Politecnico di Torino

Politecnico di Torino

Master's Degree in Ingegneria Informatica (Computer Engineering)

A.y. 2025/2026

Graduation Session March 2026

Design and implementation of an interactive navigation module for the PoliTO Students App

Supervisors:

Luigi De Russis
Cristina Ferrian
Federico Cucinella
Giuseppe Piazza

Candidate:

Giacomo Belluardo

Always $trieve And Prosper

Summary

Over the past decade, the increasingly widespread use of mobile devices has radically standardized outdoor navigation thanks to the precision and reliability of global positioning systems. However, these technologies prove ineffective in indoor environments, where signals are attenuated by architectural barriers. Within the context of Politecnico di Torino, there is a need to support first-year students in orienting themselves inside buildings, both for everyday use and emergency situations. Currently, the university's digital ecosystem offers tools such as Maps on the official website and the "Places" section of the Students App, but none of these include an actual navigation functionality. Therefore, this thesis focuses on the implementation of an indoor navigation module integrated into the "Places" section of the Students App. The feature allows users to calculate a route by entering a starting point and a destination. The system leverages a complete, multi-floor floorplan mapping of indoor spaces, with segments classified by type and accessibility level. This enables preliminary filtering of paths and identification of the optimal route using a goal-oriented routing algorithm. In addition to the route itself, the module provides information such as estimated travel time and the number of stairs and elevators. The interface allows place selection through textual input or directly from the map and displays the route—once origin and destination are confirmed—divided by floors with dynamic maps. The effectiveness and usability of the interface were evaluated through a study involving a group of students, whose positive results confirmed the validity of the adopted approach. The proposed work fills a gap in the university's digital ecosystem, offering students an intuitive solution that reduces travel time and facilitates indoor orientation.

Acknowledgements

My first thanks go to my supervisor, Luigi De Russis, who supported me throughout the entire thesis, along with the ISIAD team—Cristina Ferrian, Federico Cucinella, and Giuseppe Piazza—who were always ready to help with any issue.

I am deeply grateful to my parents and my entire family, including those who are no longer with us: you believed in me since 'day zero,' despite my stubbornness and my bad days. A special thanks to the 'carusi' from back home, companions of a thousand adventures: with you, 'A casa è a casa.'

An immense thank you to the Turin boys, who managed to take my mind off the stress (often making me forget everything else!). Finally, but no less importantly, thanks to the guys from Southopia and Valdo Fusi: skateboarding and our sessions, between party and street skating, have been my essential daily outlet.

Skateboarding and fiesta 4L!!

Table of Contents

List of Tables

List of Figures

Glossary

ISIAD
: Infrastrutture Servizi Informatici e Amministrazione Digitale.

LBS
: Location Based Services.

GPS
: Global Positioning System.

IPS
: Indoor Positioning System.

BLE
: Bluetooth Low Energy.

UWB
: Ultra-WideBand.

AR
: Aumented Reality.

POI
: Point of Interest.

UI
: User Interface.

API
: Application Programming Interface.

DBMS

Database Management System.

JSON

JavaScript Object Notation.

Chapter 1

Introduction

1.1 Context

In the last decade, the increasingly widespread use of mobile devices, such as smartphones and smartwatches, and the advent of location-based services (LBS) have radically changed the approach with which users interact with the surrounding environment. While outdoor navigation, implemented by applications such as Google Maps, has by now reached standards of reliability and precision thanks to the Global Positioning System (GPS), navigation in indoor environments turns out to be still an open technological challenge and sometimes complex to realize.

Such disparity of service between outdoor and indoor navigation is attributable to the drastic attenuation of the signal caused by architectural barriers that often make GPS ineffective for precise localization (a problem known as the "last mile").

Consequently, research has concentrated on the development of alternative Indoor Positioning Systems (IPS), exploiting pre-existing or dedicated infrastructures, like Wi-Fi networks (through fingerprinting techniques), Bluetooth Low Energy (BLE), the inertial sensors of smartphones (accelerometer, gyroscope) and, more recently, Ultra-Wideband (UWB) technology and Augmented Reality (AR).

Shifting the issue into the context of the Politecnico di Torino, the development of an indoor navigation system arises from the need to have support for people who want to orient themselves or navigate within the campus. The main target consists of freshmen, often lacking familiarity with the university places; however, the solution is designed also for external visitors or occasional users.

The issue expands when considering themes of accessibility, emergencies, or momentary changes in the spaces. It is therefore necessary that the system guarantees accessible paths for users with motor difficulties. This behavior also applies to navigation towards gathering points and the identification of alternative paths when there are works or construction sites that hinder the crossing of specific

spaces.

At the current state, the digital ecosystem of the Politecnico offers the cartographic visualization of maps on the official website and within the Students App. For the purposes of the thesis, the Students App reflects the most suitable environment for the implementation of a navigation module; furthermore, it is the means most chosen by students. In detail, the app is a project realized in collaboration with ISIAD, but is structured to welcome contributions from students, both in terms of development and through validation activities like usability tests.

The app offers various functionalities useful for the management of a student's academic life, such as a section dedicated to Courses, a completely customizable agenda, and a section dedicated to Places. The Places section provides students with a set of floor plans of the different floors of the city campuses, integrating a geographic database to have the various information of the places. The maps, implemented on a cartographic base provided by Open Street Map, are complemented by a system of markers that present the main categories of POIs, allowing users a first approach to orientation inside the buildings.

A first attempt of indoor navigation implementation was proposed by an internal initiative at the Politecnico, realized as a multidisciplinary challenge and titled "*Mapping At PolitTO*"[1]. The analysis carried out by the various groups participating in the challenge highlighted the absence of the feature as a major cause of spatial disorientation, a problem that affects in particular freshmen and non-habitual users. Such inefficiency impacts negatively on the didactic logistics, causing delays or often missed attendance at lectures.

1.2 Objective of the thesis

The primary objective of this thesis is the design and integration of an indoor navigation feature within the Students App, driven by an analysis of student feedback regarding the current implementation.

The specific objectives are:

- **Requirements Definition**: Deriving system specifications directly from identified user pain points.
- **System Design**: Defining the architectural components required for feature integration, spanning from abstract modeling to code-level implementation.
- **Technical Verification**: Testing the implemented feature to gather feedback and assess system efficiency and user satisfaction.
- **Database-side Pathfinding Optimization**: Implementing database-integrated routing logic to minimize response latency and ensure high system scalability.

The thesis focuses on specific student user flows: locating remote points of interest, pathfinding for poorly signposted or unvisited locations, and facilitating orientation during floor or building transitions necessitated by class schedules. For each scenario, the aim is to generate and display an in-app minimum-distance path to effectively guide the user to their destination.

1.3 Structure of the thesis

To provide a comprehensive view of the work done, the thesis has been organized into seven distinct sections. Each chapter represents a fundamental piece in the application lifecycle, from theoretical research to practical implementation. Following the initial premises, the content is structured as follows:

Chapter 2, Background: This chapter outlines the general overview of the problem by comparing it with current solutions, before focusing on the practical application within the University premises. The investigation also takes into account prior academic work, citing and analyzing results obtained in previous theses on the same topic.

Chapter 3, Requirements: This section describes the technical requirements of the project related to both the visual interface with which the user interacts (FrontEnd) and the internal logic that drives the entire system (BackEnd).

Chapter 4, Design: This chapter analyzes the core of the design work, tracing a path from conceptual analysis to the concrete development of the solution. After defining models based on real use cases, the text moves on to the actual construction of both the FrontEnd and the BackEnd.

Chapter 5, Implementation: This section illustrates the implementation of the solution, analyzing the technologies employed, the user interface—including its specific components—and the integration necessary for data management.

Chapter 6, Evaluation and Benchmarks: A section dedicated to discussing the results obtained from the various tests performed throughout the different development phases, while also evaluating the performance of the internal logic.

Chapter 7, Conclusion and future developments: This chapter summarizes the results obtained, offering critical reflections on the work performed and focusing on potential future implementations.

Chapter 2

Background

As outlined in the introduction, the core focus of the thesis is the development and integration of indoor navigation within the official Politecnico Students App.

The aim of the work is to overcome the limitations of the application, currently restricted on providing static cartographic representations of the university facilities without any wayfinding solutions.

Rather than simply offering static points on a map, the proposed system is designed to generate dynamic routes, effectively guiding the user through the Politecnico's complex architectural layout.

2.1 Overview of the problem

Currently, to find their way around the places, students rely on the "*Luoghi*" (Places) section of the official app. Despite this section of the application allows users to get the details of a room, such as its position or its specific campus, the support it offers is incomplete and the practical navigation remains unresolved for both freshmen and more experienced students. As mentioned in the previous chapter, the difficulty of wayfinding was strongly underscored during the "*Mapping at PoliTo*" challenge dedicated to improving the University's digital services.

From the challenge results and specifically from the analyses presented by the MEW and TerraByte teams—an alarming fact emerged: current map management not only causes inconvenience but directly impacts academic activities. Students often arrive late or miss lectures entirely due to the time wasted searching for the correct room. Effectively, what is missing is a turn-by-turn system capable of bridging the gap between the static map and the physical reality of the building.

2.2 Existing solutions

Looking at the external market to find solutions for similar problems, the example of Tap My Life Srl [2] stands out.

This company has consolidated its expertise in the healthcare sector, a context that shares specific challenges with the environment of the university, such as overcrowding and structural complexity.

Their technological response relies on an IoT platform designed to extend GPS precision into indoor environments, with a specific focus on inclusivity for users with specific accessibility needs.

By applying this model to the university context, the company proposes hybrid solutions: web platforms accessible via the university official website or on-site touch-screen Totem, which automatically compute the route starting from the device's specific location.

Figure 2.1: TapMyLife logo

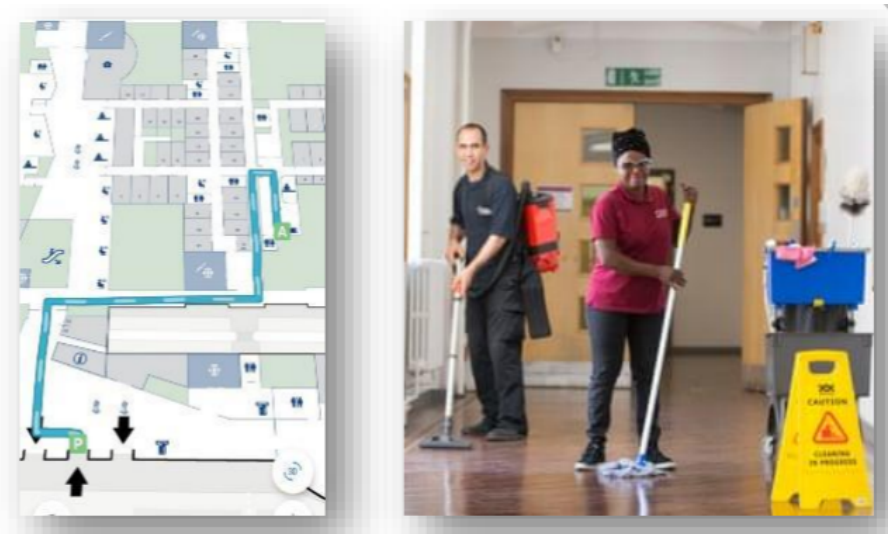

Figure 2.2: TapMyLife Navigation UI

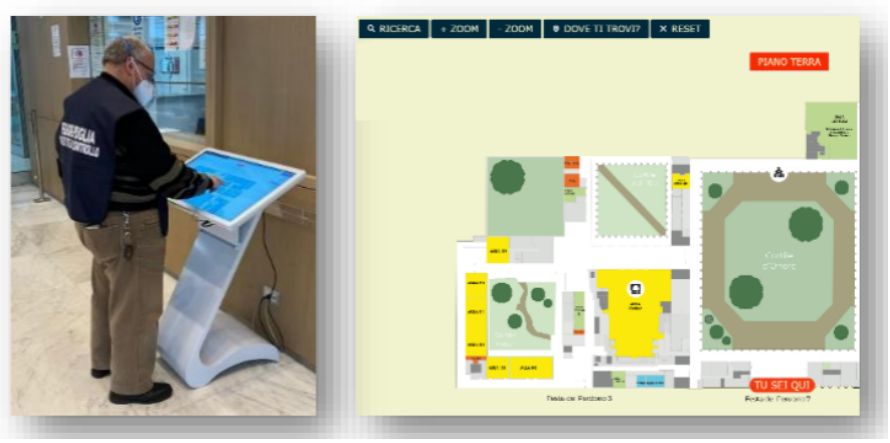

Figure 2.3: TapMyLife Totem interaction

Taking into account the Italian academic landscape, a concrete example is provided by the Politecnico di Milano[3]. The institution has integrated a detailed mapping of accessible routes into its web portal, enhancing the navigation experience with advanced search features and the inclusion of new POIs strategic for university life.

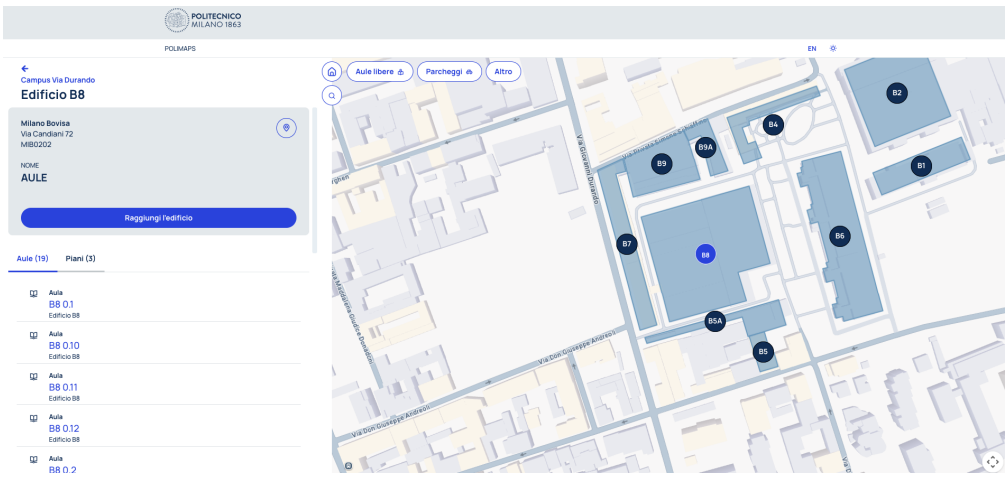

Figure 2.4: Politecnico di Milano Navigation UI

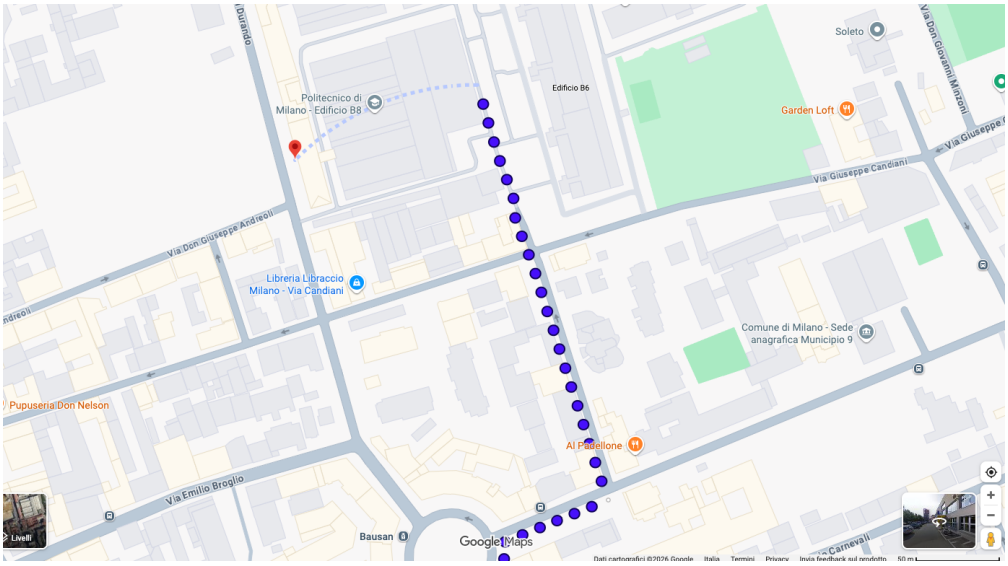

Figure 2.5: Google Maps provider after interaction with Politecnico di Milano website

2.3 Current state of the problem within the university context

In the current digital ecosystem of the Politecnico di Torino, the two reference points to orient oneself within the university are the map system of the official website[4] and the Students App, which presents the 'Places' section for the consultation of the maps and related details.

From an architectural perspective, spatial information management is centralized server-side: the University maintains building floor plans by managing CAD files through Archibus, a Facility Management platform for space management. This system associates the graphic data with a set of essential metadata, such as accessibility, visibility (public/private), floor number, building structure, and campus.

Currently, the presentation of this data for client-side use (mobile app, web) relies on an in-house developed system consisting of two distinct components. A back-office component handles—either manually or periodically upon detection of updates—the parsing and georeferencing of floor plans and metadata from Archibus, exporting them into an internal staging database. The presentation component, conversely, retrieves data via REST APIs (for metadata consultation) and handles the rendering of "*tiles*"[5] for visualization within interactive maps implemented by the client. Specifically, the tile server renders OpenStreetMap (OSM) maps on

separate layers (supporting both light and dark themes) and generates a specific layer with a transparent background for each floor to allow the overlay of the floor plans.

2.4 Solutions already available (previous theses)

QRCode thesis

Among the solutions developed internally at the university, the 2022 project by candidate Elchin Fahradov (supervisor Prof. G. Malnati) [6] is worth mentioning. This prototype relies on QR Codes to anchor the user's position within the building.

However, an operational analysis of this solution revealed substantial limitations. The primary critical issues concern the forced interaction: the student must be in the immediate proximity of a code to update their position, resulting in discontinuous navigation. Additionally, this approach requires the systematic deployment of numerous QR codes across every area of the university, inevitably leading to potential coverage gaps (blind spots).

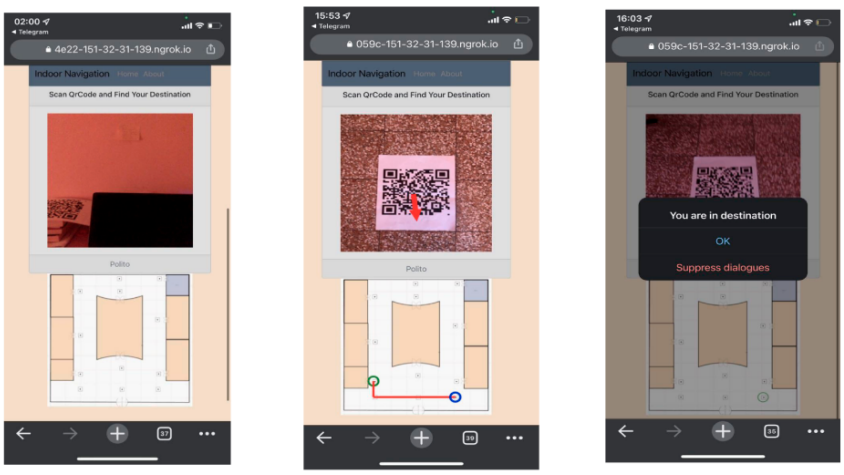

Figure 2.6: UI of the QRCode Thesis

Interdisciplinary project

Another significant proposal was developed at the Politecnico di Torino through an interdisciplinary project. The objective was to overcome the limitations of GPS/GNSS technology—effective outdoors but unreliable indoors due to signal shielding by architectural structures—by implementing a real-time positioning

system based on multilateration (deriving position by measuring distances from multiple points).

The technology selected for the indoor environment was Ultra-Wideband (UWB), chosen for its ability to guarantee high-precision positioning. This technology was paired with a specific hardware architecture comprising Raspberry Pi units coupled with a Pozyx sensor kit. The configuration requires four Raspberry Pis acting as 'anchors,' tasked with calculating the position via trilateration; subsequently, the data is transmitted by a Python script utilizing the MQTT protocol[7].

On the frontend side, the experience was integrated into the Students App using the Flutter framework, enabling the use of interactive maps via the flutter_map library. For the server side, a RESTful architecture was designed to aggregate data transmitted from the various connected sources. Although both client-side and server-side tests confirmed the technical validity of the system and compliance with requirements, the project's scalability faces significant practical hurdles, specifically the complexity of physically installing sensors throughout the university spaces and the high costs of the hardware required to cover large surface areas.

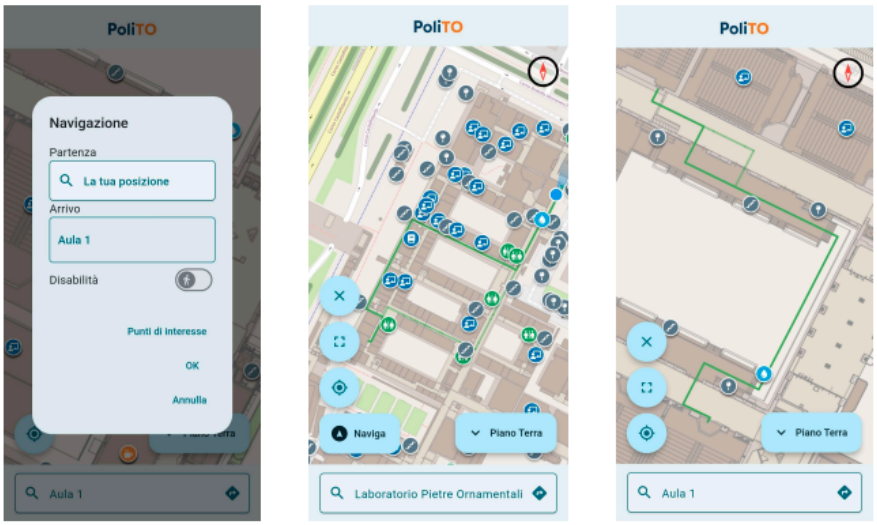

Figure 2.7: UI result of the Interdisciplinary project

Chapter 3

Requirements

This chapter lays the design foundations of the thesis, establishing the essential requirements for the development of the new indoor navigation feature. The definition of specifications is organized to reflect the software architecture, separating the needs of the user interface from those of the system logic.

Specifically, the FrontEnd section establishes how the application should behave in terms of interactivity and graphical presentation, emphasizing usability to ensure intuitive orientation. In parallel, the BackEnd section defines the technical specifications for spatial data management: from the extraction of structural information to the construction of a navigable topological model, up to the actual calculation of the optimal path.

3.1 FrontEnd Requirements

The FrontEnd requirements emerge from a User-Centered Design (UCD) approach, where the primary goal is to translate identified user difficulties into functional interface capabilities. These specifications prioritize usability and accessibility, ensuring that the new feature integrates seamlessly with the pre-existing UI/UX patterns of the Students App while adopting intuitive navigation standards familiar to modern mobile users.

3.1.1 Interaction requirements

This section is dedicated to analyzing the user interaction requirements with the application. The objective is to define the system's response logic regarding user input, ensuring a User Experience that, while maintaining consistency with what was already defined in the Students App, is intuitive.

Key elements of particular relevance include:

- **Point of Interest Selection (Origin/Destination)**: Users must have flexibility in selecting points of interest through multiple input options: list selection, search bar utilization, or direct map interaction. This approach aims to replicate interaction patterns already established by major navigation applications, reducing learning time and improving feature intuitiveness.
- **Accessibility and Inclusivity Management**: Although any public structure offers accessible routes even for people with disabilities, the system must be able to calculate routes taking into account the user's specific mobility needs. Consequently, the user interface must provide a dedicated selection mechanism: an option is needed to signal a disability that follows the standard of inclusive design, using neutral and respectful terminology that avoids any form of discrimination in the usage experience compared to able-bodied users.

3.1.2 Presentation requirements

This section addresses visual specifications governing user-system interaction. In mobile applications, presentation transcends esthetics, constituting a determining factor for data comprehension and the reduction of cognitive load.

These directives govern both esthetic and structural aspects of the Frontend, establishing a visual environment that balances appeal with functionality.

Specifically, the most relevant requirements are:

- **Route Visualization and Interactions**: Managing architectural complexity constitutes a critical presentation requirement. Given the multi-floor nature of the campus (ranging from the second basement to the fourth floor), the user interface must ensure smooth route visualization, resolving issues related to floor changes and user orientation.

  Therefore, it becomes essential to implement a visualization mode that not only displays the route, but also provides clear spatial context: the system must be able to identify and show the segment with the relative floor plan where it is located, thus facilitating orientation during indoor navigation.

  A further requirement concerns interaction with the map: the displayed route segment must be selectable, thus offering granular control over the "stages" of the movement.

- **Dynamic Route Update Upon Parameter Change**: Dynamic route updating completes the relationship between visual and interactive components. The intent is to implement a reactive architecture, capable of managing in real-time the system's behavior in response to any variation in the input state.

  Specifically, attention is focused on modifying navigation parameters: starting point, destination, and accessibility preferences (as discussed in previous

requirements). The system must be able to detect such changes and instantly trigger the route recalculation, returning a new visualization consistent with the updated data.

From the user interface (UI) perspective, this translates into the need to integrate directly interactable visual controls. These elements must allow the user to modify parameters while keeping the application's functional flow intact and ensuring smooth operational continuity.

3.1.3 Usability requirements

In conclusion of the Frontend analysis, this section addresses usability—a dimension that extends beyond technical execution to encompass overall user experience quality.

Usability represents the convergence point between the previously examined Presentation and Interaction specifications. While the latter define respectively the graphical interface and behavioral logic, usability requirements represent the measure of effectiveness and efficiency with which the user manages to interact in order to achieve their objectives.

The following criteria formalize the technical specifications according to the User-Centered paradigm: the objective is to minimize cognitive effort and implement error prevention mechanisms, thus facilitating the adoption of the new indoor navigation functionality.

Specifically, attention is directed toward:

- **Stylistic and Functional Consistency**: It is necessary that the navigation feature integrates into the existing ecosystem, inheriting as much as possible the design system of the Students App. This visual continuity is not only for aesthetic purposes, but also for functional purposes: it allows the user to apply already acquired mental patterns and models, thus reducing the learning curve in using the new feature.

- **Information Density Management**: As emerged from the analyses carried out during the interdisciplinary challenge, the representation of complex maps risks generating visual saturation. It is therefore necessary to adopt a selective approach in displaying information, showing only what is deemed relevant. Examples include the dynamic visualization of markers (POIs), vertical connections (stairs and elevators), or the visualization of the route portion that the user is traversing.

- **Route Readability and Contrast**: To ensure an experience that is free of ambiguity, it is essential that the route line is adequately distinguished from the background floor plan using the right chromatic contrast. However, this

choice must be balanced between high visibility during navigation and respect for the color palette already defined by the application's stylistic guidelines. Further attention must be given to sizing the thickness of the represented route to be evident, without overshadowing other adjacent elements, such as the markers of points of interest (POIs).

3.2 Back-End Requirements

The design of the server component is driven by a systematic analysis of the operational environment and the technical constraints imposed by the university infrastructure. This analysis is grounded in the integration of the results from the Challenge mentioned in the previous chapter and a technical assessment of the current system status. This process was structured around three points:

- **Data Infrastructure Audit**: An examination of existing CAD files and database tables was essential to understand structural data storage patterns and define the transformation logic required for pathfinding calculations.
- **User Base Profiling**: Drawing on evidence from the Challenge, a study of the student population was conducted to estimate the potential user load and relative concurrent traffic volumes.
- **Maintenance Constraint Analysis**: Observing the frequency of university floor plan updates highlighted the critical need for an automated system capable of ensuring data integrity without repetitive manual interventions.

This methodological approach ensures that the technical requirements presented below are a direct response to the operational criticalities identified in the field.

These requirements, acting as the primary architectural drivers, ensure that the system remains reliable and efficient under real-world conditions.

3.2.1 Data Maintainability

This requirement stems from the dynamic nature of the university's spatial data. Since the Cad and geometric sources can be subject to periodic structural modifications, manual updates would be unsustainable and prone to error. Therefore, the system must guarantee information consistency through an automatic regeneration mechanism from raw sources. The architecture must allow complete reconstruction of the navigation database at any time, avoiding manual incremental update procedures. This ensures immediate alignment with infrastructural modifications made by the Politecnico and guarantees data integrity over time.

3.2.2 Performance

Performance requirements are dicated by the real-time nature of mobile navigation. In a wayfinding context, the utility of the application depends on near-instantaneous feedback; high latency would render the service unusuable for a user who is moving. The backend must, therefore, guarantee adequate performance in executing the required operations, maintaining contained response times and controlled latency. The selection of algorithms and data structures must favor solutions with optimal computational complexity relative to the application domain constraints.

3.2.3 Scalability and Elasticity

Due to the potential user base, which consists of thousands of students, the issue of scalability and elasticity emerges. The system must maintain acceptable performance levels as the workload varies. The architecture must support automatic scaling mechanisms, capable of detecting load increases and dynamically allocating additional computational resources to prevent service degradation, while optimizing operational costs through appropriate instance sizing.

Chapter 4

Design

This chapter translates the theoretical requirements previously analyzed into a concrete design solution, aimed at integrating the indoor navigation module within the Students App. The entire development process has been guided by the need to rigorously satisfy the functional specifications and usability constraints established in the previous chapter.

The analysis is structured following a logical path from abstraction to software engineering:

- **Conceptual and Functional Design**: Initially, the core system functionalities are formalized, defining use case scenarios to establish how the user and system interact across various operational contexts.
- **User Interface Prototyping (UI Design)**: Subsequently, the evolution of the user interface is illustrated through the definition of graphical prototypes. This section also documents the evaluation phases and revisions made to optimize user experience.
- **System Architecture**: With the design vision established, the actual technical architecture is defined, decomposing the system into two fundamental layers: the Frontend layer and the Backend layer.
- **Communication Protocols**: The chapter presents data exchange logic, defining the APIs and methods that coordinate dialogue between the user interface and server infrastructure.

4.1 Conceptual design

The Conceptual Design section formalizes the requirements presented in the analysis of the previous chapter, synthesizing them into coherent functional modules.

This synthesis method defines the interaction modes between the user and the system through the modeling of realistic usage scenarios.

The choice of this approach is justified by the need to convert abstract concepts into rigorous technical specifications, guaranteeing that the technical design of the system is aligned with the operational requirements.

4.1.1 Definition of functionalities

In this phase, the requirements emerged from the analysis of Chapter 2 are reinterpreted and formalized in terms of the operational capabilities that the system must offer. The purpose is to reduce the gap between the abstract definition of requirements and their technical implementation through a reorganization into structured functionalities.

To this end, the requirements have been subdivided into three logical macrosections, each inherent to a specific functional domain of the system. This classification also constitutes an architectural choice aimed at favoring a modular design, allowing for the separation of concerns and promoting a clear distinction of the objectives of each component. The result is an architecture composed of interconnected but independent modules.

The derived modules are:

- **Localization and Mapping Functionalities**: manages the spatial representation of the environment and the real-time positioning of the user, ensuring the necessary spatial context for navigation.
- **Navigation and Routing Functionalities**: handles the calculation of optimal paths and dynamic re-routing, with a specific focus on accessibility and the management of temporary obstructions.
- **Advanced Search Functionalities**: allows for the filtering and retrieval of specific points of interest based on user-defined criteria, such as accessibility needs or proximity.

This setup facilitates the scalability of the system, contributing to making the entire process more structured and sustainable over time.

Localization and Mapping Functionalities

Localization and mapping functionalities provide users with an experience articulated across three main points:

- Contextual visualization of university floor plans, exposing the interactive map's main functionalities.

- Visibility of the calculated route with relevant details (number of stairs and/or elevators, actual distance).
- Floor change management, enabling visualization transition to the selected floor's plan.

Navigation and Routing Functionalities

Navigation and routing constitute the system's logical core. The system calculates the optimal route, considering any accessibility requirements specified by the user.

The functionality provides a guided route subdivided into steps, organized according to the building's different floors.

Advanced Search Functionalities

The system provides users with two Point of Interest (POI) search modalities:

- Textual search with selection from the results list
- Visual search with direct marker selection on the map, within the corresponding floor's plan

4.1.2 Definition of use cases

Use cases describe, step by step, user interactions with the application to achieve a specific objective.

Scenario definition was conducted in collaboration with ISIAD's Web and Mobile presidium domain expert, Cristina Ferrian, and UX/UI design personnel, Giuseppe Piazza.

Scenarios were defined focusing on the different interaction modalities through which users reach a specific destination.

The following actors involved in use cases were identified:

- User: individual utilizing the application to reach a location from a determined starting point.
- System: represented by the Students App, acquires user input and calculates the route.
- Database: storage system containing tables related to locations and the implemented topological model.

Subsequently, two use cases were defined, distinguished according to the user's initial interaction modality with the application.

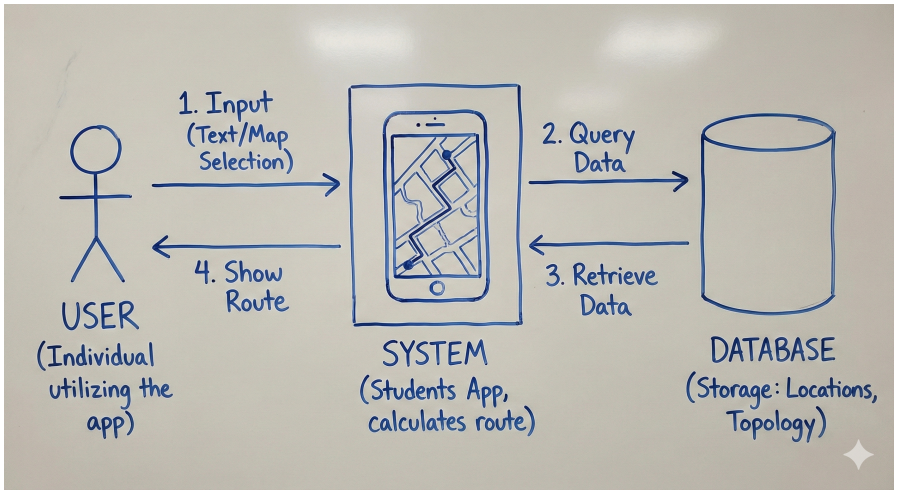

Figure 4.1: System context diagram

Use Case 1

The first case analyzes the scenario where users select the destination location directly from a list.

Users type in the text field and select the item corresponding to the desired destination.

After pressing the "*Get directions*" button, two scenarios unfold:

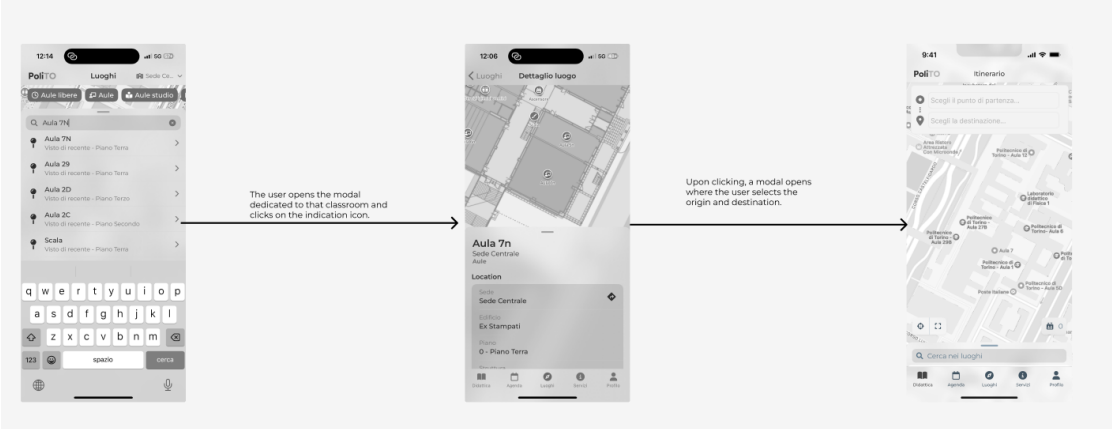

Figure 4.2: Use case 1 starting phase

- Geolocation enabled: users select the "*Your location*" item as the starting point.

Figure 4.3: Use case 1: device geolocation enabled

- Geolocation not enabled: users manually type the starting location or select it from list.

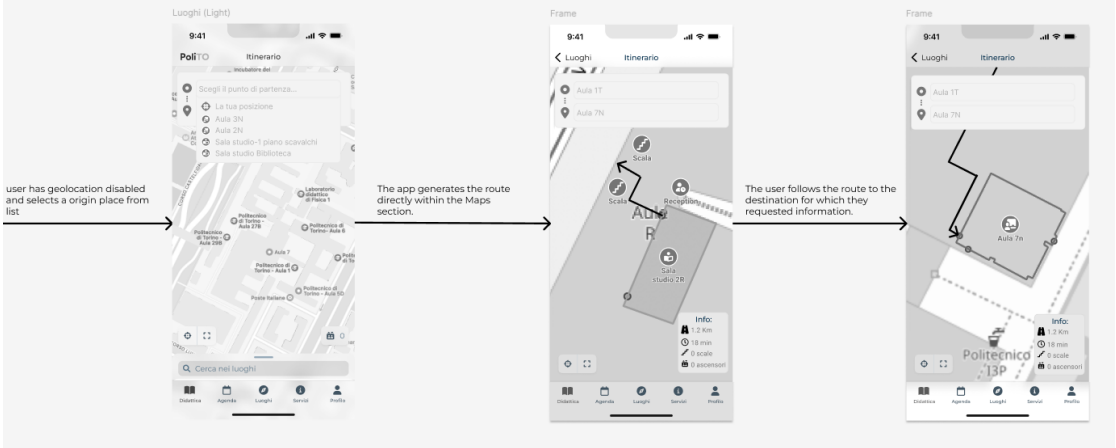

Figure 4.4: Use case 1: device geolocation disabled

Once starting point selection is completed, users generate the route and navigate through various steps, organized by floor.

Use Case 2

In the second use case, users first access the page related to a course, subsequently accessing the page of the location where the lecture is held.

After pressing the *Get directions* button, two scenarios unfold:

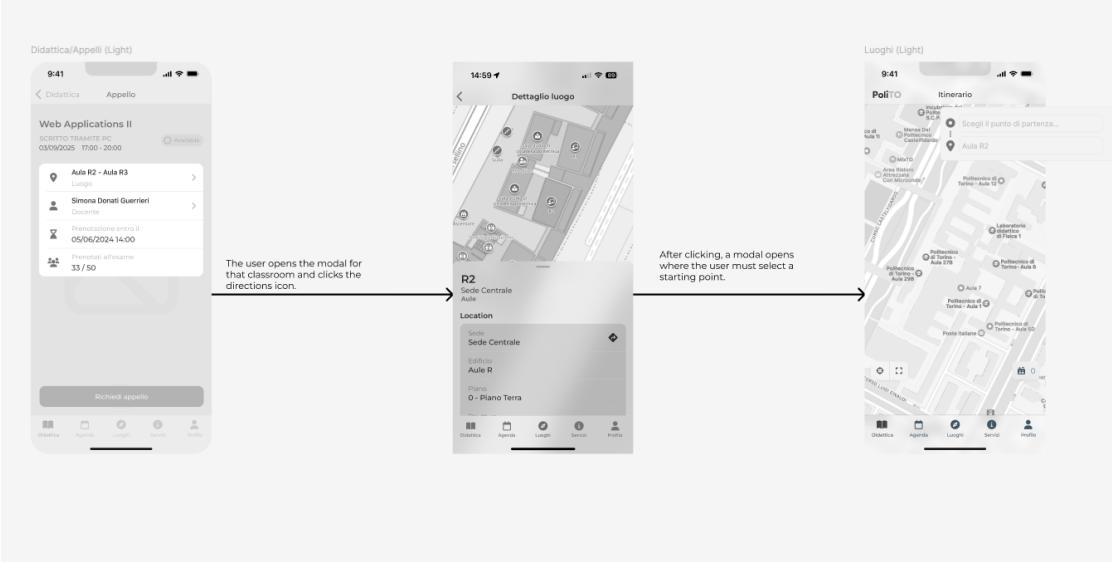

Figure 4.5: Use case 2 starting phase

- Geolocation enabled: users select the *Your location* item as the starting point.

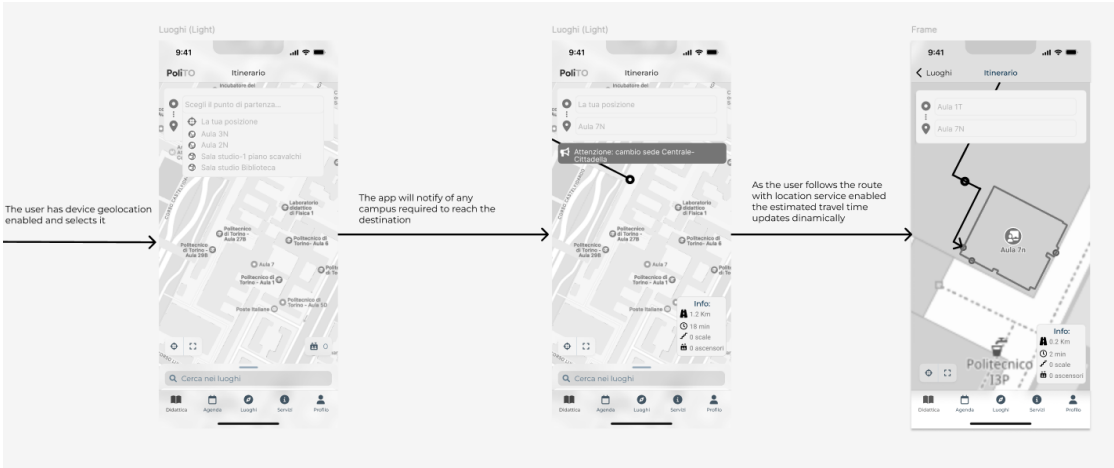

Figure 4.6: Use case 2: device geolocation enabled

- Geolocation not enabled: users manually type the starting location.

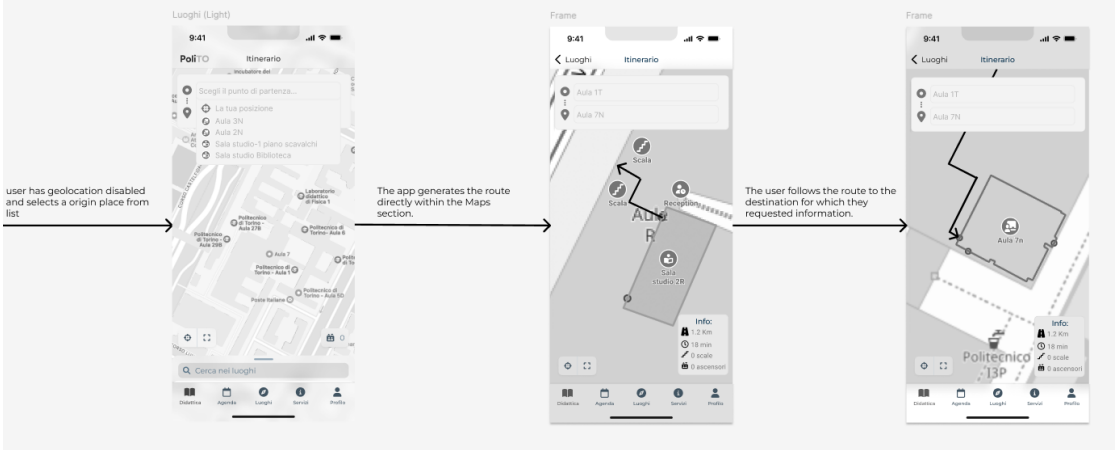

Figure 4.7: Use case 2: device geolocation disabled

Once starting point selection is completed, users generate the route and navigate through various steps, organized by floor.

4.2 User interface prototyping

Once the conceptual design phase is completed, user interface prototyping is developed.

Prototype definition constitutes the logical link between frontend requirements defined in the previous chapter and the actual interface design.

This phase's purpose is not limited to aesthetic definition but translates indoor navigation complexities into intuitive and navigable screens. The design work follows an iterative approach, starting from mock-up prototypes and proceeding through interface evaluation and revision phases.

4.2.1 Mock-up prototype

Mock-up prototypes constitute high-fidelity models that exclusively show the application's visual appearance, without interactive functionalities. Their purpose is representing aesthetic appearance and graphical element disposition.

The designed mock-up screens are presented below.

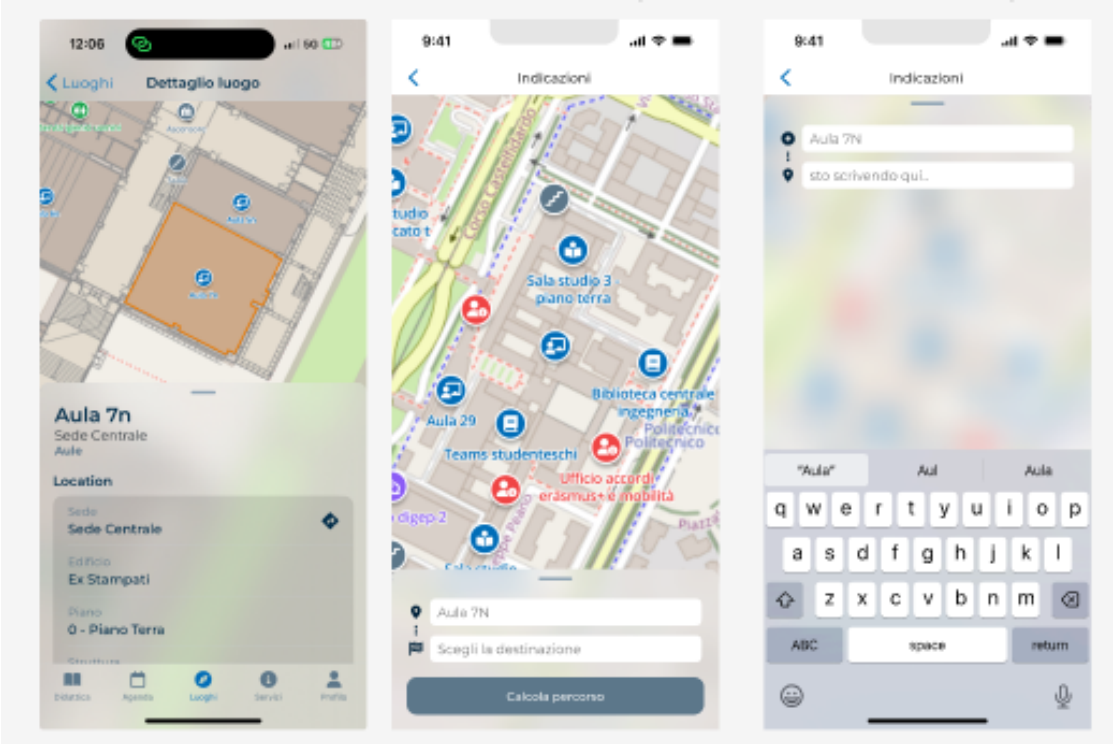

Figure 4.8: Mock-up prototype: Indications Screen

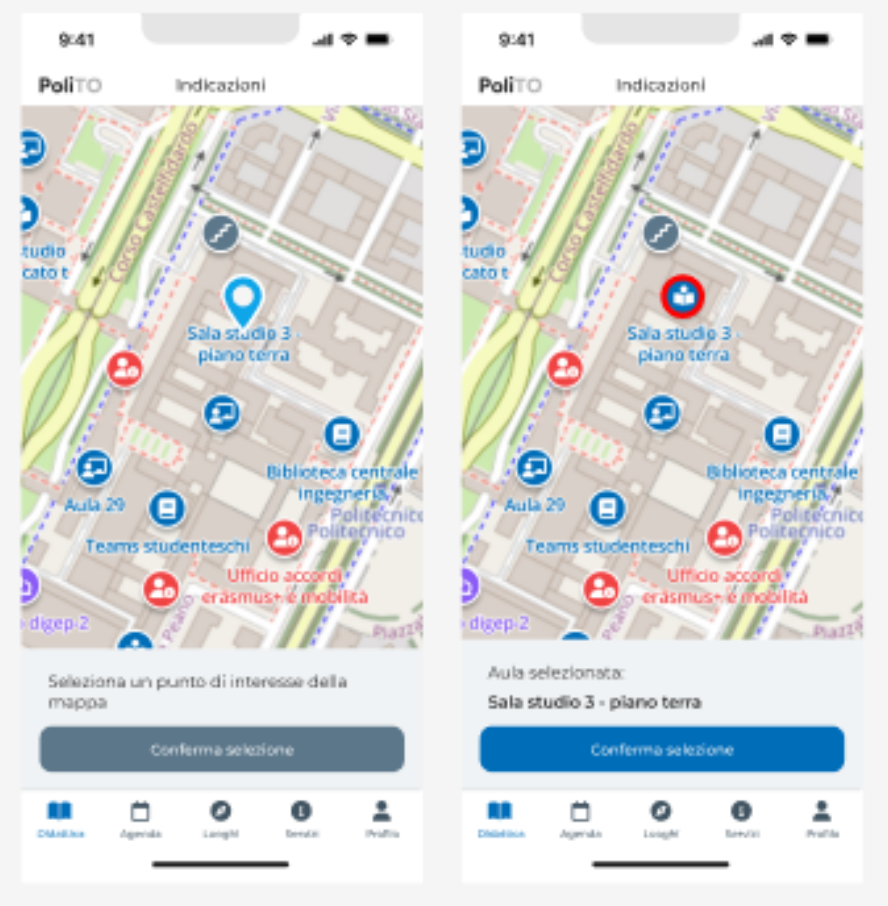

Figure 4.9: Mock-up prototype: Selection of the POI from Map Screen

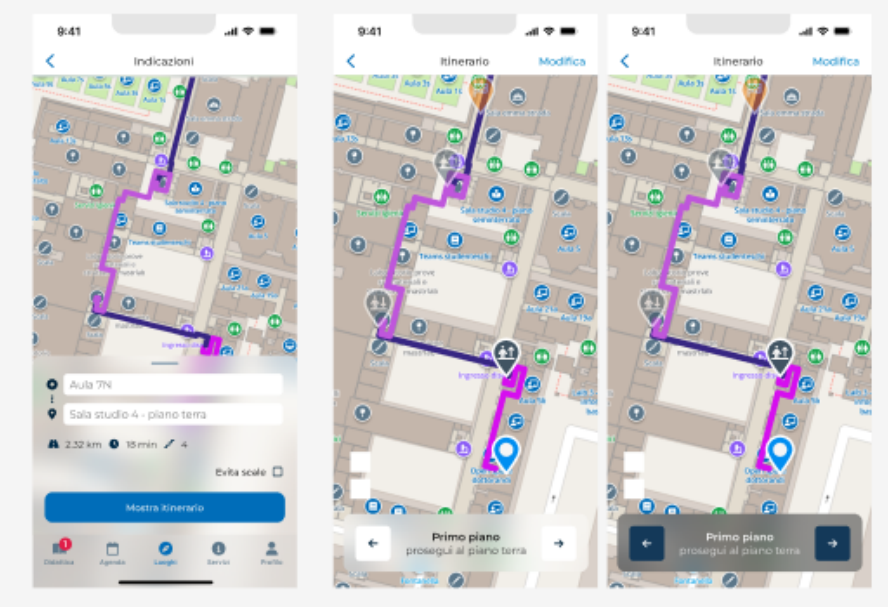

Figure 4.10: Mock-up prototype: Itinerary Screen

4.2.2 Interface evaluation and revision

Once mock-up prototype and use cases realization was completed, a prototype evaluation and revision phase was conducted.

Revision focused primarily on the functional purpose of elements composing the screens.

For example, in the sketches shown in the use cases section, the disposition of textual inputs for entering departure and destination, together with route statistical information, occupied an excessive portion of available space.

Additionally, the evaluation process highlighted the absence of a modality for selecting points of interest directly from the map (as specified in the previous chapter's requirements). This functionality was therefore added to the mock-up prototype following evaluation.

4.3 FrontEnd design

With visual appearance and logical flows consolidated through the prototyping phase, this section describes the mobile client's software architecture design.

FrontEnd design was guided by the need to manage map rendering complexity and related information, while ensuring performance fluidity.

Implementation adopts the Component-Based architectural pattern (also known as Service Layer pattern)[8], aimed at separating user interface from business

logic. This architectural choice is motivated by code scalability and maintainability requirements.

Mobile client design is articulated into the following components:

- **Data Layer**: module delegated to Backend communication. Performs API calls to obtain location information and generate the optimal route, also managing cache to improve performance.
- **Places Feature Module**: module dedicated to managing the Places feature, including the different related screens (visualization of one or multiple locations).

Implementation integrates a dedicated service layer for client-server communication, interfacing with Backend APIs to receive the optimal route.

Parallel to logical integration, user interface intervention provides for Places Feature module expansion, translating Mock-Up prototype additions into code. Development focuses on implementing three screens:

- **Indications Screen**: screen constituting the control panel for itinerary generation. Offers a scrollable component containing two textual input fields for setting departure and destination; interaction with such fields activates dynamic expansion of a list for location selection. In compliance with interaction requirements, the screen integrates a checkbox for activating accessibility filters.
- **Selection from Map Screen**: screen designed as an alternative to textual input. Invoked when users choose the list item that enables selection by directly interacting with the map. The view permits free visualization of the floor plan, with floor change option, and allows POI selection by clicking directly on its respective marker.
- **Itinerary Screen**: screen accessible once the optimal route response is received from the BackEnd. Visualization logic decomposes the itinerary into segments, organized by floor. Interaction is guided by a component located in the lower part, equipped with directional arrows for sequential segment navigation (Previous/Next). A relevant aspect is automatic synchronization between interface and map: transition to a new segment automatically loads the floor plan corresponding to the selected global itinerary portion.

4.4 BackEnd design

Parallel to client development, system design requires a solid server-side infrastructure. This section describes Backend design, namely the logical and physical architecture that enables spatial data processing.

While the Frontend manages interaction, the backend constitutes the system's decision-making engine. Design challenges addressed in this domain concern algorithmic efficiency, data integrity, and response performance.

The following analysis focuses on three particular points:

- Preliminary Analysis of the Database
- Data Modelling and Navigation Graph Modelling
- Navigation Algorithm and Logic

4.4.1 Preliminary analysis of the database

The preliminary analysis constitutes a technical assessment of the database, necessary to ascertain the compatibility of the existing schema with computational routing constraints. This phase is not limited to a simple data survey; rather, it is aimed at isolating and examining the tables dedicated to graph management, verifying that their structure allows for the efficient application of pathfinding algorithms.

Furthermore, this examination enables the identification of potential information gaps or structural inconsistencies, defining the integrations required to achieve a coherent topological representation. The analysis thus serves as an operational prerequisite for optimizing subsequent import procedures and ensuring that the system provides high-performance responses to spatial queries.

The design phase begins with data structures to be implemented on the database, considering two distinct operational phases related to navigation data (graphs):

- **Data Import**: procedure through which geometric data is extracted from digital floor plans (CAD format). Specifically, the university is structured into different sites, where each is divided into buildings, each in turn subdivided into floors. Each digital floor plan therefore identifies a building's floor.
- **Runtime Processing**: phase in which database querying occurs to generate dynamic outputs in response to client requests.

The objective is optimizing the database so that the DBMS's spatial extension can execute routing calculations with maximum efficiency.

4.4.2 Data modeling and navigation graph modeling

The data model is designed implementing specific relational entities, beginning with geometric data import from CAD sources, which populate the tables constituting the navigation graph.

In information extracted from CAD, labels are found in the form of metadata, assignable to terminal segments of various graphs. Exploiting this characteristic, "virtual" segments are created between terminal segments to connect different buildings and floors. Identifying these connections requires processing metadata associated with terminal segments and matching between corresponding labels. Given the high computational overhead of this operation and high access frequency during routing queries, link segments are stored in a materialized view named Links, allowing pre-calculation of inter-floor and inter-building connections.

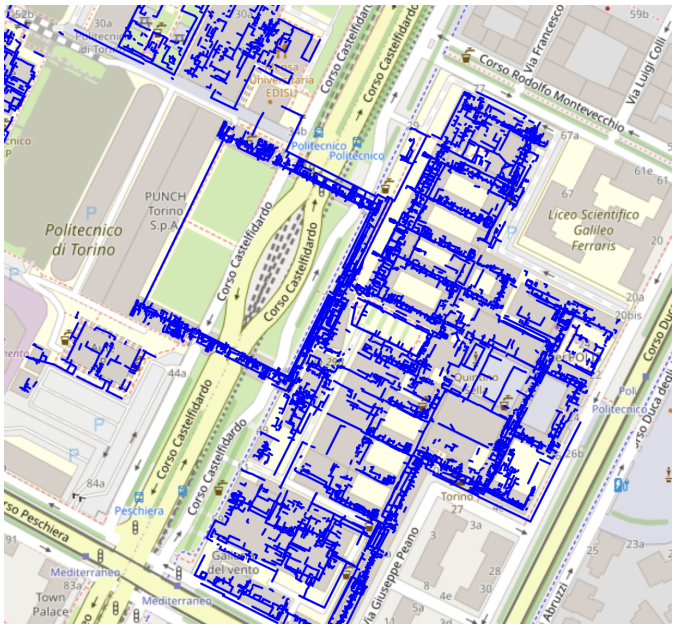

Figure 4.11: Snapshot of inter-floor and inter-building connections

Regarding the processing phase, the creation of two materialized views for vertexes and edges representation is chosen.

This architectural choice is justified by performance requirements: unlike traditional views, materialized views pre-calculate and physically store query results, guaranteeing lower data access times during calculation operations.

4.4.3 Navigation algorithms and logic

Based on the results of the preliminary analysis, the system logic was designed with a focus on transforming raw data into a functional topological model. Initial development focused on generating inter-floor (stairs and elevators) and inter-building connection segments, which were implemented through materialized views to optimize data access times and accelerate computational operations.

The core of the architecture lies in the graph-based modeling of the entire infrastructure. The system extracts the relevant set of edges by filtering them based on the geometric coincidence between the network nodes and the coordinates of the starting and destination points. Path identification is performed using the A* algorithm, which was preferred over alternatives such as Dijkstra due to its superior computational efficiency in geospatial contexts. By integrating a spatial heuristic based on coordinates and building levels, A* enables the identification of the optimal path while minimizing both Euclidean distance and overall computational cost.

Once the path is generated, a post-processing phase is applied to make the data fully usable by the user interface. At this stage, the "raw" path is enriched with informative metadata essential for the user experience, including:

- **Vertical connection quantification**: identifying the specific number of stairs and elevators to be utilized;
- **Steps detection**: flagging the presence of ramps or steps, which is vital information for ensuring accessibility for users with reduced mobility.
- **Private Access Indicator**: flags whether a route segment is publicly accessible or subject to restricted (private) access.

Finally, the system performs a decomposition of the path into ordered macro-segments. Each segment is uniquely identified based on its respective floor level and its classification as a public or private section. This organization allows for a coherent and chronological visualization of the journey across the various levels of the Politecnico structures.

4.5 Integration between Front-End and Back-End

With frontend and backend designs completed, a communication protocol for integration is defined. Integration is managed through an API (Application Programming Interface) designed to provide route computation services directly to the client.

Communication architecture is based on message exchange: the client sends a request to the Backend containing necessary data, while the server processes data and returns a response containing requested metadata.

To ensure correct data interpretation, a Data Contract for the Backend is defined. The response structure is in JSON format and includes:

- **Geospatial Segmentation**: route coordinates logically subdivided by floor membership, enabling the client to render exclusively the segment pertinent to the displayed floor.
- **Navigation Metrics**: quantitative metadata such as total route distance.
- **Inter-floor Connection Counters**: number of vertical connection elements (stairs, elevators).
- **Accessibility Metrics**: granular information (number of steps) necessary for users with active accessibility filters.

This standardization ensures that server-side processed information is correctly mapped into screens defined in Frontend design, guaranteeing coherence between calculation and visualization.

Chapter 5

Implementation

This chapter outlines the technologies and frameworks employed for solution realization, differentiating between Frontend and Backend components. Implementation specifics and rationale behind architectural selections are provided for each technology.

5.1 FrontEnd technologies

The Students App Frontend development leverages the following technologies:

- **React Native**: An open-source framework developed by Meta Platforms that enables cross-platform application development for Android and iOS using JavaScript or TypeScript combined with React and native platform APIs. TypeScript was adopted for this project, consistent with the existing Students App implementation. TypeScript represents a JavaScript superset introducing explicit types for variables, parameters, and return values, while supporting object-oriented programming features. This choice ensures enhanced code robustness and maintainability, fundamental characteristics in large-scale projects.
- **Figma**: A collaborative web-based platform for interface design, utilized for prototype, wireframe, and mockup creation.
- **Mapbox**: A cartographic service provider offering APIs for integrating customizable maps into web and mobile applications.
- **PostHog**: An all-in-one open-source platform for behavioral analysis of applications and websites, integrating analytics tools, session replay, and feature flags.

- **Microsoft Clarity**: A behavioral analysis tool enabling identification and analysis of usability issues through heatmaps and session recordings.

5.1.1 UI components

The interface component realization process unfolds through two distinct phases:

- **Prototyping Phase**: Mockup definition of new screens using Figma software. To guarantee visual coherence with the existing interface, reference was made to the Politecnico di Torino's official Design System available on Figma.
- **Development Phase**: Developed component implementation from the preceding phase using React Native[9] and TypeScript[10], integrating them directly into the existing Students App project.

The React Native libraries employed for new screen implementation are listed in the following table:

Library	**Primary Functionality**
rnmapbox	Provides JavaScript/TypeScript components for Mapbox map integration
turf/bbox	Supplies functions for calculating bounding boxes of GeoJSON objects
fortawesome	Delivers official FontAwesome components, including icons
posthog-react-native	Integrates PostHog platform functionalities into React Native projects
microsoft/react-native-clarity	Integrates Microsoft Clarity functionalities into React Native projects

Table 5.1: React Native libraries employed

5.2 BackEnd technologies

5.2.1 Frameworks and languages

The backend relies on the existing PostgreSQL DBMS that represents the map visualization infrastrucure, deployed via Docker containers. Two fundamental extensions were integrated into the database:

- **PostGIS**: A spatial extension for PostgreSQL that enables geographic data management, providing a GIS (Geographic Information System) standards-compliant system for geometric and geographic object analysis and processing.

- **ogr2ogr/GDAL**: A command-line interface (CLI) tool, part of the GDAL/OGR library, designed for converting and manipulating geospatial vector data. It facilitates translation between various formats (e.g., GeoJSON) and supports direct interaction with spatial databases such as PostGIS.
- **GRASS**: Open-source software suite for the management, processing, and advanced analysis of geospatial data (Geographic Resources Analysis Support System).
- **PGRouting**: An extension enriching PostGIS/PostgreSQL databases with geospatial routing functionalities, enabling optimal path calculation through native SQL functions.

5.2.2 Database implementation

The routing functionality implementation exploits PostGIS and PGRouting[11] extensions through a structured SQL query calculating the optimal path between two rooms and returning a GeoJSON[12] representation.

The query is articulated through CTEs (Common Table Expressions) subdividing computation into logical phases:

1. Possible Path Generation (‘all_paths‘)

```
1         SELECT *
2         FROM pgr_aStar(
3             'SELECT id, source, target,
4               calcs.cost,
5               calcs.cost reverse_cost, x1, y1, x2, y2
6               FROM polito_graph_edges edges,
7             LATERAL (
8                 SELECT $prefer_public AS prefer_public
9             ) opts,
10            LATERAL (
11                SELECT cost * ((1000* opts.prefer_public * edges.
    private)+1) AS cost
12            ) calcs
13            WHERE 1=1',
14            (SELECT array_agg(vid) FROM polito_graph_vertexes pgv
15                WHERE pgv.building_id = :s_blid AND pgv.floor_id =
     :s_flid AND pgv.room_id = :s_rmid),
16            (SELECT array_agg(vid) FROM polito_graph_vertexes pgv
17            WHERE pgv.building_id = :d_blid AND pgv.floor_id = :
    d_flid AND pgv.room_id = :d_rmid),
18            true
19          ) rt
20          LEFT JOIN polito_graph_edges ge ON rt.edge = ge.id
```

This CTE computes all possible paths between origin and destination vertices using the A* algorithm implemented by PGRouting's 'pgr_aStar' function. The function receives as parameters the graph edge selection query and arrays of origin and destination vertices, identified through building_id, floor_id, and room_id.

A pivotal component is the inner query within the 'pgr_aStar' function. Specifically, the focus is directed towards the 'preferPublic' variable, a parameter that acts as a selector for the utilization of private segments within the topological graph.

By default—as exemplified by the student profile—the 'preferPublic' variable is set to true. Whenever the pathfinding process involves potentially traversing private segments (such as laboratories located within departments where access is not consistently guaranteed), the system implements an edge-weighting logic defined by a `LATERAL` clause named "calcs":

```
LATERAL (
    SELECT cost * ((1000 * opts.prefer_public * edges.private) +
    1) AS cost
) calcs
```

This operation assigns a corrective weight—specifically, a factor of 1000—to the segment cost (represented in this study by physical distance) whenever a segment is flagged as "private" and `prefer_public` is enabled.

In practical terms, the algorithm preferentially excludes these segments, selecting them only if they represent the sole viable option to reach the destination. Finally, to ensure data integrity in the final output, the PostGIS `ST_Length` function is applied to the entire generated path. This ensures that the heuristic weight used during the navigation phase does not compromise the actual precision of the total path length returned to the user.

2. Optimal Path Selection ('shortest_path')

```
          SELECT start_vid, end_vid,
            RANK() OVER (ORDER BY MAX(agg_cost) ASC) AS rank
          FROM all_paths
          WHERE edge = -1
          GROUP BY start_vid, end_vid
```

This CTE identifies the minimum-cost path by applying a ranking function based on aggregated cost (‘agg_cost‘). The path with rank 1 corresponds to the optimal solution.

3. Microsegment Extraction (‘all_segments‘ and ‘path_segments‘)

```
1    all_segments AS ( -- all "micro"-segments of the (shortest)
     path
2         SELECT path_seq, geom, fn_bl_id, fn_fl_id,
3                   no_disab, private, vert, reverse_cost,
4              SUM(CASE WHEN vert = 1 THEN 1 ELSE 0 END) OVER (ORDER
     BY path_seq)
5              + SUM(private_changed) OVER (ORDER BY path_seq) -- sum
      1 when private changes:
6         AS seq_path_id -- segment-pack identifier
7           FROM (
8               SELECT path_seq, geom, fn_bl_id, fn_fl_id,
9                 no_disab, private, vert, reverse_cost,
10                CASE
11                  WHEN LAG(private) OVER (ORDER BY path_seq) <>
     private
12                  THEN 1 ELSE 0
13                END AS private_changed
14            FROM all_paths
15            JOIN shortest_path sp ON all_paths.start_vid = sp.
     start_vid AND all_paths.end_vid = sp.end_vid
16            WHERE rank = 1
17            -- removing inter-floor links as they are not needed
     for visualization
18            AND fn_fl_id NOT IN (SELECT fn_fl_id FROM link_floors)
19          ) t
20          ORDER BY path_seq
21        ),
22    path_segments AS ( -- segments to actually pass to the
     frontend (non-inter-floor), merged by floor
23          SELECT ps.*,
24          als_min.geom start_segment,
25          als_max.geom end_segment
26          FROM (
27            SELECT
28              als.fn_fl_id,
29              als.vert,
30              als.seq_path_id,
31              min(path_seq),
32              max(path_seq),
33              ST_LineMerge(ST_Collect(als.geom)) AS geom,
```

```
34              STRING_AGG(DISTINCT als.fn_bl_id, '-') FILTER (WHERE
    als.fn_bl_id !~ '[A-Z]{2}_[A-Z]{3}\d{2}_[A-Z]{2}_[A-Z]{3}\d{2}
    ') AS bl_id
35            FROM all_segments als
36            GROUP BY als.fn_fl_id, als.vert, als.seq_path_id
37            HAVING STRING_AGG(DISTINCT als.vert::text,'') <> '1'
38            ORDER BY MIN(als.path_seq)
39          ) ps
40          JOIN all_segments als_min ON als_min.path_seq = ps.min
41          JOIN all_segments als_max ON als_max.path_seq = ps.max
42        )
```

The 'all_segments' CTE extracts all microsegments of the optimal path (rank = 1), excluding inter-floor connections that should not be visualized in the frontend.

To ensure a coherent partitioning of the navigation path into macro-segments distinct by both floor level and access status, a unique identifier named 'seq_path_id' was developed. The generation logic for this identifier is essential for managing state transitions within a building's layout.

The calculation leverages PostgreSQL Window Functions[13], advanced tools that enable computations across a set of rows related to the current record. Specifically, the system operates as follows:

- **State Transition Detection**: Utilizing the 'LAG' function, the system compares the 'private' attribute of the current segment with that of its predecessor in the navigation sequence. Any variation, such as a transition from public to private access or vice versa, is captured and stored by the `private_change` attribute.
- **Isolation of Alternating Sections**: The 'seq_path_id' is generated by combining the value of the `vert` parameter with a cumulative sum of the detected changes. This mechanism is essential for managing complex scenarios on the same floor, such as a "private-public-private" sequence. Without this logic, the two non-adjacent private segments would be incorrectly merged into a single block, whereas this approach maintains them as distinct entities.
- **Sequence Integrity**: This methodology ensures that the user interface receives a logically structured and ordered dataset. This facilitates a clearer understanding of the journey and any associated access constraints for the end user.

Then, the 'path_segments' CTE aggregates microsegments by floor using the 'ST_LineMerge' function, producing segments actually transmitted to the frontend. Purely vertical segments (stairs, elevators) are excluded, and start and end points of each aggregated segment are identified.

4. Accessibility Element Counting

The CTEs 'steps', 'links_stairs' and 'links_elevators' calculate respectively the number of steps, stairs and elevators present on the path, fundamental information for accessibility management.

5. Final Result Construction

The query's main body constructs the final result in GeoJSON format using the 'json_build_object' function, producing a FeatureCollection[14] containing:

- Aggregated metrics (total distance, stairs, elevators, steps)
- Array of floor-based segments, each with geometry, start/end points, and segment identifier

The use of 'json_build_object' ensures that the result is usable directly by the frontend without additional transformations.

A schematic overview of the query logic is provided in the following.

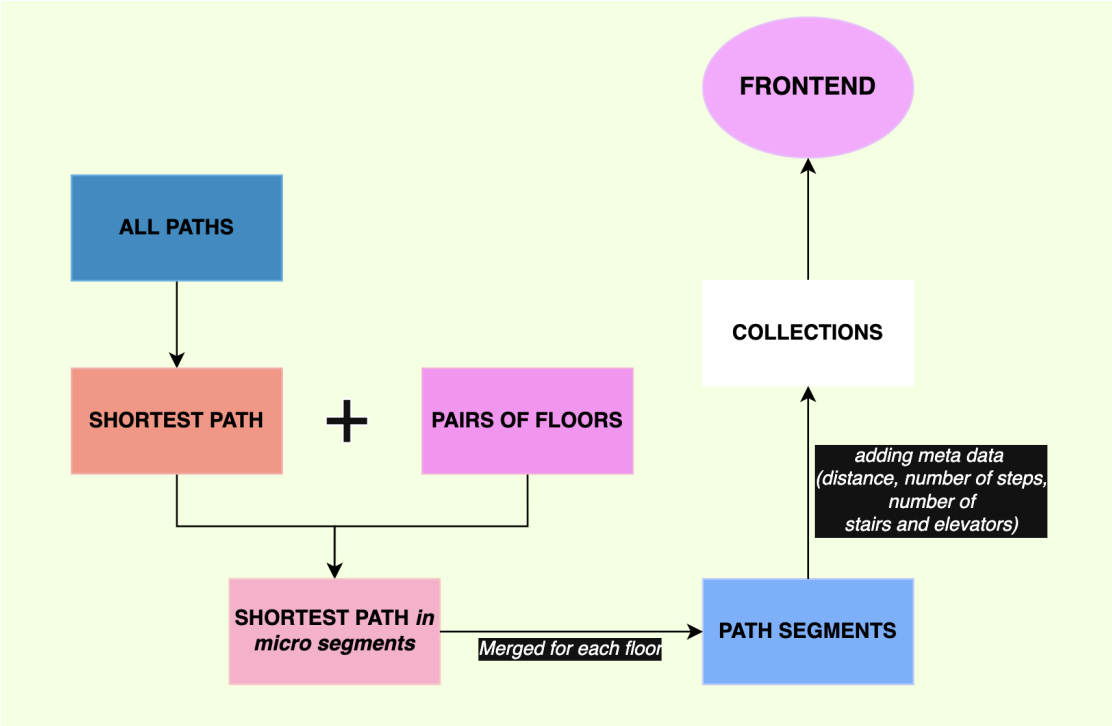

Figure 5.1: Overview of the query logic

5.2.3 APIs and backend communication

FrontEnd-BackEnd communication occurs through REST APIs documented via OpenAPI specification. For this project, the following endpoint was defined:

```
/directions/{startPlaceId}/to/{destinationPlaceId}:
  get:
    tags:
      - Places
    summary: Get path between two places| Ottieni il percorso tra due locali
    operationId: getDirections # this will be the function name available on the TypeScript side of the app
    parameters:
      - $ref: '#/components/parameters/startPlaceId'
      - $ref: '#/components/parameters/destinationPlaceId'
      - name: avoidStairs
        in: query
        description: Avoid stairs in the path
        required: false
        schema:
          type: boolean
          default: false
      - name: usePrivate
        in: query
        description: Use private paths (e.g., through buildings) if user can have access to them
        required: false
        schema:
          type: boolean
          default: false
    responses:
      200:
        description: Success
        content:
          application/json:
            schema:
```

```
30                  type: object
31                  properties:
32                    data:
33                      $ref: '#/components/schemas/
    NavigationResponse'  # inside NavigationResponse,
     it will contain a series of fields (geojson
    route, distance, duration, etc.)
34                  required: [ data ]  # geojsonroute
    will be my feature collection
35          400:
36            $ref: '#/components/responses/
    BadRequestResponse'
37          404:
38            description: Path not found
39            content:
40              application/json:
41                schema:
42                  $ref: '#/components/schemas/Error'
```

The endpoint accepts as path parameters the identifiers of origin (*startPlaceId*) and destination (*destinationPlaceId*) rooms, along with an optional query parameter (*avoidStairs*) for accessibility constraint management.

Responses are structured as follows:

- **200 (Success)**: returns a JSON object conforming to the *NavigationResponse* schema containing the calculated path
- **400 (Bad Request)**: indicates invalid input parameters
- **404 (Not Found)**: indicates the impossibility of finding a valid path between specified points

Response Structure

```
1     NavigationResponse:
2       type: object
3       properties:
4         type:
5           type: string
6           enum: [FeatureCollection]
7         totDistance:
```

```
8             type: number
9             description: Total distance of the route
    in meters.
10          stairsCount:
11            type: number
12            description: Number of stairs crossed in
    the route.
13          elevatorsCount:
14            type: number
15            description: Number of elevators crossed
    in the route.
16          stepsCount:
17            type: number
18            description: Number of steps crossed in
    the route.
19          features:
20            type: array
21            items:
22              $ref: '#/components/schemas/
    NavigationResponseFeature'
```

The ‘NavigationResponse‘ schema represents a GeoJSON FeatureCollection containing:

- Aggregated path metrics (total distance, stair/elevator/step counts)
- Array of features representing path segments subdivided by floor

Each segment is defined by the ‘NavigationResponseFeature‘ schema:

```
1     NavigationResponseFeature:
2       type: object
3       properties:
4         type:
5           type: string
6           enum: [FeatureCollection]
7         startPoint:
8           $ref: '#/components/schemas/GeoJSONPoint'
9         endPoint:
10          $ref: '#/components/schemas/GeoJSONPoint'
11        segmentId:
```

```
12            type: integer
13            description: Identifier of the segment in
    the path.
14          private:
15            type: integer
16            description: Identifier of private segment
     in the path.
17          features:
18            $ref: '#/components/schemas/
    GeoJSONFeatureGeometry'
```

Each segment contains:

- Unique identifier (‘segmentId‘)
- Start and end points (‘startPoint‘, ‘endPoint‘) utilized by the FrontEnd for rendering with specific graphical styles
- Array of GeoJSON features (‘features‘) containing aggregated microsegment geometries

Microsegment geometries are defined according to the ‘GeoJSONFeatureGeometry‘ schema:

```
1     GeoJSONFeatureGeometry:
2       type: object
3       properties:
4         type:
5           type: string
6           enum: [Feature]
7         geometry:
8           type: object
9           properties:
10            type:
11              type: string
12              enum: [LineString]
13            coordinates:
14              type: array
15              items:
16                type: array
17                minItems: 2
18                maxItems: 3
```

```
19                  items:
20                    type: number
21            required: [type, coordinates]
22          properties:
23            type: object
24            properties:
25              bl_id:
26                type: string
27              fn_fl_id:
28                type: string
```

The data structure adopts the GeoJSON format (RFC 7946), an open standard for representing spatial geometries in JSON. This choice guarantees direct compatibility with the Mapbox library of React Native employed in the FrontEnd for cartographic rendering.

FrontEnd-BackEnd Integration

For FrontEnd-BackEnd communication management, the TanStack Query library [15] (previously React Query) was integrated. The adoption of this solution is motivated by the necessity to separate client state management from server state.

Unlike traditional solutions treating remote data as static after initial loading, TanStack Query implements an asynchronous management pattern based on:

- Intelligent caching strategies with configurable time-to-live
- Automatic invalidation mechanisms and background revalidation
- Concurrent request deduplication
- Automatic retry management in case of failure
- Optimistic UI updating

This approach enables maintaining the user interface synchronized with the backend without impacting UI thread performance, guaranteeing a fluid user experience even under variable network latency conditions.

Chapter 6

Evaluation and Benchmarks

Usability evaluation represents a fundamental phase in the interface development process, as it allows verification of the effectiveness, efficiency, and satisfaction with which users achieve their objectives. In the context of the Students App application, a usability test was conducted with the objective of validating the system's user experience, identifying potential issues, and collecting suggestions for future improvements. The methodology adopted for planning and conducting usability tests is based on the heuristics written by Jakob Nielsen[16].

The test involved a representative group of real users (students from Politecnico di Torino) and was designed to simulate realistic situations and frequent activities. The data collected during the sessions provided both quantitative and qualitative information, useful for refining the UI side of the feature and the server-side logic.

6.1 Test Objectives and Methodology

The objective of the usability test was to validate the functionalities implemented in the Students App application and ensure that they were intuitive, accessible, and consistent with end users' expectations. As in any usability test, the purpose was not to evaluate the participants, but rather the interface: every difficulty or error was considered an indicator for system improvement.

The test sessions were conducted in-person, in the form of individual interviews (one-to-one mode), with an average duration of 15-20 minutes.

Each test session involved the participation of three team members: one person (the thesis author) with the role of facilitator, i.e., the person who dialogues with the participant and guides them through the required tasks, the remaining two (specifically two ISIAD members, Giuseppe Piazza and Gaia Lecis) collected data and notes regarding the actions performed by the tested users, difficulties, and any timings.

6.2 Participants Description

The test sample consisted of eight students selected based on availability and diversity in terms of degree program and language. The selection included users with different levels of familiarity with the Students App application, including students who prefer the teaching portal, in order to obtain a more representative evaluation of usability for users with different backgrounds.

Participant	**Faculty**
Participant 1	Ingegneria Informatica (Computer Engineering) L-8(DM270)
Participant 2	Ingegneria Biomedica LM-21(DM270)
Participant 3	Ingegneria Gestionale L-9(DM270)
Participant 4	ICT for Smart Societies (ICT per la società del futuro) LM-27(DM270)
Participant 5	Ingegneria Biomedica LM-21(DM270)
Participant 6	Mechatronic Engineering (Ingegneria Meccatronica) LM-25(DM270)
Participant 7	Ingegneria Aerospaziale L-9(DM270)
Participant 8	Ingegneria Informatica L-8(DM270)

Table 6.1: Table of the participants for the usability test

6.3 Verified Use Cases

During the test sessions, participants were presented with two usage scenarios articulated in specific tasks. The scenarios simulate representative activities of the student's daily experience, with the objective of verifying the complete functional coverage of the feature and the intuitiveness of the interactive flows.

Use Case 1 : "*Find the path to reach Room 13B starting from Room 9S*"

ID	Objective	Epected Success Criterion
Task 1	Enter the Places screen	Access the Places screen
Task 2	Search for Room 13B	User searches for the room using the searchbar
Task 3	Get Directions	User must click the get directions button
Task 4	Directions screen	User must be on the Directions screen
Task 5	Select Room 9S	Type the room name and select the item in the opened dropdown
Task 6	View route preview	User must click the "*Compute path*" button
Task 7	Itinerary screen	User must click the "*Show Itinerary*" button
Task 8	Follow the itinerary	User must be on the itinerary screen and navigate through it

Table 6.2: Table of tasks for Use Scenario 1

Use Case 2 : "*Your friend is in the room next to Bar Denise. Starting from your current position, find the path to reach them.*"

ID	Objective	Expected Success Criterion
Task 1	User must select either Bar Denise or a nearby room	Search via searchbar or select by clicking on the map
Task 2	Get Directions	User must click the get directions button
Task 3	Directions screen	User must be on the Directions screen
Task 4	Search for your position or the room closest to you	User must select the "Select from map" item
Task 5	View route preview	User must click the "*Compute path*" button
Task 6	Itinerary screen	User must click the "*Show Itinerary*" button
Task 7	Follow the itinerary	User must be on the itinerary screen and navigate through it

Table 6.3: Table of tasks for Use Scenario 2

For the first usage scenario, the following aspects were observed and analyzed for each task:

- Average Duration
- Success rate of individual tasks
- Misclicks
- Orientation rate

For the second usage scenario, given the high degree of freedom in choosing the starting point, and considering the advantage of repeating some steps that were repeated in the previous scenario, the following aspects are analyzed for each participant:

- Orientation rate
- Wayfinding success rate

The final results for both use cases were obtained from PostHog since, in the first approach, Clarity was unable to perfectly sample some user actions, while with a second attempt using PostHog, a more accurate measurement of the parameters was obtained.

6.4 Execution

The testing phase was conducted over a 14-day period at the Politecnico di Torino premises. The sessions primarily took place within common areas and student lounges; these environments were selected as they represent real-world use cases, characterized by a high density of Points of Interest (POIs) and a complex yet familiar topology for the student population. Such conditions provided an ideal setting to verify the effectiveness of the navigation features in a practical context.

Each individual session lasted between 15 and 20 minutes, during which participants interacted with the new feature implemented in the *Students App* following a three-phase structured protocol:

- **Operational Task Execution**: users were required to complete specific tasks corresponding to the proposed use cases, thereby testing the system's response in real-world navigation scenarios.
- **Quality of Service Assessment**: upon completion of the tasks, participants were asked a series of targeted questions designed to measure the relevance of the results obtained, the clarity of the instructions, and the overall fluidity of the user experience.
- **Feedback Collection**: at the end of the session, open comments and suggestions were gathered to identify potential improvements or any new requirements that emerged during actual use.

This structured approach ensured that the qualitative and quantitative data collected were both consistent and representative of the actual user needs within the university ecosystem.

6.5 Results and Observations

The analysis of data collected during the usability testing sessions allowed quantification of the feature's usability level, highlighting strengths and issues.

For the first usage scenario, each task was evaluated by calculating the success percentage over the total number of participants. The following are the results for each task of the first scenario:

- **Task 1 - Places screen access**: 8 participants (100%). All participants completed the task without difficulty.

- **Task 2 - Search for Room 13B**: 8 participants (100%), average time 23s. Task completion presented no issues, confirming the ease of the task itself. Execution time depends on the speed of searching for the target location.

- **Task 3 Get Directions** : 6 participants (75%), average time 7s. 25participants did not complete the task correctly, highlighting poor affordance of the directions icon for initiating the navigation process.

- **Task 4 - Directions screen transition**: 6 participants (75%), average time 290ms. Completed by all participants who passed Task 3, with reduced transition times.

- **Task 5 - Room 9S selection**: 6 participants (75%), average time 9s. Completed by all participants who passed Task 3. Execution time corresponds to the duration of searching for the starting room via searchbar.

- **Task 6 View route preview**: 6 participants (75%), average time 6s. Completion time reflects some participants' expectation for automatic route calculation after filling in origin and destination.

- **Task 7 Itinerary screen**: 6 participants (75%), average time 12s. The higher completion time is attributable to the fact that some participants used the route preview as a navigation tool before accessing the detailed guided itinerary.

- **Task 8 - Itinerary navigation**: 6 participants (75%), average time 48ms. Completed by participants who passed Task 7. The reduced execution time reflects the simplicity of the task.

Overall, the first usage scenario achieved a total conversion rate of 75% with an average conversion time of 57s.

The following shows an image of the schema provided by the PostHog library. This schema is called a Funnel and represents a strategic model, often implemented in the marketing world, that realizes a strategic path aimed at improving User Experience.

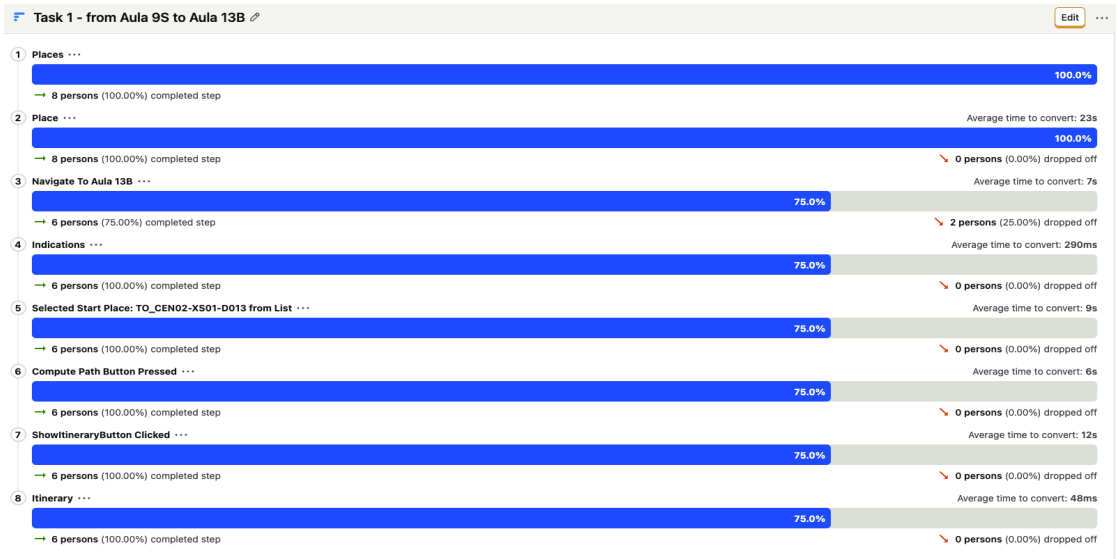

Figure 6.1: PostHog Funnel generated

For the second scenario, the results of tasks 1 and 4 are presented, characterized by multiple completion options:

- **Task 1 - Destination selection (Bar Denise or adjacent room)**: 87.5% of participants used the searchbar to search for the starting location.
- **Task 4 - Starting point selection**: 87.5% of participants correctly used the direct selection functionality from the map. times.

Improvement Proposals

***Get Directions* button optimization**. 50% of participants highlighted difficulties in transitioning to the Directions screen due to poor visibility of the "*Get Directions*" button. Two solutions are proposed: repositioning the button near the textual indication of the position, or associating an explicit textual label with the icon.

Preferential indoor routing implementation. The introduction of a selectable option (analogous to the "*Avoid stairs*" functionality) is proposed, which would allow users to specify preference for exclusively indoor paths, avoiding transit through courtyard areas. This functionality has limitations in cases where the route requires connection between separate buildings of the same campus.

Route completion visual feedback. During navigation in the Itinerary screen, participants did not consistently detect the disabling of the "*Next*" button

upon reaching the last stop. Implementation of a modal notification is proposed to confirm route completion and arrival at destination.

Origin-destination selection flexibility. The current flow requires selection of the "Get Directions" button from the destination location, automatically setting the latter as the target. Removing this constraint is proposed, allowing free completion of origin and destination without a predetermined order.

Routing graph topology correction. During three test sessions, the system erroneously returned the absence of valid paths. In collaboration with Cristina Ferrian, corrections were made to the CAD files representing the topological graphs of the structures, resolving inconsistencies in the navigation graph.

6.6 Benchmarks

In order to ensure the robustness of the implemented feature, an additional test aimed at evaluating the API's performance was carried out. Specifically, the analysis was based on a throughput benchmark of the endpoint, interpreted as the system's ability to process a determined number of requests per unit of time.

In the context of this thesis, throughput is defined as the HTTP request completion rate during the benchmark observation interval. This value is defined by *Little's Law*[17], expressed by the formula:

$$\textbf{Throughput} = \frac{WIP}{Cycle\ Time} \tag{6.1}$$

Where

- **WIP (Work in Progress)**: represents the number of requests simultaneously active during the test.
- **Cycle Time**: represents the time required to complete a single request.

The benchmark execution was developed in a Python script that leverages the concurrency primitives of the **asyncio** library combined with the **aiohttp** client/server framework. The library adopts a Cooperative Multitasking model based on a single thread and an Event Loop. Compared to traditional blocking architectures, aiohttp's asynchronous solution enables the management of thousands of concurrent connections while maintaining a reduced memory footprint. This characteristic is fundamental for maximizing throughput in I/O bound scenarios,

proving superior compared to synchronous solutions like Django, with equivalent hardware resources.

The benchmark results are reported below, distinguishing between the two operational scenarios.

- **Caching disabled**: each request involves complete processing.

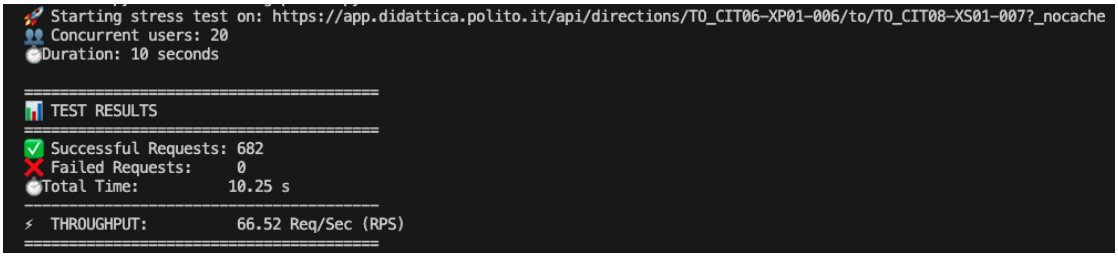

Figure 6.2: Benchmark results with cache disabled

- **Caching enabled**: the system serves the response by retrieving data from temporary memory, reducing the Cycle time and consequently increasing the Throughput.

Figure 6.3: Benchmark results with cache enabled

Chapter 7

Conclusion and future developments

This work addressed the development of an indoor navigation feature for the Students App of Politecnico di Torino, filling a gap in the university's digital ecosystem and providing a concrete solution to orientation issues identified through the Challenge analysis.

The development process was structured in four main phases: requirements analysis, design, implementation, and experimental validation. In the analysis phase, functional and interface requirements were identified through examination of issues reported in the Challenge and analysis of target user needs. This phase produced detailed specifications for both the application and backend levels. The design phase followed an iterative approach based on use case definition and prototyping. Mockups were developed for the user interface and the server architecture was designed. The design also defined the client-server communication protocol through RESTful API interfaces. The implementation realized the functionality by integrating it into the existing Students App ecosystem. On the client side, the user interface was developed in React Native for iOS, while on the server side, the routing and path calculation system was implemented. Frontend-backend integration was achieved through APIs that handle routing requests, retrieval of structure information, and user preference management. The validation phase involved testing with real users from Politecnico di Torino. The results showed an overall positive response, with satisfactory task completion rates and usability metrics. However, the analysis revealed areas for improvement: system feedback during route generation requires optimization in user communication, and the routing algorithm needs extensions to handle advanced preferences, such as minimizing outdoor paths in adverse weather conditions.

7.1 Future developments

The presented work constitutes a first implementation stage (Proof of Concept) for the integration of indoor navigation in the Students App. Although the prototype validates the proposed approach, the architecture offers wide margins for extension, both in terms of system scalability and logical refinement. Future developments aim to bridge the functional gap with established outdoor navigation platforms, increasing the application's value and encouraging its adoption by users. Specifically, the identified development directions concern: advanced parametric modeling of route preferences, extension of compatibility to the Android ecosystem (cross-platform porting), integration of real-time positioning systems (IPS), and progressive alignment of User Experience to current market standards. The identified future development directions include:

- **Advanced modeling of navigation preferences**. From a user experience improvement perspective, a significant evolution will consist in the dynamic remodulation of traversal costs for multi-floor connections. This intervention aims to implement a more "human" path calculation, favoring consistency of the chosen means and avoiding fragmentation between stairs and elevators. Furthermore, the system will be able to integrate adaptive convenience thresholds, suggesting the use of stairs for movements of few floors and reserving elevators for longer routes, thus simulating a decision-making process optimized for daily use.
- **PoliTo-Mappe compatibility**. The extension of the feature to the maps offered on the official website of the Politecnico di Torino allows a notable extension to more devices. It permits users to orient themselves freeing themselves from the exclusive use of the app to orient themselves, but accessing also from any browser. Furthermore, this extension guarantees the navigation experience inside the University in both its reference points for orientation inside the spaces.
- **Real-Time Indoor Positioning implementation**. To overcome the limitations of the current static route visualization, the integration of real-time positioning functionality is planned. This update will allow offering contextual (context-aware) indications during the journey. The implementation will require expanding the Students App's capabilities through the adoption of specific indoor localization protocols and the revision of routing algorithms, introducing dynamic management of deviations from the optimal path and live updating of the user interface.
- **Path management in the presence of temporary obstacles**. The university spaces are often subject to maintenance work that implies the

potential presence of construction sites or temporary interruptions of certain paths. The introduction of a new segment filtering logic would allow for the identification of path sections that are impassable due to obstacles, enabling the generation of alternative routes. Furthermore, this scenario would entail the addition of specific visual elements to clearly indicate to the user that the generated path includes deviations from the optimal one, similar to widely used road navigation solutions (such as Waze), which highlight sections affected by roadwork using a different color palette.

- **Alignment with mainstream navigation paradigms**. The implementation of advanced weight modulation, previously discussed, enables the generation of alternative routes (multi-routing). This evolution would allow the system to propose different path options based on specific constraints, such as minimizing outdoor crossings (a feature explicitly requested during usability tests to ensure fully indoor paths). This approach functionally aligns the Students App with market standards (e.g., Google Maps). The adoption of familiar interaction patterns and interfaces allows leveraging users' pre-existing mental models, drastically reducing cognitive load and lowering the learning curve necessary for using the system.

The implementation of these improvements would allow the system to reach functionality levels comparable to commercial outdoor navigation solutions, increasing Students App adoption and improving the overall experience of students on the university campus.

Appendix A

Usability Test Script

UX TEST: Navigazione e Wayfinding

Matricola ID: _____ **Data:** _____

Task 1

Obiettivo: Trova il percorso per raggiungere l'aula 13B partendo dall'Aula 9S.

In che piano si trova l'aula 13B?

☐ Piano Terra

☐ Primo Piano

☐ Secondo Piano

☐ Altro: _____

In che piano si trova l'aula 9S?

☐ Piano Terra

☐ Primo Piano

☐ Secondo Piano

☐ Altro: _____

Indica la sequenza dei piani che devi percorrere per raggiungere l'aula 13B:

Task 2

Obiettivo: Il tuo amico si trova nell'aula accanto al Bar Denise. Partendo dalla tua posizione attuale, trova il percorso per raggiungerlo.

UX TEST: Navigazione e Wayfinding

StudentID: _____ **Date:** _____

Task 1

Goal: Find the route to reach Classroom 13B starting from Classroom 9S.

On which floor is Classroom 13B located?

☐ Ground Floor

☐ First Floor

☐ Second Floor

☐ Other: _____

On which floor is Classroom 9S located?

☐ Ground Floor

☐ First Floor

☐ Second Floor

☐ Other: _____

Indicate the sequence of floors you need to go through to reach classroom 13B:

Task 2

Goal: Your friend is in the classroom next to Bar Denise. Starting from your current location, find the route to reach them.

Bibliography

[1] CLIK - Politecnico di Torino. *Mapping Polito Spaces*. 2025. URL: https://clik.polito.it/en/challenge/mapping-polito-spaces/ (cit. on p. 2).

[2] TapMyLife Srl. *TapMyLife-Indoor navigation, indoor tracking, RTLS in strutture sanitarie*. URL: https://www.tapmylife.com/ (visited on 01/21/2026) (cit. on p. 5).

[3] Politecnico di Milano. *Polimaps*. URL: https://maps.polimi.it/ (visited on 01/21/2026) (cit. on p. 6).

[4] Politecnico di Torino. *Mappe - Torino*. URL: https://www.polito.it/mappe (visited on 01/22/2026) (cit. on p. 7).

[5] OpenStreetMap Wiki. *Raster tiles*. URL: https://wiki.openstreetmap.org/wiki/Raster_tiles (visited on 01/15/2026) (cit. on p. 7).

[6] Elchin Fahradov. *Indoor Navigation based on the plain browser*. M. Eng. thesis. Turin, Italy, July 2022. URL: https://webthesis.biblio.polito.it/23562/ (cit. on p. 8).

[7] Fernand Meyer Kals Mekki Eddy Bajic. *Indoor Positioning System for IoT Device based on BLE Technology and MQTT Protocol*. Apr. 2019. URL: https://www.researchgate.net/publication/332523730_Indoor_Positioning_System_for_IoT_Device_based_on_BLE_Technology_and_MQTT_Protocol (cit. on p. 9).

[8] Maria DiCesare. *What is Component-Based Architecture?* Sept. 2025. URL: https://www.mendix.com/blog/what-is-component-based-architecture/ (cit. on p. 24).

[9] Inc. Meta Platforms. *React Native*. URL: https://reactnative.dev/ (visited on 01/20/2026) (cit. on p. 31).

[10] Microsoft. *TypeScript*. URL: https://www.typescriptlang.org/ (visited on 01/20/2026) (cit. on p. 31).

[11] pgRouting. *pgRouting - Open Source Routing Library*. URL: https://pgrouting.org/ (visited on 12/29/2025) (cit. on p. 32).

[12] GeoJSON. *GeoJSON*. URL: `https://geojson.org/` (visited on 12/29/2025) (cit. on p. 32).

[13] PostgreSQL. *Window functions*. URL: `https://www.postgresql.org/docs/current/functions-window.html` (visited on 02/27/2026) (cit. on p. 35).

[14] Dmitry Sobolevsky. «Feature and FeatureCollection in GeoJSON». In: *Medium* (Mar. 2023) (cit. on p. 36).

[15] TanStack. *TanStack Query*. URL: `https://tanstack.com/query/v5/docs/framework/react/react-native` (visited on 12/29/2025) (cit. on p. 41).

[16] Jakob Nielsen. «10 Usability Heuristics for User Interface Design». In: *NN/-Group* (Apr. 1994). Updated Jan. 2024 (cit. on p. 42).

[17] Kevin Sookocheff. *Using Little's Law to Measure System Performance*. URL: `https://sookocheff.com/post/modeling/littles-law/` (visited on 01/19/2026) (cit. on p. 49).