University POLITEHNICA of Bucharest
Automatic Control and Computers Faculty
Automation and Industrial Informatics Department

New Integrated Services for Orientation and Accessibility in a Smart Campus with Multiple Locations

A dissertation submitted to the Doctoral School of Automatic Control and Computers University Politehnica of Bucharest in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Drd.Ing. Ioan DAMIAN

Bucharest 2023

ABSTRACT

The issue of orientation and accessibility is present especially in a large public space. This thesis approaches this issue from a pedestrian routing perspective, which comes to the aid of newcomers, persons with special needs, and protection against the risks of pandemics. A thorough investigation of the state of the art showed that there are different solutions regarding this problem, with a possible direction towards a smart campus concept by employing maps and guiding tools, with the help of different technologies. However, each approach tries to tackle a different problem, making difficult the process of centralization for the end user. Thus, the research objectives of this thesis are to add personal contributions regarding services that improve the overall experience of the end user, to mitigate standard risk situations with better integration strategies, to employ community- and data-driven information, and to create the configuration for validation. The research method chosen for this thesis considers general scenarios to determine end-user experience, public space representation, service-oriented architecture, multicriterial routing service employed with *policies* resources, and the development of a platform to validate them. The original research contributions refer to developing a model inspired by weighted graphs to support multicriterial pedestrian routing, integration of real-world user needs with *policies* resources, the definition of the general workflow, and detailed architecture for a platform to support routing services. The proposed *policies* are resources that change the underlying graph model based on users' needs, making the system user driven. The thesis proposed algorithms for these *policies* were implemented and validated. They target community votes, avoiding crowded areas, avoiding polluted areas, sheltering from unfavorable weather conditions, and considering accessibility for reduced mobility needs. The correspondent services that offer end users the experience of orientation and navigation were developed under a new platform. The end user calls the pathfinding algorithm exposed by this platform to obtain orientation information. Moreover, the user is provided with generated navigation recommendations in textual and graphical forms, based on the route resulting from the execution of the pathfinding algorithms. The new platform was implemented and validated with test scenarios against conditions based on real-life loading requirements. These requirements were deducted by researching real buildings' layouts and the real numbers of participants at different universities. The results obtained from performance testing prove that the platform is viable and can serve between 1000 and 5000 virtual users (numbers comparable to real-life scenarios) with average request times between 500ms and 4500ms and below 4% percentage of failed requests.

ACKNOWLEDGMENTS

My thanks go to my Scientific Coordinator Professor Anca Daniela Ioniță, for her professionalism, extensive knowledge, patience, and finesse. Her relentless advice impacted enormously my thought and my Ph.D. studies.

I would like to thank my committee that guided me in the last years, for their extensive involvement and constructive criticism, Professor Mariana Mocanu, Professor Silvia Anton, and Professor Adriana Olteanu.

I would also like to thank my colleagues Florin Lăcătușu and Marian Lăcătușu for their unique points of view, and for our conversations and collaboration.

Also, I would like to thank my mother, father, my aunt, Ana, and my sisters, Ecaterina and Elena, for their patience with me and continuous encouragement.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Context and Motivation

University campuses are vast and unknown places for a fresh student, very different from a high school setting [1]. It is overwhelming being in a vast and complex place like a university campus for the first time, especially when you need precise and predictable indications of how to reach a specific class in a specific building. It is highly important to be able to orientate in these circumstances because it provides you with a sense of belonging and certainty to plan your activities. Nowadays we have many smart devices that can provide us with complex and accurate information, and yet we can get lost in a new place or not be able to predict our steps in real life based on information from a smart solution. The concept of the smart campus should provide a holistic approach to the academic experience, where students or other stakeholders could benefit from the aid of different technologies, and navigation and orientation benefits could be provided under this concept as well. However, we have to consider the fact that many universities were built when the "smart" concept was not yet created [2]. Thus, the architecture of the buildings or the connections between them were not created considering modern factors (like inclusivity, scaling with information technology, etc.). For example, the most impacted persons trying to navigate a complex building topology are persons with locomotor disabilities.

The general solution is to offer stakeholders information in the form of maps and guiding tools, to support navigation and decision-making. Yet, because every university campus is a unique place, the requirements to build such a solution must be custom-made for that specific university. The complexity that a university campus implies stands in the diversity of environments (outside vs. inside), the different level (stairs, floors, etc.), and the difficulty of some topologies vs. others. Current research brings solutions to some aspects of the issue, all in the direction of the "smart" campus, such as the landscape of the campus [3], i-campus [4], smart city [5][6], event-based navigation [7], iBeacon navigation [8], a mobile social network in a smart campus [9].

These applications that provide users with real-time directions about their navigation routes are a state of practice. However, pedestrians mostly rely on traditional orientation methods, although getting to the destination in large public spaces such as transportation hubs, commercial centers, and university campuses, has become more and more complicated and it sometimes leads to delays with unpleasant effects. The various solutions to this problem are generally focused on specific requirements, separately treating problems such as pedestrian safety, optimum wayfinding, or routing of emergency personnel.

An integrative solution is proposed in this thesis in the form of a services platform, by using general services accessible to all the users on a university campus. The services presented in this thesis are completely built by the author of this thesis (based on typical data for a smart campus), among those, there are information integration services with external services (they act as consumers of third-party services). All services are based on a light model of the campus (which represents the topology of the campus), a model which is also presented as a contribution in this thesis, with the main purpose of flexibility and versatility (it takes into consideration future changes of the services and the addition of new services).

Typically, the role of humans is to consume services from various cyber-physical systems characteristic of smart buildings or a smart city in general. Yet, recent approaches show that their role can be substantially extended to become contributors to the systems, by providing machine-readable information, making or validating decisions, interacting with the system, or even assuming the role of actuators [10]. This active presence of human agents leads to the development of socio-cyber-physical systems (SCPS). Such an example is given in [10] for planning the evacuation routes in emergencies when relevant information should not only be acquired from sensors, but also from people in the affected area. The human inputs and their contributions to making and following decisions may be integrated into the overall controlled system in a large variety of ways, such as inputs, perturbations, actuators, or feedback. People can thus intervene in multiple ways in the coupling between perception and action [11].

The thesis is placed in the context of providing navigational and accessibility directions to persons inside a university campus. This is important for offering visitors customizable experiences, by considering a range of possible needs and preferences as inputs of the routing services. It mainly targets newcomers, who require navigational information paired with customization criteria. Relevant examples are available for smart campus models that must be person-centric [12], where the benefit to people is put above other technology-driven reasons. Software services allow people to acquire a more accurate perception of the public space, and to stay in contact with other people. This kind of perception has similarities with the case of robots that need to navigate in the same environment as people [13]; the idea is that a person's perception and actions are influenced by other persons in the same environment and the technology embedded in the navigation solution. Another area that impacted our study is the intelligent decision component; a person-centric smart campus application should provide the user with the safety and understanding of their surroundings, especially in the case of navigation and accessibility—an idea that is more broadly approached in [14].

We investigated solutions to help people based on personal specific needs, for example, the impossibility to use stairs or to go through narrow doors, the preference for a less crowded or less polluted path, the necessity to avoid rain, etc. These concerns of accessibility and epidemiological safety are timely and need to be treated in an integrated way. Nonetheless, we want these services to be community-driven, an approach that has been widely adopted for car routing applications. A public space topological map contains a limited number of access points, as entrances. One can consider this as a mathematical finite set because it is difficult to add "real" resources (to infinity) in a public space that is already established; for example, a new room cannot be easily added inside a building, as it would require it to be built physically. Nonetheless, the layout of a public space can withhold a multitude of configurations, i.e., a building can be renamed, a large room can be divided into smaller (or laboratories) rooms, etc. It is also possible to have various preferences or limitations that further complicate the routing, e.g., someone with locomotory issues cannot follow the same path as the others, or, in the case of the COVID-19 pandemic, people need to interact less and still conduct their daily duties [15]. We want to address these diverse criteria in an integrated way, but still, allow flexibility concerning people's choices. This is possible based on a general-purpose graph modelling that is dynamically adapted according to various considerations such as: taking into account the feedback from other pedestrians walking in the same public space; trying to avoid crowded areas at all costs; avoiding any polluted areas based on data originating from sensors; not getting exposed to the rain by asking for real-time data from weather APIs; and not being able to use stairs because of temporary or permanent special locomotory needs [16][17]. They correspond to policies that can be applied to the routing services; when many pedestrians request routing directions in multiple manners (with 0-policy, 1-policy, 2-policy, etc.), the service recalculates the weights inside the graph for each invocation, so each person has a particular output tailored by his or her special needs.

2 Analysis of the State of the Art

In this section, we included an analysis of the state of the art across all the related fields to this thesis. This chapter aims to put in perspective what others had accomplished and how we can improve on top of their research, or originally use this research in very specific parts in this thesis.

2.1 Route Planning

Route planning is made between remote locations that do not have the same address. There is significant scientific work regarding the algorithms for navigation using GPS (Global Positioning System); most of them are based on analyzing and processing graph data for maps, but also other real-time data regarding crowds, traffic jams, accidents, and other events that may influence the selected route. They are multi-objective algorithms, trying to find a route that will not only minimize the distance, but also the time, or a route selected to go through certain points on the map (touristic objectives or other locations). The algorithms in this area are oriented toward four types of travel: wheel vehicles (cars, trucks) with graph-based pathfinding algorithms [18], railway (trains), which are simpler to solve and based on a graph theory approach [19], sea travel (ships) based on a multi-scale visibility graph [20], and foot travel (pedestrians) based on graph wayfinding on cognitive principles [21].

2.2 Mobility and Accessibility in a Public Space

Inclusive navigation for stakeholders. For mobility and orientation inside a public space (particularly a campus), there are several ways to go from one place to another: using bicycles, electric scooters, personal electric vehicles [22], or travelling as a pedestrian. Another specific issue is route generation for people with disabilities, e.g., visually impaired persons [23].

Hierarchical indoor visibility-based graph. Based on a hierarchical indoor visibility-based graph (HiVG) for navigation guidance in multi-story buildings [24], WRLD is an example of a specialized software solution for navigation and orientation (indoors and outdoors) in real-world environments [25].

Indoor positioning. Indoor positioning system can be implemented using a K nearest neighbor algorithm against a floor map layout [26], [27]. Other solution involves Bluetooth receptors and Beacons [28].

IoT technology. Long Range (LoRa) technology [29] can be used to implement an IoT solution based on environment variables. The IoT technology is used in multiple related applications, as summarized in [30].

Connectivity infrastructure. The idea of a smart university campus is approached by Fortes et al. as “Smart Tree” [31] [32].

Mobile device interactions. The interface with the user is on mobile phones most of the time, but in some situations, to reduce the costs, a set of embedded indicators can be used for outdoor or indoor navigation [33], (for freshmen) [34].

2.3 Avoiding Crowds

A more recent topic is related to disease control, particularly the COVID-19 pandemic. It is especially relevant for public spaces, where crowds are very likely to form. This problem introduces a supplementary set of constraints regarding path planning and tracking, whereas it does not fundamentally modify the algorithms and methods used in route planning in public spaces such as campuses. Of course, when trying to control the spread of a disease one must impose a set of restrictions on traffic; in this case, one refers to pedestrian mobility and not to vehicles.

University campuses are places where COVID-19 transmission can have a high impact on stakeholders due to the mobility of the affected persons. There is a dependency between pedestrian dynamics and epidemiology [35], aerosol transmission [36], the mathematical modelling of disease spreading [37], and the risks of students’ exposure to COVID-19 in a university building [38] or other urban settlements [39].

From the point of view of mobility and route planning, disease control influences the way such algorithms are defining the routes, e.g., by indicating different routes for pedestrians inside buildings [40], [41].

Software and hardware application design can offers strategies on how air sensors can be used to indicate accurately the risks of COVID-19 [42]. Bidila et al. state clearly that it is easy to violate procedures (without intention) and that in absence of testing (for people and surface) the risk of contamination is difficult to compute. They propose a low-cost and implementable solution (IoT with a dual monitoring system of air quality) to warn people when their local density becomes a risk factor.

2.4 Trends for the User Interface

In this subsection, we present research made related to the trends for the user interface, based on our research on the following topics, we remark:

- Graphic User Interface is not always superior to Text User Interface [43]

- Interface for navigation – continuous repositioning of users to the next intermediary point [44]
- Social inclusive public space – space adaptation to opposing exclusivity [45]
- Spatial concepts and relationships – user-centric space design [46]
- GPS vs indications from people – often complementary [47]
- Directional map information order – to streamline information assimilation [48]
- Relational Database System for Images [49]
- APIs – practical reuse and productivity [50]

2.5 Lessons Learned

In our research, we followed how others achieved similar goals to this thesis. Thus, the main topics we researched are route planning, mobility in public spaces, avoiding crowds, and user experience. Route planning was researched to get a high-level point of view towards routing in a global scope, not only pedestrian routing. Mobility in public spaces was researched with an accent on pedestrian navigation and how others are using various technologies to achieve navigation in such spaces. Avoiding crowds was researched as an extension and opportunity to make users feel secure during pandemic times; it is an extension because one of the original contributions of this thesis is a mechanism to avoid crowds in a public space. User experience was researched to understand how users can be offered all the details and consume the services we efficiently developed for this thesis.

The result of this analysis is that a system that uses extensively different technologies (mostly smart technologies) is not only hard to implement in the real world (offering thin chances for a real implementation) but also hard to implement and rely on in vast public spaces such as universities campuses over extensive periods. Moreover, the user and the administrator of the public space require complex actions to consume, respectively to produce data.

The parameters we deducted from our research to use for our global thesis scope, is that technology should be consumed by the users more efficiently, under the forms of services, in particular web services, over minimal and easy-to-understand visual interfaces. These services should have the option to customize the experience for each user, and these services should be mobile, thus accessible over mobile data on personal devices.

The decision for this thesis's original contributions was to create a user interface that provides the user with visual images of the layout of the public space that contains general indications for traversal. These images are created based on the images uploaded by an administrator in the platform and on top of it, the platform will create visual directions for the user, customized by her or his needs. Images will be stored on a server and be able to be fetched by the user at any time. Nonetheless, the user will be offered simple indications in the user interface (intuitive text to compensate for the visual part of navigation).

3 Research Objectives

In this section, we discuss the problematics of research, desired results, and specifications of development for our objectives, which together will provide the global objective of the thesis.

3.1 Services for Improved Accessibility in a Public Space

Universities and other public spaces are places with many architectural complexities. Therefore, various complications may be perceived by the participants inside these public spaces. The main difficulty for the end-user is the complexity of the navigation. Navigation in a public space means traversing it to reach a destination. The stress of the time limit is also added to the navigation complexity. The time limit stress is increased if other factors (external or internal) are present. An external factor could be that the public space is hard to navigate because it has a complex architectural layout. An internal factor could be that the user is suffering from locomotory issues. If these complications are not dealt with, the end-user will feel stressed and will have an overall bad user experience.

A user with locomotory issues has other needs if she or he must traverse the public space than a user that does not suffer from those. This particular user must be provided with different navigation directions than a user that does not suffer from those. If we do not make this distinction, the user that suffers from locomotory issues will have a bad user experience or, worse, her or his safety would be compromised. Every user must be treated with equality and inclusion in the mind.

Another problem when offering navigation services to users is on what channel are the services delivered. Because of the nature of the services that need to be provided to the end user, these solutions must be accessed easily and at any time.

The complete *problem* that needs to be solved is: to provide participants in the public space with services that take into consideration the possibility of participants having locomotory issues.

The *solutions* are: navigation services that can be accessed easily and at any time by the participants and offer customized solutions for the participants' needs.

The *specifications* are: to offer these services on mobile phones connected with mobile data.

Thus, the user will act as a client and the services will be provided by a server that is a different entity from the client. We built our solution around mobile phones and mobile data because they are popular, users already know how to operate them, and the services we provide do not need to include extensive training for the user.

3.2 Services for Reducing Pandemic Risks

Public spaces are usually visited by large crowds. But large crowds are susceptible to pandemics, like the one we have been through, the SARS-COV-2 pandemic.

Crowds in a public space can build spontaneously, especially in a small building, and this crowd would provide a bigger probability of transmission. It would also impact free navigation from one point of public space to another.

The complete *problem* is: to extend navigation services to take into consideration the impact of crowds inside the public space and provide alternative routes for the users to reach their destination.

The *solutions* must shelter the users from the potential pathways that contain crowds, thus reducing the risks of pandemics. The pathways provided by the navigation service in these cases try to guide people through portions of the public space that were not visited lately (if they were not visited lately there is a smaller chance of crowds building up there spontaneously).

The *specifications* must extend the functionality of the navigation service. A module that can be opted by the user's choice, thus user-driven, a module that automatically increases the crowd factors inside a public space. The crowd factors automatic update is used to show the portions of the public space that can have the potential to build large crowds, where the risks of a pandemic increase substantially.

3.3 Data- and Community-Driven Pedestrian Routing

Participants in a public space can be looked at as a community. This community is diverse, and each participant has needs that are different from other participants. The public space might not be updated to each of the user's needs, but users can drive the representation of the public space. For example, users should be allowed to vote if a portion of the public space respects some community-set criteria.

Data for the representation of the public space can be uploaded from other sources as well. The reality of the public space does not change, but the circumstances for it do. An example would be the weather conditions for the public space. In this case, the public space does not change, the architectural layout is still the same, but some architectural features can be impacted by weather. Another example would be the air quality inside the public space.

The complete *problem* is to build solutions to adapt external data and community data to extend the experience of the user inside the public space.

The *solutions* are services that allow integrations in the platform, to extend the representation of the public space.

The *specifications* are to allow the administrators to integrate data from external sources and help the end-users. The end-user is invoking the base navigation services with the options that point to external or internal data.

3.4 Configuration and Validation for a University Campus

As an example of public space, we choose a university campus for our proof-of-concept. This choice comes as the output of an analysis depending on two factors: configuration and validation.

Given our academic inclinations, we can model the services for a university campus in an effective manner. Also, a university campus can include various participants, thus multiple needs. The structure of the university campus is mostly static, meaning the architectural features do not change dynamically depending on certain events. A university campus can have multiple buildings and most probably, different administrators.

The complete *configuration problem* is to create an extensible configuration of the university campus that can support users' needs.

The *configuration solution* is to allow the administrators to decide the granularity of architectural details the model for the university campus will have when used by the end users.

The *configuration specifications* are to allow the administrator to upload configuration details regarding the real university campus, and configuration details that must follow guidelines supported by the platform.

The complete *validation problem:* the configuration of the public space is valid if the user can use the information provided by the platform, to decide while inside the university campus.

The *validation solution:* to build on top of the configuration initiated by the administrator in the configuration step.

The *validation specification:* to create services that are decoupled from the configuration code and object of the public space.

4 Research Method

In this section, we discuss general scenarios that we analyzed to determine the final form of the services we offer to the end user. We discuss the service-oriented approach we chose to achieve the design for these services as well as their implementation. We discuss multi-policy pedestrian routing where we explain what the *policy* means for our platform and how it helps the user to have a customized experience using the navigation solution. We discuss the validation of this thesis contributions for a university campus and why we chose to focus on this topic for this thesis and the proof-of-concept presented later in this thesis.

4.1 Scenario Analysis

People easily navigate public spaces they are very familiar with. However, when removing that familiarity, navigation and orientation become complex. To understand the scenarios a person is faced with in a public space that is not well known to her or him, we have to analyze what a public space looks like in the first place, and then what is the purpose of that public space.

Navigation prerequisites scenarios. To navigate a public space, people need to know where they are going (know their destination/recognize they arrived at their destination) and know what direction they should have (to be pointed in the right direction/recognize subsequent intermediary points along the waypoints of interest).

Navigation complexity scenarios. The complexity of the public space dictates the complexity of the navigation for the user. Not knowing how to navigate different kinds of complexities could imply an additional cost (lost time, or lost orientation); this is why destination recognition and direction matter in our platform design.

Custom-use complexity scenarios. Different users' needs involve other types of difficulties that a navigation platform should consider. Multiple scenarios can be mentioned here, such as locomotory complexity (that should be the most important aspect for a person with locomotory issues), crowd avoiding complexity (in case users want to avoid crowds that are created without a known-before pattern inside a public space), third party complexity (in case users want to get a correct recommendation for their navigation, considering data that originates from external providers, e.g., weather information).

4.2 Public Space Representation

At the center of the platform described in this thesis is the modelling of the public space. This model must express the reality and offer this information for building the necessary services for the users. This information must be easily stored, maintained, and used for navigation algorithms. These are the reasons why we chose our model for the public space to be a general-purpose weighted graph.

- A general-purpose weighted graph is a structure that is created around the idea of neighbors.
- A general-purpose weighted graph has standardized algorithms for querying a path inside it. This implies that algorithms are well-known and tested, thus, the end users will be reliably provided with navigation information. A general-purpose weighted graph has multiple ways to be configured: nodes, edges, and weights.
- A general-purpose weighted graph can then be created from multiple data points, in our proof-of-concept presented in the following chapters we describe how from simple static data represented in an Excel sheet in a particular format, we built the underlying graph model used by all platform services.

The granularity of the graph representation can be set by the needs of the public space or expanded in the future. In our proof-of-concept we chose the granularity of the basis graph model as follows:

- the graph nodes represent each architecture topology like rooms, hallways, stairs etc.
- edges between nodes are direct abstract connections (can be read as one node is a neighbor of another)
- the graph weights represent the users' needs (they are updatable quantifications regrading what the user needs from the navigation)

4.3 Service-Oriented Design

Our design is based on exposing services for wayfinding, to help people choose the most appropriate route for walking through a public space and getting to their desired destination. The core of the system is the capability to model the real-world space in a graph-like structure, to enable the adaptability of routing algorithms, as detailed by Costa et al. in [51]. The route adopted by each person influences the overall pedestrian flow, and the routing services also take into account the feedback given by the community of people who benefit from such services (Figure 1).

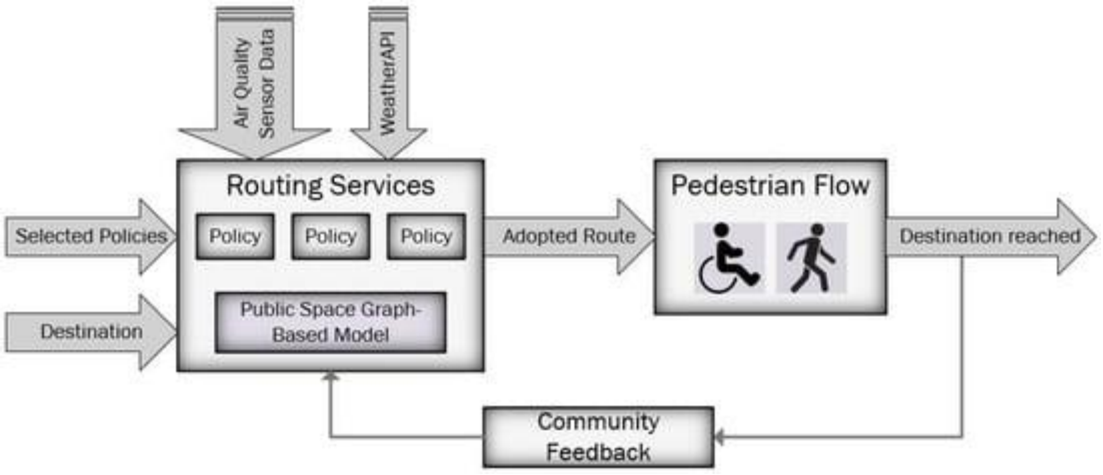


Figure 1. Pedestrian routing system.

The routing service takes as input the nodes generated by administrators and external parameters, such as weather, epidemiologic risk, personal issues such as locomotory difficulties, or emergencies. For this reason, the nodes must be created in conformity with a model that supports such policies; therefore, these policies need to be known before the actual real-world resources are modelled as nodes. For the system to be more robust, these policies represent a resource inside the system, exposed by other services; they should be a source of compliance for the nodes that are created by invoking them. The services are written in a client–server architecture that expose HTTP endpoints.

4.4 Multi-Policy Pedestrian Routing

Our design is based on the idea that a public space that offers a platform for navigation and orientation needs to take into consideration end-users needs. And needs from users are technically infinite. As much as we can presume the needs of the user, the truth will always be held by the end user.

Multiple graph models. One way to satisfy all needs is to create multiple graph models of the public space, multiple models for the weights inside the graphs, and/or multiple algorithms that could transform all data into an ordered sequence that can be provided to the user to navigate with. This method would include a lot of storage structures and costs, more maintenance on the models and more specialized information that an administrator would have to continue maintaining it, thus more training. This approach (Figure 2) presents an issue on how to solve the integration of all different graph models (that represent the needs of users), if a user has multiple needs (for example a user wants the less crowded and with the least locomotory difficulties).

Centralized graph model with dynamically generated weights. Another way to satisfy all needs is to create a base graph model to represent a public space first and to enable weights to represent the needs of the users. A great weight on an edge from the graph would mean a greater cost for the user to use in the real world what is represented by that portion of the public space. Thus, weights are dynamically generated, and user driven. Meaning the user selects when she/he wants to travel the public space considering several needs. Selecting multiple needs upfront and not relying on a different model for different needs is a more scalable and modular approach than the former one presented. This approach resembles a multi-filter set on a search (of a database for example).

The latter approach, based on a graph with dynamically generated weights, was applied in this thesis (Figure 3). The manner we enable multiple needs of the user to exist in the platform is by creating *policies*. *Policies* are resources that are stored in the platform and describe how the weights of the graph are affected if the end user invokes the same *policies.* We called them policies because they have the form of a contract, a contract that is created by a party (administrator), a contract that has points of interest like where it gets the data that represents the needs of the user, a contract that describes how it processes the data to be understood by the underlying object which is the graph model and the navigation algorithm, and a contract that can be invoked by the end-user to be provided with the benefits of that contract.

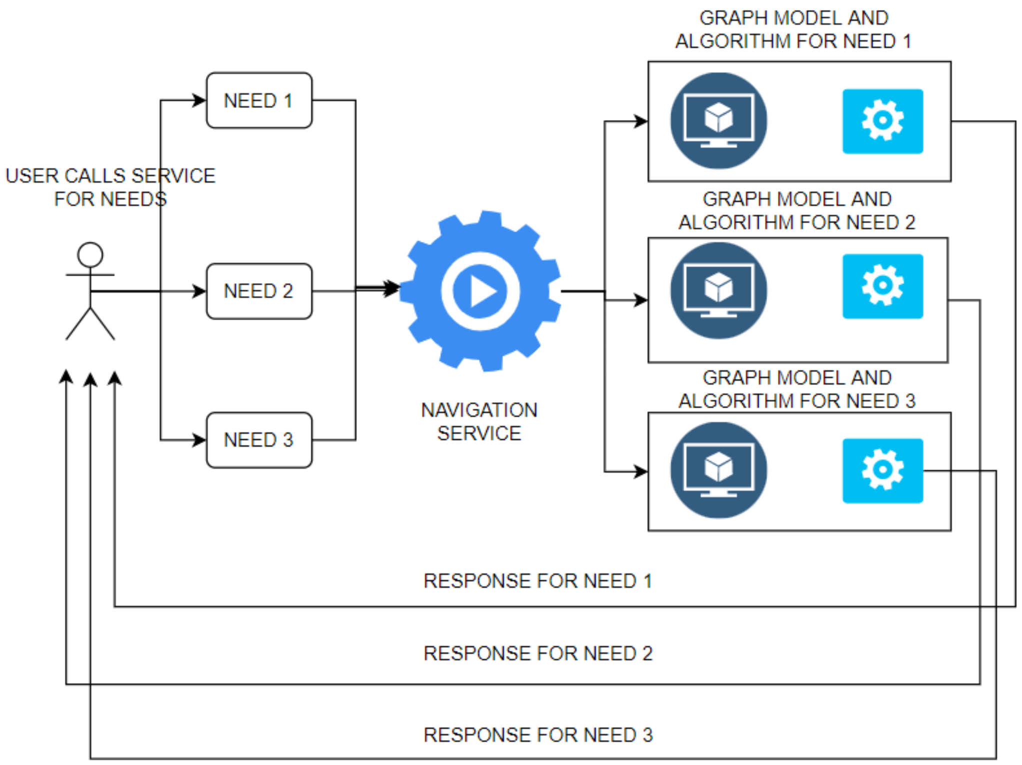

GRAPH MODEL AND ALGORITHM FOR NEED 1
NEED 1
USER CALLS SERVICE FOR NEEDS
GRAPH MODEL AND ALGORITHM FOR NEED 2
NEED 2
GRAPH MODEL AND ALGORITHM FOR NEED 3
NAVIGATION SERVICE
NEED 3
RESPONSE FOR NEED 1
RESPONSE FOR NEED 2
RESPONSE FOR NEED 3

Figure 2. The user has to call services for each need.

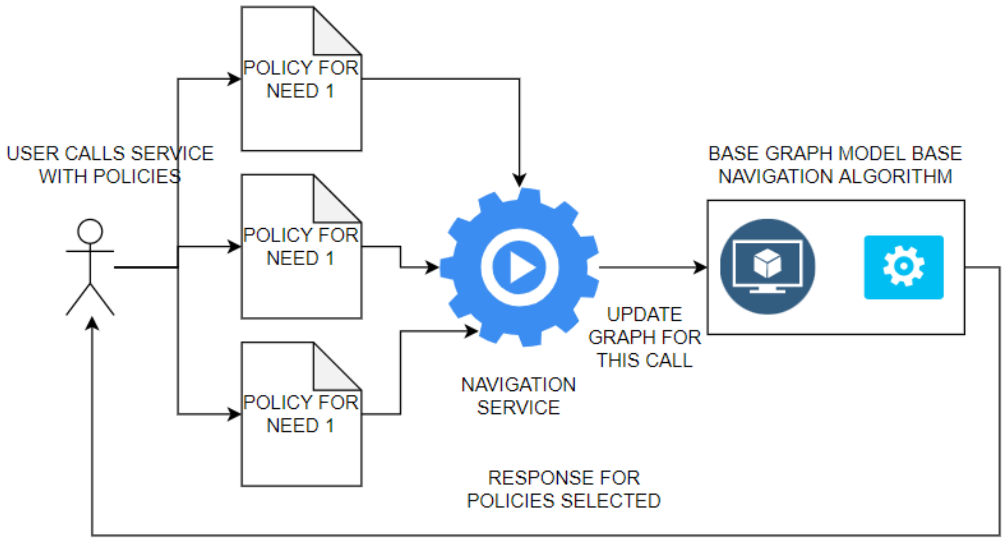

POLICY FOR NEED 1
USER CALLS SERVICE WITH POLICIES
POLICY FOR NEED 1
POLICY FOR NEED 1
BASE GRAPH MODEL BASE NAVIGATION ALGORITHM
UPDATE GRAPH FOR THIS CALL
NAVIGATION SERVICE
RESPONSE FOR POLICIES SELECTED

Figure 3. Centralized graph model for particular policies.

4.5 Validation for a University Campus

There are many types of public spaces that may have been chosen for our proof-of-concept to validate our system design, but we choose a university campus to verify and validate our system is more efficient due to several considerations.

University campuses are not as closed as public spaces (police station/hospital), because they are still encouraging free navigation, they do not serve the purpose of mass entertainment (mall) and participants on a university campus do not indulge in spending time. University campuses are vast areas where participants usually know only a part of it. University campuses are places where a participant arrives with a defined and time-bound task. This implies urgency in resolving the task.

University campuses combine different categories of architectural topologies that are of interest to the participants. These architectural topologies could ease or make the navigation of participants more difficult. Therefore, there is a need to validate the proof-of-concept for a university campus for users' preferences.

Participants in a university campus have different tasks day by day and different needs that are either of personal nature or objective nature (created by the university campus itself). These needs are resolved with the help of policies, so that each participant feels included in the university campus and has a good experience traversing this public space.

5 Research Contributions on Orientation and Accessibility Services

In this section, we discuss the scientific contributions built on top of the details presented in the previous sections:

- we chose a model that supports route finding based on multicriterial weighted graphs.
- we model the public space as a graph, and this is beneficial for route finding.
- we discuss the integration of multiple criteria to this graph, how we model criteria to fulfil users' needs and how it fits against a navigation scenario inside a public space.
- routing policies that are derived from the real-world needs of users and how they can extend the purpose of the platform.
- the general workflow inside this platform where we describe how administrators and end-users interact with the platform.
- the overall architecture that supports the outlined design and how services provide and consume data inside the platform, as well as the visualization service for end-users.

5.1 Route Finding Based on Multicriterial Weighted Graphs

We choose to represent a real-world public space as a graph, since graph theory already provides algorithms that follow the behavior in the real world, where navigation from the current location (node) towards a destination (node) is a set of steps that can be modelled with graph vertexes.

A one-criterion routing algorithm operates based on a weighted graph that only takes into consideration the amount of consumption that it takes to traverse the resources (nodes). Let us analyze an example of a weighted graph, where A, B, and C are nodes and X, Y, and Z costs to navigate from A to B, B to C, and A to C, respectively. If on the route A → B → C, the cost is $X + Y$, and it is more economical than choosing A → C that costs Z, then the chosen route should be A → B → C.

This approach can be employed as a multicriterial algorithm. The changes towards the multicriterial algorithm must be reflected both on the algorithm and on the representation; the nodes should have special properties that the algorithm should take into consideration. Thus, the weights of the dependencies (vertexes) become functions of the properties of the connected nodes, as represented in Figure 4. Some properties of the nodes might not be interesting for certain stakeholders. The nodes must have multiple properties that represent the interests of certain users; such for a newsletter subscription, all people have interests, but often they are different from one to another, and should not impact other people's interests. A user may "call" the multicriterial algorithm with his or her particular interests (nodes properties); thus, the multicriterial algorithm calculates the output in a customizable manner.

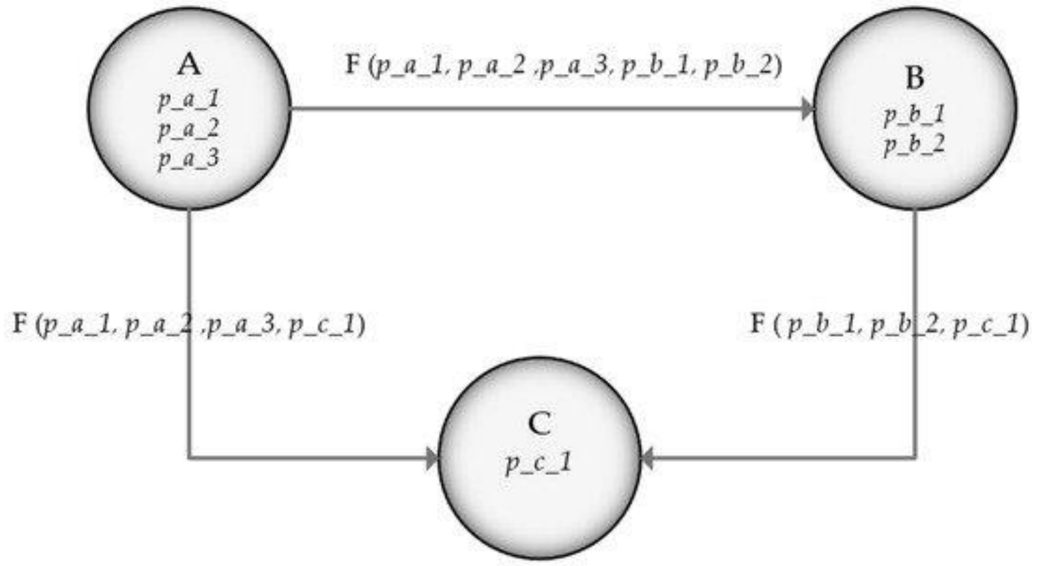

A
p_a_1
p_a_2
p_a_3
F (p_a_1, p_a_2 ,p_a_3, p_b_1, p_b_2)
B
p_b_1
p_b_2
F (p_a_1, p_a_2 ,p_a_3, p_c_1)
F (p_b_1, p_b_2, p_c_1)
C
p_c_1

Figure 4. General weighted graph for multicriterial routing algorithm.

In a generally weighted graph with three nodes, each connection between the nodes has a cost known as weight in the graph theory. In the multicriterial approach, one can assume that there is a function cost for computing the weight. Figure 4 depicts a weighted graph with three nodes, in a similar connection configuration as in the example above. The difference is that the cost to traverse nodes is no longer standalone information. Each node has designated properties. These properties on each node represent how difficult it is to access this node. To traverse from one node to another, a cost function that integrates the properties of the two nodes must be taken into consideration. This approach lets us represent how hard it is to access a resource and abstract the cost between two such resources. We consider that each of these properties has a numerical value, and the function F is the SUM function, meaning that it adds all the properties and virtually changes the weights of the graph. If these properties change dynamically, the weights of the graph also change dynamically. If we abstract these properties under certain categories (or criteria), the users can call the multicriterial routing algorithm giving these inputs. This means that the user wanting to navigate in the building/campus/public space can minimize the impact for the given criteria (since the base algorithm minimizes the navigation cost).

5.2 Graph-Based Model of the Public Space

The building graph model, represented in Unified Modeling Language, is given in Figure 5. Public space includes several buildings and several connections between the buildings that are on-site. A building's attributes include namespace (i.e., the name of the building), the number of nodes (corresponding to architectural features), and the number of connections between the nodes in the same building. A connection has two properties (source and target) to represent the neighbors of a node or building. Each node has node properties (see "*" to show the multiplicity > 0 in Figure 5), including the type of node (hallway, stairs, room, etc.) and policies. A policy is defined by name, data source (where the data come from, to build the particularities of the policy), and "dataManipulation" (a procedure for how to interact with the data from the data source).

As explained previously, a node's properties should be grouped under common criteria, to provide the user with the possibility to invoke these criteria when calling the routing algorithm (otherwise the algorithm cannot centralize information and create the weights on the graph); thus, one should employ a specific abstraction to describe this requirement. For this purpose, we define a series of *policies* to be available for the pedestrian who wants to obtain routing directions to reach the desired destination. The *policies* are defined in our approach as rules on how the multicriterial algorithm should compute the weights on a graph, where to retrieve or store information about nodes, and how information from nodes should be used while the multicriterial algorithm is running.

Such *policies* are applied to the entire graph and then they activate certain properties for each node; the attribute "nodeProperties" from Figure 5 can be populated with data concerning the chosen *policies*, to be then considered in the cost of each connection, by using the SUM function on all data quantified from *policy* data. In this way, the algorithm minimizes the overall cost and outputs the least cost route. Nonetheless, other properties of nodes can be populated with access data that will be provided to the user as the route generator output.

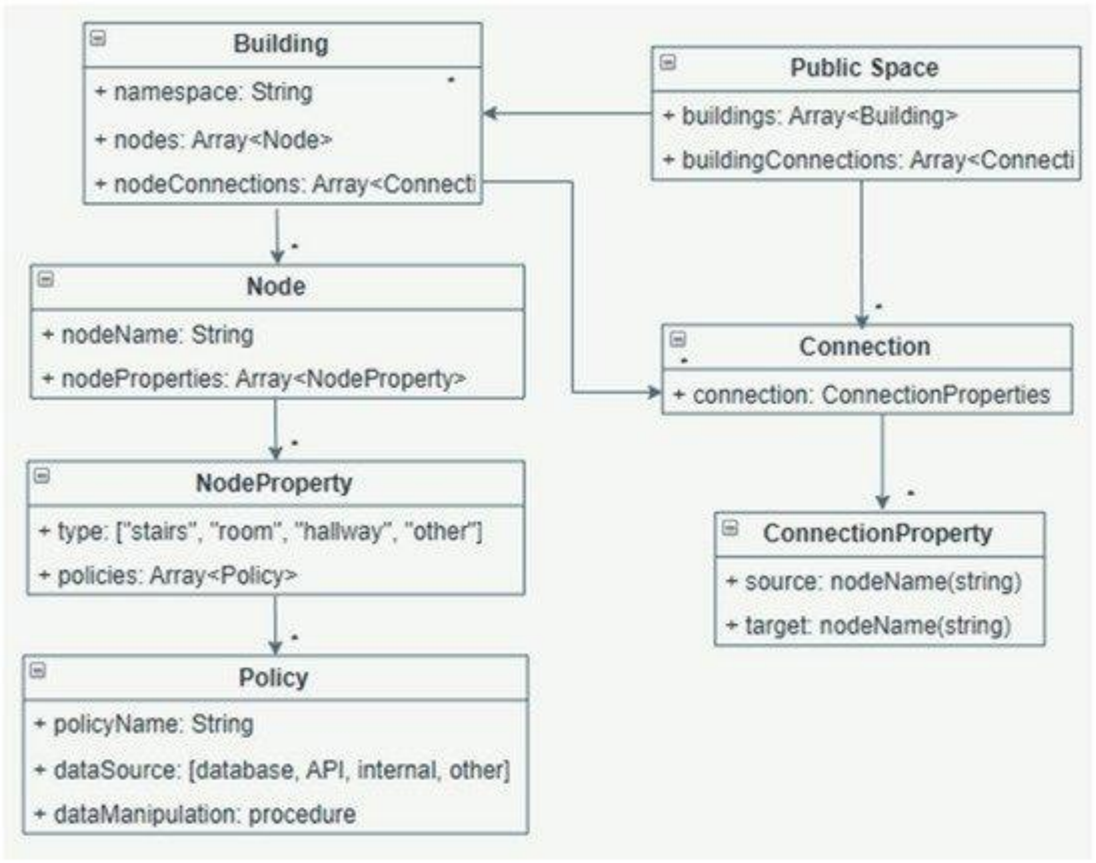

Building
+ namespace: String
+ nodes: Array<Node>
+ nodeConnections: Array<Connecti
Public Space
+ buildings: Array<Building>
+ buildingConnections: Array<Connecti
Node
+ nodeName: String
+ nodeProperties: Array<NodeProperty>
Connection
+ connection: ConnectionProperties
NodeProperty
+ type: ["stairs", "room", "hallway", "other"]
+ policies: Array<Policy>
ConnectionProperty
+ source: nodeName(string)
+ target: nodeName(string)
Policy
+ policyName: String
+ dataSource: [database, API, internal, other]
+ dataManipulation: procedure

Figure 5. Building graph model with policies attached.

5.2.1 Navigation scenario for public spaces

To analyze several scenarios, we will refer to Figure 6. This figure depicts the plan of a hypothetical building with 2 entries connected to the outside of the building. The entries are named “Entry 1” and “Entry 2”. “Hallway 1” is connected to both entries. “Room A” and “Room B” are connected to “Hallway 1”, and the two rooms are not connected. For this case, it is needed a model for the reality of the public space. A model that can be easily extendable if an architectural feature would be created or destroyed in the reality. And a model that could run navigation algorithms for the user. Such a model is a graph. A graph is an efficient representation of reality if correct assumptions are made. A graph model made for the hypothetical building is shown below in Figure 6 next to the hypothetical building.

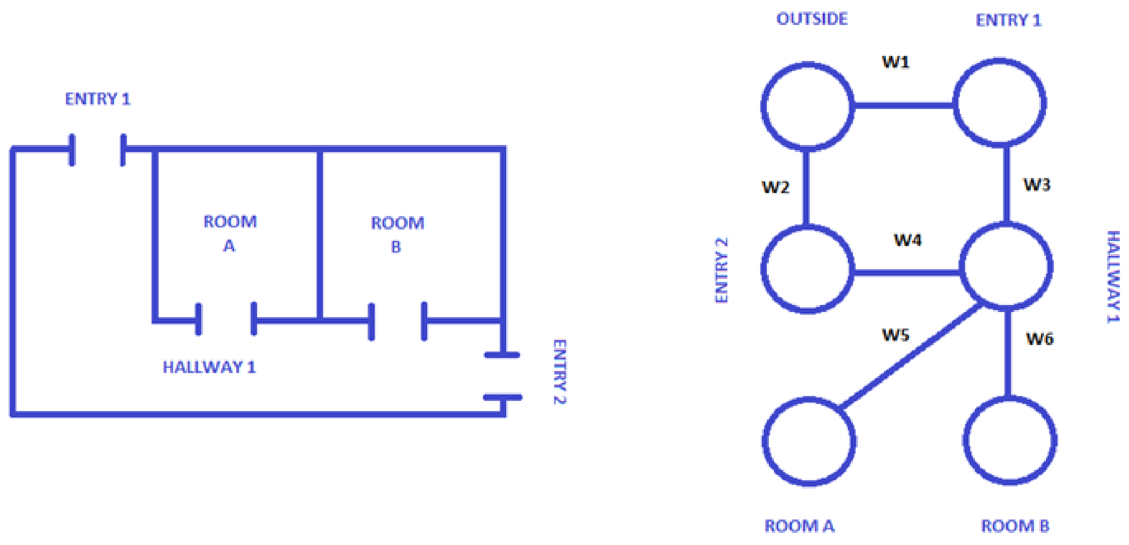


Figure 6. Hypothetical building and it’s respective Graph model.

The W1, W2, … W6 notations on the graph figure from Figure 6 represent the weights of the graph that are attributed to the edges. The way we worked with weights in this model is to assume weights are a general description of *cost*. The *cost* is quantifiable, which makes it measurable and easy to integrate with other costs. This abstraction makes it possible to extend the functionality of the services that run on top of this modelling, which is important for the overall thesis. For a simple navigation scenario where one user needs to traverse the public space, the weights can be considered a *constant cost*. And a navigation algorithm based on a graph representation (like Dijkstra) is possible to output the shortest path possible between the starting point of the navigation, and the destination set by the user.

5.3 Routing Policies

Typically, a route-generating algorithm for a graph follows the principle of reducing the cost of travel; this can be realized as the shortest path, or as the most cost-efficient path, choosing the route with the least cost to arrive at a destination. Our algorithm adopts the cost-efficient approach for the following reasons:

- a cost-efficient path algorithm works on top of a weighted graph.
- a weighted graph can be created to quantify the difficulty of accessing a node.
- we integrate more diverse information (based on the users' needs) in the weight value for every vertex (policies, optionally used).

We decided to look at a node as a stand-alone concept such as a web page, and let the node dictate how hard it is to be accessed, based on policies that are attached to it. The path-generating algorithm looks at every node in the graph and decides, based on the policies given at the algorithm call if passing through that node is desirable or not. These policies are quantified and provided as a weight addition in every vertex that the node has, allowing us to use well-established graph path generation algorithms.

Policy 1. Consider the community votes. A voting policy can offer to the general user the ability to vote if a resource respects certain criteria. The votes in time add up and are quantified via a mean value and considered when policy is requested.

Policy 2. Avoid crowded areas. Another policy, also driven by the community of pedestrians using the routing service, considers the crowding factor. When users ask the system to obtain a certain path, it means that, when accessing the constituent real-world locations, they occupy the correspondent graph nodes. Many users might need to traverse some sections in the real world, which are overlapped with other users' sections, resulting from their need to find their way; therefore, crowds can form spontaneously. The implementation of this policy is based on the outputs of previous users' routing needs, by increasing the weight of the constituent nodes. Thus, when the next user calls the routing service with the policy to avoid crowds, the weights on the links of the previously visited resources are increased, and the algorithm provides an alternative route that is more cost-efficient (where the cost regards how crowded the space is). After a given time interval, the weights on the nodes are automatically decreased if no longer provided by the routing service, meaning that the correspondent locations become less crowded.

Policy 3. Avoid polluted areas. This policy is data-driven and can be applied if the public space is provided with sensing devices to detect various physical quantities to characterize the air quality.

Policy 4. Shelter from unfavorable weather conditions. This policy also ensures a data-driven routing, based on data originating from external providers, such as weather forecast providers. The routing service takes into consideration minimizing the effect of external factors created by weather conditions.

Policy 5. Consider accessibility for reduced mobility needs. This policy holds information about the accessibility of public space locations that may create difficulties for users with reduced mobility needs. When this policy is used in the routing algorithm, the calculated route eliminates nodes or connections that lack accessibility, or at least the user is presented with information regarding the potential risks in the real world.

5.3.1 Multi-policy pedestrian routing algorithm

0-Policy Routing Algorithm. By default, the underlying graph of a building has the weights of all connections equal, because the nodes represent physical locations that are very close to each other. This applies in the case of a 0-policy routing request, where the cost-efficient algorithm prioritizes the path that cumulates a minimum value, i.e., the shortest path. Nonetheless, the users indirectly increase the weights on the paths that were given as routing solutions if the user follows the routing directions. This applies to 0-policy routing as well, because in case a user wants to avoid crowded places, it is necessary to know what locations (graph nodes) were the most visited, even if the users who requested the routing service have not selected Policy 2. If a node is present in a routing path output calculated in the recent past, the crowding-factor property is increased with a fixed amount; if a node has not been present in an output routing path, the crowding factor is decreased with a fixed amount.

Multi-Policy Routing Algorithm. In the case of the multi-policy routing algorithm, each node has specific properties that follow the policies, and these properties are used to update the weights between nodes. The cost-efficient routing algorithm outputs the path that cumulates a minimum value, which may not always be the shortest path, as it was in the case of the 0-policy request, but the one that is the most convenient to the user given his or her needs. Each policy looks at the specific node property/properties that indicate it. The policy looks at the paired properties on each node; Algorithm 1 shows that the system always listens to the user's requests and delivers the output of the routing service calls. Certain details are necessary to compute the nodes' properties regarding how crowded they are:

- *crowdProperty* – designated property on the node for the crowd-avoiding policy
- *increaseCrowd* - the amount that the *crowdProperty* value on a node should increase after it was given as a routing solution and the directions are followed by the user.
- *crowdTimeStamp* - when the *crowdProperty* was last updated.
- *currentDate* – the current date when the user call is made

Algorithm 1 Multi-policy pedestrian routing algorithm

```
Inputs: increaseCrowd
while routing service provided do
get from user call: start, finish, policies
for policy in policies:
switch policy:
Policy1: call community feedback algorithm
Policy2: call crowd avoiding algorithm
```

```
Policy3: call pollution avoiding algorithm
Policy4: call weather conditions algorithm
Policy5: call reduced mobility algorithm
end switch
receive customized_graph
end for
call multicriterial_route_finding_algorithm(customized_graph)
send directions to user
get from user call: accepted_directions
if accepted_directions
for element in path do
  if element.crowdTimeStamp is_aprox currentDate
    element.crowdProperty = element.crowdProperty + increaseCrowd
    element. crowdTimestamp = currentDate
  end if
end for
end if
end while
```

When the user calls the routing service, the system takes from the call details for start (the point from which the navigation begins) and finish (the destination of the navigation). It computes the path against the newly updated graph (*customized_graph*) taking as input start and finish and sends the user the result obtained with the multicriterial routing algorithm. After this, for each node in the path given to the user (if the node property *crowdTimestamp* is approximatively equal to the *currentDate* timestamp), the *crowdProperty* for this node is increased with *increaseCrowd* (configured to tell the *crowdProperty* by how much it should be increased after it was visited), and the *crowdTimestamp* of this node is overwritten with the value of the *currentDate*.

5.3.2 Community feedback algorithm

After using the path given by the navigation service, the user can give some feedback if the path was helpful and easy to use for her/his needs. This feedback takes the form of a vote, where the user can vote from an interval, for example, 1 to 10, where 1 means the user would most likely never want this path a second time she/he calls the navigation service, and 10 means the user would most likely want this path a second time she/he calls the navigation service. These votes are stored in the platform with the total number of votes (incrementally after each provided path is given to the user, and the user votes it). When the next call for the navigation service is made with the community feedback policy, the service will take into consideration previous feedback from users and will increase the weights of the nodes that were included in the navigation service output that the user voted with low desirable scores. Thus, the navigation service will compute a route that intersects as little as possible the nodes that were not voted favorable by past users.

Algorithm 2 Community Feedback algorithm

```
Inputs: graph, communityFeedbackList, limit1, limit2, limit3, limit5, limit6
for node in graph do
get from communityFeedbackList nodeVoteScore
for connection in nodeConnections do:
if nodeVoteScore > limit1 and nodeVoteScore < limit2
addingUpNorm = 1
end if
if nodeVoteScore > limit2 and nodeVoteScore < limit3
addingUpNorm = 2
end if
if nodeVoteScore > limit3 and nodeVoteScore < limit4
addingUpNorm = 3
end if
if nodeVoteScore > limit4 and nodeVoteScore < limit5
addingUpNorm = 4
end if
if nodeVoteScore > limit5 and nodeVoteScore < limit6
addingUpNorm = 5
end if
weightConnection = weightConnection + addingUpNorm
end for
end for
return graph
```

5.3.3 Crowd avoiding algorithm

The method we model the crowd factor inside a public space is to increase the weight of the graph accordingly to a *crowd factor*. This means that each user that traverses the graph, increases the weights of the edges of the graph (representing the connection between the real architectural features) that were visited. With the increased weights of the edges stored in the graph, the navigation algorithm that runs on top of the graph will output another route than a route for a previous user (if the real public space is constructed to offer it).

In our application, the graph crowd factor adds up each time, a user calls and accepts a route provided by the navigation service. Because after some time the real architectural features could be considered not occupied, the crowd factor has time to leave (TTL) attached to it. If that time passes and no other navigation service call returns an architectural feature that previously increased the connected weights, then, the increased weights are decreased by a *leave factor*.

The Crowd Avoiding Algorithm corresponds to Policy 2. As resulted from the previous description of the Multi-Policy Routing algorithm, even if there is no policy selected, the underlying graph is always updated with the increased weights on the sections that were most visited by other pedestrians in a certain period. Algorithm 3 configurations:

- *crowdProperty*—the crowd value on the node (the bigger the value the most visited the node is).
- *crowdTimeStamp*—when the *crowdProperty* was last updated.
- *crowdTimeFrame* – a time interval after which one considers that the *crowdProperty* value on a node should be decreased.
- *decreaseCrowdFactor* – how much a *crowdProperty* on a node should decrease after the *crowdTimeframe* expires.

Algorithm 3 Crowd avoiding algorithm

```
Inputs: decreaseCrowdFactor, crowdTimeFrame, graph, limit1, limit2, limit3, limit4, limit5, limit6
for node in graph do
   get from Policy2: crowdProperty, crowdTimestamp
   timeDifference = currentDate – crowdTimestamp
   timeDifferenceFrame = timeDifference / crowdTimeFrame
   decreaseCrowdProperty = decreaseCrowdFactor X timeDifferenceFrame
   crowdProperty = crowdProperty – decreaseCrowdProperty
   if crowdProperty < 0
   crowdProperty = 0
   end if
```

```
for connection in nodeConnections do:
if crowdProperty > limit1 and crowdProperty < limit2
addingUpNorm = 1
end if
if crowdProperty > limit2 and crowdProperty < limit3
addingUpNorm = 2
end if
if crowdProperty > limit3 and crowdProperty < limit4
addingUpNorm = 3
end if
if crowdProperty > limit4 and crowdProperty < limit5
addingUpNorm = 4
end if
if crowdProperty > limit6
addingUpNorm = 5
end if
weightConnection = weightConnection + addingUpNorm
end for
end for
return graph
```

The algorithm proceeds with computing *crowdProperty* as the current *crowdProperty* minus *decreaseCrowdFactor*, multiplied by the result of *currentDate* minus *crowdTimestamp*, and divided by *crowdTimeframe*. In this manner, the *crowdProperty* is always updated based on how much time has passed since the node was visited. If after this operation *crowdProperty* is less than 0, it means that a long time has passed since the node was visited, hence it is a viable node to be visited for users who specifically chose to avoid crowds. After *crowdProperty* for this node is computed, one updates the weight for each connection of this node. The five limits may be set to adjust the weights by adding up a norming number, with respect to the degree of how much crowding is considered appropriate.

5.3.4 Pollution avoiding algorithm

The pollution-avoiding policy increases the paired property of the nodes based on data gathered from sensors; thus, a sensor value is mirrored on the paired node property. In the same way, as explained in the above paragraph, the weights of the node with sensor policy are updated accordingly when Policy 3 is invoked by the user, to favor the paths that contain nodes with the smallest pollution values. Algorithm 4 configurations are *sensorDataSource* (data source providing the readings from the air quality sensors); *sensorValue* (actual value converted to *addingUpNorm* which is used for graph weight update).

Algorithm 4 Pollution avoiding algorithm

```
Inputs: graph, limit1, limit2, limit3, limit4, limit5, limit6
for node in graph do
get from SensorPolicy: sensorDataSource, sensorValue
sensorValue = call sensorDataSource
for connection in nodeConnections do:
if sensorValue > limit1 and sensorValue < limit2
addingUpNorm = 1
end if
if sensorValue > limit2 and sensorValue < limit3
addingUpNorm = 2
end if
if sensorValue > limit3 and sensorValue < limit4
addingUpNorm = 3
end if
if sensorValue > limit4 and sensorValue < limit5
addingUpNorm = 4
end if
if sensorValue > limit5 and sensorValue < limit6
addingUpNorm = 5
end if
weightConnection = weightConnection + addingUpNorm
end for
end for
return graph
```

Generally, a multi-policy algorithm needs norming. Our solution proposes that each data value is mapped to a discrete value from the set {1, 2, 3, 4, 5}. In Algorithms 2, 3, 4, 5 and 6, these norming numbers are added to the weights of the graph, thus representing desirability levels (1—very desirable, 2—desirable, 3—neutral, 4—undesirable, 5—very undesirable).

5.3.5 Weather condition shelter algorithm

The weather policy can be invoked to help the users to cover unfavorable weather conditions because the outside is more affected by the weather conditions, which means resources that are located outside of buildings will be affected by the weather the most. The navigation service that is invoked with this policy will send an inquiry about the weather conditions to an external source (most likely a weather API). If this weather API will find favorable conditions that support navigation outside of buildings, then the outside resources that support such a navigation will be returned to the path for the user.

The description of the navigation service using this algorithm is in Algorithm 5 bellow:

Algorithm 5 Weather condition shelter algorithm

```
Inputs: graph, WeatherAPI, limit1, limit2, limit3, limit4, limit5, limit6
for node in graph do
get from WeatherAPI weather
for connection in nodeConnections do:
if connection == outside
if weather > limit1 and weather < limit2
addingUpNorm = 1
end if
if weather > limit2 and weather < limit3
addingUpNorm = 2
end if
if weather > limit3 and weather < limit4
addingUpNorm = 3
end if
if weather > limit4 and weather < limit5
addingUpNorm = 4
end if
if weather > limit5 and weather < limit6
```

```
addingUpNorm = 5
end if
end if
weightConnection = weightConnection + addingUpNorm
end for
end for
return graph
```

5.3.6 Reduced mobility algorithm

Public spaces should be inclusive spaces for any participant to navigate safely. This is another scenario that we considered in our platform. This is where our abstract view on weight as cost inside the graph helps support these scenarios. The cost could be considered by an end-user as the difficulty of traversing a portion of public space. Difficulty can come from many criteria in the case of people affected by locomotory issues. For example, stairs with no ramps, heavy doors, different height levels on the floor, etc. These difficult architectural features are included in the graph model, in a way that can increase the weights of the edges connected to those.

For a scenario where one user tries to access from “Outside” “Room A”, the graph model supports the integration with a *difficulty map*. If we suppose the door of “Entry B” is heavy and the door of “Entry A” is much suited for the locomotory issues, then the path containing “Entry A” will be less costly thus, it will be provided to the end-user.

To establish the difficulty level of each architectural topology that represents a node in our graph model, a platform administrator must upload to the platform what we refer to above as a *difficulty map*. This *difficulty map* is created by assessing the real danger of an architectural topology. To create such a *difficulty map*, a platform administrator could make use of multiple manuals that are given to assess the risks of a building for users with locomotory issues. In our proof of concept, we used the guidelines of [45].

Algorithm 6 Reduced mobility algorithm

```
Inputs: graph, locomotoryModel, limit1, limit2, limit3, limit5, limit6
for node in graph do
get from locomotoryModel locomotoryDifficulty
for connection in nodeConnections do:
if locomotoryDifficulty > limit1 and locomotoryDifficulty < limit2
addingUpNorm = 1
end if
```

```
if $locomotoryDifficulty > limit2$ and $locomotoryDifficulty < limit3$
$addingUpNorm = 2$
end if
if $locomotoryDifficulty > limit3$ and $locomotoryDifficulty < limit4$
$addingUpNorm = 3$
end if
if $locomotoryDifficulty > limit4$ and $locomotoryDifficulty < limit5$
$addingUpNorm = 4$
end if
if $locomotoryDifficulty > limit5$ and $locomotoryDifficulty < limit6$
$addingUpNorm = 5$
end if
$weightConnection = weightConnection + addingUpNorm$
end for
end for
return $graph$
```

5.4 General Workflow for the Platform

In our assumption, the data for a public service must be driven by the community that benefits from it. Therefore, our platform has services that allow the administrators of any public space to upload the version of the public space they need for their communities. Thus, we do not create a static public space representation, but the administrator must create it. Once the representation is created, it is uploaded to the platform that converts it to the graph model where other graph-related operations, such as pathfinding, can run. The responsibility of the platform is to create the conversion service. In other words, the map model, because it maps values understood and created by the administrator to computation objects that could not be understood. The only way to respect a correct flow of information between the administrator and the platform is a set of rules or standards that can be easily understood by the administrators and easily implemented by the platform.

The workflow presents 3 phases:

- bootstrap (that creates the computation resources)
- consumption (that consumes the API and services)
- maintenance (that implies all activities to keep the platform up-to-date and relevant) [52].

5.4.1 Visualization design for the bootstrap phase

In our proof-of-concept, we set those rules around the visualization of public spaces building plans and the vicinity of architectural topologies indicated in those building plans. To be specific, we implemented services to upload raster file images that are a scale representation of the buildings' floors and an Excel file that represents what architectural topologies are in the vicinity of other architectural topologies and at what coordinates, on which building's plan. The same uploaded images will be used by the navigation services to draw on top of them directions for the user's call. We chose Excel because it is easier to use for most people, including administrators of public spaces. Any data provided by the administrator is used by the platform to create all the necessary support objects for the main service which is the navigation service.

5.4.2 Platform phases. Example of a university campus

The *bootstrap phase* is where the administrator uploads via specialized platform service all the necessary data into the platform to create the base graph model and the visual maps. For this phase, the platform requires data that is easily comprehended by humans. At this moment the platform can understand Excel files that store the intersections of the real public space resources and on what plan they are found. This plan must correspond with the images of the plans that are uploaded initially. The Excel file has a simple structure that can be easily created and understood by humans (apart from how the base graph model looks like, or the node properties map), and the visual maps are simple image files that are named in the same way they are referenced in the Excel file for the correspondence. This is the most critical phase, without this phase, the users will not be able to call the path service to get a route and an intuitive visual map through the public space.

The *consumption phase* is where the users are using the platform to find ways to move around the public space, where the most important service is the path service that will provide them with the route to be taken, human-readable directions, and visual maps. This path service can be called with policies to consider what the user needs most when traversing the public space.

The *maintaining phase* is the phase where the resources generated from the bootstrap phase are enhanced and maintained up to date. This phase is where the policies are created (from external resources, like weather APIs, or internal resources, like crowding in the public space) to offer the user a personalization option. Figure 7 displays the main operations the administrators and the users can make on this platform, without the initial input from the administrators on the platform, the user cannot start consuming its services.

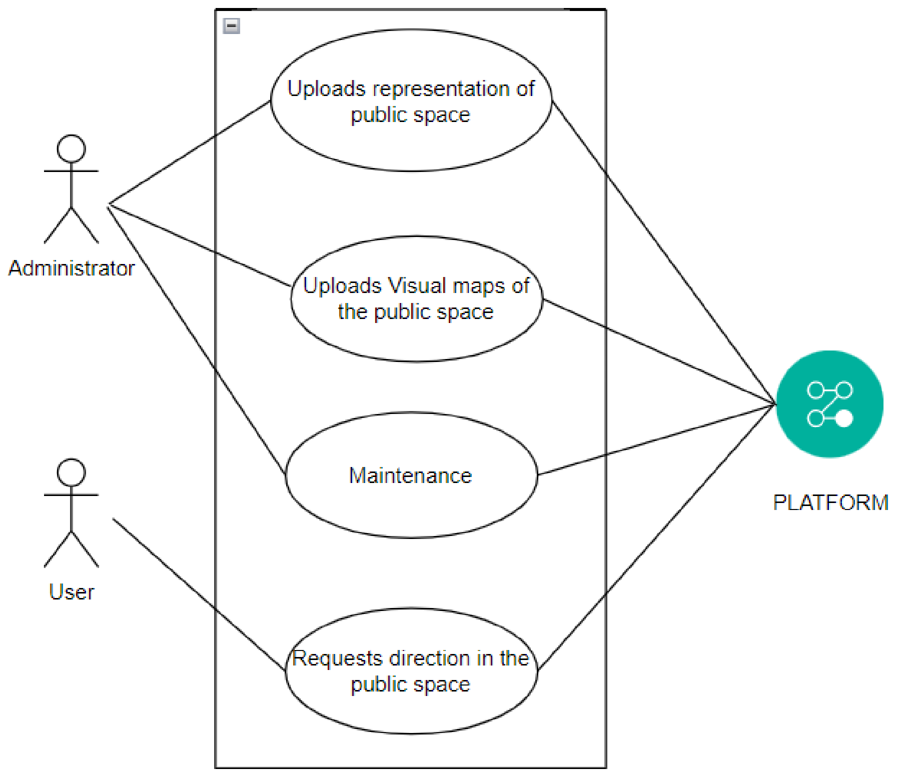

Uploads representation of public space
Administrator
Uploads Visual maps of the public space
Maintenance
User
Requests direction in the public space
PLATFORM

Figure 7. Administrator and User main operations with the platform.

5.5 Integrated System for Orientation and Accessibility

The software architecture we chose for our design is based on a Service Oriented Architecture (SOA) with the help of RESTful services. We chose to build the platform based on SOA because it follows our design explained above that can be summarized as service-centered. The building block of an SOA application is the service, which is a logical module that can fulfill the same business function in a repeatable and predictable manner. We used RESTful web services that can communicate with each other via an HTTP client. We chose to use RESTful web services, because we considered the fact that mobile devices are popular nowadays and these devices can easily communicate via an HTTP client, thus the entry point for the end users into the platform is less costly or unfamiliar.

5.5.1 Software architecture

The system is designed with 3 sections depicted in Figure 8. One section is used for user interaction with the platform. Another section is used for the development of platform RESTful web services which are consumed by the User Interaction with the platform. Another section is used for data sources that are consumed by the RESTful Webservices section. Each section communicates with the other to fulfill the need of the end-user. Without one of these three sections the value would not be provided to the user, or it would be provided with erroneous data.

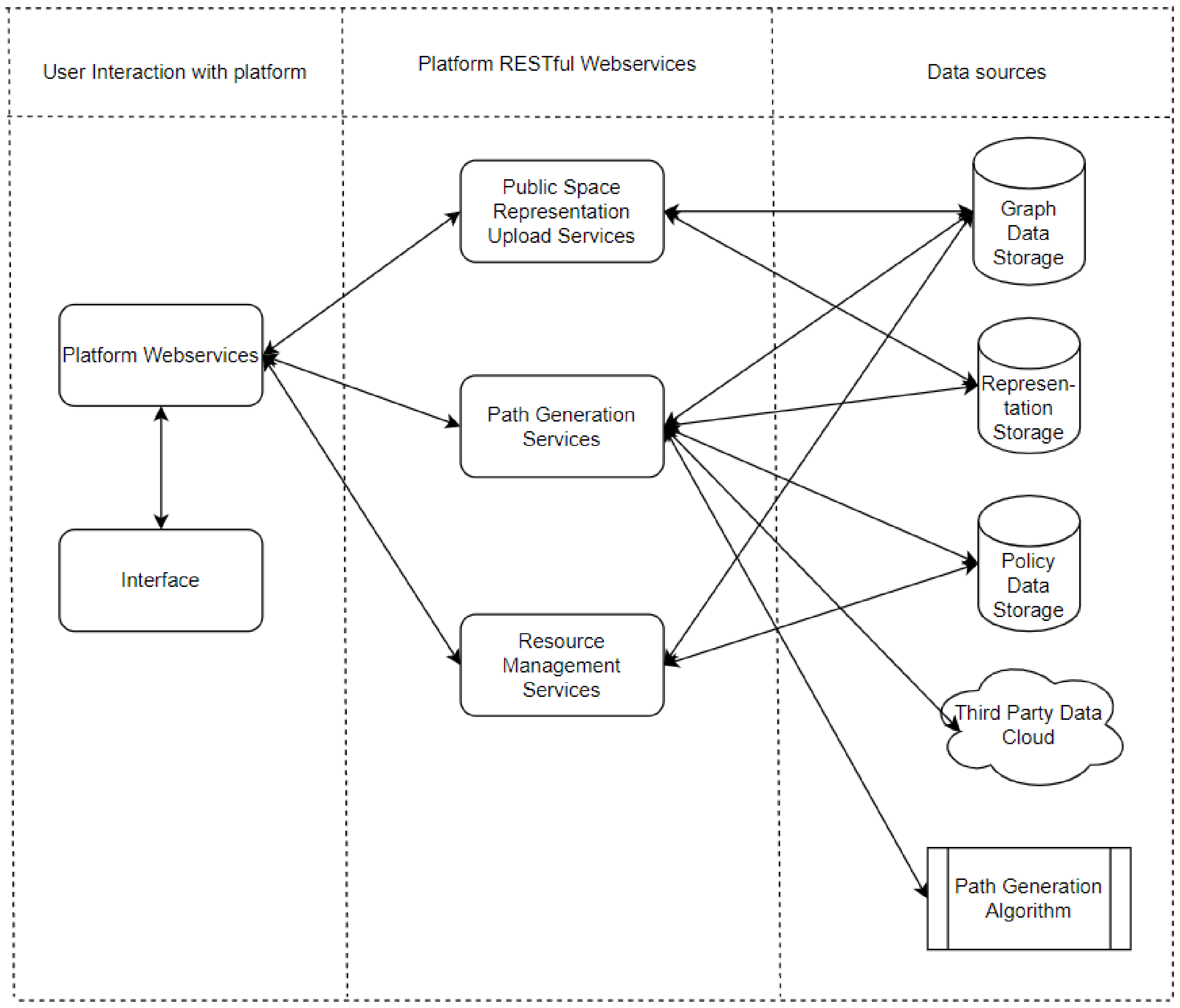

User Interaction with platform
Platform RESTful Webservices
Data sources
Public Space Representation Upload Services
Platform Webservices
Path Generation Services
Interface
Resource Management Services
Graph Data Storage
Represen-tation Storage
Policy Data Storage
Third Party Data Cloud
Path Generation Algorithm

Figure 8. System as Service Oriented Architecture.

The Interface helps the user and the administrator to reach the logic that the platform exposes. The interface is used for dual scope. The first scope provides data to the platform web services, and it consumes data from the platform web services. At this level, data is provided and consumed via HTTP calls.

Platform Webservices are the endpoints that communicate with the backend of the platform where the whole logic is created as RESTful web services. The platform Webservices are made to consume data from the Interface and provide it to the backend and to consume data from the backend and provide it to the Interface.

Public Space Representation Upload Services have the purpose of taking the representation data for the public space (Excel, image files) provided by the administrator in the bootstrap phase of the platform.

Path Generation Services take inputs from the user interface and in conjunction with the data stored in the Graph Data Storage, Representation Storage, and Policy Data Storage, Third Party

Data Cloud sends it to the Path Generation Algorithm to return the path between the inputs given by the user. Which is then pushed to the users as a response to their requests.

Resource Management Services take input from the administrator and update the information stored in the Data Source section, in particular to Graph Data Storage and to the Policy Data Storage.

Graph Storge is technologically independent, it can be stored on a Relational Data Base, or in an object in memory and disk.

Representation Storage deals with the storage of image files that represent building plans. Because these images must be sent to the user's interface for easy consumption, they must be shipped there with the help of the same web services the user is calling when sending requests.

Policy Data Storage is a storage of resources like the Graph Storage. These resources store specific logics that is used by the Path Generation Algorithm to update the base graph in memory.

Third Party Data Cloud are resources stored possibly at the creation of the policies. Path Generation Service will communicate to these Third-Party Data Cloud, to gather the information that is mapped by the policy which references it.

Path Generation Algorithm is used to create a route through the graph version provided by Path Generation Service (it could be the base graph in a 0-policy scenario or a weights-updated base graph in an n-policy scenario). This algorithm is a standard Dijkstra Algorithm that only needs as input traditional graph data and the points to create the route in between.

5.5.2 RESTful services

RESTful Services are used to transport data between the user and the platform. Each RESTful Service has an endpoint and a data standard that needs to be respected for the data to travel inside the system. The endpoint is of URL type and the standard is, generally in our proof-of-concept, a JSON format, but we have cases when the standard must be file format.

Public Space Visual Maps Upload Service technical definition is described in Table 1. The scope of this service is to take from the administrator an image that represents the public space, like a scale image of a building floor and store it on the platform.

Table 1. Public Space Visual Maps Upload Service Technical Overview

Method	POST
Endpoint	/mapUpload
Data Type	image type
Return	200 OK – for accepted image 400 Bad Request – for unaccepted image
Scope	Upload image files representing the public space to the platform.

Public Space Meta-Data Upload Service technical definition is described in Table 2. The aim of this service is to take from the administrator a document that describes the layout of the building to create the graph model for it, the end goal being to run a navigation algorithm on top of it.

Table 2. Public Space Meta-Data Upload Service Technical Overview

Method	POST
Endpoint	/metadataUpload
Data Type	text type
Return	200 OK – for accepted document 400 Bad Request – for unaccepted document
Scope	Upload meta-data document representing the public space to the platform.

Additional to these 2 core services for the Public Space Representation Upload Services exists an Extra Information Upload Service. Extra Information Upload has the scope to add data in a document format, which can be used to describe the public space but is not as important as creating the base graph model.

Table. 3 Extra Information Upload Service Technical Overview

Method	POST
Endpoint	/extrasUpload
Data Type	text type
Return	200 OK – for accepted document 400 Bad Request – for unaccepted document
Scope	Upload extra information document representing the additional data (i.e. locomotory difficulty) about the public space to the platform.

Path Generation Services are services that help the end user retrieve the most important aspect of this platform, which is the navigation and orientation information that can be used effectively in the real world. Path Generation Services contain 3 core services: Route Service, Directions Service, Map Service.

The aim of the Route Service is to take the input of the user under a JSON format that contains the start line, the finish line, and any optional policies opted by the end user. The technical definition of the Route Service is found in Table 4.

Table. 4 Route Service Technical Overview

Method	GET
Endpoint	/route
Data Type	JSON type
Return	200 OK – route between start line and finish line
Scope	Provide the user an ordered sequential route. Containing start line, ordered intermediary points, finish line.

Directions Service technical definition is described in Table 5. The aim of the Directions Service is to provide the user with syntactic directions to follow the ordered sequential route provided by the Route Service.

Table 5. Directions Service Technical Overview

Method	GET
Endpoint	/directions
Data Type	JSON type with Route Service return
Return	200 OK – route between start line and finish line with syntactic directions
Scope	Provide the user an ordered sequential route with syntactic directions.

Map Service technical definition is described in Table 6. The aim of the Map Service is to provide the user with visual maps that can be followed to reach the destination.

Table 6. Map Service Technical Overview

Method	GET
Endpoint	/map
Data Type	JSON type with Route Service return
Return	200 OK – visual map with route drawn between start line and finish line
Scope	Provide the user visual map with the route the user must take.

Resource Management Services are services used to maintain the resources of the platform, such as nodes of the graph or policies. Thus, Resource Management Services contain 2 core services, Graph Resource Management Service and Policy Management Service.

The technical definition of Graph Resource Management Service is described in Table 7. The input for this service is a node that needs maintenance.

Table 7. Graph Resource Management Service Technical Overview

Method	POST
Endpoint	/graph
Data Type	JSON type with resource to change
Return	200 OK – if the resource was changed 400 Bad Request – if the resource was not found
Scope	Provide the administrator the possibility to update small details regarding the graph model.

The technical definition of Policy Management Service is described in Table 8. The scope of the Policy Management Service is to add and change policies in the platform.

Table 8. Policy Management Service Technical Overview

Method	POST
Endpoint	/policy
Data Type	JSON type with policy to change or create
Return	200 OK – if the policy was changed or created 400 Bad Request – if the policy was not found
Scope	Provide the administrator the possibility to create and maintain policies in the platform.

5.5.3 Recommendation system

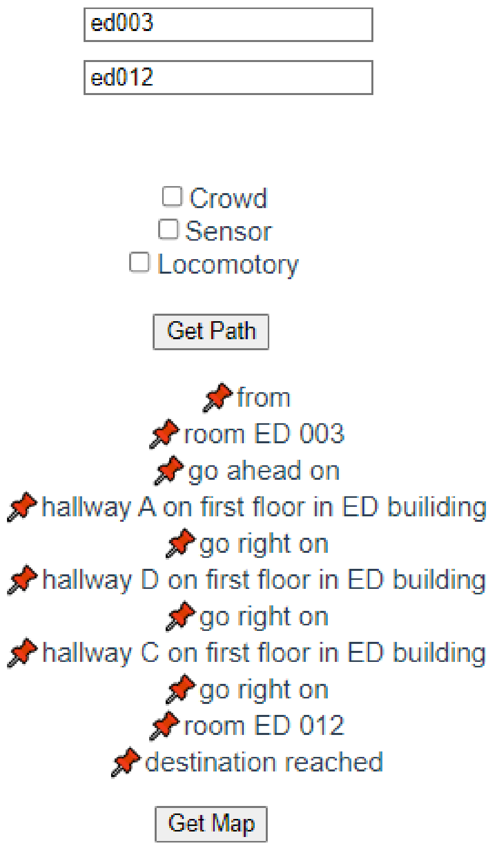
The end goal of the platform is to offer the end users the ability to help themselves with the decision-making process. The issue we are approaching in this thesis implies that an end-user is faced with the unknown. To give some examples, either the user does not know how to reach a destination, or what route is the safest. This aspect prevents the user to know upfront what her or his options are to orientate. Therefore, when the platform offers the output for the navigation service call, it has to present it in a clear, easy-to-follow, easy-to-trust manner. Trust is an impactful element in the experience with any system, especially one that supports decision-making and recommendations [53].

In this platform case, from the technical point of view, the service that the end user is most interested in is the navigation service. The navigation service runs on top of the underlying graph model of the public space. The navigation service also updates the weights of the underlying graph based on the policies user invokes. From the user experience point of view, the orientation could be overwhelming if she/he is new in that public space. For this issue, we use the visual maps the administrator used in the bootstrap process, visual maps that are connected to the nodes that the navigation service (via a recommendation system) uses and provides back to the user in the form of indications. On top of these visual maps, the recommendation system draws arrows from the start to the next intermediary position, then to the next intermediary position, and so on until the finish position.

If the visual map has extensive details like the name of the architectural features, the user can use them to find out that she/he is on the right path, also the user is syntactically given text instructions with details like “turn left”, “turn right”, “go ahead”, etc.

Textual recommendations. Figure 9 displays the user interface for a navigation service call from room ED003 to room ED012 in a proof-of-concept application created for our thesis scope for the ED building of the Faculty of Automatic Control and Computers from the University Politehnica of Bucharest. Figure 10 gives the output of the navigation service to the user in text format with syntactic details, for example “from room ED 003”, “go ahead on”, or “go right on”, etc. These details help the user to orientate and have a general direction for traversing the public space if some indications in the real world (like room names) are missing.

Visual recommendations. Figure 10 displays a route inside the ED building of the Faculty of Automatic Control and Computers from the University Politehnica of Bucharest that is provided to the end-user that requested details in Figure 9. This kind of visual map with red arrows are provided to the user when she/he calls the navigation service.



ed003

ed012

☐ Crowd
☐ Sensor
☐ Locomotory

from
room ED 003
go ahead on
hallway A on first floor in ED builiding
go right on
hallway D on first floor in ED building
go right on
hallway C on first floor in ED building
go right on
room ED 012
destination reached

Figure 9. Developed User Interface with navigation details.

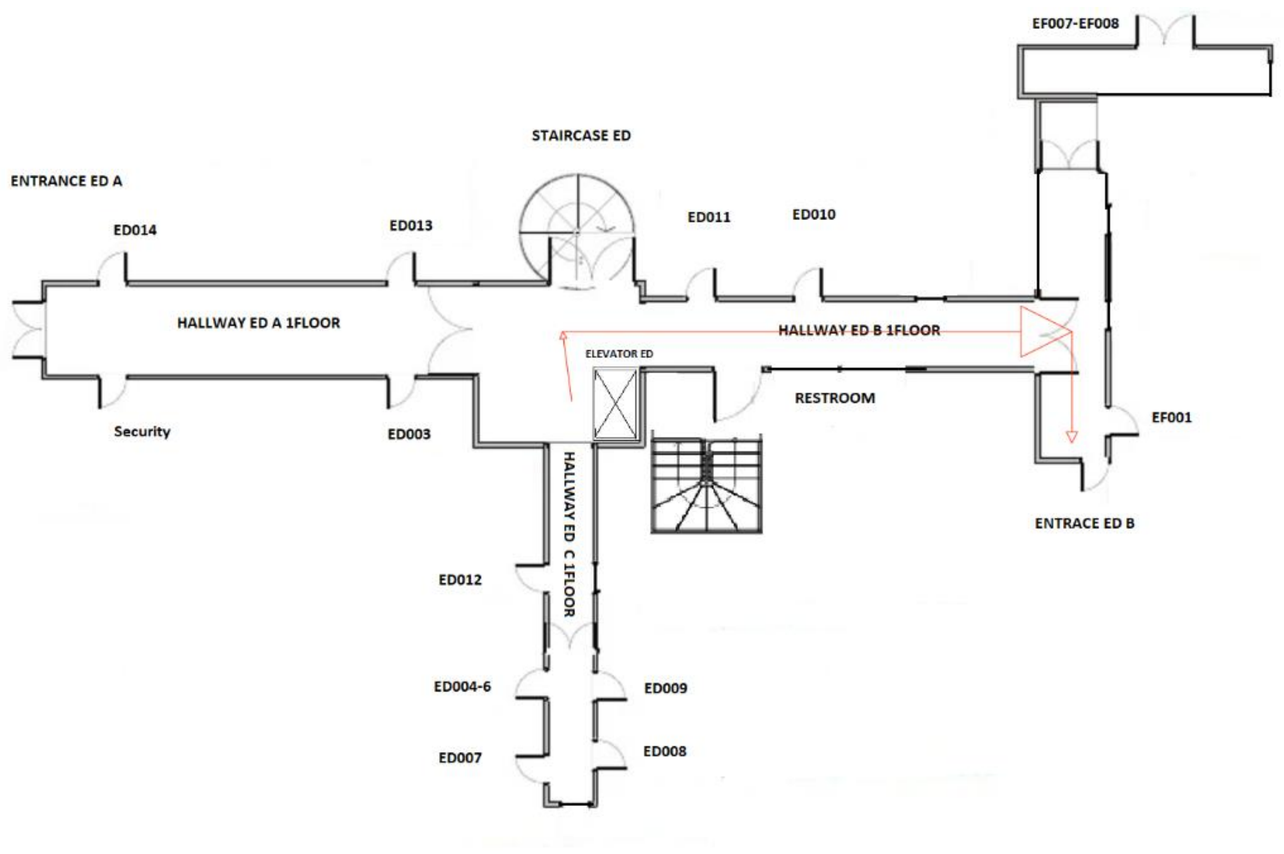


Figure 10. Developed visual map with route inside a real public space.

6 Evaluation and Validation of the Integrated Orientation and Accessibility Services

In this section we discuss how we conducted verification and validation for our platform. We discuss the evaluation method we used for this orientation and accessibility platform, discussing what computational resources we used and what cases we chose for the validation. We discuss results for the validation platform (set up for the Faculty of Automatic Control and Computers), the algorithms we used, performance and visual experience for the end user. We discuss the results and how we can assess what it would be different in other real-life-scenarios.

6.1 Evaluation Method

The development of these routing services relies on the Service-Oriented Architecture (SOA) and the webservices are based on a RESTful approach. The administrators can access an interface to map the real-world resource objects (that represent points of interest, i.e., physical locations that are important for the public space) to the computational ones inside the graph. We refer to these computational objects as resources, which are dynamic objects that can be interrogated and updated, but they represent a real-world object at the same time. It is first important to establish a general-purpose lightweight graph model, in order to have the option to easily adapt the resources in the future. This general-purpose graph operates with concepts such as resources and dependencies (nodes and vertexes, respectively). The resources need to have the possibility to evolve in time and provide information to the stakeholders. The configuration provides resources that support multiple properties, based on all stakeholders' needs (policies). We shifted from a standalone cost (which in the end would be a particular value stored in a database) to a cost integrated in the resource properties to store it with other properties a physical resource might have in a database, and to expose it as a resource on REST webservices. The connection data remain unaffected.

6.1.1 Performance evaluation

For the testing scope, we modeled a building in our faculty with the method described in Chapter 5. The resulted graph contains 107 nodes and 338 connections between them. The performance tests were executed for 1000, 2000, 3000, 4000, and 5000 Virtual Users (VUs); the selection of these numbers of users are further discussed in Section 6.3.2 related to Real-Life Loading Requirements. These VUs are spawned in the system with the total number desired for VUs, divided by 10 per second. For example, when testing the system with 1000 VUs, the spawn factor is 100 VUs/s, for 2000 VUs it is 200 VUs/s, and so on. After reaching the desired number of VUs in the system, the test continues for 60 s, and after that the VUs are evicted from the system; thus, the system does not receive requests any longer. The requests are sent to the system after a random time, between 0.5 s and 2 s.

The tests were run on a machine with Intel(R) Core (TM) i5-1035G1 CPU and 8 GB RAM, to evaluate whether it can deal with the stress from real users on a real-world architectural topology, thus offering a reliable service. The users call the services to compute the route between two points, with a random start and finish taken from the nodes list. This thesis presents the results for three of the test cases, executed with the setup described above:

- The 0-policy routing.
- The 1-policy routing for *Policy 2. Avoid crowded areas*.
- The 2-policy routing for *Policy 2. Avoid crowded areas* + *Policy 3. Avoid polluted areas.*

6.2 Results

6.2.1 Example for the locomotory policy applied for a university campus

This section describes our approach to solving accessibility and navigation needs for public spaces, considering possible locomotory constraints of users. It presents how to model a public space, how to make this model extensible, how to integrate the policy concept, and how to provide information to the end-user.

The test cases described in this thesis are indoor and intra-level navigation inside the ED building (between two floors, one on top of the other). The purpose is to verify that the services provide the user who does not opt for a locomotory policy with the shortest path toward the destination, and the user who opted for a locomotory policy with the safest path, including warning messages about the existing risks.

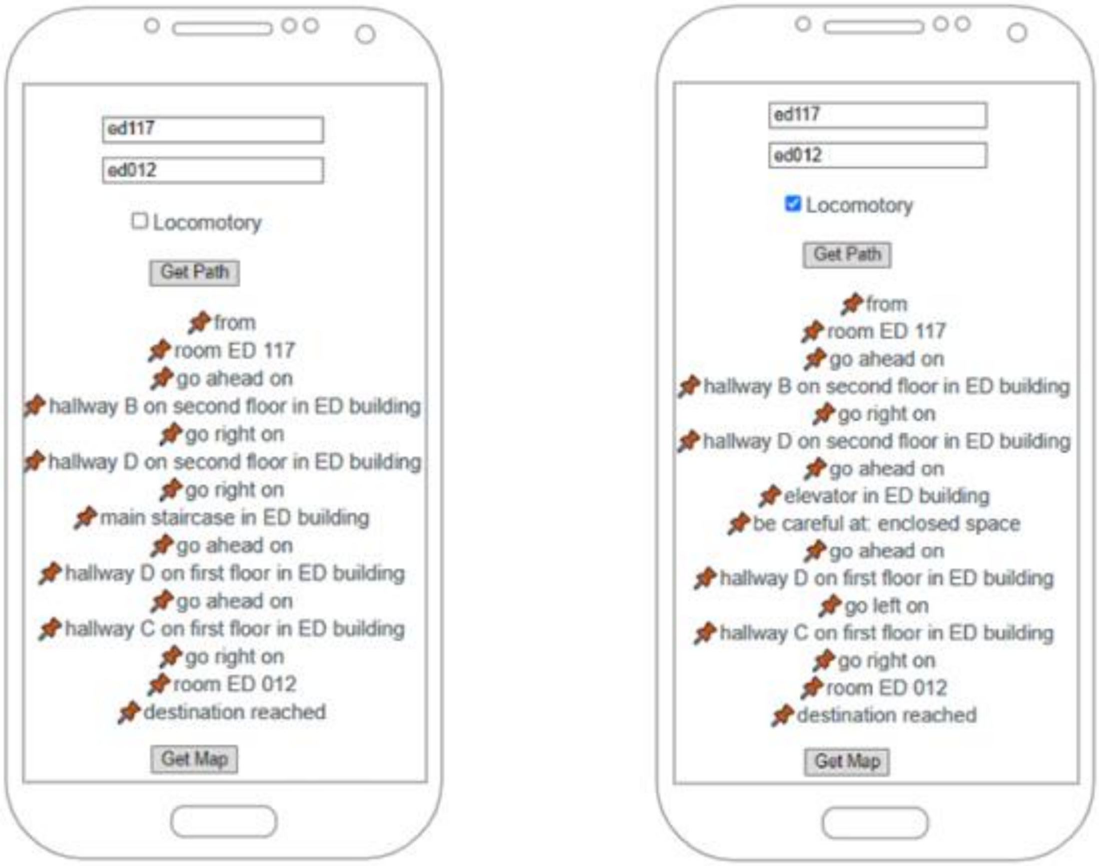
Let us consider two test cases, one with no policy selected and one with the locomotory policy. Both have the same start and destination points sent to the path generation algorithm, to obtain a comparison between the zero-policy and one-policy scenarios. The evaluation is based on:

- Criterion 1. The length of the recommended path
- Criterion 2. Whether or not there are edges that are impossible to access for a person with disabilities, e.g., stairs without a ramp, or doors that are too narrow for wheelchairs.
- Criterion 3. The difficulty level from the point of view of persons with locomotory impairments, due to ramps, heavy doors, or narrow spaces.

The chosen start point in our tests is room ED117, situated on the first floor of the ED building, connected to Hallway B from the first floor. The chosen destination point in our test is room ED012, situated on the ground floor of the ED building, connected to Hallway C on the ground floor. Figure 11 displays the output of the path generation algorithm without the option for the locomotory policy, and the output with the option for the locomotory policy.

Interpreting Figure 11, one observes that the directions provided to the user suggest him/her to take the shortest path in the graph, corresponding in reality to the main staircase in ED, which is closer to ED117. This solution responds to Criterion 1, but it fails to fulfill Criterion 2, therefore it is not suitable for a pedestrian with locomotory impairments. Then, the navigation continues to the destination on the ground floor of the building.

Interpreting Figure 11, note that the directions provided are beneficial for a person with locomotory impairments; they guide the pedestrian on the safest way (Criterion 3) and not on the shortest one (Criterion 1). In this case, the service returns a path including an elevator, which is more suited to the person with locomotory issues, and Criterion 2 is fulfilled.



ed117
ed012
☐ Locomotory

from
room ED 117
go ahead on
hallway B on second floor in ED building
go right on
hallway D on second floor in ED building
go right on
main staircase in ED building
go ahead on
hallway D on first floor in ED building
go ahead on
hallway C on first floor in ED building
go right on
room ED 012
destination reached

ed117
ed012
☑ Locomotory

from
room ED 117
go ahead on
hallway B on second floor in ED building
go right on
hallway D on second floor in ED building
go ahead on
elevator in ED building
be careful at: enclosed space
go ahead on
hallway D on first floor in ED building
go left on
hallway C on first floor in ED building
go right on
room ED 012
destination reached

Figure 11. The path between ED117 and ED012. Comparison between 0-policy and locomotory policy.

Nonetheless, information about the risks is provided; the risk of the elevator is “close space”, as defined by the locomotory policy model.

6.2.2 Performance results

To better visualize how the services perform under different policies, Figure 12 depicts a graph that shows the average response time against VUs for the three test cases mentioned. One can notice that the best average response time curve belongs to the 0-policy test; for this one, the services use the least computational effort. For the cases with 1-policy and 2-policy, the average response time curves become higher with the number of users (e.g., for 4000 VUs and 5000 VUs); however, even for this load, the differences between 0-policy, 1-policy, and 2-policy are under 1500ms, keeping the performance still acceptable for the end user.

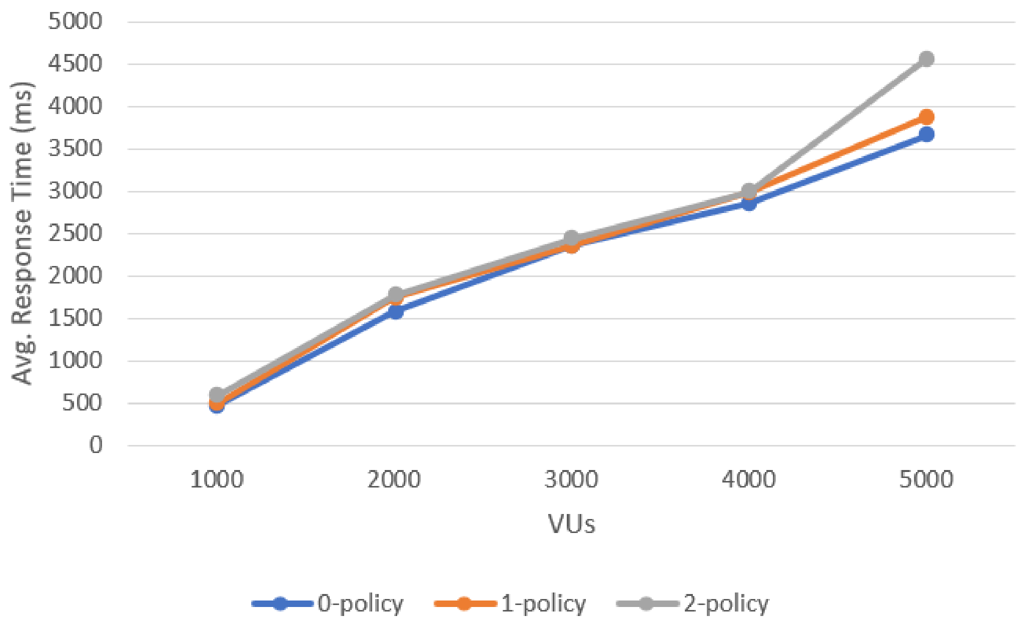


Figure 12. Average response times for 0-policy vs 1-policy vs 2-policy.

Another factor that directly impacts the end user is the failure rate of the routing services. In Figure 13, one can observe failure rates (%) against the number of VUs calling the routing services for the three test cases considered. The failure rates are in general under 4%, and higher for more VUs. For different numbers of policies applied, the values of the failure rate are rather close to each other, proving that the system is built to handle all cases in the same manner, without a strong impact from VUs and the computational effort involved.

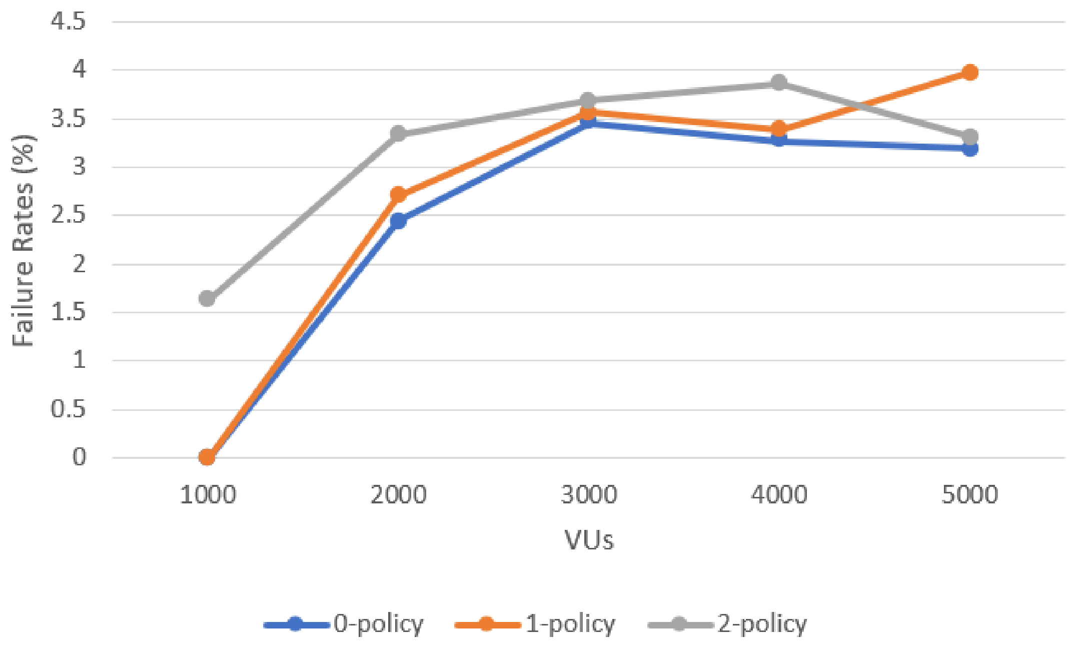


Figure 13. Failure rates for 0-policy vs 1-policy vs 2-policy.

6.3 Discussion

6.3.1 Results interpretation

To analyze if the obtained results indicate a good user experience of the routing services, let us analyze Figure 12 and Figure 13. We notice that there are differences between 0-policy, 1-policy, and 2-policy testing results but, in each case, the difference between 2000, 3000, 4000, and 5000 VUs is negligible regarding the response time; the failure rate is 0% for 1000 VUs, in the 0-policy and 1-policy scenarios, and under 4% for a larger number of users in all scenarios. The requests per second are substantially similar for all three test cases, assessing that the routing service can handle these sorts of loads reliably. The response time indicator can be assessed in comparison with a typical web page surfing, where 1 to 2 s to load the content represent the user preference, and maximum 5 s to load the page is also considered acceptable, according to [54]. Our services are more computing extensive than a simple web page retrieval, but they remain within the same preferred values. A more modular software architecture would offer the possibility to scale different modules. Scaling and distribution can offer better performance results, especially in production [55]; moreover, the way to store the underlying graph impacts how one retrieves data from it.

6.3.2 Real-life loading requirements

The number of virtual users used for testing was deducted from studying real-life requirements for various universities, in order to test the application as closely as possible to reality.

Real-life requirements for a university campus may result from studying the admission rates in each year from some well-known universities in the world. University of Cambridge had 3528 newcomers in 2019, 3465 in 2018, and 3480 in 2017, according to [56]. University of Oxford admitted 3280 new students in 2019, 3309 in 2018, and 3270 in 2017 [57]. University College London had 9145 newcomers in 2019, 6110 in 2018, and 5885 in 2017. Eidgenössische Technische Hochschule Zürich had 3357 newcomers in 2020 [58]. Imperial College of London had 3045 freshmen in 2019, 2845 in 2018, and 2795 in 2017 [59]. University of Edinburgh received 7344 newcomers in 2019, 6346 in 2018, and 6221 in 2017 [60].

Then, let us also distribute the total number of students against the total possible number of years needed to obtain a bachelor’s degree. In this manner, one can extrapolate that from a university that has approximately 25,000 students, such as the one used for our tests, on a median span of 4 years for bachelor’s degree, 6000 new students come each year.

Considering that not everyone accesses the routing services at the same time, the selection of loads used in our tests, i.e., the number of Virtual Users, follow the reality as closely as possible. For a different kind of public space, such as a transportation hub, these requirements may differ.

7 Conclusion

To conclude, the problem of pedestrian routing generally covers the following aspects: how to model the graph that corresponds to the real-world space, how to traverse the graph according to various cost functions, and how to consider other specific concerns for certain categories of people. It needs to map abstract mathematical knowledge to the concrete architectural elements, the circumstances of the moment, and the different pedestrians' needs. This thesis proposed a service-oriented design that considers all these aspects for delivering customized routing directions in a public space, by choosing from a set of policies. Two of them are driven by the community, by considering the votes of other pedestrians walking in the same public space, or by avoiding the places frequented by many other people. The other two are driven by data originating from multiple sources, such as sensors for measuring air quality or external weather monitoring services. In addition, a supplementary policy considers special needs for persons with reduced mobility.

When adding supplementary policies, the tests for up to 5000 Virtual Users showed a small difference in response times (less than 1 s) and also in failure rates (less than 0.8%)—even if the execution was not carried out in a high-performance environment. Therefore, providing such software services, including the flexibility in choosing specific combinations of policies, is accessible when managing a public space. The loading tests proved that the routing services can support high loads of requests in a public space infrastructure modelled with a medium-sized granularity.

Our recommendation when implementing this kind of platform is to start with the most suited graph model that represents the real public space. The representation granularity matters, and it is better to conduct an analysis beforehand, to determine what the specific points of interest are and to set the edges connecting the points of approximately the same size as in reality. Furthermore, it is recommended to determine the types of persons who use the public space. Lastly, our recommendations for assuring the right performance in very large public spaces are in the direction of an appropriate modularization strategy, deployment, and scaling, to offer even better performance and user experience under high loads.

7.1 Summary of the Original Contributions

The original contributions presented in this thesis fall in the categories of navigation and orientation services for smart public spaces. To summarize our original contributions, we mention the questions that determined us to develop them.

A high-level view of personal contributions presented in this thesis are:

- Analysis of the state of the art for navigation and orientation software
- Definition of a set of scenarios for navigation and orientation inside a university campus
- Propose an approach for pedestrian routing based on multiple policies (crowd-avoiding, locomotory constraints, community votes, avoid polluted areas, avoid unfavorable weather conditions)

- Design the algorithm for policy-based graph routing
- Create REST services to manage a user-centric navigation platform
- Integrate external information to be integrated with the recommended navigation for the end user
- Elaborate a model for an example of the university campus
- Configure the algorithms for accessibility
- Develop a service-oriented platform that integrates the services to manipulate models, maps, and flexible navigation solutions
- Validate the proposed approach for a university campus
- Evaluate the performance
- Discuss the opportunity of applying this approach based on the study of existing campuses needs

7.2 List of Publications

1. **Ioan Damian**, Anca Daniela Ionita, Restful Services for Orientation and Accessibility in a University Campus; University Politehnica of Bucharest Scientific Bulletin (2021)– Series C Rank: Q4 (Electrical Engineering and Computer Science); Impact factor 0.37; **WOS:000628640200003**

2. **Ioan Damian**, Marian Lacatusu, Anca Daniela Ionita, Florin Lacatusu, Software Services to Support Faculty Management in Times of Pandemic, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943

3. **Ioan Damian**, Anca Daniela Ionita, Silvia Oana Anton, Community - and Data-Driven Services for Multi-Policy Pedestrian Routing, *Sensors* (2022) Rank**: Q2** (Engineering, Electrical & Electronic), Impact factor: 3.576, **WOS: 000817661800001**

4. Florin Lacatusu, **Ioan Damian**, Anca Daniela Ionita, Marian Lacatusu, Smart Building Manager Software in Cloud, University Politehnica of Bucharest Scientific Bulletin (2021)– Series C Rank: Q4 (Electrical Engineering and Computer Science); Impact factor 0.37; **WOS: 000741473700003**

5. Marian Lacatusu, Florin Lacatusu, **Ioan Damian**, Anca Daniela Ionita, Multicloud Deployment to Support Remote Learning, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4601-4606, IATED Digital Library, doi: 10.21125/inted.2021.0936

6. **Ioan Damian**, Marian Lacatusu, Florin Lacatusu, Anca Daniela Ionita, Web Services for Guiding Persons with Locomotor Impairments in Public Spaces, *2022 26th International Conference on System Theory, Control and Computing* (ICSTCC), 2022, pp.639-644, IEEE, INSPEC, doi:10.1109/ICSTCC55426.2022.9931861, **WOS: 000889980600107**

7. Marian Lacatusu, Anca Daniela Ionita, Florin Lacatusu and **Ioan Damian**, "Decision support for multicloud deployment of a modeling environment," 2022 IEEE 18th

International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951

8. Florin Lacatusu, Anca Daniela Ionita, Marian Lacatusu, **Ioan Damian** and Daniela Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," 2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978

7.3 Future Perspectives

To talk about future perspectives, we must start with the original contributions presented in this thesis. These are the points we wish to evolve in the future, and we see it is possible at this moment to do so.

The small footprint application is a strong point for our thesis, but the real benefit can only be observed in production. This means that a platform with our design must be deployed to a Cloud solution, thus a Cloud deployment strategy must be formulated.

A core feature of the platform remains the model related to the graph-like representation. This feature is not meant to be changed for reasons we discussed above in the modelling chapters. A future development is to represent the public space with components from CAD, for which other modelling modules must be implemented. What could change is to add more navigation algorithms, which may be general-purpose so they could be integrated with the policies of the platform.

The policies aspect of the platform must not be changed, it is a core feature and the only way it could integrate multiple interests of the users inside the platform. What could be extended concerns the services that model the interaction with the reality and the storage and use inside the platform.

Another future perspective is to develop system-based push-notification that can help stakeholders be more aware if a certain portion of the public space is no longer accessible, for example, if a hallway was closed during the day, and maybe in the morning was still open.

A future perspective regarding the experience for the users is to integrate the platform with smart device platforms. Smart devices like smartwatches are still components that could benefit from technical advancement in the next years, and we believe that integration to these is an important aspect, since they are particularly popular amongst younger generations.

References

1. Ioan Damian, Anca Daniela Ionita, Restful Services for Orientation and Accessibility in a University Campus; University Politehnica of Bucharest Scientific Bulletin (2021)– Series C (Electrical Engineering and Computer Science)
2. Florin Lacatusu, Ioan Damian, Anca Daniela Ionita, Marian Lacatusu, Smart Building Manager Software in Cloud, University Politehnica of Bucharest Scientific Bulletin (2021)– Series C (Electrical Engineering and Computer Science)
3. R. Drober, Campus Landscape – Functions, Forms, Features, John Wiley & Sons, 2000, pp. 10-60
4. M. Hamblen, "Just what is a smart City?," 2015. [Online]. Available: https://www.computerworld.com/article/2986403/just-what-is-a-smart-city.html. [Accessed 17 December 2020]
5. T. Nam et al, "Conceptualizing smart city with dimensions of technology, people, and institutions" in 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times, 2011.
6. S. Bendre et al, "Event Based Campus Navigation System" International Journal of Computer Science and Information Technologies, vol. 7, no. 1, pp. 462-464, 2016.
7. D. Merode et al, "Smart Campus based on iBeacon Technology" in International Symposium on Ambient Inteligence end Embedded System, 2015.
8. J. Yim et al, "Design and Implementation of a Smart Campus guide Android App", in Appl Math Inf Sci, vol. 8, no. 1, pp.47-53, 2014.
9. L. Kwok, "A vision for the development of i-campus," Smart Learning Environments, vol. 2, pp. 1-12, 2015.
10. Calinescu, R.; Cámara, J.; Paterson, C. Socio-cyber-physical systems: Models, opportunities, open challenges. In Proceedings of the 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems, SEsCPS@ICSE, Montreal, QC, Canada, 28 May 2019; pp. 2–6.
11. Martensen, B. The Perception-Action Hierarchy and its Implementation Using Binons (Binary Neurons). In Proceedings of the 10th Annual International Conference on Biologically Inspired Cognitive Architectures (BICA), Seattle, WA, USA, 15–18 August 2019; pp. 489–500.
12. Ebrahim, A. People-Centric Smart Campus. In Proceedings of the 2021 International Conference on Transport and Smart Cities (ICoTSC 2021), Frankfurt, Germany, 17 December 2021; pp. 264–267.

13. Sánchez, C.M.; Zella, M.; Capitán, J.; Marrón, P.J. From Perception to Navigation in Environments with Persons: An Indoor Evaluation of the State of the Art. Sensors 2022, 22, 1191.

14. Coito, T.; Firme, B.; Martins, M.S.E.; Vieira, S.M.; Figueiredo, J.; Sousa, J.M.C. Intelligent Sensors for Real-Time Decision-Making. Automation 2021, 2, 62–82.

15. Ioan Damian, Marian Lacatusu, Anca Daniela Ionita, Florin Lacatusu, Software Services to Support Faculty Management in Times of Pandemic, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943

16. Ioan Damian, Anca Daniela Ionita, Silvia Oana Anton, Community- and Data-Driven Services for Multi-Policy Pedestrian Routing. *Sensors*. 2022, 22(12):4515. doi: https://doi.org/10.3390/s22124515

17. Ioan Damian, Marian Lacatusu, Florin Lacatusu, Anca Daniela Ionita, Web Services for Guiding Persons with Locomotor Impairments in Public Spaces, *2022 26th International Conference on System Theory, Control and Computing* (ICSTCC), 2022, pp.639-644, IEEE, INSPEC, doi:10.1109/ICSTCC55426.2022.9931861

18. Ntakolia, C.; Iakovidis, D.K. A swarm intelligence graph-based pathfinding algorithm (SIGPA) for multi-objective route planning. Comput. Oper. Res. 2021, 133, 105358.

19. Wang, D.; Chen, X.; Huang, H. A graph theory-based approach to route location in railway interlocking. Comput. Ind. Eng. 2013, 66, 791–799.

20. Wu, G.; Atilla, I.; Tahsin, T.; Terziev, M.; Wang, L.C. Long-voyage route planning method based on multi-scale visibility graph for autonomous ships. Ocean Eng. 2021, 219, 108242.

21. Kielar, P.M.; Biedermann, D.H.; Kneidl, A.; Borrmann, A. A unified pedestrian routing model for graph-based wayfinding built on cognitive principles. Transp. A Transp. Sci. 2018, 14, 406–432.

22. Zhao, J.; Fang, Z. Research on Campus Bike Path Planning Scheme Evaluation Based on TOPSIS Method: Wei'shui Campus Bike Path Planning as an Example. Procedia Eng. 2016, 137, 858–866.

23. Najjar, A.B.; Al-Issa, A.R.; Hosny, M. Dynamic indoor path planning for the visually impaired. J. King Saud Univ. Comput. Inf. Sci. 2022, in press.

24. Zhou, Z.; Weibel, R.; Richter, K.-F.; Huang, H. HiVG: A hierarchical indoor visibility-based graph for navigation guidance in multi-storey buildings. Comput. Environ. Urban Syst. 2022, 93, 101751.

25. WRLD Environments Cases. Indoor Navigation. Benefits and Use Cases. Available online: https://www.wrld3d.com/blog/indoor-navigation/ (accessed on 20 March 2021).

26. Wang, J.; Hu, A.; Liu, C.; Li, X. A Floor-Map-Aided WiFi/Pseudo-Odometry Integration Algorithm for an Indoor Positioning System. Sensors 2015, 15, 7096–7124.

27. Prandi, C.; Delnevo, G.; Salomoni, P.; Mirri, S. On Supporting University Communities in Indoor Wayfinding: An Inclusive Design Approach. Sensors 2021, 21, 3134.

28. Ferreira, J.C.; Resende, R.; Martinho, S. Beacons and BIM Models for Indoor Guidance and Location. Sensors 2018, 18, 4374.

29. Mataloto, B.; Ferreira, J.C.; Resende, R.; Moura, R.; Luís, S. BIM in People2People and Things2People Interactive Process. Sensors 2020, 20, 2982.

30. Lin, Y.-B.; Chou, S.-L. SpecTalk: Conforming IoT Implementations to Sensor Specifications. Sensors 2021, 21, 5260.

31. Fortes, S.; Hidalgo-Triana, N.; Sánchez-La-Chica, J.-M.; García-Ceballos, M.-L.; Cantizani-Estepa, J.; Pérez-Latorre, A.-V.; Baena, E.; Pineda, A.; Barrios-Corpa, J.; García-Marín, A. Smart Tree: An Architectural, Greening and ICT Multidisciplinary Approach to Smart Campus Environments. Sensors 2021, 21, 7202.

32. Tseng, K.-H.; Chung, M.-Y.; Chen, L.-H.; Chang, P.-Y. Green Smart Campus Monitoring and Detection Using LoRa. Sensors 2021, 21, 6582.

33. Dasler, P.; Malik, S.; Mauriello, M.L. "Just Follow the Lights": A Ubiquitous Framework for Low-Cost, Mixed Fidelity Navigation in Indoor Built Environments. Int. J. Hum.-Comput. Stud. 2021, 155, 102692.

34. Chou, T.-L.; ChanLin, L.-J. Augmented Reality Smartphone Environment Orientation Application: A Case Study of the Fu-Jen University Mobile Campus Touring System. Procedia-Soc. Behav. Sci. 2012, 46, 410–416.

35. Li, C.-Y.; Yin, J. A pedestrian-based model for simulating COVID-19 transmission on college campus. Transp. A Transp. Sci. 2022, 1–25.

36. Zhao, M.; Zhou, C.; Chan, T.; Tu, C.; Liu, Y.; Yu, M. Assessment of COVID-19 aerosol transmission in a university campus food environment using a numerical method. Geosci. Front. 2022, 101353.

37. Muller, K.; Muller, P.A. Mathematical modelling of the spread of COVID-19 on a university campus. Infect. Dis. Model. 2021, 6, 1025–1045.

38. Bartolucci, A.; Templeton, A.; Bernardini, G. How distant? An experimental analysis of students' COVID-19 exposure and physical distancing in university buildings. Int. J. Disaster Risk Reduct. 2022, 70, 102752.

39. von Seidlein, L.; Alabaster, G.; Deen, J.; Knudsen, J. Crowding has consequences: Prevention and management of COVID-19 in informal urban settlements. Build. Environ. 2020, 188, 107472.

40. Polanco, L.D.; Siller, M. Crowd management COVID-19. Annu. Rev. Control 2021, 52, 465–478.

41. Geneletti, D.; Cortinovis, C.; Zardo, L. Simulating crowding of urban green areas to manage access during lockdowns. Landsc. Urban Plan. 2022, 219, 104319.

42. T. Bidila, R.N. Pietraru, A.D Ionita, A. Olteanu "Monitor Indoor Air Quality to Assess the Risk of COVID-19 Transmission" 23rd International Conference on Control Systems and Computer Science Technologies, CSCS 2021 ; : 356-361, 2021.

43. Chen JW, Zhang J. "Comparing Text-based and Graphic User Interfaces for novice and expert users." AMIA Annu Symp Proc. 2007 Oct

44. Julian Keil, Dennis Edler, Lars Kuchinke, Frank Dickmann "Effects of visual map complexity on the attentional processing of landmarks". PLoS One 2020 March

45. Landman K. "Inclusive public space: rethinking practices of mitigation, adaptation, and transformation". Urban Des Int. 2020

46. Van Hoogdalem, H., Van Der Voordt, T. J. M., & Van Wegen, H.B.R. "Comparative floorplan-analysis as a means to develop design guidelines." Journal of Environmental Psychology, 1985

47. Yu Li, Weijia Li, Yingying Yang, Qi Wang "Feedback and Direction Sources Influence Navigation Decision Making on Experienced Routes". Front. Psychol., September 2019

48. Hermann Bulf, Maria Dolores de Hevia, Valeria Gariboldi, Viola Macchi Cassia "Infants learn better from left to right: a directional bias in infants' sequence learning". Sci Rep 2017

49. Chang, N. S., & Fu, K. S. "A relational database system for images." Lecture Notes in Computer Science, 1980

50. Mark Wallis, Frans Henskens, Michael Hannaford "A Distributed Content Storage Model for Web Applications". Conf. INTERNET 2010

51. da Fontoura Costa, L.; Oliveira, O.N., Jr.; Travieso, G.; Rodrigues, F.A.; Boas, P.R.V.; Antiqueira, L. Analyzing and Modeling Real-World Phenomena with Complex Networks: A Survey of Applications. Adv. Phys. 2011, 60, 329–412.

52. Bauer, V., & Heinemann, L. "Understanding API Usage to Support Informed Decision Making in Software Maintenance." 2012 16th European Conference on Software Maintenance and Reengineering

53. Nunes, I., Jannach, D. A systematic review and taxonomy of explanations in decision support and recommender systems. User Model User-Adap Inter 27, 393–444 (2017)

54. Bartuskova, A.; Krejcar, O. Loading Speed of Modern Websites and Reliability of Online Speed Test Services. In Computational Collective Intelligence. Lecture Notes in Computer

Science; Núñez, M., Nguyen, N., Camacho, D., Trawiński, B., Eds.; Springer: Cham, Switzerland, 2015; Volume 9330.

55. Alshinina, R.; Elleithy, K. Performance and Challenges of Service-Oriented Architecture for Wireless Sensor Networks. Sensors 2017, 17, 536.

56. UCAS. End of Cycle 2016 Data Resources DR4_001_02 Main Scheme Acceptances by Provider. Available online: https://www.ucas.com/data-and-analysis/ucas-undergraduate-releases/ucas-undergraduate-end-cycle-data-resources/applicants-and-acceptances-universities-and-colleges-2016

57. University of Oxford. Annual Admissions Statistical Report: May 2020. Available online: https://www.ox.ac.uk/sites/files/oxford/AnnualAdmissionsStatisticalReport2021.pdf

58. UCAS. 2020 Entry Provider-Level End of Cycle Data Resources. Available online: https://www.ucas.com/data-and-analysis/undergraduate-statistics-and-reports/ucas-undergraduate-end-cycle-data-resources-2020/2020-entry-provider-level-end-cycle-data-resources

59. ETH. ETH in Figures. Available online: https://ethz.ch/en/the-eth-zurich/portrait/eth-zurich-in-figures.html

60. University of Edinburgh. Undergraduate Admissions Statistics. Available online: https://www.ed.ac.uk/student-recruitment/admissions-advice/admissions-statistics