- Supporto Git: L'integrazione con Git[6] permette una gestione e condivisione del progetto efficace su diverse piattaforme, attraverso un sistema di controllo della versione.
- Gradle e Export: Per la gestione delle dipendenze, librerie e build, Android Studio utilizza Gradle [7].
  Gradle semplifica il processo di compilazione, testing e distribuzione delle applicazioni Android, permettendo inoltre la generazione di file APK.

4.1.2 Packages

L'organizzazione di un progetto creato con Android Studio è fondamentale, in quanto suddividere l'applicazione in varie sezioni con ruoli differenti permette una gestione semplice ed immediata.
I package sono strutture organizzative modulari utilizzate per raggruppare e incapsulare risorse o codice dello stesso tipo.
Possono essere considerati come delle vere e proprie cartelle strutturate in modo gerarchico, dove il contenuto di un sotto-package ha una responsabilità diversa da un altro.
La struttura gerarchica è identificabile attraverso la seguente notazione puntata: "com.example.app", dove "com" rappresenta il padre della gerarchia, "example" è un figlio di "com", ma padre di "app" e così via (esempio presente nella figura 4.1). L'utilizzo dei package svolge un ruolo rilevante nella progettazione di un'applicazione, particolarmente quando quest'ultima è composta da molte classi.

4.1.3 La Suddivisione delle Responsabilità

L'applicazione trattata in questo elaborato utilizza in modo estensivo i package, traendo fortemente ispirazione dalla struttura fornita dalla demo di Navigine. Ogni package identifica precisamente il ruolo e la tipologia delle classi Java e permette una buona suddivisione delle responsabilità. Sono presenti, partendo dalla struttura "com.navigine.navigine.demo" visualizzabile nella figura 4.1, i seguenti principali package:

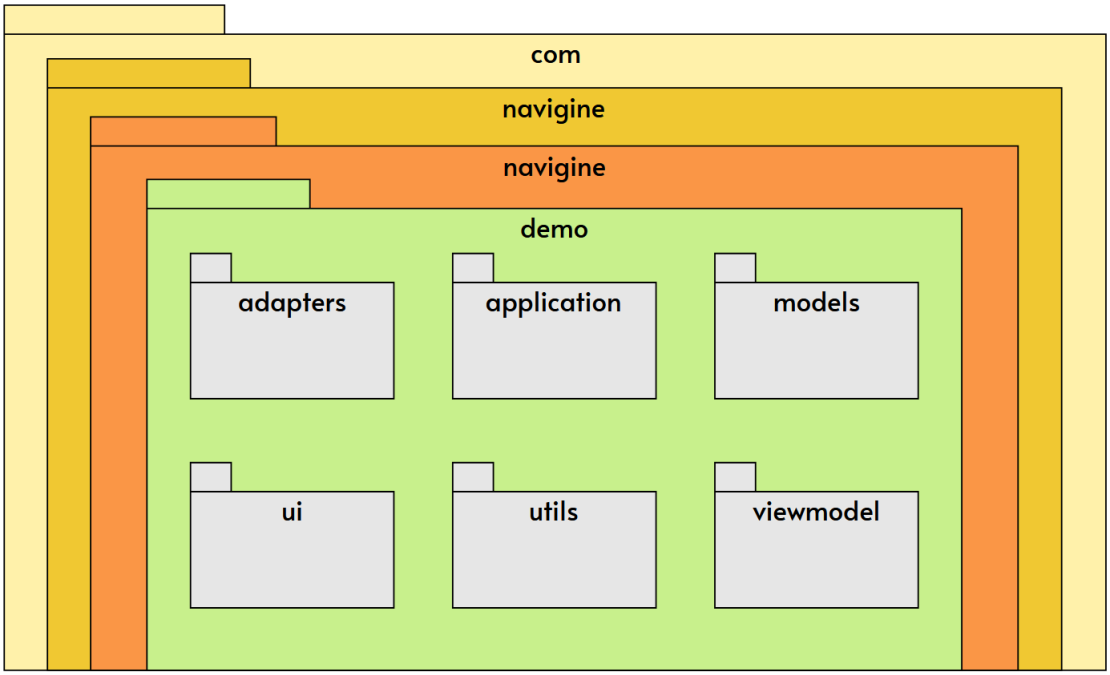

Figura 4.1: Struttura generale dei package descritti dell'applicazione

- Adapters: i sottopackages e le classi contenute all'interno di questo package si occupano di essere degli Adapters, delle classi specifiche per la gestione dell'associazione dei dati all'interfaccia utente, utilizzando classi di base come RecyclerViews e ViewHolders.

- Application: package molto semplice con una classe utilizzata come punto di ingresso dell'applicazione. Essa svolge diverse funzioni importanti durante il ciclo di vita dell'app come la gestione delle notifiche.

- Models: è composto da classi che definiscono la struttura di un tipo di dato. Il package si occupa quindi della rappresentazione dei dati, come quelli che identificano un utente.

- Ui: le risorse e le classi contenute in questo package si occupano della organizzazione dell'interfaccia grafica, in particolare della presentazione. Generalmente in ogni applicazione sono presenti sottopackage che contengono:

  - Activities: componenti principali dell'interfaccia utente (UI) e utilizzate per mostrare generalmente fragments. Nell'app sono presenti due attività, dove la prima si occupa di gestire l'UI della schermata di Login e la seconda del resto dell'applicazione.

  - Fragments: sono componenti modulari che possono essere utilizzati

all'interno di un'attività per creare UI riutilizzabili. Nell'applicazione sono presenti diversi fragment che differenziano la UI in base alle funzionalità offerte.

- Utils: questo package contiene classi che non hanno un vero e proprio ruolo ma forniscono funzioni e strutture utilizzabili da altre classi del progetto.
- ViewModel: il package che evidenzia l'utilizzo del pattern MVVM. La classe al suo interno rappresenta un ViewModel condiviso tra più componenti dell'applicazione e svolge diverse funzioni legate alla gestione del posizionamento dell'utente.

4.2 Le Principali Funzionalità

L'applicazione offre diverse funzionalità ben distinte e intuitive. Sono presenti due principali schermate che vengono mostrate all'utente: la schermata di login, che permette l'accesso ai servizi di navigazione, e la schermata di navigazione in cui l'utente può utilizzare i restanti servizi offerti.

4.2.1 Login Activity

La LoginActivity gestisce l'interfaccia utente della schermata di login. Vengono effettuati controlli su molteplici permessi al fine di poter effettuare la navigazione e forniti all'utente i componenti inizializzati dell'interfaccia.
L'utente può concettualmente effettuare il login inserendo la propria email e password negli appositi form di testo, oppure può registrarsi premendo sull'apposita scritta e utilizzando il menù a tendina. Le funzionalità specifiche di login e registrazione tramite email e password sono attualmente dei mockup, ovvero presenti solo a livello grafico. Il vero e proprio login è effettuabile tramite l'userHash, un codice univoco fornito da Navigine per accedere ai servizi dell'API.

4.2.2 Main Activity

La MainActivity si occupa della gestione e dell'utilizzo dei fragment principali dell'applicazione. È presente una barra di navigazione sulla parte inferiore dello schermo che si occupa di navigare tra i vari fragment.

Locations Fragment

È il fragment che si occupa di offrire all'utente la possibilità di scegliere una location, ovvero un'edificio dove effettuare la navigazione. Le location sono presentate con una lista ed è possibile, utilizzando il campo apposito, ricercare per nome la location desiderata.

Navigation Fragment

Il NavigationFragment è il fragment più importante di tutta l'applicazione in quanto offre all'utente i servizi di navigazione per l'edificio scelto. L'interfaccia utente mostra la mappa dell'edificio con i relativi punti di interesse e la posizione attuale dell'utente, fornendo la possibilità di ingrandire o rimpicciolire la visuale.
Sulla sinistra è presente la scelta del piano, attraverso una lista che modifica la mappa visualizzata in base alla scelta dell'utente, mentre sulla destra, oltre ai bottoni d'ingrandimento, vi è un pulsante che imposta la visuale rispetto alla posizione dell'utente.
Tramite la barra di ricerca, localizzata sulla parte superiore dello schermo, è possibile filtrare i punti di interesse (POI) per tipologia e nome. I POI sono interagibili e forniscono diverse informazioni, con la possibilità di navigarci dalla posizione in cui si è attualmente.
Durante la navigazione è indicata una stima del tempo e della distanza rimanente alla destinazione.

Debug Fragment

Fondamentale per la build sviluppatore, il DebugFragment fornisce importanti informazioni allo stesso. In particolare vengono testati i sensori del dispositivo mobile e la ricezione dei segnali da localizzatori di vario tipo. Questo fragment verrà nascosto durante la creazione della build finale destinata agli utenti.

Profile Fragment

Il ProfileFragment permette la gestione del profilo utente creato tramite la LoginActivity. Sono mostrate informazioni riguardanti l'utente come l'userHash ed un menù dove è si possono esprimere, attraverso degli switch, delle preferenze nei percorsi. Gli switch attualmente, come per il sistema di account, sono solamente dei mockup grafici. Le preferenze sono modificabili

manualmente dallo sviluppatore andando ad alterare i pesi di ogni percorso di navigazione.

4.3 Risultati e Considerazioni

4.3.1 Valutazione dell'applicazione

Concluso il primo ciclo di sviluppo, l'applicazione è stata sottoposta ad un'analisi per verificarne le potenzialità. In particolare, è stato effettuato un approfondito testing della navigazione e della sua affidabilità utilizzando i iBeacon[8] forniti dall'università. Dopo alcune modifiche effettuate ai percorsi su più piani (sono stati aumentati i pesi) attraverso la dashboard di Navigine, l'app si è rivelata affidabile e precisa, permettendo la navigazione tra punti di interesse nell'edificio scelto, con supporto a più piani.

4.3.2 Testing della Localizzazione

La localizzazione dell'utente con Navigine è stata testata utilizzando gli iBeacon di Estimote[4] Developer Preview Kit Model, compatibili con iPhone 4S, 5, 5C, 5S, iPad Mini di terza e quarta generazione, Android 4.3+.
Nello specifico, sono stati considerati in fase di testing beacon fisici, ma anche beacon virtuali, i quali si sono rivelati un'ottima alternativa gratuita.

iBeacon Fisico

L'iBeacon fisico fornisce, se analizzato con applicazioni in grado di raccogliere informazioni sui localizzatori come "Locate Beacon"[13], i seguenti dati:

- UUID
- Major
- Minor
- RSSI
- Accuracy
- Proximity
- Distance (in metri)

UUID, Major e Minor consentono l'identificazione univoca del beacon, mentre i dati rimanenti forniscono informazioni di prossimità e precisione captate dai sensori del localizzatore.

iBeacon Virtuale

L'iBeacon virtuale è implementato attraverso uno smartphone, utilizzando l'applicazione proprietaria di Estimote, per Android e iOS[3], navigando nella sezione "Configuration" e successivamente selezionando "Virtual Beacon".
Al fine di poter essere configurato correttamente, il beacon virtuale necessita di essere identificato con il seguente UUID:

- *8492E75F-4FD6-469D-B132-043FE94921D8*

Se analizzato come descritto precedentemente, l'iBeacon virtuale fornirà lo stesso insieme di dati di un iBeacon fisico.

4.3.3 Confronto con Obiettivi Prefissati

L'obiettivo dell'applicazione è quello di fornire una navigazione affidabile all'interno degli edifici del campus UniMib. Le funzionalità basilari che permettono all'applicazione di essere considerata valida sono state implementate e sono eventualmente estendibili.
La presenza di un sistema di navigazione funzionante tramite punti di interesse risulta essere una soluzione coerente alle premesse esposte.
Attualmente non sono stati implementati approfonditamente il sistema di account, in quanto il login viene effettuato ancora utilizzando l'userHash identificativo fornito da Navigine, ed il sistema di preferenza dei percorsi, che richiede la modifica manuale del peso dei percorsi di navigazione. Queste funzionalità possono essere introdotte agevolmente in sviluppi futuri attraverso servizi esterni come Firebase Authentication [5] e l'estensione della struttura evidenziata dai package.

Capitolo 5

Conclusioni e Sviluppi Futuri

Nello sviluppo di questa tesi, si è esplorata l'importanza della navigazione indoor nel contesto del campus UniMiB e sviluppato un'app di navigazione indoor per agevolare gli spostamenti all'interno degli edifici universitari.
La navigazione indoor svolge un ruolo cruciale nel migliorare l'esperienza degli utenti all'interno del campus universitario. Grazie all'applicazione, gli studenti, il personale e i visitatori sono in grado di orientarsi con maggiore facilità,risparmiando tempo prezioso nella ricerca di aule, uffici o servizi all'interno degli edifici universitari.

5.1 Risultati

Attraverso l'implementazione e la valutazione dell'applicazione di navigazione indoor, si sono osservati diversi risultati positivi. Gli utenti, inclusi gli studenti, il personale e i visitatori, possono individuare con precisione le aule, gli uffici e i servizi desiderati, risparmiando tempo prezioso.
Navigine risulta in una soluzione ottima per un contesto universitario, soprattuto per la documentazione, precisa e rigorosa, e una demo open source disponibile su GitHub.
I risultati ottenuti confermano l'importanza della navigazione indoor in un campus universitario come quello dell'università Bicocca di Milano e dimostrano l'utilità dell'applicazione sviluppata nel migliorare l'efficienza e la comodità degli spostamenti all'interno degli edifici universitari.

5.2 Sviluppi Futuri

In futuro si potrà espandere questa soluzione su diversi aspetti, tra i quali sicuramente:

- Aggiunta di una versione alle piantine che verranno aggiornate qualora l'applicazione rilevi che quelle scaricate siano obsolete.
- Integrare i POI con informazioni e funzionalità aggiuntive anche in collaborazione con una API dell'università, come prenotazione del posto in aula studio, informazioni dei professori nelle loro aule e orari della segreteria.
- Inserimento di QR-Code per ogni POI che, se inquadrato, apre l'applicazione e inizia la navigazione verso il punto selezionato.
- Modifica dinamica e personale dei pesi degli archi in base alle preferenze utente (come "Evita strade affolate" o "Evita le scale")
- Aggiunta di un sistema di account più user-friendly, con l'utilizzo di email e password.

Appendice

Questa appendice ha il compito di essere un riferimento immediato per futuri sviluppatori, i quali vorrebbero poter implementare un'applicazione di navigazione indoor utilizzando le procedure, tecniche e strutture evidenziate in questo elaborato. In seguito è presentata la procedura completa e sintetizzata per l'utilizzo dei servizi di Navigine.

Account e Procedure Preliminari

L'utilizzo di Navigine richiede necessariamente la creazione e l'utilizzo di un account sull'apposita piattaforma proprietaria [19].
La registrazione è effettuata tramite email e password, con l'accettazione dei termini di servizio di Navigine.
Una volta effettuata la registrazione saranno necessarie alcune risorse al fine di poter utilizzare i servizi offerti da Navigine, in particolare:

- Planimetria Mappe: Navigine non fornisce un sistema di realizzazione di mappe, quindi andranno preparate in precedenza utilizzando software specifici [21].
- Beacons o Localizzatori: necessari per poter localizzare l'utente all'interno del proprio edificio. Navigine supporta diversi tipi di Beacon e sistemi di localizzazione che saranno approfonditi in seguito.

Upload e Impostazioni Mappe

Completate le procedure preliminari, si può iniziare ad analizzare ed utilizzare i servizi offerti da Navigine. In primo luogo, al fine di poter utilizzare la planimetria dell'edificio, bisogna effettuare l'upload delle immagini sulla piattaforma.
Navigine supporta tre distinti formati utilizzabili come location, ovvero:

- Immagine PNG: formato di immagine generalmente di alta qualità e con supporto alla trasparenza.
- Immagine JPG: formato di immagine di media qualità compresso e poco oneroso in termini di spazio.
- Immagine SVG: formato di immagine vettoriale che utilizza una rappresentazione matematica. Questo permette, tra numerosi vantaggi, di non perdere qualità nell'ingrandimento della immagine.

La dimensione massima di ogni immagine non deve superare i 5MB.
Al momento dell'upload sarà possibile modificare numerosi parametri, utilizzando anche le impostazioni avanzate fornite. In seguito un elenco delle impostazioni fondamentali:

- Name: il nome della location, ovvero del proprio edificio.
- Floor Name: il nome del piano caricato. Utile nel caso si voglia caricare un edificio composto da più piani.
- Address: indirizzo della location. Una volta inserito sarà possibile visualizzare la location sulla mappa presente sulla destra delle impostazioni e posizionare a piacimento l'immagine della planimetria, alterando ulteriori impostazioni come longitudine e latitudine.

Gestione degli Edifici

L'ambiente di dashboard fornito da Navigine permette una approfondita gestione di edifici e piani, fornendo allo sviluppatore numerosi strumenti al fine di progettare una navigazione funzionale e di semplice utilizzo.
Gli strumenti sono accessibili attraverso una barra presente sulla parte superiore dell'interfaccia dove è possibile creare nuove location o piani come descritto nella sezione precedente, ma soprattutto alterare lo stato della mappa utilizzando i seguenti strumenti:

- Transmitters: è possibile aggiungere sulla propria mappa la posizione dei Beacon/localizzatori.
  La piattaforma supporta numerosi tipi di localizzatori tra cui iBeacon, Wifi, Wifi RTT, Eddystone, BLE o Locators. Nella realizzazione della applicazione descritta in questo elaborato il servizio è stato testato utilizzando dei iBeacons[8].

- Barriers: permette di delimitare le zone percorribili utilizzando delle barriere, le quali sostanzialmente sono dei muri virtuali.
- Routes: strumento fondamentale in quanto permette di tracciare, utilizzando semplicemente il proprio mouse, tutti i percorsi di navigazione sulla propria planimetria. Ogni percorso può essere modificato successivamente alla creazione, con la possibilità di modificare il tipo di percorso, ovvero bidirezionale, forward e backward , oppure andando ad alterare i pesi, in modo da implementare per esempio, delle preferenze di navigazione, assegnando un peso maggiore ad un percorso meno preferito. Le modifiche vengono effettuate deselezionando tutti gli strumenti attivi e cliccando sopra un percorso.
- Venues: sono i punti di interesse (POI) inseribili nella location. Ogni POI possiede diverse proprietà che possono essere modificate come il nome, la classe (ovvero l'icona che appare sulla mappa) o la descrizione. Le venues quindi offrono un discreto sistema di personalizzazione dei punti di interesse.
- Elevation: permette la gestione dei percorsi di navigazione per più piani. Verrà approfondito in seguito.
- Zones: è possibile inserire delle zone con colori differenti sulla mappa.

Ogni elemento è posizionabile e modificabile attraverso l'utilizzo del mouse e della sezione "Edit" presente sulla barra nella parte superiore della dashboard.

Gestione dei Piani

La gestione dei piani offerta dalla piattaforma si presenta molto semplice ma inizialmente non intuitiva.
Deselezionando tutti gli strumenti in uso e premendo su uno spigolo (intersezione di più percorsi o termine di un percorso), tracciato attraverso lo strumento Routes, sarà possibile aggiungere un punto di elevazione o un punto di uscita/entrata dell'edificio.
Un punto di elevazione è identificabile attraverso la spunta selezionata su "Elevation" mentre se selezionata su "Entry/Exit" identificherà un'entrata/uscita. Inoltre sono entrambi identificati ulteriormente da un nome e da un'icona verde presente sulla mappa.
Una volta inseriti molteplici punti di elevazione sarà possibile utilizzare lo strumento Elevation.

Quest'ultimo permette di tracciare in modo analogo allo strumento Routes dei percorsi che, in questo caso, andranno a collegare i punti di elevazione su piani differenti.
E' consigliato dare ai percorsi realizzati con lo strumento Elevation un peso sufficientemente maggiore rispetto ai percorsi tracciati normalmente.

Integrazione dei Servizi

Al fine di importare nell'applicazione le location realizzate sulla piattaforma di Navigine sarà necessario ottenere l'userHash dell'account, ovvero un codice univoco a dodici caratteri. Esso è reperibile accedendo alla dashboard, navigando sulla barra a sinistra presente nell'interfaccia. Premendo sulla voce "Profile" saranno disponibili alcune informazioni del proprio account tra cui l'userHash.
L'userHash verrà quindi utilizzato come identificativo, permettendo l'accesso a tutti i contenuti presenti sull'account. È possibile verificare la funzionalità e la correttezza dell'utilizzo degli strumenti forniti da Navigine utilizzando l'applicazione demo fornita su GitHub [20] per Android e iOS, dove è anche presente il necessario per integrare l'API e l'SDK (Software Development Kit) di Navigine nella propria applicazione.
In seguito riportato un'esempio di codice per l'integrazione dell'userHash nell'applicazione demo:

```
1  // Esempio della modifica
2  // Questa classe gestisce le infomazioni di accesso dell'utente (che
       sono poi anche mostrate nel layout della visualizzazione del profilo
       )
3  // Al momento sono presenti, eccetto per l'userHash, dei dati fittizi.
4  public class UserSession {
5
6      public static String USER_HASH       = "0000-AAAA-0000-AAAA";
7      public static String USER_NAME       = "user";
8      public static String USER_COMPANY    = "company";
9      public static String USER_EMAIL      = "default@mail.com";
10     public static String USER_AVATAR_URL = "";
11     public static String LOCATION_SERVER = BuildConfig.
       DEFAULT_SERVER_URL;
12 }
```

Bibliografia

[1] Android Studio. *Android Studio and App Tools*. URL: https://developer.android.com/studio (visitato il 05/07/2023) (cit. a p. 27).

[2] API_Typing. *API In Informatica: Cosa Sono E Che Vantaggi Portano - DigitalDojo.it*. Section: Programmazione. 9 Giu. 2022. URL: https://digitaldojo.it/api-informatica/ (visitato il 21/06/2023) (cit. a p. 6).

[3] Estimote. *Estimote App*. URL: https://apps.apple.com/us/app/estimote/id686915066 (visitato il 07/07/2023) (cit. a p. 33).

[4] Estimote. *Estimote UWB Beacons*. URL: https://estimote.com/ (visitato il 07/07/2023) (cit. a p. 32).

[5] Firebase Authentication. *Firebase Authentication*. URL: https://firebase.google.com/docs/auth?hl=it (visitato il 05/07/2023) (cit. a p. 33).

[6] Git. *Git*. URL: https://en.wikipedia.org/w/index.php?title=Git&oldid=1162889449 (visitato il 05/07/2023) (cit. a p. 28).

[7] Gradle. *Gradle Build Tool*. URL: https://gradle.org/ (visitato il 05/07/2023) (cit. a p. 28).

[8] iBeacon. *An iBeacon primer for indoor localization: demo abstract*. URL: https://dl.acm.org/doi/10.1145/2674061.2675028 (visitato il 04/07/2023) (cit. alle pp. 32, 37).

[9] IBM_API_Intro. *Cosa è un'API (Application Programming Interface) | IBM*. URL: https://www.ibm.com/it-it/topics/api (visitato il 23/06/2023) (cit. a p. 6).

[10] IndoorAtlas. *IndoorAtlas*. IndoorAtlas. URL: https://www.indooratlas.com/platform/ (visitato il 23/06/2023) (cit. a p. 11).

[11] Java. *Java (programming language)*. URL: https://en.wikipedia.org/w/index.php?title=Java_(programming_language)&oldid=1160904541 (visitato il 05/07/2023) (cit. a p. 27).

[12] Kotlin. *Kotlin (programming language)*. URL: https://en.wikipedia.org/w/index.php?title=Kotlin_(programming_language)&oldid=1163315849 (visitato il 05/07/2023) (cit. a p. 27).

[13] Locate Beacon. *Locate Beacon*. URL: https://apps.apple.com/us/app/locate-beacon/id738709014 (visitato il 07/07/2023) (cit. a p. 32).

[14] MapsIndoors. *Amazing indoor maps solutions with MapsIndoors*. URL: https://www.mapspeople.com/mapsindoors (visitato il 21/06/2023) (cit. a p. 7).

[15] Mazemap. *Digital Wayfinding & Timetable Integration for Universities*. URL: https://www.mazemap.com/industries/educational-institutions (visitato il 22/06/2023) (cit. a p. 9).

[16] Mazemap_heatmap. *Occupancy Monitoring with Heatmaps*. URL: https://www.mazemap.com/solutions/heat-maps (visitato il 22/06/2023) (cit. a p. 9).

[17] MVVM_Pattern. *Android MVVM Design Pattern | DigitalOcean*. URL: https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern (visitato il 23/06/2023) (cit. a p. 16).

[18] Navigine. *Open Source Indoor Positioning System | Indoor Positioning Algorithm by Navigine*. URL: https://navigine.com/open-source/ (visitato il 21/06/2023) (cit. a p. 12).

[19] Navigine Developers. *Indoor Mapping Software | Indoor Navigations*. URL: https://navigine.com/developers/ (visitato il 02/07/2023) (cit. a p. 36).

[20] Navigine GitHub Account. *Navigine GitHub Account*. URL: https://github.com/Navigine (visitato il 03/07/2023) (cit. a p. 39).

[21] Planner5D. *Planner 5D: House Design Software | Home Design in 3D*. URL: https://planner5d.com/ (visitato il 27/06/2023) (cit. alle pp. 21, 36).

[22] Situm. *Indoor Positioning System*. Situm. URL: https://situm.com/en/technology/indoor-positioning-system/ (visitato il 22/06/2023) (cit. a p. 10).

[23] Steerpath. *Indoor navigation & Maps - Steerpath*. URL: https://www.steerpath.com/products/maps-indoor-navigation (visitato il 21/06/2023) (cit. a p. 9).

[24] Wikipedia. *Componente fortemente connessa*. In: *Wikipedia*. Page Version ID: 130705264. 28 Nov. 2022. URL: https://it.wikipedia.org/w/index.php?title=Componente_fortemente_connessa&oldid=130705264 (visitato il 04/07/2023) (cit. a p. 25).

[25] Wikipedia. *Indoor-Positioning-And-Navigation-Algorithms/standalone_ algorithms at master · Navigine/Indoor-Positioning-And-Navigation-Algorithms*. GitHub. URL: https://github.com/Navigine/Indoor-Positioning-And-Navigation-Algorithms (visitato il 28/06/2023) (cit. a p. 24).