

ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA  
CAMPUS DI CESENA

---

---

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA  
Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

## **Visualizzazione real-time di dati ambientali su mappe: il complesso Navile come caso di studio**

Elaborato in:  
Programmazione di Sistemi Mobile

**Relatore:**  
**Dott.ssa Catia Prandi**  
**Co-Relatore:**  
**Dott. Gianni Tumedei**

**Presentata da:**  
**Ines Fraccalvieri**

**Sessione II**  
**Anno Accademico 2022-2023**



*Ai miei nonni,*

*So che oggi, come ogni giorno, sarete con me,  
anche chi in questo momento è lassù.*

*Questa Tesi è per voi ...*



# Introduzione

AlmaMap Mobile è un'applicazione, scaricabile su dispositivi mobili con sistema operativo iOS, che nasce con l'intento di promuovere la consapevolezza per un processo decisionale informato nel campus di Navile dell'Università di Bologna. Questa tesi documenta lo sviluppo dell'applicazione che si interfaccia con il sistema AlmaMap già esistente per reperire i dati relativi alle mappe e ai parametri ambientali all'interno del campus. Il server di AlmaMap comunica a sua volta con il sistema di Building Information Modeling (BIM) installato nel campus; esso è il processo di creazione e mantenimento di una rappresentazione digitale di un bene fisico e delle sue caratteristiche. Sfruttando il BIM, il personale del campus Navile è in grado di accedere a dati in tempo reale come temperatura ambiente, umidità, composizione dell'aria e altro ancora, semplificando così l'ottimizzazione dell'utilizzo delle risorse ambientali. Il mantenimento dei parametri in tempo reale è reso possibile attraverso l'utilizzo di sensori IoT, installati anche nel Distretto Navile dell'Università di Bologna. Si aprono così interessanti possibilità di combinazione dei dati rilevati in tempo reale dai sensori con un algoritmo di calcolo del comfort, dando agli utenti la possibilità di vederne in anticipo il livello di comfort dell'ambiente. Questa interfaccia tra AlmaMap e sistema BIM è il cuore pulsante di tutto il processo, che permette di fornire dati sempre in tempo reale su tutte le zone del campus.

Nel corso di questa tesi vedremo, suddivisi nei capitoli di appartenenza, gli aspetti fondamentali su cui si basa AlmaMap. Nel primo capitolo vedremo i vari mattoncini che vanno a formare il sistema, ovvero i singoli elementi

presi in causa, tutta la parte del contesto e background di AlmaMap. Verranno trattati diversi argomenti, i principali saranno: Smart Environment, Building Information Modeling e Data Visualization. Nel secondo capitolo approfondiremo l'analisi e la progettazione dell'app, per avere un quadro generale su come verrà progettata e sviluppata l'applicazione, in cui verranno esposte le scelte attuate nell'ambito del design, architettura e tecnologie utilizzate. Si conclude con lo sviluppo proprio ed effettivo che appartiene al terzo ed ultimo capitolo, dove verranno mostrate le scelte fatte per quanto riguarda l'implementazione del codice.

# Indice

<b>Introduzione</b>	<b>i</b>
<b>1 Contesto e Background</b>	<b>1</b>
1.1 Smart Environment . . . . .	1
1.1.1 Mobile Computing . . . . .	3
1.1.2 Sensori . . . . .	4
1.1.3 Smart Buildings . . . . .	4
1.1.4 Applicazioni dello Smart Environment . . . . .	5
1.2 Building Information Modeling . . . . .	8
1.2.1 Dimensioni del BIM . . . . .	10
1.2.2 Information Model . . . . .	12
1.2.3 BIM - API . . . . .	12
1.3 Data Visualization . . . . .	14
1.3.1 Dati in tempo reale . . . . .	16
1.3.2 Human-Computer Interaction . . . . .	17
1.3.3 Mappe . . . . .	18
1.3.4 Mappe e Data Visualization . . . . .	19
1.3.5 Mappe e Dati in tempo reale . . . . .	20
<b>2 Analisi e Progettazione</b>	<b>23</b>
2.1 Analisi . . . . .	23
2.1.1 Requisiti funzionali . . . . .	27
2.1.2 Requisiti non funzionali . . . . .	29
2.2 Design . . . . .	30

2.2.1	Architettura . . . . .	31
2.2.2	Design Dettagliato . . . . .	32
2.2.3	Tecnologie utilizzate . . . . .	34
2.2.4	Mockup e idee di progettazione . . . . .	37
<b>3</b>	<b>Sviluppo</b>	<b>39</b>
3.1	Software e ambiente di sviluppo . . . . .	39
3.2	Implementazione . . . . .	40
3.2.1	Organizzazione del codice . . . . .	40
3.2.2	Database . . . . .	41
3.2.3	API . . . . .	42
3.2.4	Mappa interattiva . . . . .	43
3.2.5	Legenda . . . . .	49
3.2.6	Informazioni e azioni contestuali . . . . .	50
3.2.7	TabView . . . . .	52
3.2.8	Visualizzazione rapida del campus . . . . .	52
3.2.9	Visualizzazione delle Informazioni degli Spazi . . . . .	55
	<b>Conclusioni</b>	<b>59</b>
	<b>Bibliografia</b>	<b>61</b>

# Elenco delle figure

1.1	Ubiquitous computing interactions . . . . .	2
1.2	Categorie dello Smart Building . . . . .	5
1.3	Fasi di sviluppo BIM . . . . .	10
1.4	Dimensioni del BIM . . . . .	11
1.5	Variazione media della temperatura . . . . .	16
2.1	Planimetria Distretto Navile . . . . .	26
2.2	Diagramma dei casi d'uso completo . . . . .	28
2.3	Pattern MVVM . . . . .	32
2.4	Parametri per calcolo del comfort . . . . .	35
2.5	Mockup pagina della (1) mappa e della (2) legenda . . . . .	38
2.6	Mockup pagina degli (1) edifici e dei piani, e pagina dei (2) sensori . . . . .	38
3.1	Diagramma ER . . . . .	41
3.2	Esempio di come le entità vengono definite nel database . . . . .	43
3.3	Implementazione richiesta dei dati tramite API . . . . .	44
3.4	Gestione del mancato collegamento al server . . . . .	45
3.5	Implementazione del server e gestione richieste HTTP . . . . .	45
3.6	Mappa esterna del campus . . . . .	46
3.7	Mappa interna del campus . . . . .	47
3.8	Funzione per ingrandire l'immagine . . . . .	48
3.9	Funzione per rendere l'immagine interattiva . . . . .	48
3.10	Elemento legenda definito in Core Data e XCode . . . . .	49

3.11 Collegamento con database per definizione della legenda . . . . .	50
3.12 Un elemento della legenda, che mostra il segnalino, lo sfondo degli elementi e i nomi della leggenda sia in italiano che in inglese. . . . .	50
3.13 Pagina per la visualizzazione della legenda. . . . .	51
3.14 TabView visualizzata dall'utente . . . . .	52
3.15 Implementazione TabView in XCode . . . . .	53
3.16 Visualizzazione rapida degli edifici del campus . . . . .	53
3.17 Visualizzazione rapida dei piani del campus . . . . .	54
3.18 Visualizzazione rapida degli spazi del campus . . . . .	56
3.19 Implementazione della lista degli edifici . . . . .	56
3.20 Implementazione della riga dell'edificio . . . . .	57
3.21 Implementazione della lista degli spazi . . . . .	57
3.22 Visualizzazione dei dati raccolti dai sensori . . . . .	58

# Capitolo 1

## Contesto e Background

### 1.1 Smart Environment

Si possono trovare diversi modi per definire uno *smart environment*, ma è generalmente descritto come un ambiente fisico in cui le tecnologie consentono alle persone di sperimentare ed interagire con lo spazio e i dati, mantenendo collegati i computer e tutti i dispositivi considerati *smart*, ovvero intelligenti, alle nostre attività quotidiane. Con ambiente intelligente si può individuare una stanza, un edificio o un intero complesso di infrastrutture [1].

Possiamo inoltre definire l'*ubiquitous computing* come integrazione funzionalità di connettività in tutti gli oggetti nel nostro ambiente in modo che possano interagire tra loro ed automatizzare le attività di routine. Si vogliono dunque integrare microprocessori ad oggetti che vengono utilizzati nella nostra quotidianità per farli comunicare in modo efficace e veloce mantenendoli sempre attivi e connessi alla rete [2].

Gli studi della tecnologia ci hanno portato a voler mantenere tutti i dispositivi connessi tra di loro con il concetto di "*Ovunque, ad ogni ora, in ogni posto e con ogni dispositivo*", vista come l'interazione tra ambiente fisico, persone, sensori e dispositivi. Le persone vivono in un ambiente fisico e indossano o trasportano dispositivi mobili, mentre tutto ciò che ci circonda è composto da oggetti che possono essere dotati di sensori. Esiste un ambito

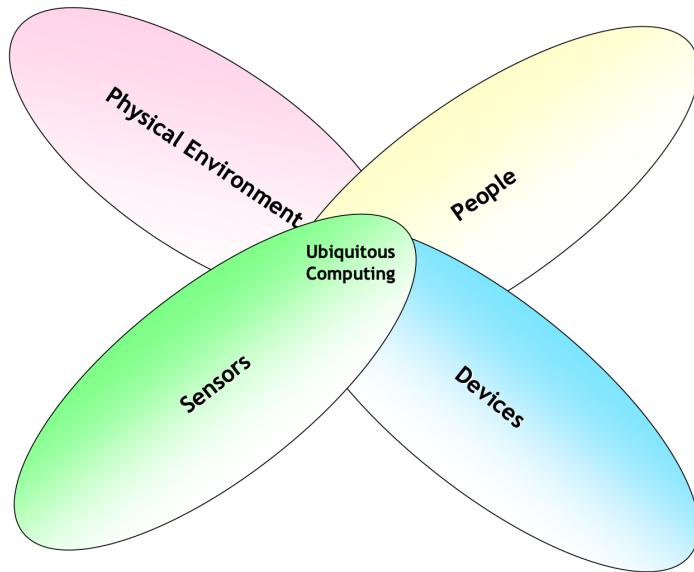


Figura 1.1: Ubiquitous computing interactions

di ricerca in cui si studiano i principi e le metodologie per la creazione di ambienti intelligenti, noto come *ambient computing* o intelligenza ambientale. Possiamo utilizzare l’ambient computing per aiutarci a trasformare enormi volumi di dati in indicazioni utili e interpretabili per facilitare la nostra vita. Partendo dall’ambiente fisico ci permette di creare stanze in cui i dispositivi interagiscono tra loro, adattandoli alle preferenze e alle esigenze di ogni utente. Questa tipologia di ambiente viene definita come smart environment. Sono state superate numerose sfide per poter realizzare ambienti intelligenti. Si è cercato di unire diversi fattori molto importanti per la realizzazione di configurazioni dinamiche dei dispositivi con una estesa scalabilità; tra queste troviamo l’operatività Multi-Part, Multi-Device e Multi-Vendor.

Uno smart environment può essere classificato in base a tre principali approcci [3]:

- Technology-driven: che trasforma gli oggetti e gli spazi comuni integrandoli con sensori IoT che comunicano attraverso Internet, memoriz-

zano le informazioni in un cloud e successivamente vengono visualizzati e mostrati agli utenti.

- Smart city: un ambiente intelligente può essere visto come una città intelligente su piccola scala, che ospita molti dispositivi e connessioni, fornisce una serie di funzionalità e offre agli utenti innumerevoli servizi.
- Promuovere la sostenibilità: un ambiente intelligente ha lo scopo di fornire una migliore qualità di vita agli utenti. Controllando il proprio consumo energetico e l'efficienza dei dati ambientali raccolti, riduce gli sprechi e utilizza le risorse in modo più efficace.

### 1.1.1 Mobile Computing

Al giorno d'oggi si ha l'introduzione di una nuova figura: l'utente *nomade*. La tendenza, infatti, è quella di introdurre dispositivi diffusi nell'ambiente e dispositivi mobili o indossabili, così da mantenere l'utente sempre connesso. Di conseguenza i dispositivi necessitano di maggiori capacità, introducendo così i telefoni cellulari con il collegamento a Internet, memorie di archiviazione e capacità di comunicazione. In questo caso si ha una cooperazione esplicita tra utente e Smart Environment, in quanto l'utente interagisce con il dispositivo mobile per eseguire un compito che va ad influire sull'ambiente che ci circonda.

Se il dispositivo è indossato dall'utente o utilizzato in maniera implicita, si passa nel campo del wearable computing. L'obiettivo è quello di connettere l'utente a un ambiente personale adatto a colui che ne fa uso. Possiamo definite quindi gli *Smart Objects* come oggetti in grado di connettere le persone e l'ambiente [4]. Al giorno d'oggi sono molteplici gli oggetti che usiamo nella nostra quotidianità, come ad esempio gli Smart Speakers, altoparlanti dotati di software avanzati come gli assistenti virtuali; Lampadine intelligenti, danno la possibilità di essere gestite da remoto dall'utente, dunque anche quando ci si trova fuori casa; gli Smartwatch, veri e propri orologi che contengono di-

verse funzionalità come i contapassi, visualizzazione delle notifiche e sensori per la misurazione dei battiti cardiaci.

### **1.1.2 Sensori**

Ciò che ci fornisce il collegamento tra il mondo fisico e quello digitale sono i sensori; essi, utilizzati in grandi quantità, catturano le caratteristiche dell’ambiente circostante. Si potrebbe dunque definire il sensore come *un dispositivo che percepisce una proprietà fisica e trasmette il risultato a una misurazione. Un sensore mappa il valore di alcuni attributi ambientali a una misurazione quantitativa*[5]. I sensori che si trovano nell’ambiente acquisiscono informazioni e dati dall’ambiente che li circonda, vengono elaborati e successivamente contestualizzati e mostrati all’utente.

### **1.1.3 Smart Buildings**

Per quanto riguarda gli edifici intelligenti, uno studio [6] ha stabilito che le caratteristiche possono essere classificate nelle seguenti aree:

- Risposta climatica: è la capacità di rispondere alle condizioni climatiche esterne e di conseguenza identificare il miglior profilo operativo, diminuendo al minimo la domanda di energia e generare energia rinnovabile. Le principali tecnologie utilizzate per questa categoria sono i sistemi IoT e i servizi di previsioni meteorologiche.
- Grid response: è la capacità di un edificio di compiere azioni in conseguenza ai segnali provenienti dalla rete, solitamente con l’obiettivo di massimizzare l’efficienza a scala di quartiere o di città. Ciò è possibile grazie alla generazione di energia rinnovabile e alle infrastrutture di misurazione avanzate.
- User response: un edificio può offrire molte funzionalità che hanno un impatto diretto sulla vita quotidiana degli utenti, e le loro interazio-

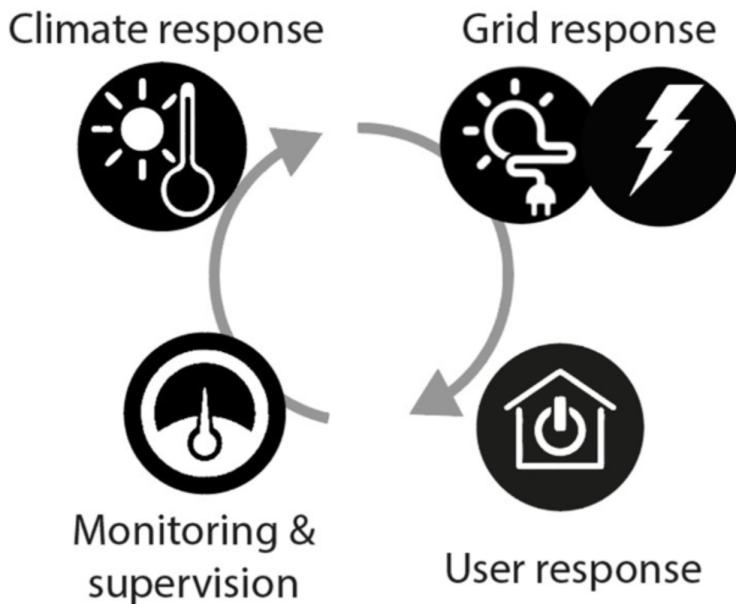


Figura 1.2: Categorie dello Smart Building

ni con il sistema possono a loro volta offrire dati che possono essere utilizzati per migliorare manualmente o automaticamente il sistema.

- Monitoraggio e supervisione: è la capacità di effettuare il monitoraggio in tempo reale delle operazioni dell'edificio e del comportamento dei suoi utenti, con la possibilità di eseguire attività di manutenzione e identificare immediatamente i guasti nel sistema.

#### 1.1.4 Applicazioni dello Smart Environment

Negli ultimi anni sono state sviluppate diverse applicazioni *Smart Environments* in diversi domini. Esse utilizzano sistemi di Context Management e schemi di dati. I vantaggi che apporta l'impiego di tale applicazione, sono in termini di sostenibilità, produzione di energia rinnovabile e interazione con l'utente. Si prova quindi a testare tutte queste tecnologie all'interno dei campus universitari portando alla realizzazione dei cosiddetti *smart campus*,

con l'obiettivo di migliorare l'esperienza di studio e la condivisione degli spazi comuni. I campus universitari sono adatti a testare queste tecnologie grazie alla disponibilità di utilizzo di sofisticate tecnologie di smart environment e la presenza di una comunità studentesca dotata di dispositivi intelligenti e disposti a partecipare ai progetti di studio. In tali ambienti, il ruolo degli utenti sta diventando sempre più rilevante, passando da beneficiari passivi di servizi a partecipanti attivi, aiutando così a migliorare sempre di più l'esperienza di chi si trova all'interno di un ambiente intelligente.

Gli autori di *Campus Energy Education Dashboard* (CEED)[7] hanno creato un sistema che raccoglie e visualizza i dati sul consumo di energia per migliorare l'efficienza energetica e aumentare la consapevolezza degli utenti del campus. l'obiettivo è in particolare quello di monitorare la vivibilità delle aule.

Con l'evoluzione della pandemia di Covid-19 iniziata nel 2019, si è spinto molto sulla ricerca e sullo sviluppo nella realizzazione di infrastrutture per monitorare e rilevare le distanze fisiche tra le persone negli spazi indoor. Questo controllo è stato fondamentale all'interno dei campus, perché si aveva la presenza di molta gente che proveniva da diverse città, e il rischio di diffusione del virus era più elevato. Uno dei dati che più si studiavano era il numero di dispositivi connessi ad un access point, che permette di fornire una stima sul numero di persone presenti all'interno di un'aula [8].

Per dimostrare le capacità di un ambiente intelligente basato sull'IoT, Chiara Ceccarini [9] ha progettato e sviluppato un sistema smart campus presso l'Università di Bologna, sede di Cesena, che utilizza sensori per raccogliere dati in tempo reale su vari parametri ambientali. Sono stati condotti diversi sondaggi e hanno mostrato che la comunità del campus comprendeva il potenziale del concetto in termini di consapevolezza ed era disposta a contribuire con idee e progetti da inserire nell'infrastruttura.

La *Georgia Tech Aware Home* [10] fu uno dei primi e più completi di questi

progetti smart home; era parecchio avanzato per i tempi in cui venne realizzato. La Aware Home Research è dedicata all'esplorazione multidisciplinare delle tecnologie e dei servizi in casa. Questo progetto prevede diverse iniziative: servizi per le persone anziane, strumenti per la famiglia, applicazioni per aiutare gli operatori sanitari di bambini con disabilità dello sviluppo, e molti altri ancora. Alcuni servizi sviluppati sono stati:

- Digital Family Portrait: supporto per la comunicazione familiare.
- Memory Mirror: per la gestione dei farmaci ed utilizzando la visione artificiale si cerca di stimare il rischio dell'anziano di cadere in situazioni naturali.
- AudioNotes: centro messaggi per la famiglia.
- Baby Steps: un'applicazione per aiutare i genitori a monitorare i progressi di sviluppo del loro bambino.
- Pervasive Dietary Advisor: che monitora la salute delle persone con diabete di tipo 2 dopo che hanno lasciato l'ospedale.
- Smart Energy: soddisfare le esigenze rispetto al consumo energetico dei vari elettrodomestici.

A testimoniare l'importanza crescente dello smart environment, un progetto europeo lanciato nel 2011, che tutt'ora viene portato avanti, è “Smart Cities and Communities” [11], una partnership sostenuta dalla Commissione Europea che riunisce città, industrie, piccole imprese, banche e altri stakeholders. Lo scopo è quello di finanziare le località europee che si distinguono per la riduzione dei consumi e la pianificazione di uno sviluppo sostenibile; parliamo di città che riescono a trovare soluzioni integrate e più sostenibili alle sfide del contesto urbano che provengono da diversi settori, come quelli legati all'energia, la mobilità, i trasporti e le ICT.

Le tecnologie urbane intelligenti possono dare un contributo significativo allo sviluppo sostenibile dei Paesi Europei. Una smart city è un'entità che

utilizza efficacemente le ICT per integrare le esigenze della propria comunità, in termini di energia e altre utenze come protezione ambientale, mobilità e trasporti oltre a servizi per i cittadini (sanità, istruzione, servizi di emergenza, ecc.). Ciò aumenterebbe anche la diffusione di tecnologie e soluzioni intelligenti nelle comunità rurali, contribuendo così anche allo sviluppo delle imprese. Sono diversi i criteri di valutazione della *smartness* [12] delle città europee: Smart Economy, diffusione di modelli di business innovativi; Smart People; Smart Governance, partecipazione e trasparenza sono al centro della smart city; Smart Mobility, sviluppo di un sistema di trasporto moderno e sostenibile; Smart Environment, gestione delle risorse naturali e la cura del patrimonio ambientale; Smart Living, ovvero la dimensione che si concentra sulla qualità della vita.

## 1.2 Building Information Modeling

La rappresentazione digitale delle caratteristiche fisiche e funzionali di una struttura è definita come Building Information Modeling (BIM). E' una risorsa condivisa per ottenere informazioni su una struttura che farà da base per le decisioni da dover prendere [13]. BIM cerca di far collaborare le diverse parti del ciclo di vita del progetto per poter inserire, estrarre, aggiornare o modificare le informazioni contenute al suo interno.

Può essere, inoltre, inteso anche come un processo oltre che un software, assumendo significati diversi in base al punto di vista da cui si guarda il progetto:

- Il progetto: tutti i dati vengono forniti e condivisi dai partecipanti al progetto.
- I partecipanti: è il processo per la definizione del progetto, per concepire, progettare e gestire la struttura.
- Team: BIM rappresenta la progettazione integrata, che sfrutta le soluzioni tecnologiche e incoraggia la creatività.

Il software può essere utilizzato sia da individui che da organizzazioni per pianificare, progettare, costruire, gestire e mantenere edifici e altre infrastrutture fisiche, tra cui strade, ferrovie, ponti e porti, nonché impianti idrici, elettrici e del gas, oltre a tanto altro.

Grazie all'approccio sistematico, il BIM permette un'analisi attenta e precisa di tempi, costi e geometrie garantendone un maggiore controllo. I vantaggi del BIM sono frutto della connessione di team, workflow e dati per l'intero ciclo di vita del progetto, dalla progettazione e ingegnerizzazione fino alla costruzione e messa in funzione, allo scopo di lavorare con più efficienza e ottenere risultati migliori.

Il processo del BIM supporta la creazione di dati che possono essere utilizzati durante l'intero ciclo di vita di un progetto.

- Progettazione: Ottimizza la pianificazione combinando i dati reali per generare modelli dell'ambiente di costruzione e naturale esistente.
- Costruzione: In questa fase vengono eseguiti i processi di progettazione concettuale, analisi, creazione dei dettagli e documentazione.
- Produzione: Le informazioni di logistica della costruzione del progetto vengono condivise con i professionisti e gli appaltatori vengono coinvolti per garantire i massimi livelli di sincronizzazione ed efficienza.
- Manutenzione: I dati BIM si estendono anche alla gestione operativa e alla manutenzione degli asset costruiti.

Prendendo come esempio la costruzione di un edificio, un software BIM offre supporto nella gestione dei costi e informazioni. Con alcuni software BIM si può ottenere una simulazione della costruzione del fabbricato prima che venga effettivamente eseguita, con lo scopo di trovare eventuali problemi di sicurezza, errori di programmazione e/o possibilità di prefabbricare beni fuori sede, insieme ad altro ancora. Possono essere integrati inoltre, dei

sensori per fornire informazioni in tempo reale sulla struttura, così da rendere l'analisi del suo funzionamento e della manutenzione più semplici al team di sviluppo.

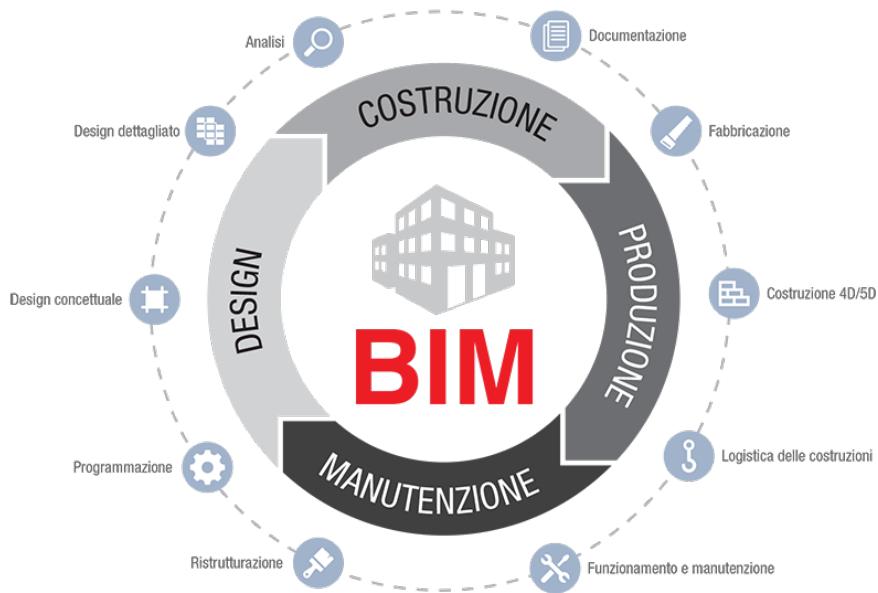


Figura 1.3: Fasi di sviluppo BIM

### 1.2.1 Dimensioni del BIM

I modelli BIM sono spesso divisi in dimensioni. In un progetto tradizionale, le informazioni sono rappresentate in disegni 2D e testo. La geometria nel 2D dei disegni è descritta con linee e curve. Quando si verificano cambiamenti, spetta ai membri del team ricordarsi di dover modificare quantità, testo, linee e curve.

Con BIM possono essere aggiunti ulteriori dimensioni come ausilio per la progettazione [14]:

- Terza Dimensione: Un modello BIM 3D contiene oggetti 3D che costituiscono il modello informativo. Questi oggetti rappresentano l'edificio

o gli spazi degli edifici, in una realtà virtuale. Contengono informazioni, come minimo, su lunghezza, larghezza e altezza.

- Quarta Dimensione - Tempo: La quarta dimensione si riferisce all'aggiunta del tempo al 3D. Questo approccio rende possibile visualizzare l'intera costruzione del progetto o solo alcune fasi di esso e vedere quali tempistiche delle attività influenzano il flusso di lavoro.
- Quinta Dimensione - Costi: La modellazione 5D o stima basata su modello, è il modello 4D aggiungendo i costi. Consente di stimare la quantità e i costi dei materiali presi in considerazione.
- Sesta Dimensione - Gestione: La sesta dimensione è dedicata alla gestione delle strutture e delle informazioni del modello, per ridurre i costi durante il ciclo di vita del progetto. A volte i modelli BIM 6D sono indicati come Asset Modelli Informativi (AIM).
- Settima Dimensione - Sostenibilità: Questa dimensione è dedicata alla stima dell'impatto ambientale e alla sostenibilità del progetto, cercando di migliorare l'efficienza energetica della struttura.

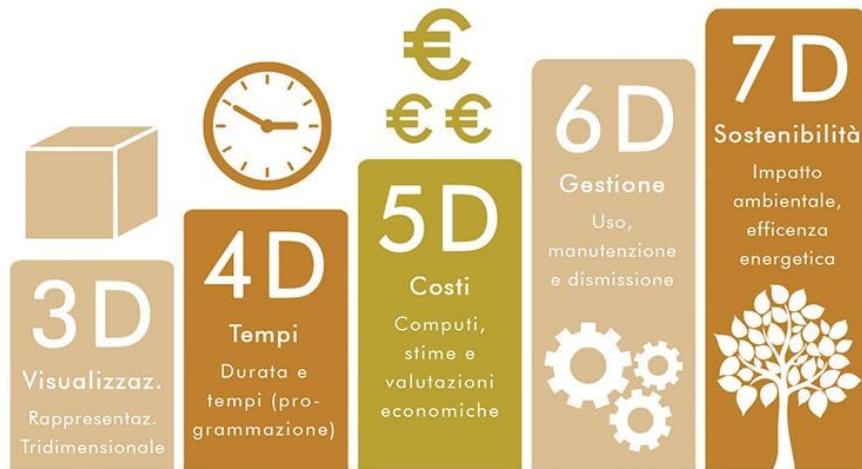


Figura 1.4: Dimensioni del BIM

### 1.2.2 Information Model

Un modello BIM può essere pensato come un ”modello informativo”, information model, una ricca fonte di dati contenenti documenti grafici, non grafici e collegati. Il modello è sviluppato progressivamente nel corso del ciclo di vita del progetto. Tipicamente un modello passerà attraverso una serie di iterazioni. In primo luogo come modello di intento progettuale o Project Information Model (PIM) che viene sviluppato durante la fase di progettazione [14].

Questo modello viene quindi sviluppato in un modello di costruzione virtuale, in genere quando la proprietà viene trasferita dal team di progettazione ai fornitori della costruzione. Infine l'AIM (Asset Information Model) è stato sviluppato per l'uso all'interno delle fasi di funzionamento e di utilizzo.

I modelli informativi sono design orientati agli oggetti, costituiti da questi ultimi o da elementi che hanno caratteristiche fisiche e funzionali ad essi collegate, così come la relazione con oggetti e spazi vicini. Questi oggetti contengono informazioni che abilitano diversi tipi di analisi da eseguire, come la quantificazione del materiale richiesto. I modelli informativi di una singola struttura sono riuniti nel Common Data Environment (CDE) per presentare il modello unico dell'asset.

### 1.2.3 BIM - API

I componenti del BIM spesso fanno uso di formati proprietari e risulterebbe complicato integrarli con progetti esterni, per questo motivo si può ricorrere all'utilizzo di altri sistemi, come ad esempio API web, per permettere a diverse applicazioni di interagire tra di loro.

Un'interfaccia di programmazione dell'applicazione (API Application Programming Interface) è un'interfaccia che aiuta gli sviluppatori a connettere insieme diverse applicazioni. Le API sono un collegamento “neutrale”, tra un'applicazione che fa una richiesta (client) e l'applicazione che fornisce la risposta (server), indipendente dal linguaggio di programmazione e dalle spe-

cificità delle applicazioni che le API collegano. Un'API è un insieme di regole che definiscono il modo in cui i computer o le applicazioni comunicano tra loro. Si trovano tra un'applicazione ed il server Web e agiscono come un livello intermedio che elabora il trasferimento dei dati tra i sistemi.

1. Un'applicazione client avvia una chiamata API. Richiesta per recuperare informazioni, viene elaborata da un'applicazione al server Web tramite l'URI (Uniform Resource Identifier) dell'API.
2. Ricevuta una valida richiesta, l'API esegue una chiamata al programma esterno o al server Web.
3. Il server invia una risposta all'API con le informazioni richieste.
4. L'API trasferisce i dati all'applicazione richiedente iniziale.

Sia che si desideri gestire strumenti esistenti o progettarne di nuovi, è possibile utilizzare un' API per semplificare il processo. Le API abilitano l'integrazione, in modo che le piattaforme ed applicazioni possano comunicare tra loro senza problemi. Offrono flessibilità, consentendo di stabilire connessioni con nuovi programmi, offrire nuovi servizi e, inoltre, le API creano un ulteriore livello di protezione tra i dati ed il server.

Al giorno d'oggi, la maggior parte delle API sono API Web che espongono i dati e le funzionalità di un'applicazione su Internet. Le principali API che possiamo trovare sono: API aperte, open source, a cui è possibile accedere con il protocollo HTTP; API dei partner, esposte a o da business partner strategici; API interne, che restano nascoste agli utenti esterni; API composite, che combinano più API di servizio o dati.

Sono stati, inoltre, sviluppati alcuni protocolli per fornire un insieme di regole definite che specificano i comandi e i tipi di dati accettati: SOAP (Simple Object Access Protocol) è un protocollo API creato con XML, che consente di inviare e ricevere dati tramite SMTP e HTTP; XML-RPC si basa su uno specifico formato di XML per trasferire i dati; JSON-RPC è un protocollo simile a XML-RPC, ma questo utilizza JSON; REST (REpresentational

State Transfer) è un insieme di principi dell’architettura API Web, che significa che non esistono standard ufficiali ma deve rispettare determinati vincoli architettonici.

Uno dei maggiori vantaggi dell’implementazione di un’API all’interno del BIM è la capacità di rendere facilmente disponibile un ”deposito” di dati. Si avrà un maggiore controllo sulla diffusione delle informazioni, e si potrà anche creare un database centrale delle migliori informazioni. Implementando un’API puoi consentire ad altri utenti di accedere a una funzione e tenere traccia, ad esempio, dei materiali per altri progetti allo stesso modo. Questo non solo assicura che le informazioni siano centralizzate, ma garantisce anche che altri team contrassegnino correttamente i loro materiali. Creando un’API, puoi semplificare la comunicazione e la collaborazione tra team e reparti, facilitando i flussi di lavoro e integrare diversi sistemi. Tutti questi vantaggi possono portare ad un aumento della produttività, oltre che alla possibilità di integrare il BIM con altri software e progetti [15].

### **1.3 Data Visualization**

Si può definire la *Data Visualization* come la rappresentazione dei dati al fine di visualizzare le informazioni nel modo più chiaro ed efficiente possibile, sfruttando le capacità cognitive dell’essere umano, creando dei tool per aiutarci con la consapevolezza e il supporto per le decisioni da prendere, evidenziando pattern o anomalie presenti nei dati. Questi ultimi possono essere raccolti attraverso diverse fonti, come ad esempio dei sensori, e la loro visualizzazione sta diventando sempre più importante, entrando a far parte della nostra quotidianità [16].

La raccolta e la rappresentazione dei dati è divisa in diverse fasi, nelle quali vengono immagazzinati ed elaborati per ottenere tutte le informazioni utili da fornire all’utente. Il processo, secondo un insieme di criteri ben definiti, può essere eseguito in due maniere: automatica o manuale. Bisogna innanzitutto capire qual è l’obiettivo da raggiungere e qual è il target di

utenti a cui si vuole puntare. E' molto importante rappresentare i dati nella maniera più corretta, in base a ciò che si vuole mostrare e mettere il focus su determinate informazioni [17].

La rappresentazione dei dati va tuttavia preceduta da una fase di analisi che consiste nella seguente serie di operazioni [3]:

- Raccolta di informazioni attraverso diverse fonti.
- Esaminazione e mantenimento solo delle informazioni utili.
- Mapping in cui i dati vengono suddivisi per caratteristiche simili.
- Ricerca degli estremi.
- Raggruppamento e ordinamento secondo determinati criteri.

Una volta che abbiamo raccolto i dati e scelto il target del cliente a cui si vuole arrivare, è molto importante anche il contesto in cui i dati vengono visualizzati. Spesso senza il contesto corretto i dati forniti possono essere privi di significato o non si punta il focus corretto su ciò che vogliamo che si presti attenzione [18]. Prendiamo come esempio la figura 1.5 mostrata: senza specificare il contesto l'immagine è priva di significato. Se sapessimo che il grafico raffigura le variazioni di temperatura media annua durante gli anni, possiamo dedurre che probabilmente c'è stato qualcosa che ha fatto sì che questo fenomeno aumentasse nel tempo. Se inoltre, aggiungessimo l'informazione che dagli anni 1960 in poi si ha una grossa crescita degli impianti industriali, possiamo arrivare alla conclusione che l'aumento della temperatura media annua è data anche dall'inquinamento creato dagli impianti industriali.

Sono molteplici i possibili modi per rappresentare i dati raccolti. Uno dei fattori più importanti è il mantenimento della visualizzazione dei dati il più semplice e chiaro possibile. Più sarà difficile la loro comprensione e la loro visualizzazione, più sarà complicato per l'utente prendere determinate decisioni. La soluzione più comune per la rappresentazione dei dati è l'utilizzo

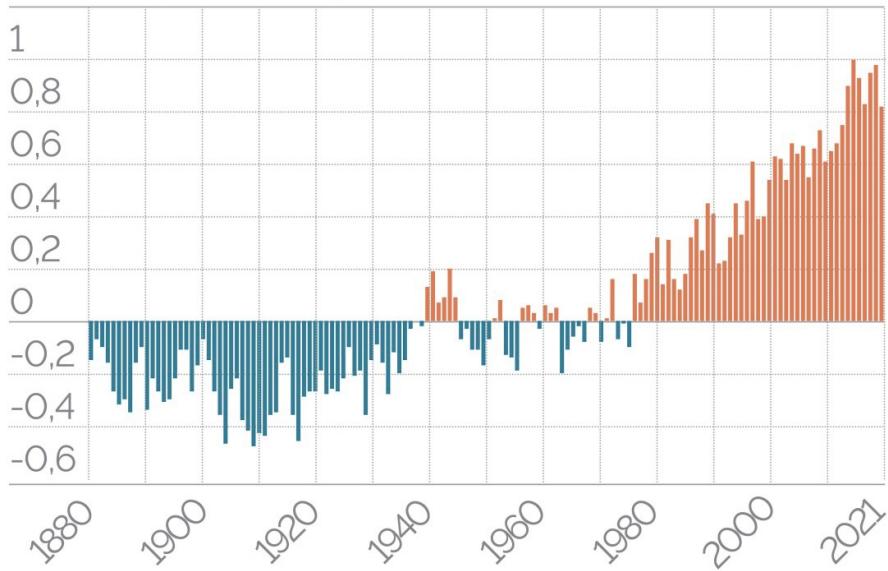


Figura 1.5: Variazione media della temperatura

dei grafici. Questi ultimi sono molto utili perché offrono una vasta quantità di opzioni per la visualizzazione delle informazioni. Trovare il grafico giusto per il lavoro giusto è molto importante e aiuta a trasmettere una comprensione immediata delle informazioni [17].

Anche i colori che vengono utilizzati hanno un'importanza notevole quando dobbiamo rappresentare i dati. Si deve fare molta attenzione sull'accessibilità e sul significato di ciascun colore. Per assicurare una buona accessibilità dei nostri contenuti il contrasto del colore del testo rispetto allo sfondo ha una notevole rilevanza per la visualizzazione delle informazioni; esistono alcune categorie di utenti che possono avere difficoltà nella lettura di testi con poco contrasto. Inoltre, i colori vengono spesso associati a determinati significati, ad esempio, il rosso viene accostato a situazioni di pericolo o anomalie.

### 1.3.1 Dati in tempo reale

Il Real-Time computing, noto anche come reactive computing, viene utilizzato per descrivere un sistema informatico che reagisce agli eventi eseguen-

do attività entro un intervallo di tempo specifico. L’intervallo di tempo per l’esecuzione di queste azioni è dell’ordine dei millisecondi. Il rispetto di questi intervalli di tempo specifici è fondamentale per l’efficacia dei sistemi in tempo reale. Se un sistema in tempo reale non riesce a eseguire azioni nell’intervallo prestabilito, può risultare inadeguato o inutile. Inoltre, un sistema può essere definito quasi in tempo reale se trascorre un certo lasso di tempo tra un evento, la sua gestione e la riproduzione dei dati raccolti, solitamente a causa di ritardi introdotti dalla rete o da pipeline automatizzate di raccolta dati.

Quando si vuole rappresentare i dati, spesso, la raccolta Real-Time delle informazioni risulta essere importante, perché i dati devono essere rappresentati in modo accurato. L’inclusione di questo sistema di visualizzazione dei dati porta ad avere molteplici vantaggi all’utente. Oltre ad essere utili per aiutarci alla scelta delle decisioni da prendere, alla possibilità di ricevere dei feedback immediati per quanto riguarda la loro correttezza, aumenta anche la consapevolezza migliorando il sistema [3].

Tuttavia, lavorare con i dati in tempo reale richiede maggiori costi e si è sempre a rischio di perdere la connessione e insieme la loro raccolta e visualizzazione per diverse cause, come ad esempio la connessione alla rete.

### 1.3.2 Human-Computer Interaction

Si è studiato che la Data Visualization e Human-Computer Interaction sono strettamente legati, poiché è stato dimostrato che indagare e sfruttare l’interazione tra gli utenti e i dati può portare a una migliore comprensione e coinvolgimento dell’utente con le informazioni rappresentate [16]. Esistono diversi modi per far interagire l’utente con i dati visualizzati: sia tramite computer, smartphone o dispositivi smart, che andando nello specifico con tastiera, mouse o interfacce per la realtà virtuale aumentata o mista e altro ancora. Oltre alle metodologie di sviluppo, ricerca anche soluzioni che permettano la validazione dei risultati ottenuti, solitamente sulla base di interviste e questionari, con l’obiettivo di ottenere il feedback degli utenti , in

modo che i loro suggerimenti possano essere integrati nel successivo ciclo di sviluppo dell'applicazione in questione [9].

### 1.3.3 Mappe

Lo strumento più utilizzato per il wayfinding sono le mappe, che vengono utilizzate per la localizzazione e la navigazione e possono offrire diverse funzionalità. Possono rappresentare qualsiasi posizione e evidenziare informazioni come punti di interesse o altro. Le mappe possono essere visualizzate in diversi formati, solitamente creati come file raster o vettoriali e successivamente salvati in formato JPG, PNG o SVG [19].

Le mappe raster non sono altro che delle immagini, formate da una matrice di punti colorati con stessa forma e dimensione; i pixel. Le mappe vettoriali invece, sono costituite da forme geometriche come punti, linee e poligoni a cui vengono attribuite determinate caratteristiche di colore e spessore. L'insieme di questi elementi geometrici viene utilizzato per rappresentare tutti gli oggetti presenti in una carta. Man mano che le dimensioni e la complessità della mappa cresce si ha la necessità di utilizzare funzionalità e formati più avanzati.

Al giorno d'oggi, le applicazioni che sfruttano le mappe sono estremamente comuni, e possono essere utilizzate per diversi scopi tra cui la navigazione, i giochi e altro.

**Wayfinding** “Way finding” significa letteralmente “trovare la via”, il termine è stato usato per la prima volta negli anni ’60 dall’architetto americano Kevin Lynch, che nel suo libro “L’immagine della città” indaga su come le immagini ambientali influiscano sulla vita delle persone [20]. Il wayfinding può essere definito come la consapevolezza di trovarsi in un determinato luogo o ambiente e sapere anche come arrivarci partendo da una determinata posizione. E’ dunque il modo per le persone di trovare il proprio orientamento, e quindi la capacità di determinare la propria posizione. Esistono molteplici

strumenti per aiutare l'utente con il suo orientamento che utilizzano diverse tecnologie a loro disposizione.

Diverse sono le tecnologie che possono essere utilizzate: GPS, un sistema di navigazione satellitare che consente agli utenti di determinare con precisione la propria posizione[21]; Wi-Fi, sistema di localizzazione che può essere utilizzato all'interno di strutture; BLE Beacon, in cui vengono utilizzati trasmettitori che appartengono alla classe BLE (Bluetooth Low Energy); RFID, localizzazione tramite RFID (Radio-Frequency IDentification) stabilisce una connessione sfruttando le radiofrequenze.

#### 1.3.4 Mappe e Data Visualization

La creazione di mappe e la visualizzazione di dati è estremamente importante in quanto cerca di rappresentare i dati in una forma che possa essere più facilmente compresa e interpretata da un pubblico non tecnico. È considerata comunque una scienza assicurarsi che le immagini siano accuratamente conformi ai dati su cui si basano [22]. Ma le mappe, come molte cose, non sono tutte uguali, alcuni stili rappresentano meglio determinati tipi di informazioni rispetto ad altri.

- Point Map: Una mappa di punti è uno dei modi più semplici per visualizzare i dati. È utile per mostrare i modelli di distribuzione e densità delle cose, ma richiede di raccogliere accuratamente le coordinate in modo da poter identificare ogni posizione con precisione sulla mappa. Questa tecnica può essere difficile da utilizzare con mappe su larga scala, poiché i punti possono sovrapporsi l'un l'altro a determinati livelli di zoom.
- Proportional Symbol Map: vengono utilizzate delle forme geometriche per rappresentare i dati in una posizione particolare. Tuttavia, in base alla dimensione e/o al colore del punto, può essere utilizzato per rappresentare più variabili contemporaneamente.

- Cluster Map: Presenta un concetto simile di utilizzo di punti di varie dimensioni e colori per rappresentare più tipi di dati in una posizione. Questa tipologia risolve il problema principale del sovraffollamento nelle mappe dei punti, ma richiede speciali strumenti di visualizzazione dei dati.
- Heat Map: Una mappa termica utilizza colori o sfumature per rappresentare valori o intervalli di valori diversi. Risulta utile per visualizzare in modo più preciso modelli di alte e basse concentrazioni di una variabile.

Un altro fattore che ci aiuta a comprendere meglio i dati che stiamo visualizzando è l'utilizzo di una legenda. Essa può essere impiegata per mostrare all'utente come vengono visualizzati vari canali come colore e dimensioni per rappresentare i dati. Molte mappe fanno uso di diverse proprietà visive per visualizzare diversi valori. La legenda mostra cosa significano queste associazioni e quindi aiuta a leggere e capire meglio il significato dal grafico.

### 1.3.5 Mappe e Dati in tempo reale

La loro combinazione può essere di grande ausilio per esaminare l'ambiente che ci circonda in tempo reale. Al giorno d'oggi, il numero di informazioni provenienti da diverse fonti è in continuo aumento, e grazie al mondo sempre più digitale e interconnesso, sta diventando molto semplice raccogliere dati sull'ambiente. La Data Visualization ci può aiutare facilitando la comprensione dei dati raccolti.

Concentrandosi sull'indoor, la visualizzazione dei dati può essere utile per visualizzare informazioni raccolte dai sensori che sono distribuiti all'interno degli edifici.

L'importanza di tale associazione, tra informazioni raccolte e visualizzazione in tempo reale di esse, si rispecchia nella possibilità dell'utente di poter tenere controllati tali dati, appunto in tempo reale. Così facendo ci permette di avere una situazione monitorata dell'ambiente che ci circonda in ogni

istante.

Grazie allo studio di Chiara Ceccarini [9] abbiamo la possibilità di toccare con mano questa particolare tipologia di progettazione; il suo sistema smart campus ci da l'occasione di raccogliere informazioni sulle aule attraverso dei sensori. Queste informazioni sono aggiornate in tempo reale; in particolare si riportano valori di luminosità, temperatura e umidità delle singole aule del campus di Cesena.

Il progetto che verrà presentato di seguito avrà come scopo la progettazione di una applicazione mobile in cui avremo la possibilità di spostarci all'interno di un campus universitario, attraverso una mappa, e visualizzare i dati ambientali in real time delle aule presenti.



# Capitolo 2

## Analisi e Progettazione

Questo capitolo inizia documentando la fase di progettazione di Alma-Map mobile, compresa l’analisi dei requisiti, la definizione dell’architettura e le tecnologie che sono state scelte per eseguire l’implementazione, che è documentata nel terzo capitolo 3, insieme a tutte le scelte rilevanti in termini di procedura di sviluppo e distribuzione.

### 2.1 Analisi

Il 14 ottobre 2021 è stato inaugurato il nuovo Distretto Navile dell’Università di Bologna, un nuovo campus che potrà ospitare circa 3.700 studenti e circa 500 docenti e tecnico-amministrativo personale a pieno regime, con poco meno di 68.000 mq di superficie edificata e oltre 52.000 mq di aree esterne (2.1).

Il nuovo polo universitario ospita le attività didattiche e di ricerca del dipartimento di chimica “Giacomo Ciamician”, di chimica industriale, farmacia e biotecnologie “Toso Montanari” e la sezione di astronomia del Dipartimento di Fisica e Astronomia “Augusto Righi”. Inoltre, diventerà la sede di vari uffici dell’Amministrazione Centrale dell’Alma Mater e ospita l’INAF Astrofisica e Space Science Observatory di Bologna e Chernkov Array Sede dell’Osservatorio (CTAO).

Il distretto è composto da cinque edifici multipiano di varia altezza e altre tre strutture multifunzionali, distribuite su una superficie di poco meno di otto ettari. Questo spazio ospita 29 aule didattiche per una capienza complessiva di 2.115 posti, 4 laboratori informatici per 254 posti, 14 aule studio con 672 postazioni e laboratori di ricerca per un totale di oltre 700 posti. Inoltre, il campus ospita una nuova biblioteca con 190 postazioni e una caffetteria con una capienza di 150 persone.

Con l'apertura di un complesso così grande l'Università ha deciso di dotare gli edifici del Campus Navile non solo di adeguate informazioni segnaletiche e indicazioni, ma anche di un set di mappe interattive, che chiunque può utilizzare per orientarsi e per ottenere informazioni sul vari spazi all'interno dell'area.

Una soluzione simile era già stata adottata nel Campus di Cesena, mostrando benefici rilevanti per la comunità universitaria. Per elaborare ulteriormente, seguendo l'inaugurazione della sede di Cesena nel 2018, il Dipartimento Informatica - Scienza e Ingegneria (DISI) di Cesena aveva realizzato un'applicazione chiamata AlmaMap [23], che fornisce tuttora, per il complesso di Cesena, funzionalità simili a quelle che sono richieste a Navile. Per questo motivo è stato designato il Dipartimento dall'Università con il compito di realizzare una nuova versione di AlmaMap , su misura per il complesso Navile.

Il campus non è stato aperto alle attività all'unisono inizialmente, ma seguendo un processo graduale, sono stati resi operativi inizialmente gli edifici Ue4 e Ue5. Per supportare la creazione e la manutenzione delle mappe del campus, l'Università ha fornito al team di sviluppo le planimetrie degli edifici del complesso, nonché informazioni sul nome e sullo scopo di ogni spazio.

Il Campus Navile è un moderno complesso, dotato fin dalla sua nascita di avanzate soluzioni di automazione degli edifici e controllo.

Ci si vuole focalizzare sull'automazione del funzionamento dei sistemi di riscaldamento, ventilazione e condizionamento dell'aria (HVAC), al fine di ottimizzare il consumo di energia e ridurre l'inquinamento. Di fatto, il

sistema BIM sfrutta un livello di sensori IoT che misura dati in tempo reale tra cui temperatura, umidità, anidride carbonica e altro, combinandoli con il programma del campus per gestire l'accensione, lo spegnimento e il funzionamento dell'HVAC [3].

Il sistema BIM è pensato per essere utilizzato principalmente da tecnici e personale ICT, pur essendo per lo più trasparente per la stragrande maggioranza degli occupanti dell'edificio. Tuttavia, il fatto che Navile BIM si basi su standard aperti per la comunicazione lo rende un'ottima piattaforma su cui costruire e possibilmente fornire valore aggiunto non solo per il personale tecnico del campus, ma per tutti coloro che sono coinvolti nella vita del complesso. Da qui è nata la seconda parte del progetto AlmaMap, i dati raccolti dai sensori consentono di gestire l'HVAC, i quali possono rivelarsi preziosi anche per gli occupanti, per aiutarli nel processo decisionale e renderli più consapevoli di ciò che li circonda.

L'Università di Bologna ha scelto di affidarsi a Siemens per l'installazione di un'applicazione di Building Information Modeling [3] basata sul sistema Desigo [24], una suite completa di strumenti e soluzioni per edifici intelligenti ad alte prestazioni. Questo sistema BIM può essere facilmente ampliato con hardware aggiuntivo o integrato con servizi esterni.

La soluzione ufficiale Desigo Building Management System installata nel campus è Desigo CC, un software GUI che si interfaccia con l'applicazione BIM sottostante per fornire dati attuali e storici raccolti dai sensori, ricevere notifiche di avviso e gestire i sistemi in tutto il complesso.

Visto il successo delle mappe visualizzate nei chioschi nel complesso Navile [3], si è pensato di realizzare una versione mobile di AlmaMap, per dare la possibilità agli studenti di avere la mappa del campus sempre disponibile dei propri dispositivi.

Il sistema AlmaMap deve implementare tre funzionalità principali. La prima è legata alla visualizzazione di una mappa interattiva del campus, per poter determinare dove si trovano gli edifici nel complesso e consentire di navigare tra i piani e vedere i singoli spazi al loro interno. La seconda mira

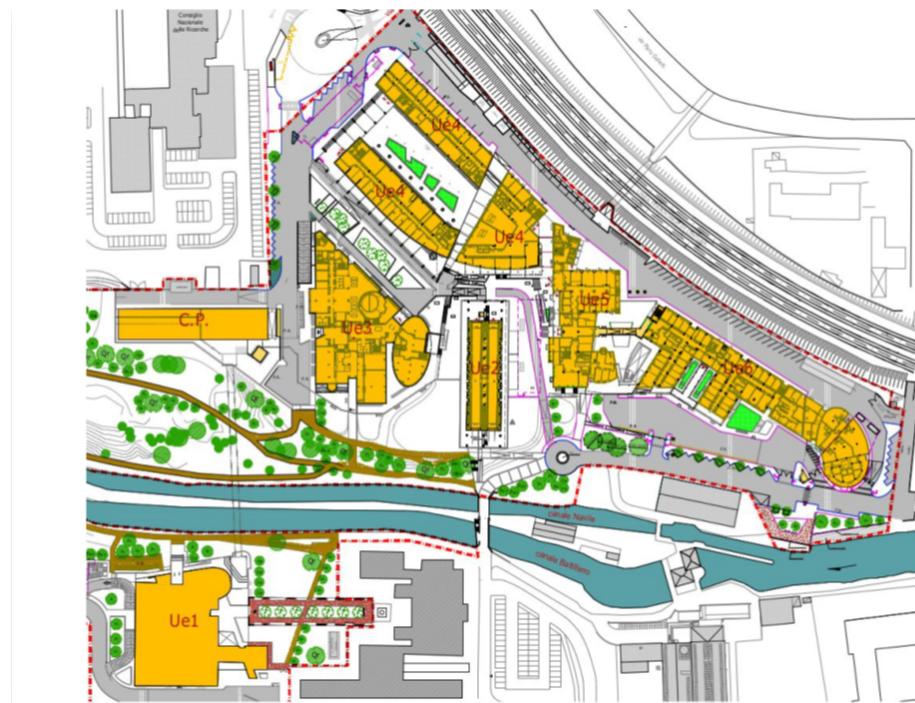


Figura 2.1: Planimetria Distretto Navile

all’ottimizzazione del tempo, fornendo una visualizzazione più rapida degli edifici senza passare dalla mappa, sfruttando l’utilizzo di un elenco. Ultima, fornire la possibilità di visualizzare i dati ambientali delle aule in cui sono presenti appositi sensori. Grazie a queste caratteristiche, sarà più semplice, sia per chi va al campus per la prima volta, sia per chi ha bisogno di visitare un’aula per orientarsi e per trovare il luogo dove si vuole andare.

Dato che l’app mobile non ha un punto di consultazione specifico, come il chiosco nel caso della versione già presente del progetto, la parte di navigazione rimane più complessa da realizzare, quindi per il momento si è deciso di ometterla, ma può sempre essere una buona idea per un progetto futuro da poter integrare.

La mappa del campus, e tutte le altre immagini utilizzate in questo progetto, che sono in formato SVG, mi sono state fornite da Gianni Tumedei, utilizzate nella sua stessa tesi [3]. La planimetria di tutti gli edifici gli era stata fornita a suo volta dall’Università di Bologna in formato DWG. Ogni

file conteneva una quantità elevata di dettagli tecnici che non necessitano di essere riprodotti nelle mappe vettoriali e rendono molto difficile comprendere la disposizione della stanza. Di conseguenza, il team di sviluppo ha dovuto ripulire i file DWG e salvandolo nel formato SVG.

Per associare un edificio o uno spazio alla forma corrispondente nell'asset SVG, la scelta è ricaduta sulla memorizzazione dell'attributo id del tag nel database. Pertanto, tutte le forme che rappresentano un edificio o uno spazio sono state create con un ID univoco.

## 2.1.1 Requisiti funzionali

### 2.1.1.1 Diagramma dei casi d'uso

Il diagramma dei casi d'uso, mostrato in Figura 2.2 e analizzato in questa sottosezione, è il risultato di processi di generalizzazione e raffinamento che hanno permesso di definire le sequenze di casi d'uso. E' cruciale per dividere l'obiettivo finale in più sotto-obiettivi e concretizzarli in caratteristiche reali che faranno parte del sistema. Osservando il diagramma in Figura 2.2 è subito evidente come siano due le modalità possibili per interagire con l'applicazione. La prima, e probabilmente più veloce, è sfogliare l'elenco degli spazi del campus, senza utilizzare in alcun modo la mappa. La seconda è navigare e interagire effettivamente con la mappa, toccando gli edifici, cambiando piano, ecc. Questa soluzione potrebbe non essere veloce come la precedente, ma consente di trasmettere una comprensione più profonda della disposizione del campus ed è probabilmente più intuitiva e divertente per l'utente finale. Un altro aspetto importante è che le due interazioni che apportano maggior valore all'utente finale sono l'ottenimento di informazioni su uno spazio.

Il diagramma dei casi d'uso, visualizzato in Figura 2.2, ha una stretta somiglianza con quello di AlmaMap versione 2.0 [3], indicando che molte funzionalità del sistema rimarranno invariate. Non sarà presente, come accennato nella precedente analisi, la navigazione interna del campus. Avere

dati da visualizzare, oltre a poter calcolare il livello di comfort, significa interfacciarsi con il BIM per ricavarne le misurazioni.

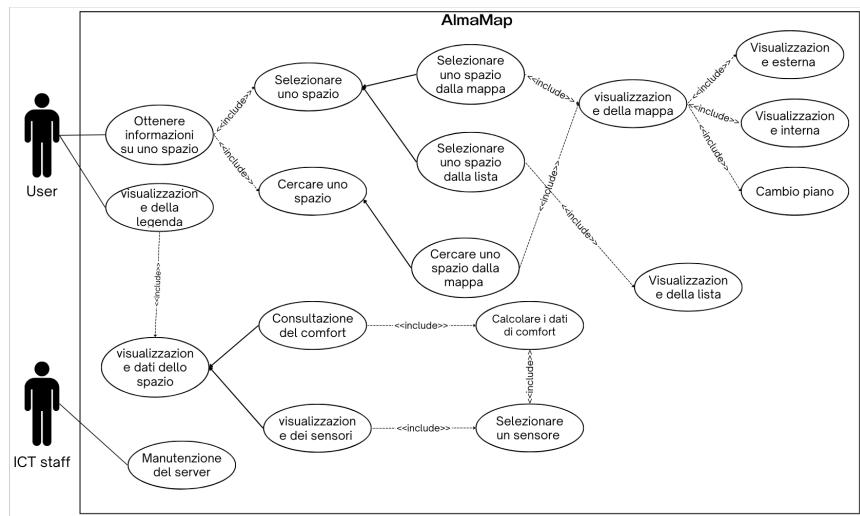


Figura 2.2: Diagramma dei casi d'uso completo

### 2.1.1.2 Caratteristiche del sistema

**Mappa del Campus** Per navigare nel campus e nei suoi spazi, la mappa è suddivisa in due visualizzazioni: esterno per il campus, interno per ogni edificio e piano e fornire un livello di dettaglio basato sulla vista corrente. Per esempio, le singole stanze non saranno visibili guardando l'esterno.

**Legenda** Utilizzata per identificare facilmente le varie aule e punti strategici del campus.

**Lista degli edifici** Per poter visualizzare rapidamente gli edifici presenti senza dover utilizzare la mappa.

**Lista delle aule** Per selezionare un'aula senza doverla trovare nella mappa. L'elenco dinamico e ricercabile degli spazi riceve un piccolo aggiornamen-

to, che visualizza un indicatore accanto a ciascuna stanza da cui il sistema raccoglie i dati dei sensori.

**Dettagli su uno spazio** Per trasmettere all’utente tutte le informazioni su uno spazio. I dettagli dello spazio dovrebbero almeno comprendere il suo nome e la classificazione, con possibilità di visualizzazione delle informazioni come una descrizione e il numero massimo di persone che può accogliere. La visualizzazione dei dettagli dello spazio ha la possibilità di mostrare informazioni aggiuntive sotto forma di dati del sensore.

**Sensori** Sono presenti diversi sensori all’interno del complesso Navile, utilizzati per ricavare dati ambientali delle aule con il fine di ottimizzare il consumo di energia e ridurre l’inquinamento. All’interno dell’applicazione, dev’essere possibile consultare i dati ambientali per le aule dotate di tali sensori.

**Interfacciamento con API di AlmaMap** Verranno utilizzate API per richiedere i dati nel backend del progetto, così da rendere più semplice l’interoperabilità. In questa sezione, inoltre, troviamo le funzionalità rivolte all’utente definite nella precedente fase di analisi vengono estese con nuove funzionalità che ruotano attorno ai dati provenienti dai sensori ambientali.

**Sensori** Ciascuna voce dovrebbe avere la capacità di visualizzare anche informazioni in tempo reale come dati storici.

**Dettagli su un sensore** Ciascuna voce del sensore dovrebbe avere la capacità di visualizzare informazioni sui dati in tempo reale e storici.

### 2.1.2 Requisiti non funzionali

**Estendibilità** Questo requisito si riferisce all’estensibilità sia in termini di nuove funzionalità che di nuovi dati per le funzionalità esistenti. Facendo

parte di un progetto universitario è probabile che il sistema sia soggetto a una serie di cambiamenti e aggiunte durante il suo ciclo di vita. Per questo motivo, deve adottare un'architettura che facilita l'implementazione di nuove funzionalità, al fine di promuoverne lo sviluppo in future tesi o attività di ricerca. Dato che gli edifici e gli spazi del campus diventeranno progressivamente operativi nel tempo, alcune aree non saranno immediatamente accessibili al pubblico. Tuttavia è necessario tener conto della loro inclusione a breve termine, e per questo motivo deve essere facile, dal punto di vista dello sviluppo, aggiungere mappe di nuovi edifici.

**Interoperabilità** Sebbene non sia richiesta alcuna integrazione con alcun servizio esterno sulla prima versione del sistema, mantenere aperta questa possibilità compensa alcuni aspetti di interessante opportunità. Ad esempio, potremmo utilizzare gli Open Data per ottenere informazioni sull'orario delle varie aule, comunicare con i sensori IoT per rilevare i parametri ambientali degli spazi del campus [23], o utilizzare esterni API per fornire le previsioni del meteo direttamente sulla mappa del campus.

**Resilienza** AlmaMap potrebbe riscontrare vari errori durante il suo funzionamento, causati da interruzioni, modifiche della rete o della configurazione e altro ancora. Il sistema dovrebbe essere quanto più resiliente possibile, evitando che gli errori sui singoli sensori influiscano sulla sua funzionalità complessiva. La fase di progettazione per AlmaMap ha vari obiettivi tra cui definire come avverrà l'interazione AlmaMap - BIM al fine di recuperare tutti i dati richiesti, decidere come elaborare i dati recuperati e come calcolare il comfort.

## 2.2 Design

Questa sezione descrive in dettaglio la procedura di progettazione che è stata adottata per il progetto. Partendo dai requisiti di sistema individua-

ti nella precedente sezione, prima da un punto di vista più generico e poi analizzando singolarmente ciascun componente.

### 2.2.1 Architettura

La struttura è suddivisa in più componenti posizionati su più livelli secondo un modello client-server. Le principali componenti che troviamo sono:

1. Server: che ospita il backend di AlmaMap applicazione e banca dati.
2. Applicazione Backend: è un web server che permette di accedere ai dati di AlmaMap tramite un set di API HTTP. Inoltre, espone tutti gli asset delle mappe, che possono essere reperiti sempre tramite richieste HTTP.
3. Database: i dati di AlmaMap saranno persistenti sul server che ospita l'applicazione di backend.
4. Applicazione Frontend: è l'oggetto di questa tesi, l'interfaccia utente lato client risiederà in un'applicazione mobile in iOS, tutte le interazioni dell'utente avverranno attraverso di essa.

#### 2.2.1.1 Applicazione frontend

I componenti definiranno il layout e lo stile dell'interfaccia utente e gestiranno la logica dell'applicazione e la comunicazione con il backend. L'esatto stile di programmazione dipenderà in definitiva dal linguaggio e dal framework scelti per eseguire la fase di implementazione, poiché molte librerie frontend sono piuttosto supponenti a questo riguardo.

#### 2.2.1.2 Utilizzo di API HTTP

Verranno eseguite richieste tramite API per richiedere la backend le informazioni da mostrare all'utente, così da avere un'applicazione facilmente adattabile agli aggiornamenti futuri, come aggiunta di nuovi edifici.

Inoltre verranno utilizzate per richiedere i dati raccolti dai sensori passando dal progetto di AlmaMap Navile 2.0 [3] già esistente, che a sua volta raccoglie i dati dal BIM Desigo sopracitato.

### 2.2.1.3 Model-View-ViewModel (MVVM) design paradigm

Il Model-View-ViewModel è un paradigma di progettazione architettonica di organizzazione del codice. Funziona perfettamente con il concetto di interfacce utente reattive. Il modello MVVM consente di separare in modo pulito la logica di business e presentazione di un'applicazione dalla relativa interfaccia utente. La gestione di una separazione netta tra la logica dell'applicazione e l'interfaccia utente consente di risolvere numerosi problemi di sviluppo e semplifica il test, la gestione e l'evoluzione di un'applicazione. Esistono tre componenti principali nel modello MVVM: il modello, la visualizzazione e il modello di visualizzazione. Ognuno ha uno scopo distinto, come illustrato in figura 2.3. Il pattern MVVM viene utilizzato in questo progetto così da tenere separata la parte dell'interfaccia utente dalla parte di richiesta e gestione dei dati da elaborare.

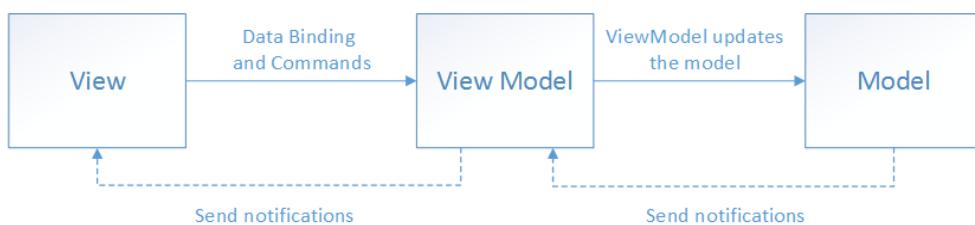


Figura 2.3: Pattern MVVM

### 2.2.2 Design Dettagliato

Questa sottosezione viene presentata un'analisi delle scelte più rilevanti in termini di design.

**Spazi** I dati principali che il sistema gestirà sono gli spazi, che possono appartenere ad un edificio e ad un piano specifici e sono identificati con un codice univoco e classificato con una categoria di legenda. Ogni spazio avrà un nome da visualizzare nell'applicazione frontend, e una descrizione opzionale, per ospitare eventuali informazioni aggiuntive ad esso correlate.

**Mappa** L'area da rappresentare sotto forma di mappe comprende l'esterno del complesso del Navile e i piani interni di ciascuno degli edifici del campus. Come formato viene utilizzato SVG, che è l'estensione vettoriale più popolare e, come tutti i formati di questa categoria, ha il vantaggio di non produrre mai immagini di bassa qualità, indipendentemente dalla risoluzione con cui vengono visualizzate. Un altro vantaggio delle immagini SVG è il fatto che le loro forme possono memorizzare metadati aggiuntivi al loro interno e possono essere consultate e controllate da fonti esterne. Questa funzionalità sarà essenziale per mostrare le informazioni sulla mappa in modo affidabile e senza fare troppo affidamento sulle coordinate. Per questo motivo è stato scelto SVG come formato per le mappe dell'applicazione AlmaMap Navile.

**Collegamento di mappe e dati** Un aspetto cruciale da definire è come creare un collegamento tra un'area della mappa e un edificio, piano o spazio archiviato nel database. Viene il formato SVG dei file delle mappe assegnando un ID a ciascuna forma nella mappa e memorizzandolo nel database al posto delle coordinate. Per la sua robustezza e semplicità è stato scelto quest'ultimo approccio.

**Recupero dei dati** L'app ovviamente recupererà i dati dal server durante il suo funzionamento. Di conseguenza, si vuole gestire il recupero dei dati al fine di:

- Offrire una buona esperienza utente
- Evitare di mostrare dati non aggiornati
- Ridurre al minimo il carico di lavoro del server

### 2.2.2.1 Calcolo del comfort

Il comfort interno è correlato a quattro parametri distinti: i) Comfort Acustico, ii) Qualità dell'aria, iii) Comfort Visivo e infine iv) Comfort Termico.

Sfruttiamo i sensori BIM per rilevare alcuni dati su tre di questi aspetti:

- Sensori del suono
- Sensori per la qualità dell'aria
- Sensori per la temperatura e l'umidità dell'ambiente

Sfortunatamente, al momento della stesura di questa tesi, il sensore BIM che fornisce misurazioni utili sui parametri di comfort visivo non sono ancora presenti. Inoltre, anche se fossero disponibili sensori di luminosità, sarebbe fuorviante includerli nel calcolo del comfort senza sapere se lo spazio in cui si trovano è occupato o meno, in quanto quando è vuoto le luci sono spente e il comfort visivo e generale ne risentono. Per questi motivi si è deciso di escludere i parametri VC dal calcolo del comfort, lasciando questo miglioramento a sviluppi futuri.

Dopo aver definito l'elenco dei parametri ambientali, il passo successivo è stato stabilire gli intervalli ottimali e accettabili per ciascuno di essi. I valori nella tabella 2.4 si ispirano alle linee guida dell'Organizzazione Mondiale della Sanità [25] per temperatura, umidità e suono, e alla Direttiva europea sulla qualità dell'aria [26] per gli inquinanti atmosferici.

I parametri ambientali rilevati dai sensori utilizzano scale e unità di misura eterogenee, pertanto è stato necessario normalizzarle prima di poterle combinare per il calcolo del comfort.

### 2.2.3 Tecnologie utilizzate

In questa sezione viene riportato l'elenco delle scelte tecnologiche effettuate per il progetto, per consentire il soddisfacimento dei requisiti sopra menzionati e l'implementazione dell'architettura definita.

Parameter	Unit	Optimal range	Acceptable range
Temperature (summer)	°C	23 – 26	21 – 28
Temperature (winter)	°C	19 – 22	17 – 24
Humidity	%	40 – 60	30 – 70
Sound	dB(A)	< 40	< 65
PM <sub>2.5</sub>	µg/m <sup>3</sup>	< 10	< 25
PM <sub>10</sub>	µg/m <sup>3</sup>	< 20	< 50
NO <sub>2</sub>	µg/m <sup>3</sup>	< 40	< 150
O <sub>3</sub>	µg/m <sup>3</sup>	< 50	< 120
CO	µg/m <sup>3</sup>	< 2000	< 10000

Figura 2.4: Parametri per calcolo del comfort

### 2.2.3.1 Server e Client

#### Node.js [27]

Node.js è uno dei framework JavaScript più diffusi per la programmazione di web app. Node.js è un runtime JavaScript guidato da eventi asincroni costruito sul motore JavaScript V8 di Chrome, e viene usato sia per lo sviluppo frontend che back end.

Node.js fornisce un'API di input / output non bloccante, ovvero le richieste di input / output vengono eseguite in background, senza provocare blocchi in attesa della risposta. Questo tipo di applicazione è detto asincrono perché le singole operazioni possono procedere indipendentemente dal flusso del programma principale.

#### Core Data [28]

Poiché l'esigenza è quella di gestire una quantità medio-piccola di dati ben strutturati, la scelta è caduta su Core Data. Internamente si appoggia per la gestione degli oggetti al modello entità relazione. Core Data si occupa

della gestione su disco, della gestione dei cambiamenti, della minimizzazione della memoria occupata e delle query su disco.

### **Swift** [29]

E' un linguaggio di programmazione solido e intuitivo, creato da Apple per facilitare lo sviluppo di app per iOS, Mac, Apple TV e Apple Watch. È semplice da usare ed è anche open source. Swift è stato progettato per coesistere con Objective-C ma anche per essere più resiliente agli errori. Tra le caratteristiche più interessanti di questo nuovo linguaggio di programmazione troviamo il cosiddetto Playground. In questo nuovo ambiente operativo, lo sviluppatore potrà vedere in diretta, o quasi, quali siano i risultati delle modifiche apportate al codice.

### **SwiftUI** [30]

SwiftUI ti aiuta a creare app su tutte le piattaforme Apple con la potenza di Swift e un codice ridotto. Offre esperienze migliori, su qualsiasi dispositivo Apple, utilizzando un solo set di strumenti e API. SwiftUI utilizza una sintassi dichiarativa, quindi si può semplicemente indicare cosa dovrebbe fare l'interfaccia utente.

### **UIKit** [31]

UIKit fornisce una varietà di funzionalità per la creazione di app, inclusi componenti che puoi utilizzare per costruire l'infrastruttura principale delle tue app iOS, iPadOS o tvOS. Il framework fornisce l'architettura di finestre e viste per implementare l'interfaccia utente, l'infrastruttura di gestione degli eventi per fornire Multi-Touch e altri tipi di input alla tua app e il ciclo di esecuzione principale per gestire le interazioni tra l'utente, il sistema e la tua app .

### **SF Symbols** [32]

SF Symbols è una libreria di iconografia progettata per integrarsi perfettamente con San Francisco, il carattere di sistema per le piattaforme Apple.

I simboli sono disponibili in nove pesi e tre scale e si allineano automaticamente al testo. Possono essere esportati e modificati utilizzando strumenti di modifica della grafica vettoriale per creare simboli personalizzati con caratteristiche di progettazione e funzionalità di accessibilità condivise. SF Symbols 5 introduce una raccolta di animazioni espansive, oltre 700 nuovi simboli e strumenti migliorati per simboli personalizzati.

### SVGView [33]

La libreria supporta i casi d'uso più comuni di elementi SVG. L'obiettivo è portare tutta la potenza di SVG sulle piattaforme Apple. Il framework può analizzare i file SVG e rappresentarne il contenuto in SwiftUI. Offre la possibilità non solo di eseguire il rendering dei file SVG, ma anche di aggiungere loro interattività, gestire l'input dell'utente e utilizzare SwiftUI per mettere in movimento la tua arte.

#### 2.2.4 Mockup e idee di progettazione

Un mockup, o mock up, è la bozza di un oggetto o di un sistema, priva delle funzioni del prodotto finale. Può essere utilizzato per scopi illustrativi o promozionali, in modo da poter presentare al pubblico un progetto in divenire, non ancora completo e non ancora lanciato sul mercato. Nelle figure 2.5 e 2.6 vengono mostrati i mockup del progetto AlmaMap mobile che sono stati implementati. Si era pensato di mettere una TabView in basso così da rendere più facile la navigazione all'interno dell'applicazione. Verranno divise le diverse pagine in base al loro scopo: i) una per mappa, ii) una pagina in cui visualizzare la legenda, iii) navigazione rapida all'interno del campus, e infine iv) la pagina in cui vengono mostrati i valori raccolti dai sensori.



Figura 2.5: Mockup pagina della (1) mappa e della (2) legenda



Figura 2.6: Mockup pagina degli (1) edifici e dei piani, e pagina dei (2) sensori

# Capitolo 3

## Sviluppo

I contenuti di questo capitolo sono strettamente correlati a quanto definito nel Capitolo 2. Verranno illustrati come sono state implementate tutte le parti più rilevanti del progetto, tra cui il soddisfacimento di tutti i requisiti funzionali che erano stati definiti precedentemente. Verranno mostrati e commentati sia parti di codice che screenshot dell’interfaccia utente, per rendere il più chiaro possibile le scelte fatte.

### 3.1 Software e ambiente di sviluppo

#### XCode [34]

Xcode è un ambiente di sviluppo integrato che consente di sviluppare, testare e distribuire app per tutte le piattaforme Apple. Swift, SwiftUI e Xcode funzionano insieme come se fossero uno solo. Il servizio Xcode si connette a un repository di controllo sorgente (Git), controlla il codice e lo integra nel progetto.

#### Visual Studio Code [35]

Visual Studio Code è un editor di codice semplificato con supporto per operazioni di sviluppo come debug, esecuzione di attività e controllo della versione. Ha lo scopo di fornire solo gli strumenti di cui uno sviluppatore ha

bisogno per un ciclo rapido di creazione e debug del codice e lascia i flussi di lavoro più complessi agli IDE con funzionalità più complete, come l'IDE di Visual Studio.

### **Git** [36]

Git è un sistema di controllo delle versioni distribuito gratuito e open source progettato per gestire qualsiasi cosa, dai progetti piccoli a quelli molto grandi con velocità ed efficienza.

### **GitHub** [37]

GitHub è un servizio web e cloud-based che aiuta gli sviluppatori ad archiviare e gestire il loro codice e a tracciare e controllare le modifiche. Git è un sistema di controllo versioni distribuito, il che significa che l'intero codice base e la cronologia sono disponibili sul computer di ogni sviluppatore, il che permette di creare facilmente ramificazioni e fusioni. E' stato utilizzato questo sistema per ospitare il repository del nostro progetto.

## **3.2 Implementazione**

### **3.2.1 Organizzazione del codice**

Il codice è organizzato in modo da tenere separata la parte di codice in cui verranno gestite le Views, quindi tutta la parte grafica visualizzata dall'utente, dalla parte delle richieste e gestione dei dati da mostrare.

Questa divisione rende più semplice per lo sviluppatore comprendere la finalità delle parti e giostrarsi più facilmente nel caso si vogliano apportare delle modifiche al codice.

Sono quattro le suddivisioni principali: i) le Entity del DB, ii) Resources in cui vengono inseriti le immagini e i file json, iii) Views dove troviamo tutte le interfacce che vengono mostrate all'utente e iv) dove troviamo tutta la gestione di background.

### 3.2.2 Database

La struttura del database Core Data, illustrata come diagramma ER nella Figura 3.1, è piuttosto semplice, con otto tabelle e nessuna relazione N-N. Nel processo di passaggio da ER alle tabelle del database effettive, la relazione Edificio-Legend è stata rimossa, poiché ogni edificio appartiene alla stessa voce Legend.

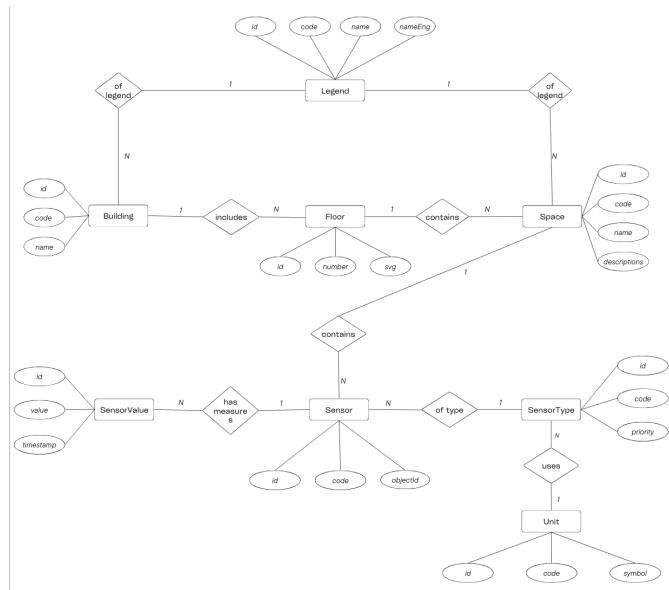


Figura 3.1: Diagramma ER

Il nostro database è formato dalle quattro tabelle che rappresentano:

- la legenda
- gli edifici
- i piani
- i relativi spazi che possiamo trovare all'interno del campus
- sensori
- la tipologia dei sensori

- unità utilizzata da ogni tipologia di sensori presenti
- valori raccolti

La prima entità, la legenda, tiene traccia di tutti gli elementi fondamentali che possiamo trovare nella mappa, passando dalle aule, alle scale d'emergenza, agli edifici. Questi ultimi sono rappresentati in un'altra tabella per poter essere facilmente aggiornata quando verranno attivati questi servizi in altri fabbricati del complesso Navile.

I dati principali che il sistema gestirà sono gli spazi, che possono appartenere ad un edificio e ad un piano specifici e verrà identificato con un codice univoco e classificato con una categoria di legenda. Ogni spazio avrà un nome e una descrizione opzionale, per ospitare eventuali informazioni aggiuntive.

Ogni spazio contiene al suo interno uno o più sensori utilizzati per la raccolta di dati ambientali, che vengono salvati in una tabella apposita. Ogni sensore è classificato in base alla sua tipologia, che può essere per il rilevamento della temperatura, dell'umidità o per altre informazioni. Tutti i dati utilizzano un'unità diversa per la loro rappresentazione, che vengono definite in una tabella a parte.

La figura 3.2 ci mostra quanto detto sopra, ovvero la definizione delle entità all'interno di XCode e Core Data, insieme ai suoi attributi e la loro tipologia di elemento.

### 3.2.3 API

L'app utilizza API per richiedere tutte le informazioni che verranno successivamente mostrate all'utente. Data la quantità relativamente piccola di record attualmente presenti, gli endpoint degli edifici, della legenda e degli spazi, restituiscono semplicemente tutte le righe del database della tabella corrispondente. In futuro, soprattutto per la tabella Spaces, se il numero di righe diventa troppo elevato per una singola richiesta, i parametri di richiesta verranno aggiunti imponendo agli utenti di specificare un criterio di ricerca.

```

8 import Foundation
9 import CoreLocation
10
11 struct Spaces : Identifiable, Decodable {
12
13     var id: Int32
14     var code: String
15     var name: String
16     var descriptions: String
17     var floorId: Int32
18     var legendId: Int32
19
20     init(id: Int32, code: String, name: String, descriptions: String, floorId: Int32, legendId: Int32) {
21         self.id = id
22         self.code = code
23         self.name = name
24         self.descriptions = descriptions
25         self.floorId = floorId
26         self.legendId = legendId
27     }
28 }
29

```

Figura 3.2: Esempio di come le entità vengono definite nel database

Nella figura 3.3 viene mostrato come viene implementata la gestione della richiesta dei dati tramite API.

Si esegue il collegamento al server e si effettua la ricerca dei dati, tenendo conto anche dei vari casi di errore che possiamo riscontrare. Nel caso in cui questa operazione non vada a buon fine, il programma è progettato per utilizzare i dati in locale, così da mantenerlo sempre in funzione, però non viene assicurato che tutte le informazioni siano aggiornate come nel server; questa operazione viene mostrata nella figura 3.4.

Il server in cui sono presenti tutte le informazioni è stato implementato su un IDE diverso, Visual Studio Code. Viene gestita la richiesta HTTP per estrarre tutte le informazioni su un determinato oggetto, come ad esempio gli edifici o i sensori presenti nel campus.

### 3.2.4 Mappa interattiva

La mappa interattiva è formata dai componenti principali esterno, interno e Floor. Il componente Esterno ha un design semplice, poiché esegue il rendering della mappa SVG del campus. Quando l'utente attiva il componente per cambiare la visualizzazione, vengono aggiornati i parametri e passa a mostrare l'interno dell'edificio. Il componente interno esegue il rendering

```

57     func fetchJsonFile(filename: String) -> Data? {
58         let urlString = "http://localhost:3000/\(filename)"
59         guard let url = URL(string: urlString) else {
60             print("Invalid URL")
61             return nil
62         }
63
64         var responseData: Data?
65         let semaphore = DispatchSemaphore(value: 0)
66
67         let task = URLSession.shared.dataTask(with: url) { data, response, error in
68             defer { semaphore.signal() }
69
70             if let error = error {
71                 print("Error fetching data: \(error.localizedDescription)")
72                 return
73             }
74
75             if let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode == 404 {
76                 print("File not found")
77                 return
78             }
79
80             responseData = data
81         }
82
83         task.resume()
84         semaphore.wait()
85
86         return responseData
87     }
88
89 }
```

Figura 3.3: Implementazione richiesta dei dati tramite API

di un componente Floor per ogni piano dell’edificio attualmente selezionato. In questo stato, gli spazi nel piano vengono dipinti con il loro colore della legenda e, quando cliccati, mostrano ulteriori dettagli sulla stanza aprendo una pagina apposita.

Nella figura 3.6 viene mostrata la mappa iniziale che viene visualizzata dall’utente. Nella parte bassa dello schermo sono presenti dei bottoni, uno per ogni **building** attualmente disponibile alla navigazione, che una volta azionati mostrerà l’interno dell’edificio scelto, partendo sempre dal piano terra. Ogni immagine SVG utilizzata può essere ingrandita per visualizzare meglio la mappa.

Una volta che si accede all’interno dell’edificio scelto, si visualizzerà la mappa partendo, come anticipato, dal piano terra. Nella parte bassa dello schermo si trovano dei bottoni per navigare tra i vari piani, e verrà colorato diversamente quello che viene azionato. In alto, inoltre, troviamo l’elemento **Back** per ritornare alla visualizzazione esterna del campus. Come mostrato in figura 3.7, i diversi spazzi presenti nel piano vengono colorati in base alla

```

13     init(fileName: String) {
14         list = [DataType]()
15 
16         if let data: Data = fetchJsonFile(filename: fileName){
17 
18             let decoder = JSONDecoder()
19 
20             do {
21                 list = try decoder.decode([DataType].self, from: data)
22 
23             } catch {
24                 fatalError("Error parsing \(fileName) as \([DataType].self): \(error)")
25             }
26         } else {
27             let fileName2 = fileName + ".json"
28 
29             guard let file = Bundle.main.url(forResource: fileName2, withExtension: nil)
30             else { fatalError("File \(fileName2) not found") }
31 
32             let decoder = JSONDecoder()
33 
34             let data: Data
35 
36             do {
37                 data = try Data(contentsOf: file)
38             } catch {
39                 fatalError("\(fileName2) not loaded: \(error)")
40             }
41             do {
42                 list = try decoder.decode([DataType].self, from: data)
43 
44             } catch {
45                 fatalError("Error parsing \(fileName2) as \([DataType].self): \(error)")
46             }
47         }
48     }

```

Figura 3.4: Gestione del mancato collegamento al server

```

18 app.get('/:filename', async (req, res) => {
19     const filename = req.params.filename;
20     const filePath = `./data/${filename}.json`;
21 
22     if (fs.existsSync(filePath)) {
23         try {
24             const fileContent = await readJsonFile(filePath);
25             res.json(fileContent);
26         } catch (error) {
27             console.log(error);
28             res.status(500).send('Error reading the file');
29         }
30     } else {
31         res.status(404).send('File not found');
32     }
33 });
34 
35 app.listen(port, () => {
36     console.log(`Server is running on http://localhost:${port}`);
37 });

```

Figura 3.5: Implementazione del server e gestione richieste HTTP



Figura 3.6: Mappa esterna del campus

legenda, e alcuni di loro, come laboratori, classi o servizi, se cliccati apriranno una schermata per visualizzare le informazioni.

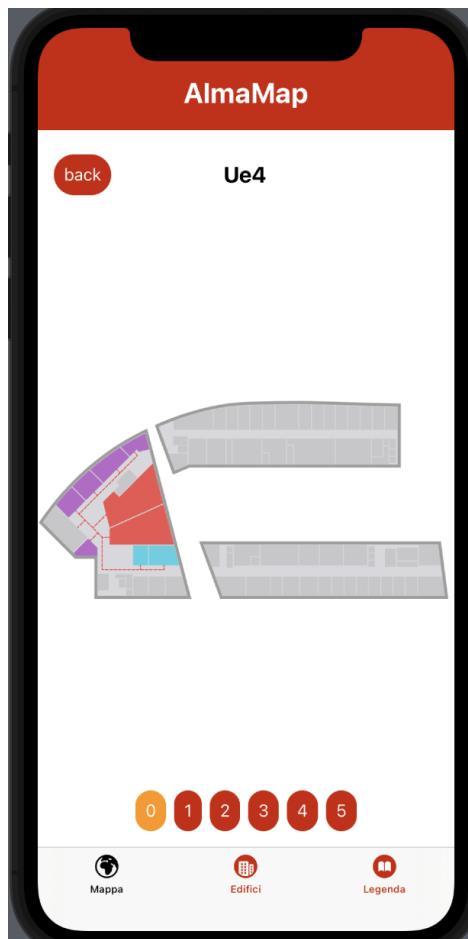


Figura 3.7: Mappa interna del campus

Nell'immagine troviamo il metodo per dare la possibilità di ingrandire l'immagine, così da renderla meglio visualizzabile.

La gestione per selezionare i vari nodi presenti nelle immagini SVG, che rappresentano gli spazi, viene implementata nel metodo in figura 3.9. Con queste righe di codice si assegna un colore ad ogni spazio presente nella mappa e inoltre attiva la visualizzazione delle loro informazioni una volta che si seleziona lo **Space** interessato.

```

66 // Computed-property per creare la MagnificationGesture
67 private var magnificationGesture: some Gesture {
68     MagnificationGesture()
69         .updating($pinchScale) { value, state, _ in
70             state = value
71         }
72         .onEnded { value in
73             let newScale = zoomScale * value
74             zoomScale = min(maxZoomScale, max(minZoomScale, newScale))
75         }
76 }
```

Figura 3.8: Funzione per ingrandire l'immagine

```

79 // Funzione per configurare SVGView
80 // Qui inserisci ogni gestione delle stanze supportate
81 private func setupView(_ view: SVGView) -> SVGView {
82     spaces.forEach{space in
83         if let part = view.getNode(byId: getId(space: space.legendId, code: space.code)) as? SVGShape {
84             part.fill = getLegendColor(type: space.legendId)
85             part.onTapGesture {
86                 spaceSelect = space
87                 if(space.legendId != 6 && space.legendId != 7 && space.legendId != 8){
88                     show = true
89                 }
90             }
91         }
92     }
93     return view
94 }

```

Figura 3.9: Funzione per rendere l'immagine interattiva

### 3.2.5 Legenda

Nelle figure 3.10 e 3.11 viene mostrato come gli elementi della legenda vengono definiti e inseriti nel database utilizzato.

In questa prima immagine 3.10 notiamo come l'elemento della legenda ha diversi attributi per descriverlo. Partiamo dall'attributo `id`, un codice univoco per rendere più facile la sua identificazione, il `code`, che può essere utilizzato anche lui per poter ricercare l'elemento, e infine abbiamo `name` e `nameEng`, utilizzati per aiutare l'utente a capire di che tipo di spazio si tratta, viene visualizzato sia il nome in italiano che in inglese, così da rendere la legenda comprensibile anche a studenti e docenti internazionali.

```

8 import Foundation
9 import CoreLocation
10
11 struct Legend : Identifiable, Decodable {
12
13     var id: Int32
14     var code: String
15     var name: String
16     var nameEng: String
17
18     init(id: Int32, code: String, name: String, nameEng: String) {
19         self.id = id
20         self.code = code
21         self.name = name
22         self.nameEng = nameEng
23     }
24 }
25

```

Figura 3.10: Elemento legenda definito in Core Data e XCode

Nella seconda figura 3.11, invece, viene mostrato come viene inserita e collegata l'entità al database specificando come chiave l'attributo *id*.

La legenda ha subito diverse iterazioni e alla fine si è stabilizzata su quella mostrata nella Figura 3.12. Questo design è stato scelto per la sua efficacia nel trasmettere tutte le informazioni necessarie per comprendere la mappa - come appare il marker, di che colore è lo spazio e cosa rappresenta la legenda sia in italiano che in inglese - in modo molto compatto e diretto.

```

8 import Foundation
9
10 import CoreData
11
12 extension LegendEntity {
13     static func fetchRequest(_ predicate: NSPredicate) -> NSFetchedResultsController<LegendEntity>{
14         let request = NSFetchedResultsController<LegendEntity>(entityName: "LegendEntity")
15         request.predicate = predicate
16         request.sortDescriptors = [NSSortDescriptor(key: "id", ascending: true)]
17         return request
18     }
19
20     func convertToUser() -> Legend {
21         Legend(id: Int32(self.id), code: self.code!, name: self.name!, nameEng: self.nameEng!)
22     }
23
24 }
25

```

Figura 3.11: Collegamento con database per definizione della legenda



Figura 3.12: Un elemento della legenda, che mostra il segnalino, lo sfondo degli elementi e i nomi della leggenda sia in italiano che in inglese.

La legenda viene visualizzata in una pagina apposita per poter ottenere in modo rapido la corrispondente descrizione del marker presente sulla mappa, come mostrato in figura 3.13.

### 3.2.6 Informazioni e azioni contestuali

Una serie di componenti semplici vengono visualizzati in modo condizionale in base a dove si trova l’utente:

- Pulsanti per spostarsi su/giù nei piani e tornare alla mappa precedente
- Etichetta indicante l’edificio e il piano attualmente visualizzati dall’utente
- Pulsante per ottenere una visualizzazione più rapida della struttura del campus
- Pulsante per avere la visualizzazione della legenda.

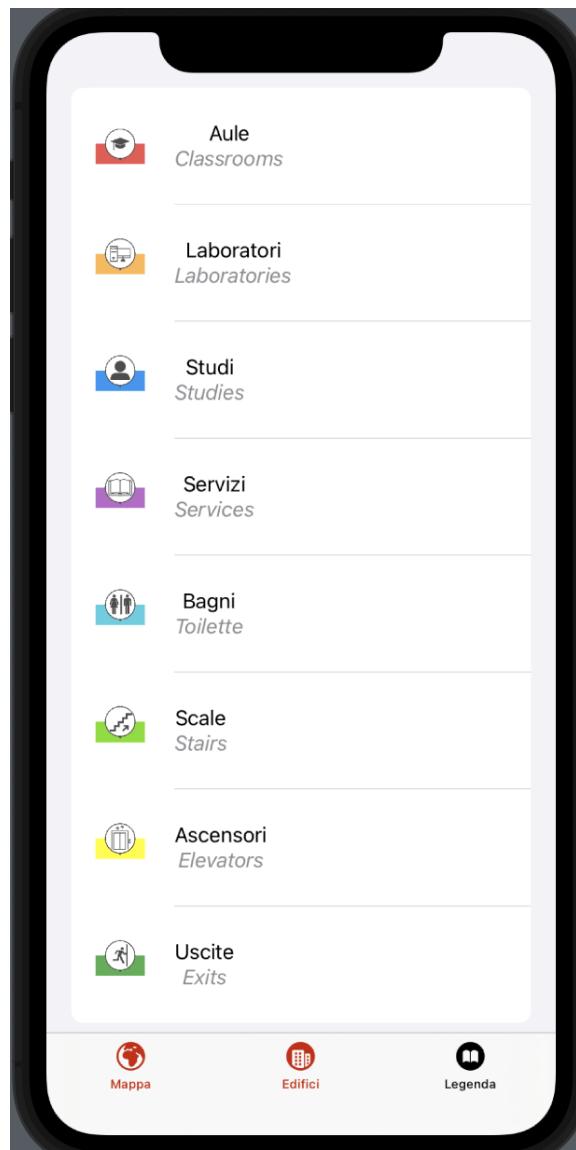


Figura 3.13: Pagina per la visualizzazione della legenda.

### 3.2.7 TabView

La barra in basso mostrata in figura 3.14 è progettata per poter avere diversi tipi di visualizzazione della struttura del campus. Nella prima pagina viene mostrata la mappa del Navile per poter navigare all'interno degli edifici e dei piani. Nella seconda pagina troviamo la struttura visualizzata come elenco degli edifici disponibili con la possibilità accedere alla lista dei vari piani e aule presenti. Nell'ultima, invece, viene mostrata la legenda dei marker che possiamo trovare nella mappa.

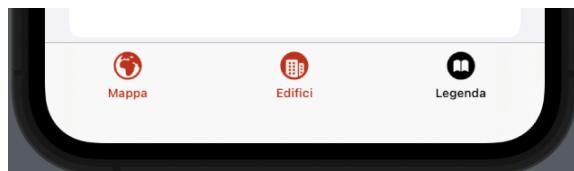


Figura 3.14: TabView visualizzata dall'utente

Il codice per l'implementazione di questa TabView è mostrato nella figura 3.15. Ogni elemento di questa barra di navigazione nel momento in cui viene cliccato rimanda l'utente alla pagina selezionata. Ogni View necessita di alcuni elementi per il corretto funzionamento, come ad esempio il collegamento alla legenda nel database per la LegendView e un viewModel, in cui si trova la gestione per il caricamento di tutte le informazioni richieste.

### 3.2.8 Visualizzazione rapida del campus

Come già anticipato, oltre alla visualizzazione del campus tramite mappa, si è pensato di poter navigare più facilmente alla ricerca degli spazi utilizzando una lista navigabile degli edifici e dei piani. Ogni edificio ha diversi piani che comprendono più spazi.

ù

In questo modo possiamo visualizzare gli spazi presenti in ogni piano e poi accedere più facilmente al loro dettaglio. Non tutti i piani in questo momento hanno spazi disponibili alla loro visualizzazione.

```
var body: some View {  
    TabView{  
        MapView(viewModel: viewModel).tabItem(){  
            Image(systemName: "globe.europe.africa")  
            Text("Mappa")  
        }  
        BuildingsView(viewModel: viewModel, building: viewModel.building.first!).tabItem(){  
            Image(systemName: "building.2.crop.circle")  
            Text("Edifici")  
        }  
        LegendView(viewModel: viewModel, legend: viewModel.legend.first!).tabItem(){  
            Image(systemName: "book.circle")  
            Text("Legenda")  
        }  
    }.accentColor(Color.black)  
    .navigationTitle("AlmaMap").navigationBarBackButtonHidden(true)  
    .navigationBarTitleDisplayMode(.inline)  
}
```

Figura 3.15: Implementazione TabView in XCode

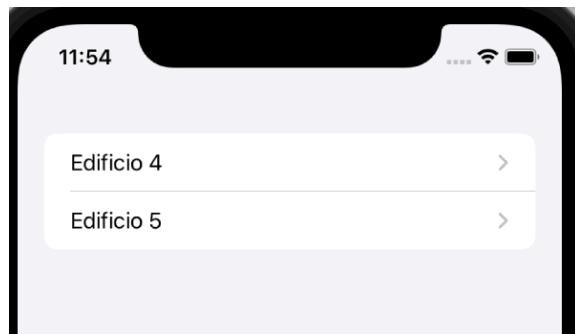


Figura 3.16: Visualizzazione rapida degli edifici del campus

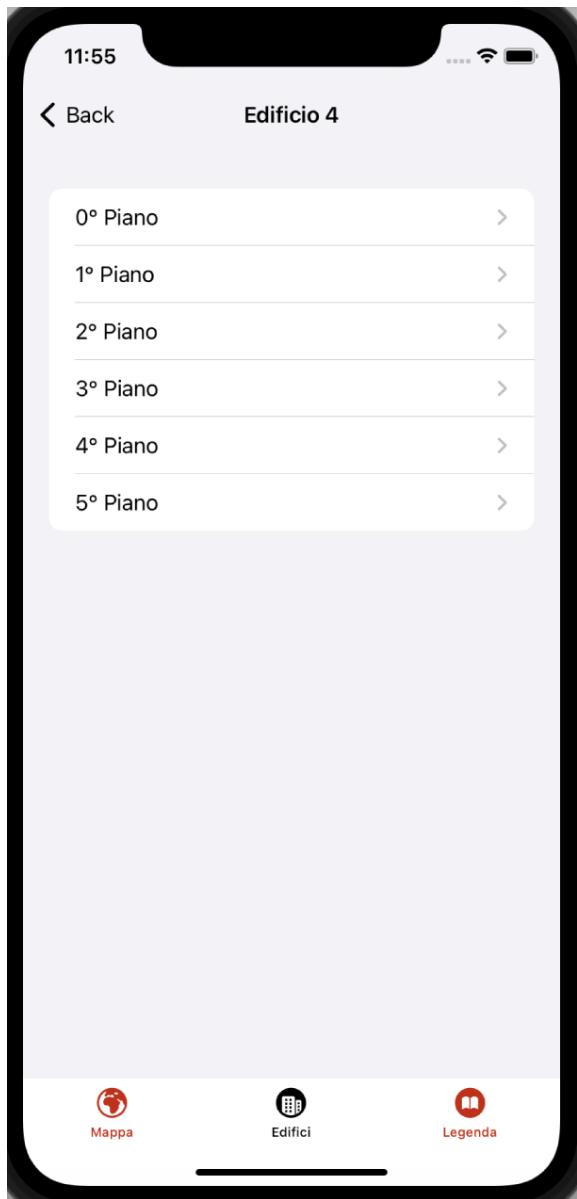


Figura 3.17: Visualizzazione rapida dei piani del campus

Come mostrato nella figura 3.18, abbiamo una `NavLink` per ogni spazio presente che rimanderà l'utente alla pagina di visualizzazione dei dettagli e dei sensori presenti in essi.

Possiamo osservare nella figura 3.19 come viene implementata la parte di visualizzazione degli edifici. Viene implementata creando una `NavLink` che ci porterà alla lista dei piani presenti, che a sua volta sarà così progettata che mostrerà tutti gli spazi in esso presente. Per rendere più leggibile il codice e anche più semplice da adattare a modifiche future, si è pensato di utilizzare una View separata per generare la riga in cui viene mostrato il nome dell'edificio. Questa operazione ci dà la possibilità di personalizzarla più facilmente.

Anche l'implementazione della lista degli spazi è molto simile, in quanto a sua volta comprende una `NavLink` che ci porterà alla pagina in cui verranno mostrate tutte le informazioni e gli eventuali sensori presenti, come mostrato in figura 3.21.

### 3.2.9 Visualizzazione delle Informazioni degli Spazi

Ogni spazio contiene diverse informazioni, ed alcuni di essi utilizzano, inoltre, dei sensori per raccogliere i dati ambientali che vengono poi mostrati all'utente. Nell'immagine 3.22 si può visualizzare come vengono mostrati all'utente i dati raccolti. Alcuni di essi, come temperatura, umidità e suono, vengono colorati in base ai loro valori. Verde se il valore è ottimale, Arancione se è buono e Rosso se è troppo alto o troppo basso.

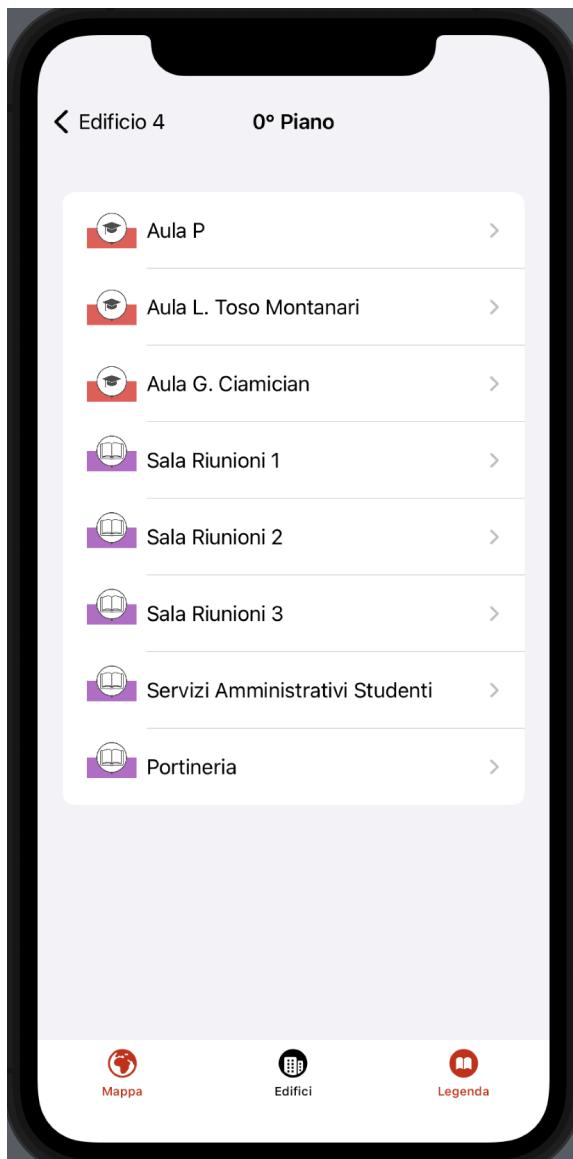


Figura 3.18: Visualizzazione rapida degli spazi del campus

```
43 var body: some View {
44     NavigationView(){
45         List(buildings){ building in
46             NavigationLink(destination: FloorView(viewModel: viewModel, building: building)){
47                 Text(building.name)
48             }
49         }
50     }
51 }
```

Figura 3.19: Implementazione della lista degli edifici

```
11 struct BuildingRowView: View {  
12  
13     var building: Building  
14     @ObservedObject var viewModel: DataLoader  
15     var body: some View {  
16         HStack {  
17             Text(building.name)  
18         }  
19     }  
20 }  
21  
22
```

Figura 3.20: Implementazione della riga dell’edificio

```
45     var body: some View {  
46         List(spaces) { space in  
47             if((space.legendId == 1 || space.legendId == 2 || space.legendId == 3 || space.legendId == 4) &&  
48                 space.floorId == floor.id ){  
49                 NavigationLink(destination: InformationSpaceView(space: space, viewModel: viewModel)){  
50                     SpaceRowView(space: space, viewModel: viewModel)  
51                 }  
52             }  
53         }.navigationTitle(String(floor.number) + "° Piano")  
54  
55     }  
56 }  
57  
58 }  
59
```

Figura 3.21: Implementazione della lista degli spazi

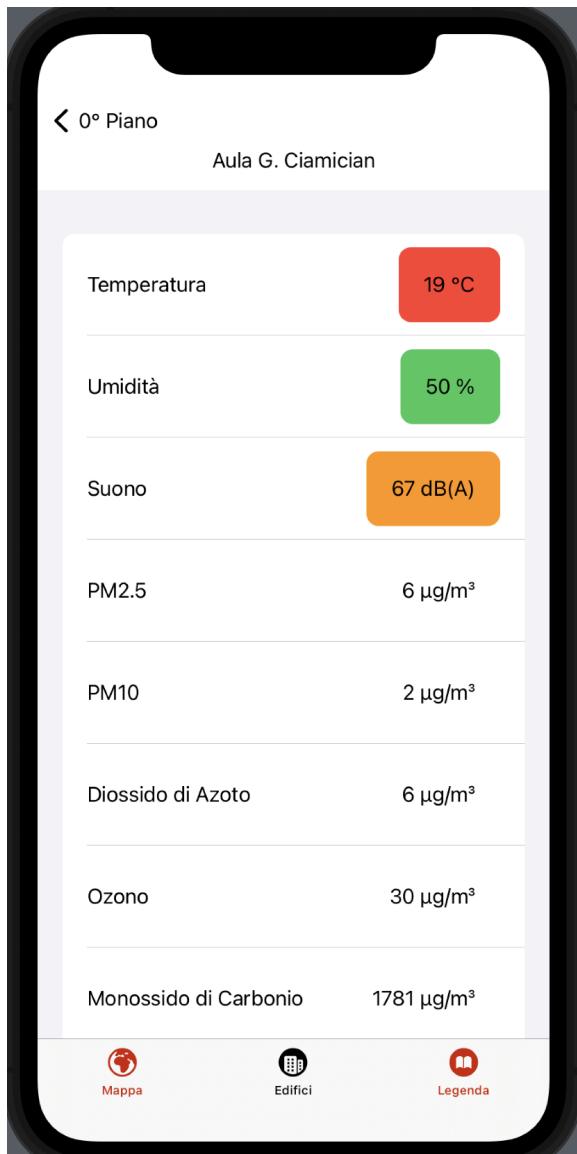


Figura 3.22: Visualizzazione dei dati raccolti dai sensori

# Conclusioni e Sviluppi futuri

Questa tesi ha presentato il sistema AlmaMap mobile. Essa nasce dalla possibilità di rendere fruibile, attraverso uno smartphone, il servizio della già presente AlmaMap senza la necessità di recarsi presso un chiosco touch del campus Navile. Il progetto inizialmente nacque in risposta ad una necessità del nascente campus che richiedeva un'applicazione per aiutare le persone con l'orientamento, fornendo mappe di tutti gli edifici e una funzione di navigazione con cui spostarsi senza smarirsi. AlmaMap mobile si serve delle interconnessioni già presenti tra l'app originale e il BIM, il quale funge da middleware tra l'applicazione e una fitta rete di sensori installati in tutto il campus, che forniscono informazioni in tempo reale sulle misurazioni dei vari parametri ambientali nelle aule e nei laboratori. Questi valori vengono poi utilizzati per calcolare il livello di comfort di ogni spazio monitorato; questi dati verranno mostrati agli utenti direttamente sul proprio smartphone come supporto al loro processo decisionale e per promuovere la consapevolezza in merito ai parametri ambientali nel campus e ad eventuali problemi di sostenibilità.

AlmaMap mobile ha lo scopo dunque di facilitare l'esperienza degli utenti all'interno del campus, senza dover necessariamente raggiungere un chiosco. La possibilità di spostare il servizio sul proprio telefono renderlo molto più utile e facile da comprendere, con la possibilità di orientarsi per le aree avendo sempre a portata di mano i valori ambientali, captati dai sensori, e le mappe tramite cui orientarsi.

**Sviluppi Futuri** AlmaMap è un sistema giovane, con molte potenziali funzionalità da implementare nello sviluppo futuro. Questa parte della tesi si concentra sulla possibilità di aggiungere al sistema funzionalità nuove, più complesse, interessanti e significative.

Una caratteristica da privilegiare è sicuramente l'implementazione della navigazione già presente nella versione AlmaMap Navile 2.0 [3] installata nei chioschi, per rendere più semplice il raggiungimento di una determinata zona o aula del campus. Al momento non c'è un metodo semplice ed efficace di localizzare lo studente all'interno del campus, motivo per cui la navigazione da mobile è stata momentaneamente omessa in questo progetto. Ma c'è spazio per migliorare ulteriormente e ciò non significa che essa non possa essere implementata in futuro. Essa porterebbe ulteriori implementazioni da sperimentare per rendere l'esperienza dell'applicazione sempre migliore e il più completa possibile.

L'integrazione AlmaMap-BIM potrebbe essere sfruttata per fornire dati sul comfort al BIM, in modo che possa utilizzarli per prendere decisioni indipendenti su HVAC e altri impianti, riducendo al minimo lo spreco di risorse e migliorando ulteriormente l'efficienza del sistema. Un'altra potenziale feature è la possibilità di fornire, tramite l'app, dei feedback sul livello di comfort indicato. Ciò permetterebbe di raccogliere dati che possano essere utilizzati per migliorare l'algoritmo, sfruttando eventualmente algoritmi di machine learning.

Un'altra interessante integrazione, che si potrebbe inserire sia nella versione mobile che nei chioschi, è un sistema analitico per raccogliere dati su come gli utenti interagiscono con le app. Sapere cosa fanno gli utenti, sempre nel rispetto della loro privacy, poiché l'interazione con l'app è anonima, può fornire spunti utili sulle funzionalità che preferiscono e sui problemi che incontrano, aprendo la strada a ulteriori miglioramenti futuri.

Ultimo, ma non per importanza, è rendere l'app più inclusiva, così da poter essere utilizzata da tutti tramite lo sviluppo nella sua controparte Android, così da permettere a molti più utenti di utilizzare questo servizio.

# Bibliografia

- [1] Chiara Ceccarini e Paola Solomoni Catia Prandi, Lorenzo Monti. Ssmart campus: Fostering the community awareness through an intelligent environment, 2020. URL <https://doi.org/10.1007/s11036-019-01238-2>.
- [2] Digital trends, 2023. URL <https://www.digitaltrends.com/computing/what-is-ambient-computing/>.
- [3] Gianni Tumedei. Towards a smart campus digital twin: Promoting awarness and sustainability through wayfinding and real-time environmental data., 2020. URL [https://www.fontbonne.edu/wp-content/uploads/2020/04/AA\\_Data\\_Visualization\\_White\\_Paper\\_Proof\\_1\\_20.pdf](https://www.fontbonne.edu/wp-content/uploads/2020/04/AA_Data_Visualization_White_Paper_Proof_1_20.pdf).
- [4] Ibm, 2023. URL <https://www.ibm.com/it-it/cloud/learn/what-is-mobile-cloud-computing>.
- [5] Sara Bartolini. Smart sensors for interoperable smart environment., 2009. URL [http://amsdottorato.unibo.it/2576/1/Bartolini\\_Sara\\_tesi.pdf](http://amsdottorato.unibo.it/2576/1/Bartolini_Sara_tesi.pdf).
- [6] J. Al Dakheel. Smart buildings features and key performance indicators: A review, 2020. URL <https://www.sciencedirect.com/science/article/pii/S2210670720305497/pdf>.
- [7] A. Sanguinetti e M. Pritoni K. Salmon, J. Morejohn. Research gate, 2021. URL <https://www.researchgate.net/publication/>

- 320866758\_The\_Iterative\_Design\_of\_a\_University\_Energy-Dashboard.
- [8] Michael Chee e Prashant Shenoy e Rajesh Balan Camellia Zakaria, Amee Trivedi e Emmanuel Cecchet. nalyzing the impact of covid- 19 control policies on campus occupancy and mobility via wifi sens- ing, 2022. URL <https://dl.acm.org/doi/10.1145/3516524>.
  - [9] Silvia Mirri e Catia Prandi Chiara Ceccarini. Designing interfaces to display sensor data: A case study in the human-building interaction field targeting a university community., 2021. URL <https://www.mdpi.com/1424-8220/22/9/3361>.
  - [10] S. N. Patel e E. D. Mynatt e B. Jones e G. Abowd J. A. Kientz, E. Prince. The georgia tech aware home, 2008. URL [https://homes.cs.washington.edu/~shwetak/papers/chi\\_ahome.pdf](https://homes.cs.washington.edu/~shwetak/papers/chi_ahome.pdf).
  - [11] European Commission. Smart cities and communities, 2023. URL <https://joinup.ec.europa.eu/collection/rolling-plan-ict-standardisation/rolling-plan-2023>.
  - [12] Centre of Regional Science. Smart cities ranking of european medium-sized cities, 2007. URL <https://joinup.ec.europa.eu/collection/rolling-plan-ict-standardisation/rolling-plan-2023>.
  - [13] National BIM Standard. Frequently asked questions about the national bim standard-united states, 2014. URL <https://web.archive.org/web/20141016190503/http://www.nationalbimstandard.org/faq.php#faq1>.
  - [14] P. Nowak e D. Philp e J. T. Snaebjornsson I. B. Kjartansdotir, S. Mordue. Building information modelling, 2017. URL <https://www.ciob.org/sites/default/files/M21%20%20BUILDING%20INFORMATION%20MODELLING%20-%20BIM.pdf>.

- [15] Lee Maguire. 4 ways apis impact the construction industry, 2022. URL <https://www.bimacademy.global/insights/digital-technologies/4-ways-apis-impact-the-construction-industry/>.
- [16] Chiara Ceccarini. Hci methodologies and data visualisation to foster user awareness. *Proceeding of the CHItaly*, 2892(1613):28–35, 2021.
- [17] Soar Higher. Seven steps to better data visualisation, 2023. [https://www.fontbonne.edu/wpcontent/uploads/2020/04/AA\\_Data\\_Visualization\\_White\\_Paper\\_Proof\\_1\\_20.pdf](https://www.fontbonne.edu/wpcontent/uploads/2020/04/AA_Data_Visualization_White_Paper_Proof_1_20.pdf).
- [18] Layers of data: How context helps make better decisions., 2023. URL <https://towardsdatascience.com/4432b855573c>.
- [19] Mappe digitali, 2023. URL [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjWrMzfifj-AhXaVPEDHSZUACkQFnoECA8QAw&url=https%3A%2F%2Fsd5bc3223aac5d21e.jimcontent.com%2Fdownload%2Fversion%2F1506005487%2Fmodule%2F13242426924%2Fname%2Fcorso%2520cartografia%25202.pdf&usg=A0vVaw3HC2LSGvLQWYViL-Y\\_eYsX](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjWrMzfifj-AhXaVPEDHSZUACkQFnoECA8QAw&url=https%3A%2F%2Fsd5bc3223aac5d21e.jimcontent.com%2Fdownload%2Fversion%2F1506005487%2Fmodule%2F13242426924%2Fname%2Fcorso%2520cartografia%25202.pdf&usg=A0vVaw3HC2LSGvLQWYViL-Y_eYsX).
- [20] Kevin Andrew Lynch. *L'immagine della città*. Marsilio Editori spa, 1960.
- [21] Rick W Sturdevant. Navstar, the global positioning system: a sampling of its military, civil, and commercial impact., 2020. URL <https://history.nasa.gov/sp4801-chapter17.pdf>.
- [22] Safe Graph. 12 methods for visualizing geospatial data on a map, 2023. URL <https://www.safegraph.com/guides/visualizing-geospatial-data>.
- [23] Catia Prandi e Silvia Mirri. Lorenzo Monti. Iot and data visualization to enhance hyperlocal data in a smart campus context., 2018. URL <https://doi.org/10.1145/3284869.3284878>.

- [24] Desigo. Desigo, 2023. URL <https://new.siemens.com/global/en/products/buildings/automation/desigo.html>.
- [25] World Health Organization. World health organization, 2023. URL <https://www.who.int>.
- [26] European Air Quality Directive. European air quality directive., 2023. URL [https://environment.ec.europa.eu/topics/air\\_en](https://environment.ec.europa.eu/topics/air_en).
- [27] Node.js. Node.js, 2023. URL <https://nodejs.org/en/about>.
- [28] Apple Developer. Coredata, 2023. URL <https://developer.apple.com/documentation/coredata/>.
- [29] Apple. Swift. un linguaggio potente e aperto a tutti per creare fantastiche app., 2023. URL <https://www.apple.com/it/swift/>.
- [30] Apple. SwiftUI, 2023. URL <https://developer.apple.com/xcode/swiftui/>.
- [31] Apple. Uikit, 2023. URL <https://developer.apple.com/documentation/uikit>.
- [32] Apple. Sf symbols, 2023. URL <https://developer.apple.com/sf-symbols/>.
- [33] GitHub. Svgview, 2023. URL <https://github.com/exyte/SVGView>.
- [34] Apple. Xcode, 2023. URL <https://developer.apple.com/xcode/>.
- [35] Visual Studio Code. Visual studio code, 2023. URL <https://code.visualstudio.com>.
- [36] Git. Git, 2023. URL <https://git-scm.com>.
- [37] GitHub. Github, 2023. URL <https://github.com>.
- [38] Alessio De Vita. *Geolocalizzazione e wayfinding indoor e outdoor per utenti con disabilità*. Addison-Wesley, 2019.

- [39] Locatify. Indoor positioning systems based on ble beacons., 2023. URL [https://locatify.com/blog/indoor-positioning-systems-ble-beacons/#:~:text=How%20does%20BLE%20based%20Indoor%20Positioning%20work%3F&text=BLE%20beacons%20are%20the%20cornerstones,see%20Trilateration%20vs%20Proximity%20detection\).](https://locatify.com/blog/indoor-positioning-systems-ble-beacons/#:~:text=How%20does%20BLE%20based%20Indoor%20Positioning%20work%3F&text=BLE%20beacons%20are%20the%20cornerstones,see%20Trilateration%20vs%20Proximity%20detection).)
- [40] RFID Global. Tecnologia rfid, 2023. URL <https://www.rfidglobal.it/tecnologia-rfid/>.
- [41] Esri, 2023. URL <https://www.esri.com/en-us/arcgis/products/mapping/overview>.
- [42] Cloud optimized geotiff, 2023. URL <https://www.cogeo.org/>.
- [43] Google maps, 2023. URL <https://www.google.com/maps>.
- [44] Openstreetmap, 2023. URL <https://www.openstreetmap.org/>.
- [45] BACnet. Bacnet, 2023. URL <https://bacnet.org>.