Towards SUMP with M-Learning

Game and Media Technologies Master Thesis project

Konstantinos Petridis
S.N.: 6570399

Universiteit Utrecht

Supervisor: Prof. dr. R.C.Veltkamp
Second examiner: Dr. ing. S.C.J. Bakkes
University Of Utrecht
Netherlands, July 2022

Preface

This paper describes the steps taken for completing a Master Thesis project, for the Game and Media Technologies Master studies of the University of Utrecht. The project is about the creation of an educational tool for Urban Planning students, so as to assist them in the learning process and instill the principles of Sustainable Urban Mobility Planning. The project consists of a mobile application, as well as a standalone pathfinding service. The collaborators of this project are the RIVM and staff of the University of Utrecht. The main supervisor of this project is Prof. Dr. Remco C. Veltkamp.

1 Introduction

Since the time that a vast majority of people moved towards the urban centers (early 1800), the need for transportation through the growing towns and cities has been and still is on the center of focus for both governing authorities and urban planners as well. Early forms of public transportation as we know it have taken place in England in 1825, with George Stephenson, who built the first steam railway (*Locomotion*) in the world by connecting Stockton and Darlington [26]. Another example would serve the first electric streetcar that was built in Richmond, Virginia in 1888.

In the beginning and in the latest years, the main issues that have to be faced from urban mobility planners, are mainly to assist as much people with their transporting needs as possible. From a recent research, it has been noticed that almost 1.3 million people use the Dutch railway, and especially the central station of Utrecht had an average of **194,385** travelers per working day, 5% more than last year [22]. With the passing of years, even more people apart from residents use the public transport, so the mobility planning has adapted and refactored with these new changes in mind.

In the latest years, significant environmental changes have taken place. Especially, large emmisions of carbon dioxide have proved to be a great problem for residents and for the quality of life in urban centers, in addition to thinning the ozone layer even further. Comparing to past times, a good deal of concepts have been introduced in the field of urban planning, concerning the aforementioned reasons. A good example would be Shanghai, in which electric buses are used for the daily transport, which has reduced the CO_2 emissions and assisted with the noise pollution reduction as well [17].

With all these concepts in mind, it can be understood that urban mobility or **S**ustainable **U**rban **M**obility **P**lanning, (SUMP for short), is a really important concept for an optimal quality of life in an urban area. The residents' needs for fast, flexible and secure transportation in combination with environmental protection are ideals that future urban planning experts and specialists should stick to.

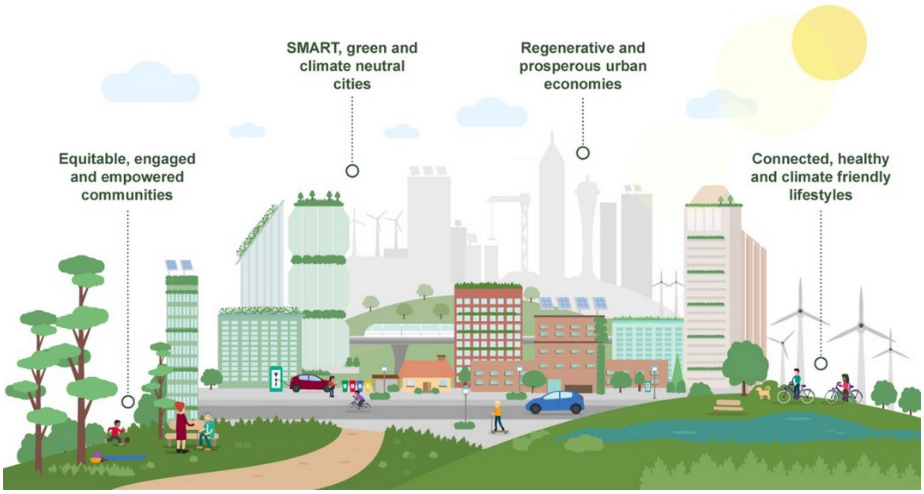


Figure 1: Main principles of Urban Sustainability. Source: https://www.rethinkingthefuture.com/sustainable-architecture/a4249-what-are-the-principles-of-urban-sustainability/

1.1 UrbanSCOPE and SUMP

UrbanSCOPE is a project that belongs to the **Erasmus+** programme, which is funded by the European Union and aims to support the education and training of a wide variety of individuals, apart from students [10]. Specifically, the motivation behind this project lies on the importance of including the core **SUMP** concepts and principles in relative studies, such as architecture and spatial planning. This should be done for the reason that students of these studies will be able to own and understand them from an early stage. Apart from students, though, what additionally motivates this project, is to bring these concepts closer to residents of urban areas, in order to help them understand sustainable mobility, and also actively participate in changes concerning the application and possible improvement in their area. In such manner and by taking into consideration the fact that UrbanScope aims *"To change attitudes and behaviours of city inhabitants in favour of sustainable mobility "*, this project's outcome will belong in the category of persuasive technologies. A persuasive technology system is one that is designed with a purpose to change the behavioral patterns of users through persuasion and social influence. To sum up, the objective of this research project is to create a persuasive system, specifically a mobile application which will provide assistance to urban mobility students, both in their educational practical work and also in convincing them to select more sustainable mobility practices (i.e. public transport, biking) [28].

For that reason, the primary aim of UrbanSCOPE is to provide a hands-on approach to students, by implementing a learning tool, so as not only to improve

the teaching quality of the specific studies but also to bring students closer to the basic concepts of sustainable development. A secondary aim of UrbanSCOPE is to further adapt this learning tool and transform it into a serious/persuasive game, suitable for children.

1.2 Sustainable Urban Mobility Planning

In this subsection, definitions of the **SUMP** concept will be provided, along with the most basic principles that comprise this concept. According to Ortuzar's article, *sustainability* is defined as a policy that promotes development, which aims to balance the relationship between humanity and nature on three aspects: *social inclusion*, *economic development* and *environmental balance*, which are the three pillars of sustainability [15].

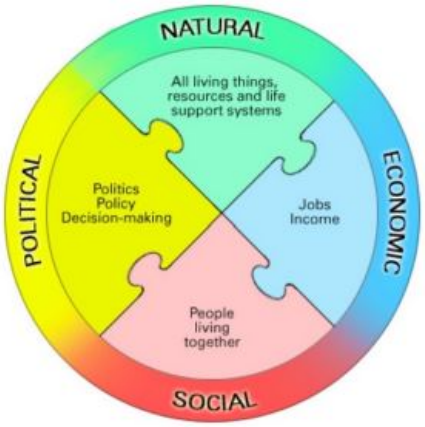


Figure 2: The four pillars of sustainability with the integration of the political component, according to Ortuzar.

As far as urban mobility is concerned, the author provides some interesting observations from reports of the Texas AM Transportation Institute, for 70 cities in the United States of America. It has been observed that drivers spend 60 hours in traffic jams which is twice as much, compared to 10 years ago. Also, due to the population increase (10% the last 20 years), the mileage of streets has been increased as well (15%). However, it has been estimated that the value of time that has been lost in traffic is around US$80 million dollars per year. Another interesting observation is that congestion times and duration of trips during traffic have also been increased, 50% and 10% respectively. The author also states that because of the increased congestion times and their intensity, negative social incidents have appeared (i.e. road rage).

The author based on such observations, points out three challenges for urban mobility. These are:

- People being excessively dependent on private cars,
- The extension of the roads/streets, wastes large amounts of land area,
- Considerate impact on the environment, due to increased CO2 emissions.

These problems are extremely difficult to be tackled or dealt with efficiently and effortlessly, since they have no coherent solutions, and some that might have been tried out, can have unforeseen consequences in the future. Moreover, these challenges are also interdependent with the social factor, something that makes them even harder to solve or tackle, as they require considerable behavioral changes from people (i.e. changes in their transportation routines). It is mentioned, however that due to the advances in computational power and analytical abilities, it is possible to create and instantiate the way cities are functioning. In such manner, innovative and more effective solutions can be constructed.

Experts of the urban mobility science are also prompted to work on real world problems, by going out in locations that face such issues. Furthermore, the general public also needs to know about these problems, so a popularization of this scientific field is necessary, and also the experts should point out and warn both the public and the respective authorities of the dangers of not acting to solve these challenges.

The article of Schipper, Emanuel and Oldenziel states that experts could find interesting solutions to the previously mentioned challenges by looking into various past practices and facts that lead to the unsustainable mobility that characterizes numerous cities the past 20 years. Again, the goal of the experts here is to provide mobility alternatives that can be inexpensive and sufficient to the people's needs, without harming the environment [11].

The major cause of the challenges that mobility experts face today, is based on old ideals of a functioning cities by past mobility planners, who believed that an ideal city would mostly use cars for transportation. The design of a vast majority of cities has followed this principle as well, where people would live in the suburbs and use their cars to go to the city's center, through radial roads. These ideals were also encouraged by actors and lobbies that had the resources and the interests to support the creation of streets and highways, specifically made for cars and the redesign of cities, in order to support car mobility. These actors and lobbies also presented the usage of cars as an inevitable future practice, and in such manner belittled past mobility practices such as cycling or walking. As a result, the plans that were created to support such visions, had little to none place for pedestrians and cyclists, as well as ignoring the massive ecological damage that would be caused due to the increased CO2 emissions and the destruction of large amounts of land.

A prime example of a U-turn to the past, as it is mentioned in the article, is to be persistent in some sustainable practices that have been existing in the past. Various post-socialist cities have a well-structured public transport system, and despite the motorization of mobility at that time, the idea of using public transport systems instead of automobiles has been practiced, accepted and became

a part of the people's culture in these cities. Other past practices such as cycling have not disappeared due to the massive changes that happened in the city centers and suburbs then. The authors also propose that ideal solutions could be derived from the combinations of such practices with technologies that emerged in the 21st century. Specifically, technologies that provide information and communication for mobility purposes could assist these past practices in an effective manner.

1.3 General thesis objective

The specific thesis project is mainly focused on the transformation of an already existing application into the learning tool required by UrbanSCOPE. The application that will serve as a starting point for the creation of the learning tool is named **MEES** (*Modelling Environmental Exposure System*) [25].

This project was created by *Team Seemless*, which focused on studying the connection between air quality and cognitive abilities. With this in mind, the team created an application that is able to measure the effects of air quality on the cognition of the application users and also to increase the awareness of the air quality in general. What's special about this application, and the reason that it was selected as the basis of the UrbanSCOPE learning tool is the route planner included in the application's functionalities. The user can enter two destinations, the start and the end of a route and the planner will calculate the route and show the total nitrogen dioxide exposure. Routes can be created for three modes: foot, bicycle and car travel. With various modifications, this route planner will assist this research project in creating routes based on specific criteria, something which is necessary for the creation of the learning tool required by UrbanSCOPE.

1.4 General Approach

The primary objective that this learning tool has is, as mentioned before, to assist university students in experiencing new urban planning methods. With such experimentation, they will gain more insight and further knowledge about the sustainable mobility principles, that are interconnected with their studies. Furthermore, their learning experience is enhanced through experimenting with different types of plannings in an urban center (i.e. route - mobility planning, planned route comparison etc.).

Another goal aimed to be achieved from this project, is for students to be able to learn and understand the various elements their local communities include. These can be split into two types, *tangible* and *intangible* elements. The tangible elements include the material elements of a town or city, such as bus stops, historical monuments, recycling bins etc. The intangible elements are mostly focused on how residents perceive certain characteristics or areas of a city. One prime example of this would be the safety a resident feels in a neighborhood.

Of course, not all of these elements will be easy to find with the assistance of online resources (**Ri**jksinstituut voor **V**olksgezondheid en **M**ilieu, RIVM for short). Since this project will address to urban planning students, and also will regard their own city as well (localized content), their assistance will be required in order to complete various information regarding the city they are currently inhabiting and studying. For this concept, interaction modules will be added in the learning tool, in order to further assist students in adding data on the element tables. With all these information collected, the tool will have a more localized content and in addition, the current and future experiments regarding urban planning and city architecture will have an initial set of data to start with.

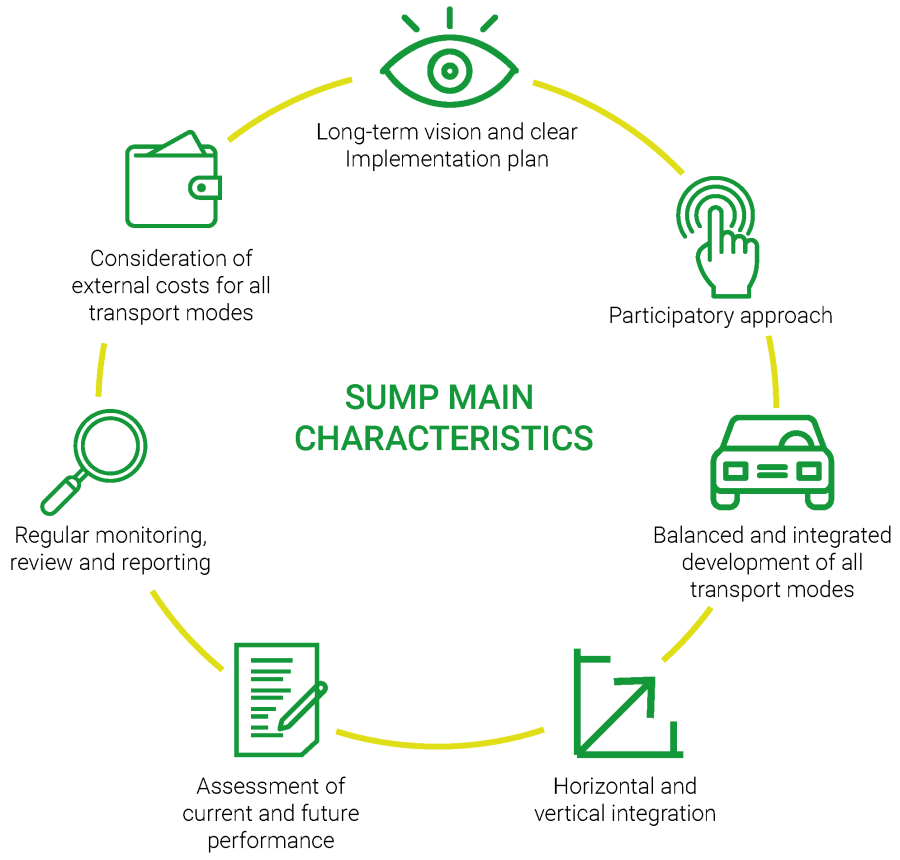


Figure 3: Main characteristics of SUMP. Source: https://sumbilbao.com/2019/en/sustainable-urban-mobility-plans-a-new-way-of-planning-urban-mobility/

2 UrbanSCOPE platform

2.1 General idea and concepts

The main idea and concept behind this platform is to assist university students in experiencing new urban planning methods. With such experimentation, they gain more insight and further knowledge about the sustainable mobility principles, that are interconnected with their studies. Furthermore, their learning experience is enhanced through experimenting with different types of plannings in an urban center (i.e. route - mobility planning, etc.).

The core concept of this application is the execution of one or different types of *scenarios*. A scenario is a problem or an issue a local community is facing, based on the principles of SUMP. For this project, the rerouting of already existing routes on a map consists one type of scenario. A Scenario contains a unique scenario ID, the ID of the user that created it, a list of Element Tables associated with the scenario, a Starting, and an Ending point.

A *criterion* is a specific type of objective that needs to be satisfied, within the application of a scenario. One scenario can have either one or multiple criteria and for the scenario evaluation, either one or multiple criteria need to be satisfied. Various criteria have been devised and proposed for the initial plans based on literature, and through testing and discussing with the project partners, some of them either will be refactored or discarded and new ones will be added. The selection of criteria is also based on the principles of sustainable mobility planning in an urban area.

The application will be mainly used by students, who initially will have to provide some information for themselves, which are:

- Name and surname
- Contact information

The functionalities of the MEES project about user profiles will be imported in this project as well. Since this type of information is confidential, it is crucial that the application complies with the *General Data Protection Regulation* (GDPR).

An important functionality of the platform is the *addition of virtual elements and types of content on the city map*, which users are able to add them on a location of their choice (see section 1.4). These virtual elements are:

- Tangible elements (i.e. bus/tram/train stops, etc.)
- Content, such as pictures of a place or notes

Of course, their relocation and deletion is also a possibility, if the user desires to change the position of an element or permanently remove it from the map. Elements are also saved in tables which contain information about them such as name, a list of (location, value) pairs, indicating the GPS coordinates of an element, and an attribute value, which is used as a weight for the route planning method.

It has to be stated that not all of these elements are easy to find with the assistance of online resources (Netherlands National Institute for Public Health and the Environment, **RIVM** for short, and **Overpass Turbo**). Sensors from this institute provide air quality data for the city of Utrecht, whereas Overpass Turbo, a web-based data mining tool for OpenStreetMaps, is responsible for providing information for both categories of elements (see Methodology section).

Since this project addresses to urban planning students and regards their own city as well (localized content), their assistance is required in order to complete various information regarding the city they are currently inhabiting and studying. With all this information collected, the tool will have a more localized content and in addition, the current and future experiments regarding urban planning and city architecture will have an initial set of data to start with.

The algorithm that is used for the planning of routes in this application requires at first the initialization of a starting and an ending location. The route includes locations from the Element table, the ones associated with the respective Scenario type chosen by the user. The function that is used for the planning takes in consideration the attribute values in the table, and provides a route as a result, based on specific modes selected by the user. These three modes will be implemented for the initial phase of the application:

- Walking
- Bicycle
- Public transport

In later versions, a car mode will be also added, since, according to literature, the rerouting of several highways or vehicle roads can prove beneficial for sustainability in transportation.

The output of the algorithm is a *route*. This route is a series of two or more interconnected waypoints, in such manner that they form a path or road on the map. In our case these waypoints are the starting and ending point, defined in the first phase of the route planning functionality and the elements/content that already exist on the map or added by the user.

The back end of the application consists of several tables, that contain various types of data. First, as far as Dutch cities are concerned a connection with the RIVM database is established in order to ensure information such as temperatures, humidity, CO2 emissions etc. Moreover, in the tables of the database, extra data regarding local information of each city are saved. Additional local details that might be missing from the information retrieved from the OpenStreetMap, with Overpass Turbo. In such manner, the localized content of the application is improved.

The first thing a user observes when the app is running, after a successful login, is the map overview of the city he/she is currently in. The map also displays the user's location and all the local information retrieved from the application's database. All the existing information from online resources and

the students' assistance, are displayed on the map, along with the additional virtual elements a user might add.

2.2 Functional design overview

2.2.1 Elements

1. **Profile**. Contains information, such as:
   - User ID
   - Name of user
   - User contact info
2. **Elements**. An element is an object which will have:
   - Location (coordinates on the map)
   - Type
   - Value
3. **Scenario**. A scenario will have:
   - Scenario ID
   - User ID
   - Element table(s)
   - Starting Location
   - Ending Location
4. **Route Planning algorithm**. The algorithm will require:
   (a) Starting and ending locations of a scenario
   (b) Locations of elements associated with a scenario
   (c) Mode

2.2.2 Back-end

There will be two types of back end users, **Administrators**, and **Planners**.

An **Administrator** can:

1. Create/remove Element Tables.
2. Add content to Element Tables.
3. Add/remove a Planner.
4. Add/Remove a Scenario.

5. Add/Remove a Criterion.

A **Planner** can:

1. Change their Profile information.
2. Create Element Tables.
3. Add content to Element Tables.
4. Add a Scenario.
5. Add a Criterion.

2.2.3 Front-end

There will be a menu that consists of the following submenus:

- Create Scenario
- Load Scenario
- Delete Scenario
- Upload Scenario
- User Settings

There will be two types of front-end users: **Planners** and **Consumers**.

A **Planner/Consumer** can:

1. Create a Scenario.
2. Select criteria for a Scenario.
3. Load a Scenario.
4. Delete a Scenario.
5. Save a Scenario.
6. Edit an already existing Scenario.
7. Add/remove elements to/from map.
8. Add/remove content to/from map.
9. Create/edit a Route Planner.
10. Select elements/content/mode for Route planner.
11. Save route created from Route planner.
12. Change user settings.

3 Related literature

3.1 Mobile applications in education literature

The major concept behind this research project is based on how mobile applications have entered the field of education and are used even more the last years.

A formal definition is provided in the paper of Drigas and Angelidakis [3]. **Mobile Learning** or **M-Learning** in short is a sub-category of e-learning and thus a sub-category of education. Since the majority of students are proficient in technology, this term is defined again as the process of learning crosswise various contexts, throughout interactions with content or other people by using personal electronic devices. The writers also state that through their research it was found that the use of mobile applications in a classroom enhanced the problem solving and collaboration skills of the students. Not only this but also the students were not as distracted as they would be in normal classroom situations, and they were able to work as a team throughout the duration of the class. The writers also state that since mobile devices are offering more portability than other learning tools, the learning locations are bound to change as well as the contexts and how the contexts influence the learners.

Ali and Blasquez in their publication in the Universities and Knowledge Society Journal, present some successful experiences with mobile applications and their impact on education [20]. They also provide various interesting facts and information about how mobile technologies will enhance the field of education and how a significant number of higher education institutes are focusing on mobile technologies. In more detail, it is mentioned that since mobile devices are portable and able to connect wirelessly to the internet, learners are able to perform various eduational tasks (i.e. research, task completion, collaborative work), without being restrained in a specific location at a specific time. The ability of the mobile devices to connect to the internet, allows learners to access resources and up-to-date material for their studies and to interact with experts, a fact that definitely enhances the educational process. Furthermore, the application of knowledge and skills to a specific context is encouraged with M-learning, since learners can study either when they are doing something else (i.e. work) either at their own space.

Taneja and Goel published a paper where they explore the moving of various universities around the globe towards mobile learning activities and technologies [24]. In more detail, because of the increased usage of mobile applications from students, universities are striving to focus their attention in exploring options for mobile learning purposes. According to a research they cite in their paper, 78% of the students own a mobile device and at least 73% of them expressed their interest in using mobile applications specific to their universities. Based on these numbers, it can be understood that universities will keep on focusing in M-learning technologies and innovations in the near future.

Moreover, M-learning is mentioned as a social phenomenon, since learners in the latest years are constantly on the move and create new contexts, add new

knowledge patterns and facilitate the educational purposes due to their interaction with other people, settings and technologies. The writers point out that with M-learning, the learner has become the center of learning and technology assists in the learning process, despite the context. Also, M-learning is helping learners from different cultures to understand and communicate with each other more effectively. However, the teachers have to be re-educated on this type of alternative education, in order to be well-prepared for an educational system, enhanced by technology. Other factors in the current education have to be refactored (i.e. ideal time length for a course), since the current generation of learners use mobile devices that provide instant information and feedback and are usually more preferred than personal computers. It is also stated that the learning process will not have a classroom as a typical studying location, but it will steadily change based on the preferences of each learner (i.e. home, park, coffee place)[20],[24].

A prime example of M-learning applications in universities is the case study performed by Dominguez, Escudero, Riera and Delgado [9]. In this paper, the writers present and analyze the results from an experiment they performed with the assistance of higher education students from the fields of Architecture and Building construction. The case study hypothesis was based on whether M-learning would assist students in creating, visualizing and adjusting models in a virtual environment before their actual construction. This offers an initial hands-on approach on the construction of such models and interaction from the students, without having to create them in the real world. The approach also enables students to share their ideas with others in their group. Their research aim was to evaluate how mobile AR (artificial reality) technologies enhance the learning process of students. The assessment had its basis on researches based on how a person interacts with a mobile device and the ability of visualising models or keeping digital notes on such devices. Two groups of students were created for the experiment, one using conventional teaching methods and the other one using AR M-learning, and the differences in the students' academic results, motivation and satisfaction levels were measured and assessed. The two research questions of the experiment were based on if there would be changes in the levels of these three variables based on the teaching scenarios. The results of the experiment showed a significant variance between the two groups, with the group using the AR technologies, having a notable increase in their academic results, and being more motivated,active and engaged throughout the workshop. The authors conclude that AR technology could complement the conventional teaching methods in an effective manner, since students are able to model virtual architectural projects on the location they want to create them and observe the different stages of the model's creation, while enhancing their communication skills and understanding.

3.2 Location-based mobile games literature

Apart from its educational purpose, the application could also be described as a **location-based game**, because players have to use the GPS interface to add various elements on the map (see section 4). The following literature will provide an insight of what comprises a location-based game, how it is slowly entering the field of education and how the pandemic of Covid-19 Coronavirus affected these games.

Some terms, definitions and explanations that really provide a valid insight of what a location-based game is are provided by Dale Leorke [19]. Networked digital technologies are utilized for interaction between people or between people and the physical environment, through the concept of a game. It is mentioned that with specialized technologies and equipment, the physical environment is intertwined with the game's virtual world along with the action of each player. As it can be easily understood from the term of these games, devices are used that can track the player's current location in physical space. For this reason, location-aware technologies can assist, some examples that are mentioned are smartphones, mobile devices, Global Positioning System location trackers or geotagged data on a web interface. Leorke pointed out that there is discourse around the impact of location based games. Some discussions were focused on how these games influence players to either engage with or disconnect them from their social or physical conditions around them. For this reason, he established two neutral statements that summarize the influence location based games have on players, and these are:

- Location-based games set the boundaries for a space suitable for playing, which is distanced and detached. This allows for player interaction which wouldn't happen under normal circumstances.
- Location based games modify everyday areas into playable ones.

The application, as mentioned in the previous subsection, is mainly focused on educating and teaching students, so it would be useful to see how location-based games are utilized and facilitated for educational purposes. According to Avouris and Yannoutsou, the **narrative structure** along with the game rules and the content can assist the learning progress in a location-based game's virtual space [5]. Because of interactions of various types i.e. event producing simulators, in-between player interactions, information generation by interacting with physical objects. The importance of narrative is stated, which according to the authors is an important tool for establish meaning and organize the experience. Its main characteristic is the *emplotment*, which is combining different heterogeneous parts (actions, events) into a consistent whole and crafting the relationships between those parts. The player is the one that completes the narrative process, because in that way a new interpretation of the world is being created and that procedure can have valuable learning experiences. The paper also emphasizes other pedagogic sides of location-based gaming which are the mobility that these games offer and allow the creation of links between activities

on the outside world and the classroom, the extension of the learning experience over time and the way such games contribute to informal learning. Role playing activities that location-based games usually offer assist players in understanding complex concepts and systems, by immersing them into "life-sized" simulations. In that manner, scientific understanding is enhanced in a powerful manner. Mobile devices assist in transforming students into players in these simulations and enable them to interact with the virtual world with inquiries and experimentation. In addition, location-based games have a strong social dimension that can offer a wide variety of skills to the players, such as teamwork, multimodal thinking, information management, civic engagement and the acceptance of diverse perspectives.

Taking into consideration the current events regarding the pandemic of Covid-19, the vast majority of doctors and health experts urge for keeping away from crowds and suggest avoiding social contact [21]. As it has been previously mentioned, one major characteristic of location-based games is the social dimension that they have and the opportunity they provide to players, who initially are strangers to each other, to socially interact. This specific aspect of these games goes against the urges and advices of the medical community, since isolation is required to stop the virus from spreading to other people [21]. According to a research performed by Laato, Islam and Laine, this timeline was specifically difficult for such games, because of the severity of the pandemic, whereas the video game industry flourished, since people had to spend more time at home. The vast majority of games offered various incentives to prevent people from going out []. The authors specifically mention that location-based games is one of the most healthy genre of games, since they motivate physical activity and social interactions. However, because of these characteristics, they took a hard hit the last two years [18]. There were major in-game changes for this genre of gaming for supporting stationary gameplay, but according to the results of the research, players did get motivated to go out and socialize to some minor extend, but players were adherent to rules and suggestions of health/government officials. Fear of missing out and deficient self regulations were factors that influenced the playing intensity [18].

3.3 Route planning literature

Since route planning plays an important role in our application, it was deemed necessary to search in literature for similar applications that use route planners based on user input or user generated content.

El Ali, in his paper [1], where he proposed a system to create walkable paths and routes in Amsterdam, based on multiple photographs that were taken in these paths, states that photographs from multiple social media services, can provide further information apart from the social dynamic aspect and the urban flow. He mentions that the intent and what people experienced in these locations can also be identified and moreover how people moved from one place to another. By using a image-sharing website called *Flickr*, that has geotagged images and photographs, the author was able to find data from multiple photographers.

Based on the notion that a person takes a photograph in an interesting location, the author devised a method that would check where photographers have been and how their movements were similar to the corresponding movements of other photographers. This method is able to devise and generate walkable routes and paths based on photographs taken in various places within a city.

This framework is definitely useful for our case, since it generates routes based on other criteria than arrival speed or efficient energy consumption. Our project here is based on generating paths based on the SUMP principles, which means that it would be useful for the students to be able to create new paths within the city. It will also be practical for the purposes of localization. Geotagged photographs will also assist in identifying the most important landmarks in each city, even from the initial phases of the implementation and without getting the localized data tables from the students.

In another research, Waschk and Kruger designed an algorithm, that generates artistic routes, with only two inputs, a starting point and a figure, chosen by the user [4]. This figure can be either a logo, a shape or a message. This algorithm computes the routes necessary in order to depict the figure input as accurately as possible. This depiction starts from the initial starting point and with a variation of Dijkstra's algorithm and metrics that minimize distances, paths and the sum of various distances put together the algorithm automatically generates the routes necessary to create the input figure. The results have shown that their algorithm performed with accurate depictions of the initial figures.

Although this research has a different scope and aim than our research project, the way that the algorithm functions could prove beneficial for generating routes automatically and without a great number of input. Also the metrics stated in this paper, will definitely be useful for this application, since they will be needed in cases of creating a route that has criteria such as fast arrival at ending point or not spending much time on foot, bike or public transport.

Another research, similar to the previous one, is the one in the paper written by Garcia, Arbelaitz, Linaza, Vansteenwegen, and Souffriau [2]. In this research the writers enhance the functionalities of Personalized Electronic Tourist Guides (PET), which is an application on handheld devices, and acts as an electronic local tourist office. This application enables tourists to see recommended routes and points of interest (POI) in a city. The routes are generated based on specifications and choices made by the users, and these routes are also open to customization. This research uses a previous review for Personalized Electronic Tourist Guides, written by Vansteenwegen and Souffriau, as a starting point.

The combined research of all writers is an enhancement on the PET applications. Specifically, it was observed that in the current applications for assisting tourists, there was no integration for public transports and no advanced heuristics were used in order to generate routes fast and efficiently. At first, a list of POIs is created based on information of the destination and the user's profile. Afterwards, by including other parameters such as restrictions from the user (i.e. budget, available time), POI information (i.e. location, prices for tickets

etc.) and transportation data (i.e. bus/tram/train schedules etc.), the system is able to generate a personalized tour route for the user. Not only that but the users have the ability to customize these routes. Inserting new POIs, deleting existing ones or reordering the POIs in the generated route are the basic functionalities for customization.

This specific framework is closely related to our own research, since our project would like to give the students the ability to create routes on a city map based on scenarios and their corresponding criteria and especially rearrange virtual elements that they might add on the map.

For the various criteria that will be used for the route planning, especially for the creation of new bike lanes on the map raster of our application, the research of Hochmair will prove helpful [13]. In this research, he proposes additional criteria for planning a bike route, apart from the most basic ones (shortest path, less time). These new criteria were a result of an internet survey, where people were asked to devise their criteria for reaching a destination in an imaginary situation. In that situation, users had to become tourists using a bicycle in a city not known to them, and they had to reach a specific destination. They were also asked to rate their criteria with a rating from 1, as not important, to 4, as very important.

This research is beneficial for our application, since criteria for creating new scenarios are needed. Although these criteria apply only to bike lanes and routes, some of them can be mutual for pedestrians or for public transport routes. With this as a starting set, multiple cases for the SUMP concepts can be covered, and this set can be expanded based on consultation from the students and discussion sessions.

4 Research questions

A wide variety of methods have been implemented in order to facilitate data input in mobile applications. Some of them are pretty straightforward (touching a text field and entering the information with a virtual QWERTY keyboard), others are more engaging for the users (using speech-to-text systems, scanning a QR code) [7].

One of the most important functionalities existing in the M-learning tool, is when a user adds an element at a specific location on the map. Since this functionality requires input from the user, three different methods have been implemented:

- Direct coordinate input,
- Finger/cursor input,
- GPS location input

4.1 Variables and suitability factors

This research study is focused on *discovering which type of input method is most suitable for adding an element on the map of the M-learning tool.* The suitability of an input method could be explained by two major factors, *functionality* and *user experience.* Both factors have smaller subcategories, so as to have a more accurate depiction of how these types of input are perceived by the users:

- Functionality
  - How accurately a task is executed (difference between given locations for elements to be placed and actual user input locations)
- User Experience
  - How intuitive a method is
  - How engaging using a method for tasks
  - How easy is a method for executing tasks

The variables that have been identified in this study are the following:

- **Independent variables**: *The three different methods that are compared in this experiment (direct input, finger/cursor input, GPS input)*
- **Dependent variables**: *The functionality and user experience factors along with their subcategories*
- **Control variables**: *The application environment remains the same for all tests that will be run, three predefined locations that are provided to users (see 5.4) .*

Since there are two categories of factors, the research questions will be two in total, as well as the null hypotheses. The null hypotheses for both categories will be presented first, so as to derive each individual research question more accurately.

- **Functionality**
  - **Null Hypothesis 1**: *All three methods that have been developed have proven to be not accurate for the task of adding an element on the map.*
  - **Research question 1:** *How accurate are the three methods that have been developed for the task of adding an element on the map?*
- **User Experience**
  - **Null Hypothesis 2**: *All three methods that have been developed have no effect on the way users perceive them in terms of ease-of-use, engagement and intuitiveness.*
  - **Research question 2:** *How users perceive the three methods that have been developed (ease-of-use, engagement and intuitiveness) for the task of adding an element on the map?*

5 Experiment Approach

5.1 Experiment Overview

A general overview of the experiment is that users will receive the application via a URL link and then will be asked to enter elements on three predefined spots, by using all three aforementioned input methods (see section 4). After that, a questionnaire will be available for them, to answer questions based on their experience with them. The steps that describe the experiment are presented in the final part of this subsection. In the following subsections, the experiment phases will be explained in more detail as well as the methods that will be used.

1. Three images with locations are provided so you can have a hint on where benches should be placed on the map raster.

2. The methods that you'll use to add three benches on the map are the following:

   - **Direct Input Method**: By using the image as assistance, find the exact coordinates of where you want to place the bench and input them in the "Add Element" window (Latitude/Longitude textboxes on the app window). Also, paste those coordinates on the textbox on the form.
   - **Finger/Cursor Input Method**: By using the image as assistance, find where your bench needs to be placed and use the "Add Element" option (straight from the map raster) to add it where you think it should be placed. Then, copy the latitude/longitude values that are shown in the "Add Element" window and paste them on the textbox on the form.
   - **GPS Input Method**: By using the image as assistance, you have to physically go to the location and when you are there, select the "Add Element on my location" option. Then, copy the latitude/longitude values that are shown in the popup window and paste them on the textbox on the form.

3. Before closing the application, click the option named "Clear All", next to the "View All" option.

4. The last parts of this experiment will include answering questions regarding the usability/engagement of the methods as well as an overall rating for the MEES application.

5.2 Important Notice

However, an important precaution for the users is that no WiFi can be used throughout the experiment. The reason for that is that during the implementation and testing phases of the project, the GPS geolocation module of the application used to calculate different positions on the map raster for a computer that was connected on the WiFi network, which were inaccurate with the actual location of the computer. So for this reason, the experiment requires the use of mobile phones and mobile data for using the application and thus, accurately finding the geolocation of the users, a necessary component of this experiment.

5.3 Placing elements on the map

As mentioned above, users will be able to add virtual elements on the map with three types of input. For the experiment, one element has been selected for placement around the map, and that is the bench, since it has been extensively used throughout the testing phases of the project, and it will be easier for the users to deal with one type of element instead of a variety of elements.

The *first input type (Direct Coordinate Input)* will be by clicking the leftmost button on the map, named "Add Element (Direct Input)" and changing the coordinates of the element in the Edit Element window. In this case, a set of coordinates has to be provided in the corresponding text fields and after clicking the Save button the element will be moved to the new provided position. For selecting a more accurate position, users will have to iterate several times to find a spot that is close to the one the image shows.

The *second input type (Finger/cursor Input)* is by using the cursor/finger input, which means that the user will have to find the location on the map and then right click on that spot (or finger input if they use a smartphone/tablet) and select the Add Element option.

The *third input method (GPS Input)* is focused on utilizing GPS capabilities, and for this reason a different Add Element option has been developed (named Add Element on your location), which adds an element on the current location of the user. This method was based on existing functionality of the MEES application. A geolocation module which was built-in the system, stores the latitude and longitude of the device in one minute intervals, and this capability has been transformed into an input method [25].

As mentioned in the first part of this section, the first and second input methods will use two images that have **predefined locations** for elements. Users will use these images as visual assistance so as to get an overall idea of where an element should be placed. For the third input method, users will have to physically go to the location the image is showing to use the GPS input method so as to add an element on that spot.

5.4 Experiment data

The data that will be gathered from this research, will be *quantitative continuous data*, because this is a *within-subjects* research test (same users providing answers for the input methods) [16]. Also, since the experiment includes three types of input methods, the answers of the questionnaire will be separated into three groups, one for each method. The test will be used to describe the usability of these methods (ease of use, accuracy, engagement, see section 4) and also to form a necessary basis so as to compare postdesign changes. For these reasons, the test is a *benchmark summative usability test* [16].

As far as the data collection is concerned, users will receive, via email, two URL links, one that redirects to the questionnaire (see next subsection) and one for the application. As for the number of participants that will be required, the rule of thumb will be followed, so **thirty users** will provide the data for this research to have meaningful and valid results [16].

As far as the dependent variables are concerned (see 4.1), there are two ways of measuring them in the experiment. For the *functionality* factors, the user input on step 2 and the coordinates for the predefined locations (which will be referred to from now on as **Initial Element Locations**) will be compared to the input that the users will provide in step 5 (which will be referred to from now on as **Actual Element Locations**). In that way, it will be possible to measure the distance between the Initial Element Locations and the Actual Element Locations. With these measurements, valid conclusions can be drawn as to which method is more accurate for placing an element on the map. The following subsection can provide more information as to how the *user experience* factors can be measured.

5.5 Questionnaire

After the completion of the tasks (adding the bench element on three different locations on the map) users will be asked to complete a usability questionnaire in Google forms [16]. The questions are derived from the previously mentioned subcategories (see section 4) and mainly based on the **System Usability Scale** questionnaire (SUS for short), and specifically the **all-positive version** [16]. According to Sauro et al., the System Usability Scale is a quick and reliable questionnaire about the perceived usability of a system. This questionnaire has been used widely in testing various components of a system, including websites and interfaces. It includes 10 questions with 5 answer options, which are on a scale from 1-5 (1 being as *Strongly Disagree* and 5 as *Strongly Agree*). As for the reason why the **all-positive version** is used, the answer is based on the fact that the respondents will not likely make any sort of mistakes when they provide their answers and researchers are prone to not make coding errors. Of course, since the questions of the SUS have a generic scope, so as to be easily applicable to all sorts of usability tests, they will be altered accordingly to fit the purposes of this paper's experiment. Users will answer these questions on how they appreciated the input methods in terms of how easy/intuitive they

were to use for completing the tasks. Also, users will be able to leave some optional comments regarding the experiment in a text field after answering the questions. There is a detailed overview of the questionnaire in the appendix session of this paper.

6 Implementation/Methods

This section focuses on providing more explanations about the frameworks and the software that has been used to implement the features that have been described in the previous section. The M-application is mainly built within the **Mendix** platform, along with a **pathfinding web service**, which is responsible for calculating paths based on NO2 pollution and on element location within the city. Various tools have been used throughout the implementation phase, specifically for retrieving locations and road topology through **OpenStreetMaps**.

It has to be mentioned that the methodology and implementation of this project proved to be an arduous task for the author, because of the various aspects of the mobile application that had to be taken care of (i.e. friendly user interface) as well as the web service (i.e. pathplanning based on a collection of elements), so the assistance of mr. Geert-Jan Giezeman[1] was requested. The initial versions of the application and the web service have been developed by the author, and mr. Giezeman took over the project's implementation when the author had to focus on setting up the research experiment. The current versions of the application and the web service are a product of mr. Giezeman's work.

Also, as it has been mentioned before, the basis of this project is an past collaboration project between bachelor students of the University of Utrecht and the RIVM. The foundation for the current project was provided by an RIVM expert.

6.1 Mendix

Mendix is a low-code development platform that enables the creation of mobile applications with the assistance of a graphical user interface. Its core functionalities are based on the principles of model-driven design, automatic code generation, and visual programming and utilize the concept of end-user development. Mendix can be used by a wide developer spectrum, and does not require any specific knowledge compared to other software platforms. Moreover, since the time needed to create applications is considerably reduced, they can be deployed and tested in a swift amount of time. Application functionalities can also be extended with custom scripts in Java and Javascript. As far as security reasons are concerned, Mendix provides options for administrator/user roles for the deployment phase of the application. Corresponding role distributions for administrators and users are also available, which are mainly used in the testing phases of the application. The MEES application has been redeveloped within this platform. The RIVM institute provided a version of MEES that had the functionalities of the initial application that is mentioned in section 1.3. The main modules that are utilized in the application are the ones below:

[1] drs. GJ (Geert-Jan) Giezeman, https://www.uu.nl/medewerkers/GJGiezeman

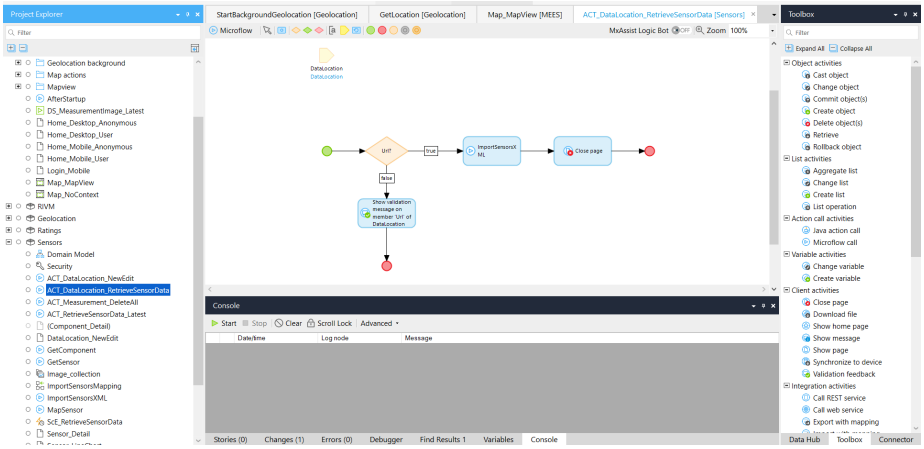


Figure 4: User interface of Mendix.

- RIVM module
- Sensors module
- Map widget module (named as MEES in the app)
- Geolocation module

In order to be more specific and clear, some additional information have to be provided for how Mendix saves information on an application's database and the way the application's functionalities can be created. Real-world objects or classes in Mendix are named as *entities*. Entities have attributes which represent some information about an entity and can be of various types (String, integer, decimal number etc.) An entity can be persistable or non-persistable, and the main difference between these categories is that the first one has the ability to create a database table for an entity whereas the second one does not.

For the functionality part, Mendix offers two ways which are the *microflows* and *nanoflows*. Microflows are used to express the logic in Mendix application in a visual manner and perform actions such as creating/updating/deleting objects, showing a page, making a choice or retrieving information. The graphical notation is based on the **Business Process Model and Notation**, which is a manner of depicting business processes in a workflow [6]. Microflows are usually composed of various *microflow elements* (not to be confused with the city elements, see 1.4), which are the following:

- **Events**: represent the starting and ending points of a microflow as well as special operations in a loop.
- **Flows**: connect various microflow elements.
- **Decisions**: create and merge choice paths.

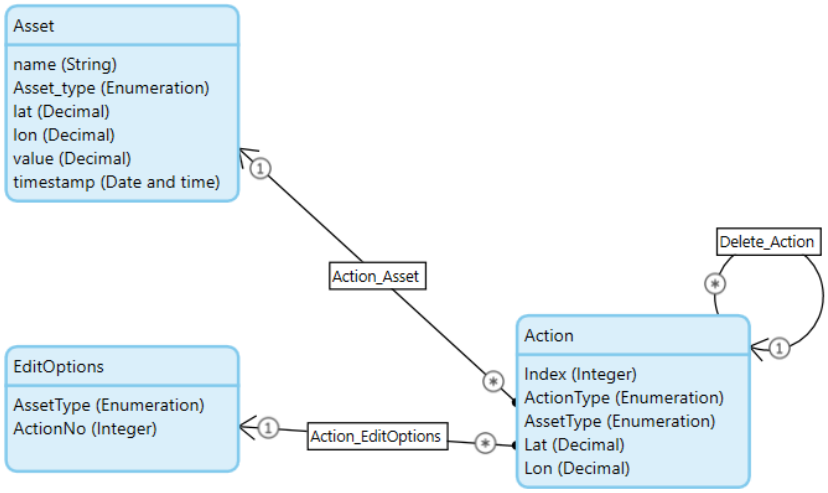


Figure 5: Domain model of the Elements module. Several entities can be linked together via *Associations* or can be linked with themselves.

- **Activities**: the actions in a microflow.
- **Loop**: used to iterate over a list of objects.
- **Parameter**: data serving as input for a microflow.
- **Annotation**: element used for comment creation.

Nanoflows are also composed with the same microflow elements, but there is a number of differences between microflows and nanoflows. One major difference between those two, is that microflows run in the runtime server, which means that they can't function if the application is offline, whereas nanoflows are able to do so. Nanoflows are run directly on the device which also provides a speed benefit for logic that does not have to run on the server. They also allow direct execution for client actions and if an error is encountered during a nanoflow runtime there's no rollback to previous changes. Moreover, differences can be noticed in the execution of expressions, based on the fact that microflows use Java libraries and nanoflows use JavaScript libraries. Lastly, if a list is changed in a sub-nanoflow, the changes won't be reflected in the original nanoflow.

The **RIVM** and **Sensors** modules are interconnected with each other. The Sensors module is responsible for receiving air quality data from sensors placed all around the city of Utrecht. These sensors retrieve data for multiple air particles that are harmful to people (NO2, NH3, PM2.5, PM10). The module has functions that retrieve data for the concentrations of these harmful particles for each sensor separately by reading XML schemas, and saving these information

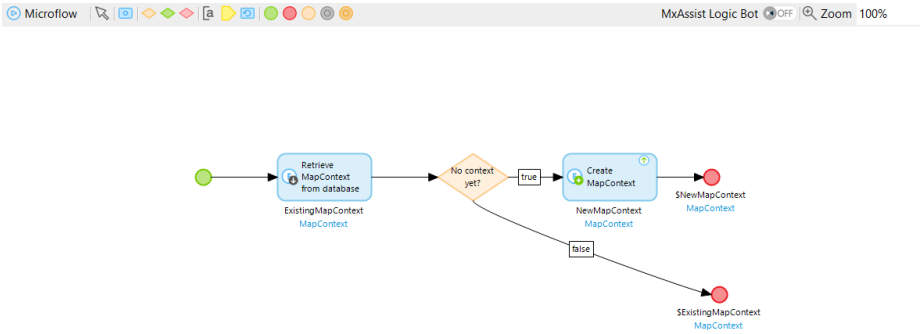


Figure 6: Example of a microflow in Mendix. This microflow is responsible for retrieving the latest map context and if none is found, then a new one is created.

on the database of the application, since the Sensor entity is a persistable one. In the Administrator mode of the application, more details can be viewed for each sensor separately (location, particle concentration etc.). The air quality data for each sensor is updated every hour.

The **RIVM** module has functionality that is responsible for turning the air quality data of the sensors into meaningful results not only for the administrators but also for the users. It includes methods and custom Java actions that utilize the locations of the sensors, as well as the concentration of the particles. It has to be mentioned that the majority of data gathered from the sensors of RIVM utilize a coordinate system named Rijksdriehoeksstelsel [14]. This is a coordinate system used in the European Netherlands, as a geographical information system and for maps of various authorities. It is mainly a Cartesian system, with the x-axis running from the west to the east and the y-axis from the north to the south. The central point of the system is the spire of the Onze Lieve Vrouwetoren ('Lange Jan') in Amersfoort, and all values are positive, with the y-axis values being greater than the x-axis values. These coordinates have to be converted to the WGS84 (latitude/lognitude) coordinate system. This is based on the fact that the map widget, which is the essential component of the application can mark and utilize locations on the raster only with this coordinate system.

The air quality information from the sensors module and the coordinate data that have been converted from the RD coordinates to the WGS84 coordinate system, are being utilized by a main script in Java, which is responsible for creating an overlay image on top of the map raster. This image is also edited by another Java action, which is responsible for changing the color of coordinates based on the amount of particle concentration in the coordinate location (blue for the least amount of concentration and red for the highest amount). In that way, the image is utilized as a heat map of harmful gas concentrations around

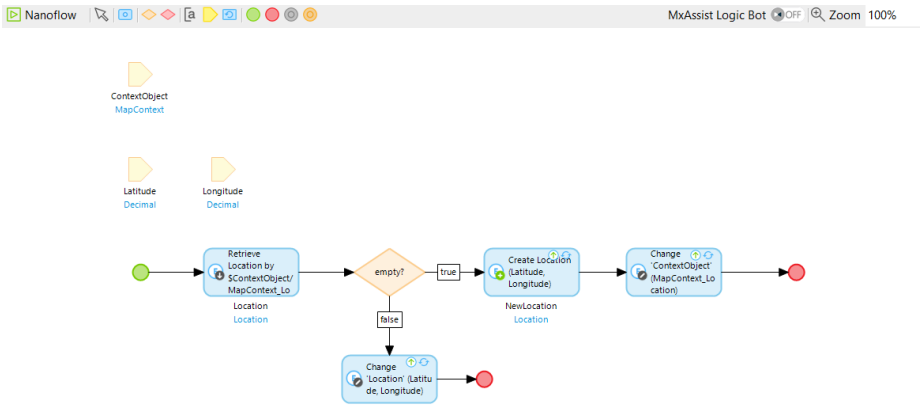


Figure 7: Example of a nanoflow in Mendix. This nanoflow is responsible for updating the location of a user on the map's context.

the city of Utrecht. This image can be created through the administrator mode, and it is automatically applied on the map raster on User mode.

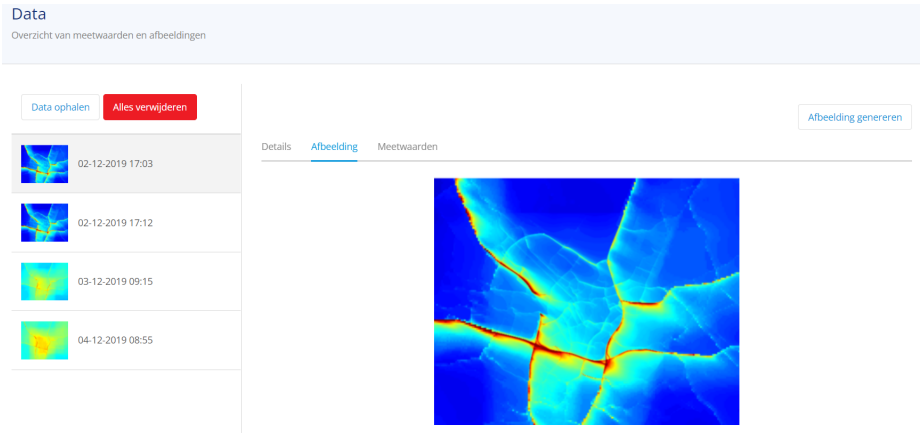


Figure 8: Administrator mode for the MEES application. Administrators can download the latest air quality data and create the overlay image for the main interface of MEES.

The main module for the application's functionalities is the **MEES** module. It includes the main pages of the application for various modes (desktop, phone) and for various user roles (User, Anonymous), scripts for assisting the tracking of the geolocation of a user and retrieving the latest overlay heatmap image. The main component of this module is the Map Widget. All the modules and functionalities of the application depend on this widget, since it will be used extensively for the experiment (see section 5). The module also includes

functions that are responsible for retrieving the context of the map from the database, which includes roads and routes that have been created by the user as well as elements that have been placed on the map. The widget hosts maps from various map providers (Google Maps, OpenStreetMap, Mapbox, HERE Maps), but OpenStreetMap has been chosen as the main provider since its data are open source and there is aplenty of documentation for retrieving information and data about road structure, city amenities and elements (see subsection **??**).

The domain model of the *MEES* module includes three entities, *MapContext*, *Location* and *Stretch*. The MapContext entity holds information about the map's state (default center latitude and longitude, number of roads created on the map and their lengths), the Location entity is utilized from the Geolocation module and the user's locations are saved over time. For the latter entity, the attribute named Timestamp is responsible for saving the date and time a location has been visited, which will be useful later for a microflow that adds elements on the latest GPS location. The last entity, Stretch, which is the ratio of the cost a selected route and the cost of the optimal route, holds information about route indexes as well as their respective lengths (enter citation here).

A useful feature that the map widget is offering is the marking of specific locations on the map. The widget is able to specify locations with two ways, either by a latitude/longitude pair or by address. For the case of the experiment the first specification of the location of a marker has been chosen. Moreover, the widget offers marker customization options, such as:

- Marker title: can be displayed in a pop-up window when the marker is clicked.
- Marker visualisation style: markers can have an icon on the map, could be one of the images Mendix offers, but also an image that the user provides. This can also be applied to a batch of marker locations on the map.
- Marker event: markers can have event handlers that will respond when the marker is clicked (i.e. opening a page).

The following modules have been created for the purposes of the experiment as well as setting up an environment for the application to be utilized at a later stage from Urban Planning students. The modules are the following:

- Elements module
- Content module
- Routeplanning module

The **Elements** module is a noteworthy module in the application as well. It includes methods for adding elements on the map raster, the pages that users can view on the application (editing/overview pages), as well as menus that extend the features of the map widget. It also includes all the element types that can be added on the map raster (see figure 11).

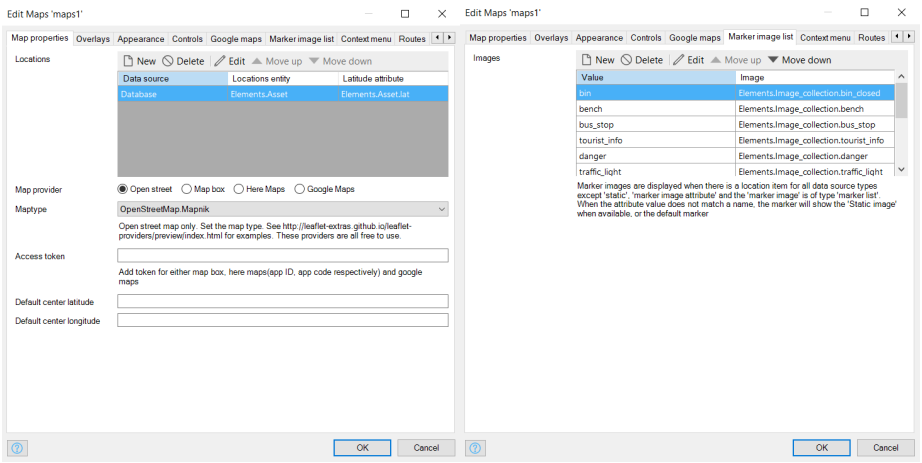


Figure 9: Map widget properties and marker initialization/customization

The methods that have been developed to add an element on the map follow the same notion, and utilize the functionalities of the map widget, specifically the marker functionality. By incorporating a *Map Context parameter* as it can be seen in the figure below (figure 12), the *latitude/longitude map properties* can be utilized in the element creation. The first activity in the microflow is retrieving the latest *Options*, which are mainly the *element type* that should be added on the map, and an *enumerator* that keeps track of the elements that have already been added. If there are no prior Options, a new Options entity is being created (see figure 13). After that, the Element object is created, based on the element type parameter of the Options object, and the Latitude/Longitude parameters of the MapContext parameter. The following activity creates a new *Action* object, which utilizes the *Options enumerator, the Element object and the Options object*. Moreover, the *Action Type* parameter is set as *Delete*, so that the user can easily delete an Element from the map, with a simple cursor click or finger tap. Subsequently, this also deletes the Action object. The final activity of this microflow increases the enumerator by one.

The same notion is followed from the method that utilizes the geolocation of the user. Instead of using the marker's latitude and longitude values, a specific microflow activity is used to retrieve all the locations of a user that have been saved in a database. The Location entity has an attribute named as *Timestamp*, which receives time and date values. By adjusting this attribute to receive the current date and time as value (the command *[%CurrentDateTime%]*), all the locations have a timestamp, which is useful in order to retrieve the user's latest position. Mendix offers the ability of retrieving a list of objects from the database, as well as sorting them based on a specific attribute of the entity (figure). For this case, the Location timestamp is utilized and the list is sorted in a descending order, which means that the first item of the list will be the latest

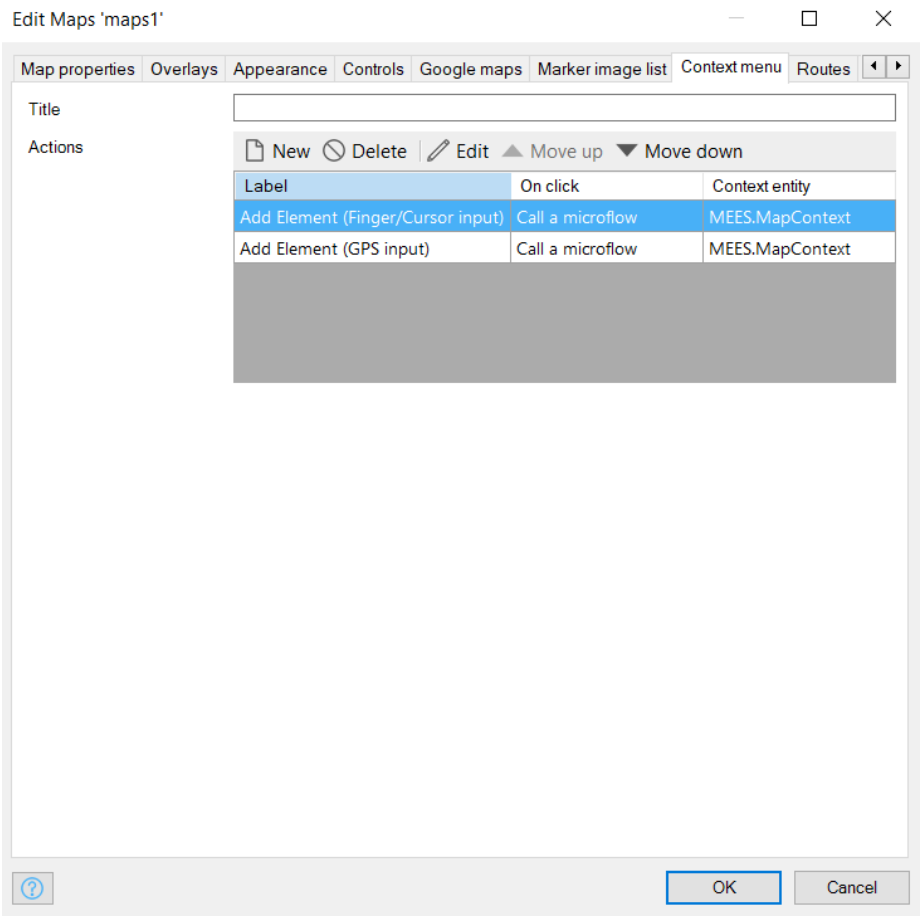


Figure 10: Map widget context menu: here a developer can add the methods that will be used on the map raster.

user location, which is saved on a variable named *GPSLoc*. After this activity, the rest of the microflow follows the same notion as previously, which is creating an Element object with the location of *GPSLoc*, and the Options that have been previously retrieved, as well as the Options enumerator incrementation.

The **Content** module bears many similarities with the Elements module, specifically with the utilization of the marker functionality of the map widget. Instead of adding Element objects on the map, users are able to add images or videos at a specific location, as well as notes about a place on the map. The microflows of this module have similar functionalities as the ones in the Elements module. The core difference is that this module is utilizing a media uploader page instead of a simple popup window, so that users will be able to upload their selected images or videos on the server. Furthermore, another notable difference

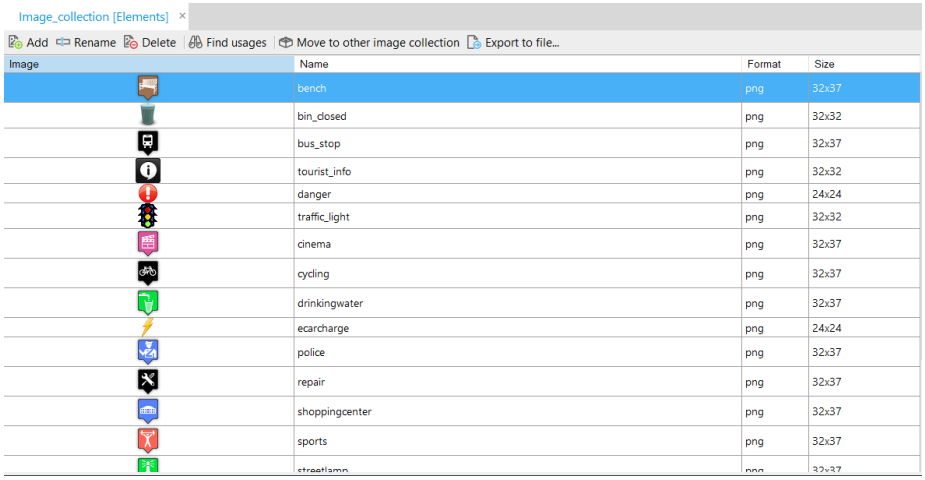


Figure 11: Mendix allows the creation of image collection objects, which are utilized from the Map widget. Here, all the elements that can be added on the map are listed.

is the way the Content element entities are initialized in the Domain model. Mendix allows generalization of entities, which means that an entity is able to *inherit* the attributes, associations and events of a more general entity, in this case of the Image entity (subsequently, FileDocument is a general entity for files in Mendix), which is already set as a default one in every Mendix project. The generalization feature is based on the Object Oriented Programming concept of *inheritance*. Thus, the entities that belong to the Content domain model inherit the attributes of *Image* for images and *FileDocument* for videos.

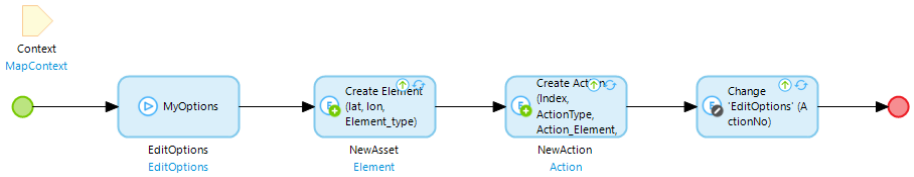


Figure 12: Microflow that adds an element on the map raster, on the latitude/longitude location selected by the user.

The **Routeplanning** module holds the entities, microflows, scripts and overview pages, responsible for creating routes and pathways on the map raster. The domain model of this module includes various entities that are responsible for holding information about a route (starting and ending locations, type of vehicle used, route overlay picture and route options) but also non-persistable entities (not saved on application's database), that are used in order to hold

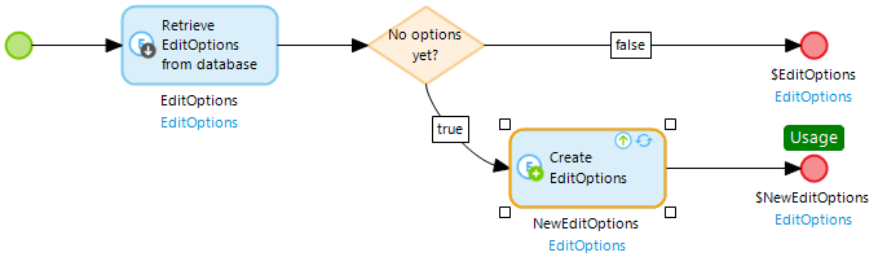


Figure 13: Microflow that retrieves existing options (element type, enumerator) from the database. If there are none, a new EditOptions object is created.

information about the route planning microflows. This is a module that includes a substantial interconnected microflows and the description of them will start from the microflow that is the starting point of this microflow chain. The diagram below can assist the readers to follow the procedure of creating a route in this module.

The starting microflow is named *AddRouteStart*, and the first activity is responsible for retrieving all the starting locations from the database in the form of a list. The following method deletes this list. This happens in order to have only one route active on the map raster, since past versions of the projects enabled multiple routes on the map, something that was cluttering the map, and was not visually helfpul for the users. The next activity is creating a RouteStart object, with the latitude/longitude pair that was selected by the user on the map. As it has been mentioned before, the connection between microflows and the map widget is achieved with the utilization of a MapContext parameter, which provides values for the latitude/lognitude pairs in runtime. The *AddRouteStart* microflow ends with the call of another one named *MakeRoute*. The same notion is followed in the *AddRouteEnd* microflow.

The *MakeRoute* is calling another microflow named *ClearLengths*, which in turn calls another one, *MyMapContext*. The latter is responsible for retrieving an already existing MapContext object (the state of the map) and if there's no existing object a new one is created and returned. Back to the *ClearLengths* microflow, the retrieved MapContext object has one of its attributes changed, the Length attribute which is set to zero. After this step, all Stretch objects are retrieved from the database and deleted in the next activity.

Back to the *MakeRoute* microflow, after the road lengths and stretches are cleared, the microflow retrieves a Start object from the database. This list has been previously cleared by *AddRouteStart* and has only one entry, which is the latest Start object that has been added by the user. After this, a check is performed if this object is empty, which terminates the microflow if it is true

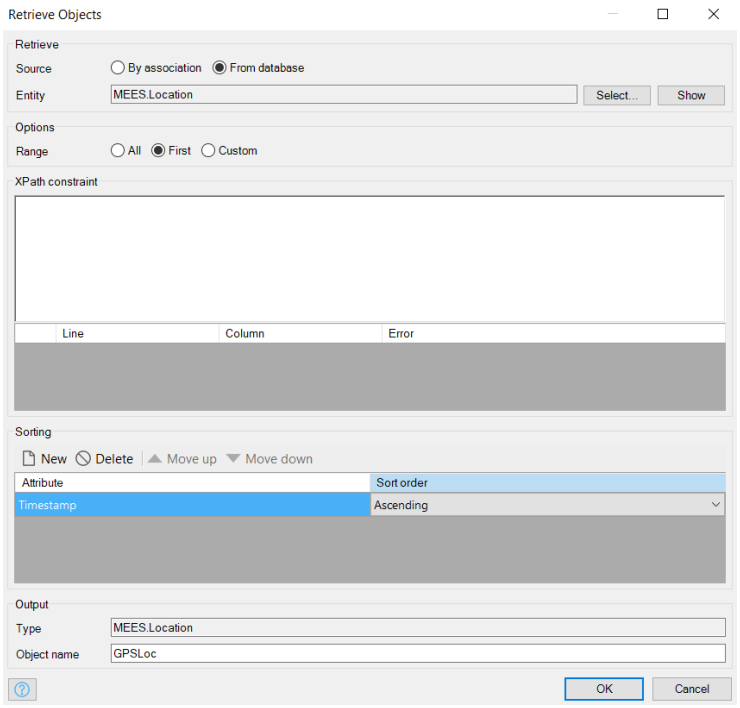


Figure 14: Retrieving objects in Mendix. This can be done via associations between entities or via the database. There are options of retrieving only one object or a custom number of them, as well as sorting functionalities based on object attributes.

and continues if it is false. Afterwards, a Finish object is retrieved from the database, as in the previous case with the Start object. A check is performed to establish if the Finish object is empty or not, which has the same results as the previous case with the Start object. After the check, the Options object is retrieved from the database, with attributes that save information about the type of vehicle used, if nitrogen dioxide should be taken into consideration, how the route should be created (direct, via a collection of elements which are retrieved from the server, via specific elements chosen by the user or via an extra road), an interval used to specify the distance between the elements and lastly the element types that will be retrieved. Another check is done to see if nitrogen dioxide is used. The Start, Finish and Options objects are provided in the following activity named *GetRoute* as parameters.

The first activity of the *GetRoute* microflow, calls another microflow, *VehicleToString*, which is responsible for retrieving the vehicle type attribute from the Options object and turn it into a String variable. Back to the *GetRoute* microflow, the second activity creates a String variable with the name of the server, which is utilized in further stages of the microflow. Afterwards, a check

is performed on an attribute of the Options object, the one about how the route should be created (named Via in the implementation). If the Via attribute is empty, then an empty ViaRoute object is created. This object belongs to a non-persistable entity in the domain model, which in later stages is used to hold information that have been retrieved from the pathfinding web service. This will be explained further below when the activities that communicate with the service are presented. If the Via attribute has the value 'Direct', then a *Call REST* activity is used, which is one of the Mendix activities that achieves communication with REST endpoints.

Before this activity is explained, it is vital to provide some information about the **REST API** and **REST** architectural style. First of all, ***RE****presentational* ***S****tate* ***T****ransfer* ***A****pplication* ***P****rogram* ***I****nterface* is an architectural style that allows software to communicate with other software either over the network either on the same device with the assistance of the HTTP protocol [12]. Clients can access the content of resources in various formats such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Six constraints have to be satisfied in order for a service to be characterized as RESTful:

- Client-Server: This style, based on the principle of concern separation (user interface and data storage concerns), assists in the independend evolution of components, meaning user interface portability to multiple systems and scalability improvements.
- Stateless: A client request must contain all necessary information for the server to understand it, and not take advantage of any stored information on the server.
- Cache: The data provided with the server's response must implicitly or explicitly label itself as cacheable or non-cacheable. If the data are indeed cacheable, it can be reused later for equivalent requests for a short time period.
- Layered system: Hierarchical layers that constrain the behavior of components. Components cannot see other layers apart from the immediate ones they interact with.
- Uniform interface: The components interface applies to the principle of generality so as to simplify the overall system and improve interaction visibility. However, four constraints have to be satisfied to achieve that:
  - Resource identification: The interface must be able to identify uniquely the resources (i.e. document, image, collection of other resources etc.) that are exchanged between server and client.
  - Manipulation of resources through representations: Resources should have unique representations (the state of a resource at any particular time) in the server response and API consumers should modify those to modify their state in the server. Resource representations consist

of *data*, *metadata* that describe the data and *hyperlinks* that help clients to move to the next state.

- Self-descriptive messages: Resource representations should include adequate information to describe how to process the message and information on additional actions that can be performed on the resource.
- Hypermedia as the engine of application state: Clients should only have the *URI* of the application (Uniform Resource Identifier) and the application should utilize hyperlinks to dynamically drive resources and interactions.

RESTful APIs that utilize the HTTP protocol usually have:

- *A base URI*, such as $http://api.example.com/$
- *HTTP methods*, such as GET,$POST$,PUT and $DELETE$
- *Media type*, to define transition between resource states.

It has to be stated though that REST is totally different from HTTP, since people usually use these terms interchangeably [23]. First of all, HTTP is a **communications protocol**, while REST is a set of attributes of a particular architecture system. Moreover, since HTTP allows the use of cookies and sessions to save state, it violates the constraint of statelesness. Finally, URLs usually allow web servers to violate the constrain of uniform interface.

Back to the Mendix application, the *Call REST* activity is a vital part for the *Routeplanning* module, because the results of the pathfinding service have to be incorporated in the Mendix application and be translated into meaningful information for the users. This activity not only allows connection and communication with a REST endpoint but also provides functionalities for handling responses or requests. The General tab of this activity holds information about the endpoint's URL, which should be in a String format and the HTTP method that will be used, which in this case are GET and POST (figure 15). The GET method is used to request data from a source and the query string is sent in the URL of the GET request. The POST method is used to send data to the server in order to create/update a resource and the data sent to the server with this request is stored in the request body of the HTTP request. The reason why the POST method was selected over the PUT method will be explained at the specific operations further below. Further options are included, like for how many seconds the application needs to wait for the REST endpoint and if proxies will be used for the request as well as client certificates.

The other important tabs that are utilized for the two HTTP methods that are used are the last two, which are named *Request* and *Response*. The GET method mainly utilizes the Response tab. In order to handle the response, Mendix offers these options:

- Apply Import Mapping: if the response is in XML or JSON format, it can be transformed directly into a Mendix object.

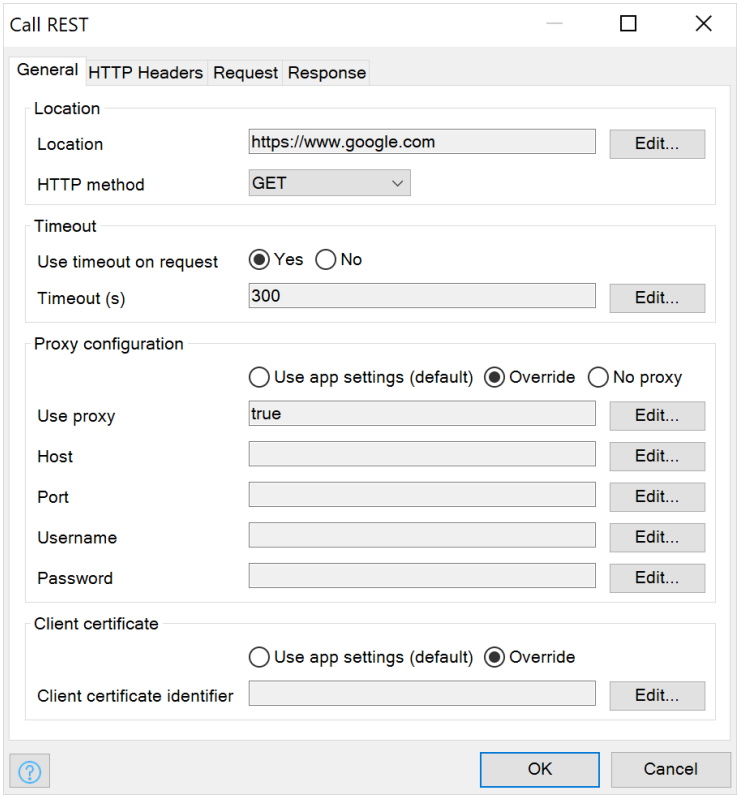


Figure 15: The General tab of the Call REST activity. Here the location of the endpoint can be specified as well as the HTTP method that will be used.

- Store in an HTTP response: successful responses can be stored in HTTP objects.
- Store in a file document: if the response contains binary content (i.e. a PDF), it can be stored in an object which inherits from the FileDocument class of the System.
- Store in a String: if the response is a String, it can be stored in a String variable.
- Do not store in a variable: if the response contains no useful information.

Since the pathfinding service's response is in String format and has to be saved in objects, so as to access the list of points that a route is comprised of, the option in the response handling list is the first one. An Import mapping object is used to define how incoming XML or JSON files can be converted into Mendix objects according to a specific XML or JSON structure. For that reason, a response from the pathfinding web service has been used to build a

JSON structure object. Mendix is able to read the structure of the file and identify the objects it encapsulates as well as their attributes and their data types. With the assistance of this JSON structure object, the Import Mapping is able to create non-persistable Mendix objects in the domain model of the module. Lastly, the response is saved in a variable of type ViaRoute and the microflow returns it as a result.

Now that it has been clarified how Mendix achieves connection with REST endpoints and interpets the results into Mendix objects, the other values of the Via variable in the *GetRoute* microflow can be explored. If the value of the aforementioned variable is *Collection*, it means that the route will be planned from a collection of elements that are already saved on the server where the pathfinding web service is hosted. For that reason, two String variables are created, one that holds the type of elements that will be used for the route planning and the other one is the interval between the elements. The *Call REST* activity utilizes the GET method and returns a result in the same manner as the previous case.

The Via variable has two other values that haven't been explored, *Elements* and *ExtraRoad*. For the *Elements* case, the user requests a route via elements that have already been added on the map raster, which means that the application will include the coordinate pairs of these elements in the request. The coordinates of the elements are retrieved by the microflow named *CoordsOfElements*. This microflow firstly retrieves all the elements of a certain type (specified in the Options by the user) and creates an empty list and an object named MyVia, which holds information about the type of vehicle as well as the interval between the elements. After that, the empty list is populated with the latitude/longitude pairs of the element list and the final activities of the microflow create the start and the end of the route objects, based on the corresponding parameters. Back to the *GetRoute* microflow, the final activity of the *Elements* case, is the *Call REST* one, but in this case the POST method is utilized. Because of that, the Request tab has to include information for the request to the service.

For this case, an *Export Mapping* object is provided as a parameter. As it has been previously mentioned for the Import Mapping objects, the Export Mapping is used to define how Mendix objects can be converted to an XML schema and it is necessary to send data to other systems in formats that can be processed by them. Likewise to the Import Mapping case, a JSON structure is needed so as Mendix can derive the structure of the mapping that has to be exported. This structure holds information about the coordinates of elements on the map raster (added by the user), the starting and ending points of the route, the vehicle that is used and the interval between the elements. The Response tab is similar to the previous two cases, where an Import Mapping object is responsible for turning the response from the web service into Mendix objects.

For the *ExtraRoad* case, a Road object is retrieved from the Options object. This road has been provided in the Options by the user and the route that will be planned will utilize this road. If the retrieved Road object is empty, then the result is an empty Route object. If it is not empty then a microflow named

CreateExtraRoad is called, and this is responsible for creating a Road object based on the road that exists in the Options object as well as the vehicle used. Then, the Start and End objects of the route are created and the new Road object is returned. Again, the *Call REST* activity is called with the POST method. For this case, the Request tab has an Export Mapping object that has been tailored by a custom JSON structure for exporting information about road objects and their attributes. The Response tab is structured similarly to the previous case.

Returning to the *MakeRoute* microflow, the microflow that is called after the execution of the previously described *GetRoute* microflow, is named as *CreateRoute*. This one is responsible for creating an empty route object at first and then retrieving the results of the previous microflow, specifically the coordinates. Based on the minimum and maximum values of the coordinate list, a microflow responsible for creating a bounding box is called within *CreateRoute*. After the creation of the bounding box, the coordinate list is scanned and every entry is saved in *RoutePoint* objects, based on the entry's index and the values for latitude and longitude. This step is necessary because the map widget is able to draw information from Mendix objects in order to present them on the map and the data received from the web service have to be processed in that manner. The newly created *RoutePoint* list is commited and this is the final activity of the *CreateRoute* microflow.

As it has been mentioned before, the results returned by the response of the web service are raw and need first of all, to be transformed into objects that can be edited and accessed by Mendix and furthermore, into meaninful results for the user. The last activities of the *MakeRoute* microflow, use the processed information from the previous activities as well as the information for the bounding box in order to connect the coordinates into a distinguishable line which is on top of an overlay, that is being placed over the map.

The last two modules of the Mendix application are focused on creating, saving or deleting data about routeplanning, specifically when the options of the route creation is about utilizing an extra road or through a collection of elements. The Roads module has functionalities for:

- Adding Start/End of extra road
- Adding an intermediate road
- Adding a connection for an extra road
- Deleting an extra road
- Showing the last created extra road, and
- Saving an extra road.

The Collections module includes two JSON structures that specify the version of the elements retrieved from the database as well as the identification of the elements (where they are located) and type of marking that should be

used on the map (i.e. bench). Based on the JSON structures, import mapping objects have been implemented to transform data from the web service into Mendix objects. The microflows in this module retrieve the up-to-date collections from the web service, the current version of the elements and refresh the already retrieved collections.

6.2 Pathfinding web service

This subsection will provide an insight on how the web service responsible for calculating routes and paths reads a request URL sent by the mobile application described in the previous section, calculates a route and returns it as a list of coordinates to the application. Firstly, the initial setup of the service and its functionality will be described. In the ends of this subsection the current state of the service will be presented.

6.2.1 Initial approach - GraphHopper

The project's requirements included a pathfinding functionality that would be able to calculate paths based on criteria specified by the user. Since Mendix offers functionality for REST services, the development or integration of a similar web service was deemed to be ideal. In the first place, an online pathfinding service was utilized, named *GraphHopper*[2]. GraphHopper is a RESTful pathfinding service that offers route planning, navigation, traffic aware route optimization functionalities and worldwide coverage based on OpenStreetMap data. The GraphHopper route planner is also open source and for utilizing the routing API, an API key is necessary. This key is generated with the creation of a GraphHopper account, which is free of charge. The main URL request that was utilized in the first stages of the pathplanning was the **GET Route endpoint**[3], and one example of this request is structured like this:

```
https://graphhopper.com/api/1/route?
point=52.089696,5.109990
&point=52.091700,5.119260
&profile=bike
&points_encoded=false
&key=api_key
```

Figure 16: GET Route endpoint request.

The *point* query parameters specify the starting and the ending point of a route, the *profile* parameter is about the vehicle that is used and can get

[2]https://www.graphhopper.com/
[3]https://docs.graphhopper.com/operation/getRoute

values based on a Routing matrix, which utilizes routing profiles from OpenStreetMaps[4] (in the project's case, the parameter has values of "bike" and "foot", for bikers and pedestrians). The *points_encoded* parameter is a boolean one and is about changing the encoding of location data in the response, since the default is polyline encoding, which is more compact than a list of points but requires special client code for unpacking. For this project, the list of coordinates is needed so this parameter is always set to false. The last parameter is the API key that was mentioned before.

The results of this query are returned in the format of a JSON file (this is the reason that Mendix was offering JSON structure creation as well as import/export mapping functionalities) and includes a list of coordinates in the form of LineString objects. For the first stages of the routeplanning service, this GET request was mainly used within the Mendix application to plan routes.

However, since the pathplanning process required the calculation of routes from point A to point B, while considering air quality data or specific locations within the city (locations specified by the user or retrieved from a server, i.e. bench locations), a custom pathfinding web service would probably be more helpful than a simple route planner that focuses only on optimizing routes based on distance. With this in mind, and also with the experience of utilizing RESTful services, the author began to implement this service after consultation from the supervisor and the RIVM expert that assisted the author in the first phases of the project.

6.2.2 Pathfinding RESTful service with Spring

The creation of this custom pathfinding service was based on a tutorial from the website of **Spring**, an open source application framework and *Inversion of Control container*[5] for the Java platform. This container provides consistent means for configuration and management of Java objects via *reflection* (ability that allows processes to examine and modify their own behavior). The creation,configuration and the calling of initialization methods of objects is managed by this container. The Spring framework is mainly utilizing the Java programming language but other languages like Kotlin or Groovy can be used as well.

Spring also offers a service that allows users to set up project dependencies, metadata, language preferences and versions of Spring Boot and Java that will be used in the project. Afterwards, users can download the ZIP file that is automatically generated and it is the archive of the web application that is configured with their choices. This service is named **Spring Initializr**[6], and it was used to create the basic dependencies and setup for the project. The pathplanning service utilized Spring Initializr, so as to create the foundation of

[4] https://docs.graphhopper.com/section/Map-Data-and-Routing-Profiles

[5] Inversion of control: custom-written code parts receive control flow from a generic framework, compared to traditional programming that custom code calls into reusable libraries for generic task execution.

[6] https://start.spring.io/

the project. The configurations were the ones below:

- Project: Gradle project (Gradle vesion 7.0)
- Language: Java (Java SDK 16)
- Spring Boot: 2.5.0 (SNAPSHOT) version
- Dependencies: Spring Web

By setting up the foundation of the project, the next step was to create the necessary classes in Java that would be responsible for controlling and representing the data received by the Mendix application as well as structuring the service's replies. The next section will describe the classes, methods and the algorithm that was used to create the first version of the pathplanning service.

6.2.3 Initial Project Structure and Methods

In this part of the paper, the foundation of the service will be presented, along with the tool that was utilized to extract necessary information about road structure in the city of Utrecht initially and then for the cities of Darmstad, Gyor and Athens. Afterwards, the main algorithm that was implemented to calculated the shortest path from point A to point B will be explained as well as the extra methods that were created in order to provide more robust results as far as the two pathplanning criteria are concerned (NO2 concentration and route between multiple element locations). A **UML diagram** will show the relationships between the classes of the initial implementation (see 30). The second criterion regarding the creation of routes between multiple element locations is tackled in the latest iteration of the route planner which will be presented after the description of the initial project.

6.2.4 Resource Representation and Resource Controller Classes

The mindset followed is the same as in the Graphhopper case: a URL with all the necessary information about a route will request the intermediate points from the web service. The service will read the URL, map the provided data to Java attributes, the algorithm will provide the result in the form of a JSON object that contains all the intermediate calculated points of the route.

For that to work, the Spring framework requires two classes, one that will be used to represent the data received from the URL (**Resource Representation**) and the other to read the URL and map data to attributes of the previous class (**Resource Controller**). Specifically, parameters in the URL will specify the latitude/longitude of the starting and ending locations of a route. For that reason, the **AStar** class (**Resource Representation** class in this project's case) was created with attributes such as:

- id: unique identifier for the request
- lat1: starting point latitude

- lon1: starting point longitude
- lat2: ending point latitude
- lon2: ending point longitude
- path: list of doubles, uses pathfinding algorithm to calculate intermediate route points

Proper get/set methods are implemented in order to retrieve or alter the appropriate attributes. The *path* variable utilizes the pathfinding algorithm from the main method of the service in order to save all the route nodes in a list. The **AStarController** class is the **Resource Representation** class. This class is utilizing the *RestController* and *GetMapping* annotations of Spring, with the latter ensuring the mapping of HTTP GET requests to /aStar are mapped to the aStar() method (see figures below). The *RestController* annotation is used to mark this class as a controller which returns a **domain object** (instance of a class related to the application's domain) instead of a view). The values of the query string parameters (lat1,lon1,lat2,lon2) are bound to the parameters of the aStar object with the *RequestParam* annotation. The *id* value is automatically generated every time the controller receives an HTTP GET request. In that manner, the client's request is automatically mapped to an object, which can be easily converted to a JSON object, with the assistance of Spring's HTTP message converter support.

```
http://localhost:8181/aStar?startX=52.0885&startY=5.1262&endX=52.0742&endY=5.1523
```

```
@RestController
public class AStarController
{
    private final AtomicLong counter = new AtomicLong();
    @GetMapping("/aStar")
        public AStar aStar(@RequestParam(value = "lat1") double lat1,
                 @RequestParam(value = "lon1") double lon1,
                 @RequestParam(value = "lat2") double lat2,
                 @RequestParam(value = "lon2") double lon2)
    {
           return new AStar(counter.incrementAndGet(),lat1,lon1,lat2,lon2);
        }
}
```

Figure 17: Initial URL format along with the structure of the AStarController class, which maps the corresponding parameters from the URL to the attributes of the AStar object.

6.2.5 Pathfinding Algorithm

The algorithm that was implemented to calculate the distances between the various points of the route is the **A* algorithm**. What sets it apart from other

similar algorithms, is the *heuristic* that is utilized for searching for the next route point (point's distance from the end point of the route). This algorithm constantly compares the distance of a point from the starting position and the ending position, with the latter having priority because of the heuristic. This algorithm not only prioritizes points that are close to the endpoint of the route but also re-calculates distances between neighboring points, if a shorter path is discovered between them. In that way, shorter paths between neighbors are constantly discovered and saved.

For the implementation of the algorithm into the service, an *open list* is required, which will keep a set of discovered points that can be (re-)expanded. This open list has been implemented as a *priority queue* which uses the *Comparator* Java interface. This interface has a function that compares two objects and returns:

- negative integer (-1), if the first object is less than the second
- zero, if both objects are equal
- positive integer (+1), if the first object is greater than the second

A class named as *NodeComparator* includes this function, which has been overridden, in order to compare route nodes, based on their distance from the starting and the ending point of the route. The sum of these distances is named as the *F-score*, and this is used to compare the two nodes. This class is utilized from the priority queue of the algorithm, in order to give priority to nodes that are closer to the endpoint of the route. The *Euclidean distance* was used as a distance measure between nodes.

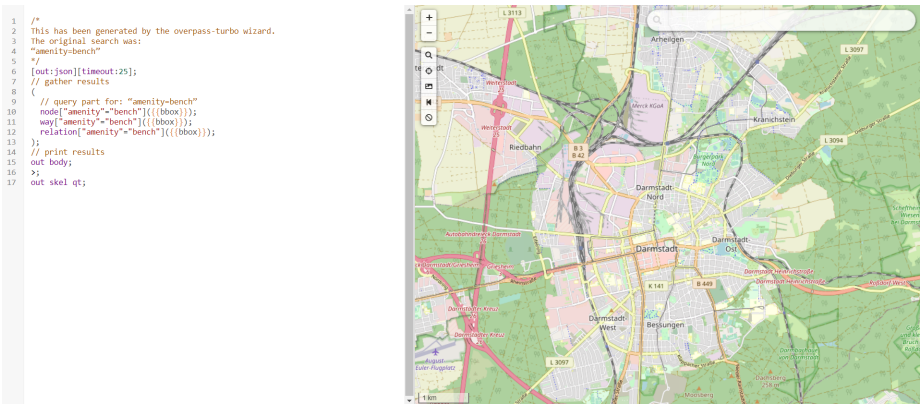
In the following part, specifically 6.2.7, the NodeComparator class is also taking the value of the NO2 concentration into consideration, about selecting a Node to be placed on the front of the queue. In that manner, distance as well as the volume of pollution in a location play an important role in which nodes are prioritized to be selected.

6.2.6 Mapping and Nodes

Before the implementation of the A* algorithm, an interconnected network of nodes is needed for the algorithm to be able to draw information about locations and calculate the distances between nodes, as well as combine them into routes. For this reason, a tool named **Overpass Turbo** was used[7]. This is a data mining tool, designed to retrieve information from OpenStreetMap, based on queries performed with the assistance of a query wizard, which uses OpenStreetMap tags. This was an iterative process since the query containing tags related to bikelanes proved to be fractured and not connected. During the testing phase of the service, routes were either incomplete or comprised of different parts. This issue made clear that the road mapping should be consistent and unified, which meant that a combination of different OpenStreetMap tags

[7] https://overpass-turbo.eu/

in a single query. After various tries and research through the forums of OpenStreetMap, an appropriate query was found that provided an interconnected network of nodes. The result was a JSON file that included all information about the bikelanes in the city of Utrecht.



```
/*
This has been generated by the overpass-turbo wizard.
The original search was:
"amenity=bench"
*/
[out:json][timeout:25];
// gather results
(
  // query part for: "amenity=bench"
  node["amenity"="bench"]({{bbox}});
  way["amenity"="bench"]({{bbox}});
  relation["amenity"="bench"]({{bbox}});
);
// print results
out body;
>;
out skel qt;
```


Figure 18: Overpass Turbo: a data mining tool for retrieving information from OpenStreetMap. Users can find the place they want to retrieve information on the map on the right side of the page (bounding box), and use the Query Wizard to create a query, based on OpenStreetMap tags.

The next part was to retrieve the node information (mainly latitude/longitude coordinates). The core structure of the JSON file is the one below:

- **Elements**: the collection of the data retrieved from OverpassTurbo. Data are separated into **types**.
  - **Way**: *Way ID* (String),*list of Node IDs* that comprise the way (List of Strings), and *Tags* that contain information about the way.
  - **Node**: *Node ID*, *latitude* and *longitude* coordinates.

```
// sharrow
way["cycleway"="shared_lane"]({{bbox}} ) ;

// CONVENTIONAL BICYCLE LANE
// lane on both sides of the street
way["cycleway"="lane"]({{bbox}});

// lane on either side
way["cycleway:left"]({{bbox}});
way["cycleway:right"]({{bbox}});

// dedicated bikeways
way["highway"="cycleway"]({{bbox}});

way["bicycle"="designated"]({{bbox}});
```

```
{
  "type": "way",
  "id": 2161182,
  "nodes": [
    7740068,
    13877901
  ],
  "tags": {
    "highway": "footway",
    "loc_name": "Wittestenensteeg",
    "name": "Jutfaseweg",
    "source": "survey"
  }
}
```

```
{
  "type": "node",
  "id": 3956709980,
  "lat": 52.0232895,
  "lon": 5.0426460
}
```

Figure 19: OpenStreetMap tags used in a query for retrieving bikelanes and structures of Way and Node elements in the resulting JSON file.

Methods in the main method of the service are responsible of parsing the JSON file and saving all information about node IDs, latitude and longitude, by creating Node objects (based on the Node Java class). The Way element is utilized mainly for assigning neighbors to each Node object, since it has a list attribute where all neighbors are stored. In that manner, Node objects are created and saved in a *dictionary* data structure, where the key for each entry is the *Node ID*. In that manner, this data structure holds geographical information about the bikelane structure of the city of Utrecht and it is easily adjustable to new changes, i.e. NO2 concentration which will be explained in the following part.

6.2.7 RIVM Sensor Data

Through the Mendix project, the website responsible for providing air quality information for the overlay image creation was used in this part of the route planning service. These data are in the form of a specific data structure, named **Esri ASCII grid** [8]. This is a raster file format specifically for encoding geographical information. This format describes a geographical place in the form of an array of equally sized square grid points that are arranged in rows and columns. Each grid point has a value that describes a geographical characteristic (slope, elevation, in this case the NO2 concentration). The cells of the grid can be referenced by their coordinate location. The Esri grid also includes information about the format of the data (header information) at the first six lines, such as number of rows/columns and more importantly, information about the coordinates of the origin of the raster. The latter, however are still in the RD coordinate system, which is widely used by the RIVM. In the main method, the website is parsed for the information and the raster's origin coordinates are retrieved and converted to the WGS84 coordinate system and data from the sensors are saved in a list.

Since the Node dictionary mentioned in the previous part has to be updated with the air quality information, an extra attribute in the Node class has been added regarding the *NO2 concentration* at that point. Since each cell in the Esri ASCII grid is representing an actual location, a method in the main class of the service is responsible for retrieving the NO2 concentration for a specific coordinate pair. As soon as a coordinate pair is parsed from the Overpass Turbo JSON file, this method is called to provide the concentration of NO2 in that spot, and is saved in the newly created Node object. In that manner, the Node dictionary holds all information necessary for providing routes with the NO2 concentration as a criterion.

6.2.8 How the service works now

The latest version of the service still utilizes the Spring framework's resource controllers and representation classes, now with some extended functionality. New classes have been added for route requests specifically for uploading element locations and requesting a route between those and another one for utilizing element locations that are already saved in the service's server. Special classes in Java have been implemented in order to save data about locations on earth, since it includes attributes for latitude/longitude, as well as functions for euclidean and *geodesic distance*[8]. These functions, in combination with a JSON parser that draws information about element locations extracted from Overpass Turbo, are utilized to precompute the aforementioned locations, which can be later used to facilitate the route planning procedure, as far as saving time is concerned.

In addition, multiple programs have been implemented and integrated in **C++**. Specifically, the *A* algorithm*, which calculates the paths based on the information retrieved from the URLs and other programs that assist in the calculation and selection of candidate nodes for the final path have also been implemented in the aforementioned programming language. Programs that facilitate in inspecting element lists or route lengths, uploading element collections on the service, computations of routes via multiple collections and also a localized version of the Graphhopper service are included in the C++ module of the service's new version.

Since multiple users will have to use the application, a module that focuses on creating passwords and assigning to accounts has been developed in **Rust**. Furthermore, a *dbus server*, short for *Desktop Bus*[9] for uploading functionalities. The initialization program of the dbus server registers the name of the URL used for the calls and creates a path for the objects to be passed from the URL. The objects implement an interface which has an Upload method. Another program in Rust opens a connection to the service bus and after that a wrapper structure is utilized so as to ease the sending of method calls to a specific destination and path.

[8] Geodetic distance is the length of the shortest curve between those two points along the surface of the earth model being used by the spatial reference system. Geodetic distance behaves well for wide areas of coverage, and takes the earth's curvature into account.

[9] Message-oriented middleware mechanism that allows communication between multiple processes running concurrently on the same machine.

7 Results

7.1 Data Preparation and Participants

For the experiment, 25 participants contributed in providing the data needed for the statistical analysis and the drawing of the results. Before analyzing the collected responses, the raw data had to be processed and properly prepared. Coordinate responses had to be separated into latitude/longitude pairs, and the corresponding displacement value had to be calculated. Responses were split and categorized for each questionnaire section and examined separately through Google Sheets. A couple of outliers were spotted during the data cleanup, but since the sample is small, it was decided not to remove them. Separate sheets were used for the splitting process of the data, and a *CSV* file was generated with the displacement values, in order to be imported to **R Studio**, to facilitate the statistical analysis of the accuracy of the input methods. Google Sheets had sufficient tools for the calculation of the **System Usability Scale** questionnaire, as well as the final three questions of the experiment questionnaire, which were based on a Likert scale of 5 levels and focused on how users perceived the MEES app overall based on these three factors:

- MEES functionality,
- User experience with MEES,
- How likely it is for users to recommend MEES to others.

The next subsections will provide more insight about the calculations and tests that were utilized in order to draw meaningful results about the accuracy of the methods as well as how users perceived the methods and the MEES application.

7.2 Accuracy Results

As it has been mentioned before, the displacement values had to be calculated before the application of any statistical tests. Based on the separated latitude/longitude pairs of the replies and on the predefined coordinates, a simple **Euclidean distance** function in Google sheets provided the necessary data to be analyzed. 25 displacement values for each method were created, so 75 displacement values in total. The first chart below (figure 20) shows the individual answers from each participant. It is also noteworthy to see which method provided the minimum displacement for each participant too, so the pie chart shows which method provided the lowest displacement value for each case (figure 21).

Since there are three different groups of data, in order to understand which method has the lowest displacement values, thus being more accurate than the other two, these groups have to be compared between them to see if there are any significant differences and if there is a reason for them to be compared. The first step is to test whether the data follow a normal distribution. The test that

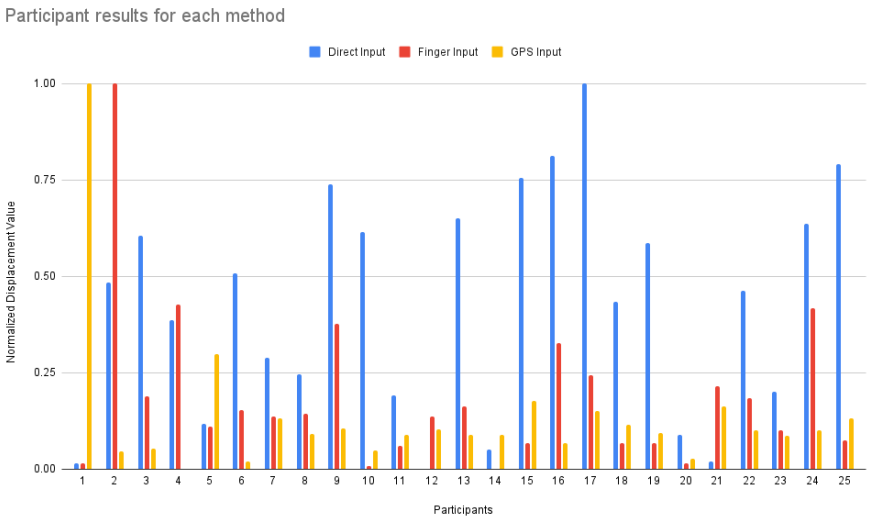


Figure 20: Results from each individual participant.

Shapiro-Wilk test for normality	*W*	*p-value*
Direct Input	0.94974	0.2474
Finger/Cursor Input	0.72791	0.00001772
GPS Input	0.46871	0.00000001863

Table 1: P-values calculated by Shapiro-Wilk test for normality.

was used for this process is the **Shapiro-Wilk test for normality** and two hypotheses have to be stated, the null hypothesis and the alternate one:

- H0: Direct/Finger/GPS displacement data are normally distributed
- HA: Direct/Finger/GPS displacement data are not normally distributed

Based on literature ([16]), the appropriate **alpha level** should be set to **0.05**. The test results for all groups are presented in the table below.

The p-value that is calculated is compared to the selected alpha level. If the p-value is **greater** than the latter then the alternate hypothesis can be rejected, which means that the data in the group that was tested are normally distributed, otherwise the null hypothesis is rejected, meaning that the data in the group that was tested are not normally distributed. Comparing the p-values from each group to the alpha level, the normality of the data for each group has been concluded, along with their distributions (figure 22):

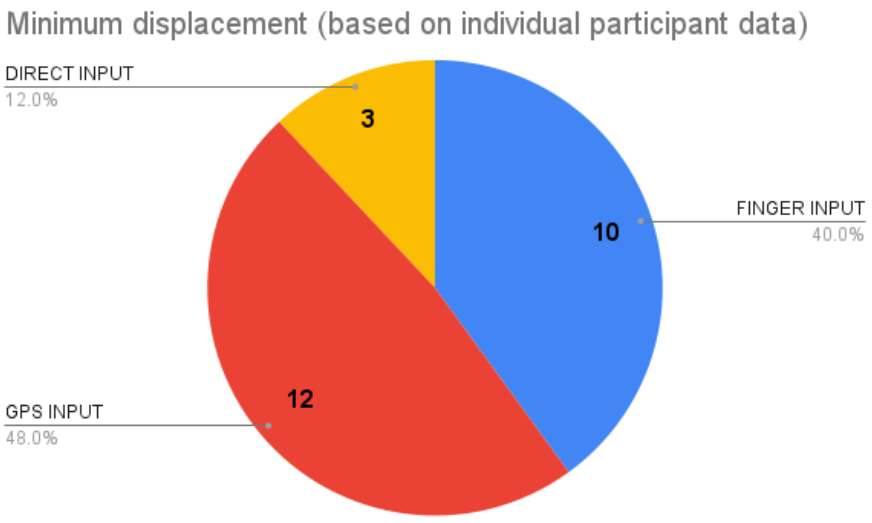


Figure 21: Minimum displacement for each participant case.

- For Direct input group: H0 is accepted and HA is rejected, so data in this group are normally distributed.
- For Finger/Cursor input group: H0 is rejected and HA is accepted, so data in this group are not normally distributed.
- For GPS input group: H0 is rejected and HA is accepted, so data in this group are not normally distributed.

The Shapiro-Wilk test has shown that two of the groups that have to be compared between them do not follow a normal distribution and one of them does. In this case, a **t-test** could not be used in order to see if there are any significant differences between those three groups. The alternative statistical test that can be used in this case is the **Wilcoxon signed rank test**. This is a non-parametric alternative to the t-test, and is usually used when the compared data groups do not follow a normal distribution. This test is focused on finding differences between the median of the compared groups. The general null and alternate hypotheses have been set in such manner as described below:

- H0: The median difference between Direct and Finger/Direct and GPS/ Finger and GPS displacements is zero
- HA: The median difference between Direct and Finger/Direct and GPS/ Finger and GPS displacements is not zero

In total, six tests were performed, two for each group comparison. The reason for that is that this test provides two ranked sums (named V in R, W in

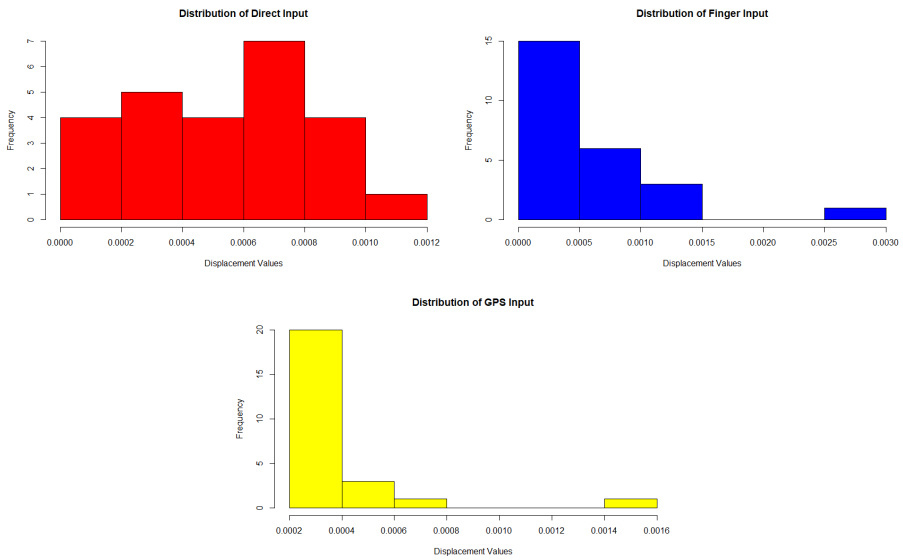


Figure 22: Distributions of data of all three groups.

Wilcoxon Signed Ranked Test no.1	*V*	*p-value*
Direct vs. Finger/Cursor	179	0.6721
Finger/Cursor vs. Direct	**146**	**0.6721**

Table 2: P-values calculated by Wilcoxon Signed Ranked Test between Direct and Finger/Cursor input groups.

statistics) and the smallest sum is selected to provide the test statistic that will in turn provide the appropriate p-value. For example, the Direct The tables below show the group comparisons and the p-values in bold, show which ones were selected to draw results about the differences between the groups. Again, the alpha level that was selected is 0.05. For this test, if the p-value that is calculated is **lower** than the selected alpha level, then the null hypothesis can be rejected, otherwise the alternative hypothesis is rejected.

The result from test no.1 (table 2) is that with a test statistic *W= 146* and a *p-value = 0.7*, the alternate hypothesis for this test is rejected and the null hypothesis is accepted, which means that there is no significant difference between the medians of those groups.

The result from test no.2 (table 3) is that with a test statistic *W= 103* and a *p-value = 0.1*, the alternate hypothesis for this test is rejected and the null hypothesis is accepted, which means that there is no significant difference between the medians of those groups.

The result from test no.3 (table 4) is that with a test statistic *W= 66* and a

Wilcoxon Signed Ranked Test no.2	*V*	*p-value*
Finger/Cursor vs. GPS	222	0.1135
GPS vs. Finger/Cursor	**103**	**0.1135**

Table 3: P-values calculated by Wilcoxon Signed Ranked Test between Finger/Cursor and GPS input groups.

Wilcoxon Signed Ranked Test no.3	*V*	*p-value*
Direct vs. GPS	259	0.008069
GPS vs. Direct	**66**	**0.008069**

Table 4: P-values calculated by Wilcoxon Signed Ranked Test between Direct and GPS input groups.

p-value = 0.008, the alternate hypothesis for this test is accepted and the null hypothesis is rejected, which means that there is a significant difference between the medians of those groups.

Based on the results of the Wilcoxon signed ranked tests, the only groups that proved to have a significant difference between them are the *Direct input group* and the *GPS input group*. Figure 23 shows the displacement values based on each participant, and it can be easily seen that in the majority of cases, the displacement values of the GPS method for each participant are lower compared to the equivalent Direct method. Figure 24 compares the two histograms of the displacement values for both methods. Again, it is clear that the majority of these values for the GPS method are between zero and 0.0005, whereas the Direct input displacement values are gathered mainly in the range between 0.0005 and 0.0010. This means that in terms of *accuracy*, **the GPS method is more accurate than the Direct input method**.

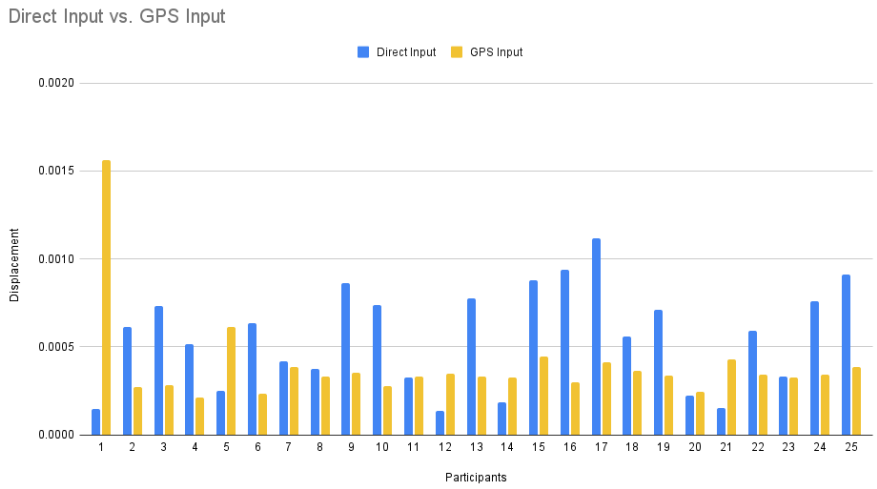


Figure 23: Direct input versus GPS input results for each individual participant.

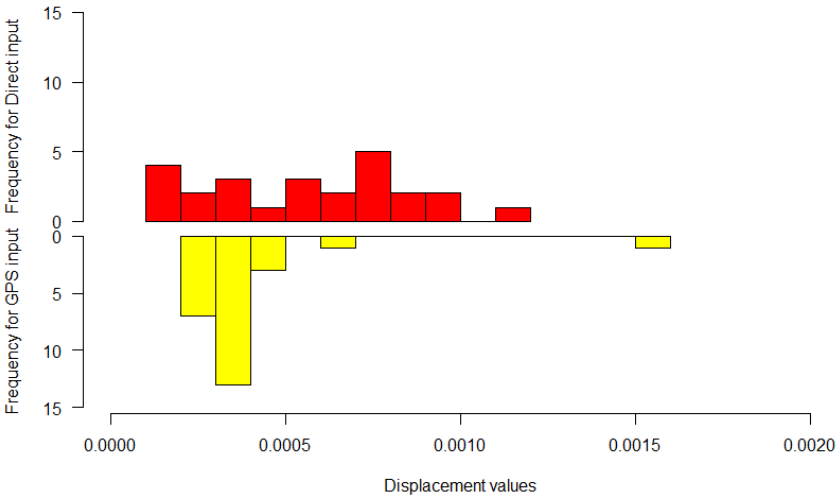


Figure 24: Direct input versus GPS input histograms compared.

7.3 System Usability Scale Results

The next part of the questionnaire is focused on retrieving answers about the way users perceive the method, specifically about the effectiveness, efficiency and satisfaction. As it has been mentioned in the Experiment section (see section 5, specifically 5.5), the **System Usability Scale** (or **SUS** for short) questionnaire has been adjusted to the needs of the current experiment. Before proceeding to the calculation of the scores, the results were cleaned up and separated into different sheets for each individual method. Firstly, the answers have to be set up in the manner of a five-scale Likert type scale, which looks like this:

- 1 - Strongly Disagree
- 2 - Mildly Disagree
- 3 - Not Agree/Not Disagree
- 4 - Mildly Agree
- 5 - Strongly Agree

The calculation of the data follows this flow:

- Calculate the sum of the odd-numbered questions and subtract 5 from the total to get X.
- Calculate the sum of the even-numbered questions and subtract that total from 25 to get Y.
- Add these two values and multiply the result with 2.5

The result from these calculations is an SUS score out of a 100. It has to be stated that this number is **not a percentage score** but a total score out of a 100. This score then is compared to the figure below (see figure **??**), so as to see in which category the score corresponds to. The average score of this test is 68/100, and results below or above this score can provide an immediate insight into the overall usability of the design solution. The results from applying these calculations to the participant data are shown in the tables below. The tables show the statistical data for each input group as well as the number of grades for each category of the questionnaire.

It can be understood from the table, that the means of each method are all below the threshold of 68/100 and also no methods have results that fall under the categories of A, B or C. This means that the methods definitely require a workaround or refactoring in regards to usability issues as well as if they are over complicated for users to use in various tasks.

SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Figure 25: SUS acceptability score.

Methods	*Direct Input*	*Finger/Cursor Input*	*GPS Input*
Mean	50.8	50.9	45.3
Standar Deviation	4.932882862	3.671398462	5.016638981
Minimum	40	45	37.5
Maximum	60	60	55
A	0	0	0
B	0	0	0
C	0	0	0
D	11	10	3
F	14	15	22

Table 5: SUS acceptability score and statistical data of the Direct Input method.

7.4 Likert-scale Results

The last section of the questionnaire has three questions regarding the functionality of MEES, the experience of the participants with MEES as well as how likely it is for them to recommend it to someone else. For these questions a five-point Likert type scale was used, similar to the one that was used for the SUS. Based on Likert scale analysis literature, mean values had to be categorized into different levels ([27]). The range of each level is calculated by using the formula below:

- (highest point - lowest point)/ number of levels used

which in our case is:

$$(5 - 1)/5 = 0.8$$

Level	Description
1 - 1.8	Strongly Disagree
1.9 - 2.7	Mildly Disagree
2.8 - 3.6	Not Agree/Not Disagree
3.7 - 4.5	Mildly Agree
4.6 - 5	Strongly Agree

Table 6: Levels of Likert scale answers.

Factors	*Functionality*	*Experience*	*Recommendation*
Strongly Disagree	0	1	0
Mildly Disagree	5	3	12
NA/ND	10	0	10
Mildly Agree	9	5	2
Strongly Agree	1	1	1
Sentiment Level	3.24	3.08	2.68

Table 7: Results from the Likert-scale calculation process.

The table after the Likert-scale levels (table 7) shows the sum of all levels of the answers that were provided as well as the overall sentiment level for each category.

The overall sentiment level is calculated by adding the individual sentiment levels and dividing them by the number of categories and the final overall sentiment of the participants for the MEES application is falling under the middle Likert-scale level (Not Agree/Not Disagree). Similar to the overall sentiment level, the levels of Functionality and Experience categories fall under the same Likert-scale level, with the Recommendation category being in the Mildly Disagree category.

The list below has a summary of all the results:

- **Functionality**
  - The displacement data for the three input groups (Direct, Finger/Cursor and GPS) have been statistically analyzed and significant differences were found *between the Direct and GPS input groups*, with the **GPS method being more accurate compared to the Direct method**.
- **User Experience**
  - The **System Usability Scale questionnaire** was used in order to understand how participants feel about using the Direct, Finger/Cursor and GPS input methods. Based on the **SUS acceptability score**, **all methods scored under the category of F**.
- **MEES Sentiment Level**
  - According to the last three questions of the questionnaire, which are based on the **Likert-type scale**, the **overall sentiment feeling** of participants about the application is **under the middle of the Likert-scale levels, specifically at Not Agree/Not Disagree**.

8 Concluding Remarks

This section will provide more insight about the results of the experiment mentioned in the previous section, as well as how the research questions have been met, based on the results. Furthermore, limitations and issues that have been encountered during the implementation and the experiment will be mentioned, so as to shed more light on how these issues influenced the progress of this thesis project. Moreover, possible future work extensions will be mentioned and suggested in order to enhance and improve the overall functionality of the MEES application.

8.1 Results and Research questions

For the *Functionality* category, based on the two hypotheses that have been set in subsection 4.1, and with the interpretation of the results from the experiment, **the Null Hypothesis no.1** can be rejected, since a significant difference has been found between two of the displacement groups, with the GPS method having the majority of lower displacement values compared to the Direct method, thus making the GPS method more accurate than the Direct one. No significant differences have been noticed between the Direct and Finger/Cursor input groups, as well as between the Finger/cursor and GPS input groups based on the statistical analysis, which proves that there is no reason for comparing groups that have no significant difference between them. In order to summarize about the first category of research questions, the **Research question no.1** can be answered effectively.

For the **User Experience** category, by reviewing the Null Hypothesis no.2 and Research question no.2 in combination with the results from the SUS questionnaire, it can be understood that the **Null Hypothesis no.2** can be rejected. The results have shown that participants were not indifferent to the three methods based on how easy it is to add elements on the map raster, how intuitive the methods are and how effectively the tasks are executed. To answer **Research question no.2**, participants were not satisfied with the three methods, since all of them scored on the lowest rank of the SUS acceptability table.

8.2 Limitations

As far as limitations are concerned, the vast majority of the issues had the application framework as the main suspect. Despite having a significant swift build time as well as fast deployment of demo mobile applications, Mendix has proven to be ineffective during the implementation of this project. In most cases, Mendix offers basic functionalities for simple mobile applications, and for more complex or custom functionalities, developers usually have to end up writing their own custom code in Java or Javascript, or implement their own REST services, as it happened with this project too. Furthermore, the documentation for numerous console errors and also for widgets proved to be lacking in multiple aspects (i.e. list of possible errors, extended functionalities etc.). As far as

the map widget is concerned, based on the Mendix documentation page, it is still under development, and also it has limited functionalities compared to map widgets from other frameworks. All these contributed to the classic "trial and error" process, which took a significant amount of implementation time. Despite lacking in documentation, Mendix has an active community forum where developers can post questions and other users answer them. Various posts of the author concerning bugs or runtime errors about the framework were answered sufficiently based on the community's answers.

Furthermore, the application's UI had some uniformity issues when it was running on different mobile phones. For example, for some old IPhone models, some UI parts could not even load, and for the majority of the participants, the Map widget was not loading properly at first. It is also worth mentioning that the MEES application is a free application on the Mendix cloud which means it will go to sleep mode after an hour of inactivity. Many participants faced this issue, which led them to restart the application and some of them did not do the experiment at first. Moreover, despite the clear instructions on the questionnaire about the way the three methods should be used, participants still faced some trouble mainly because of the way the UI is structured. These issues led to the author being present during all the experiment phases and explaining to the participants how they can complete the first part of the questionnaire. Moreover, the majority of test cases before the experiment deployment were conducted mostly on the desktop version of the application through an Internet browser.

The non-intuitive and buggy UI in combination with the low participant number (25 instead of 30, which was initially set) are definitely two of the reasons why the three methods scored so low on the SUS questionnaire. According to literature, sample sizes of n=30 tend to have more normal mean distribution plus the fact that more participants would provide a wide variety of answers and subsequently a clearer image on how the three methods are perceived from users.

8.3 Future work possibilities

A possible future enhancement for the MEES application could be the refactoring and rearrangement of the user interface, as the SUS questionnaire made abundantly clear.The opinion as well as the work a UI/UX expert could prove beneficial for MEES, since there will be more focus on creating an intuitive, user-friendly, effective, efficient and in addition uniform interface for all mobile devices. Experts on this field could definitely enhance the application's interface in a more user-friendly manner. Furthermore, such experts will be able to find what is wrong with a user interface, a method, a menu or a sidebar in a much more efficient manner. Also, the new functionalities and the refactored UI should be tested more on mobile devices of both mobile operating systems instead of desktop computers, since the devices have proved to be more complex in the way they represent the application's interfaces or functionalities not to mention the unique bugs that could appear on mobile devices. With the renewed UI, a second benchmark summative usability test should be performed, similar to the experiment of this paper, so that new answers can shed light on how the reformed interface and subsequently the input methods affect the ease-of-use, efficiency and intuitiveness.

References

[1] S.Sas A.El Ali and F.Nack. "Photographer paths: Sequence alignment of geotagged photos for exploration-based route planning". In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW* (Feb. 2013), pp. 985–994. DOI: 10.1145/2441776.2441888.

[2] M.Linaza A.Garcia O.Arbelaitz, P.Vansteenwegen, and W.Souffriau. "Personalized Tourist Route Generation". In: *ICWE 2010: Current Trends in Web Engineering* 5.3 (July 2010), pp. 486–497. DOI: 10.1007/978-3-642-16985-4_47.

[3] A.S.Drigas and P.Angelidakis. "Mobile applications within education: an overview of application paradigms in specific categories". In: *International Journal of Interactive Mobile Technologies* 11.4 (2017), pp. 17–29. DOI: https://online-journals.org/index.php/i-jim/article/view/6589.

[4] A.Waschk and J.Krueger. "Automatic route planning for GPS art generation". In: *Computational Visual Media* 5.3 (Sept. 2019), pp. 303–310. DOI: 10.1145/2441776.2441888.

[5] Nikolaos M. Avouris and Nikoleta Yiannoutsou. "A Review of Mobile Location-based Games for Learning across Physical and Virtual Spaces". In: *J. Univers. Comput. Sci.* 18 (2012), pp. 2120–2142.

[6] "Business Process Model and Notation". In: (Jan. 2014). DOI: https://www.omg.org/spec/BPMN/2.0.2/.

[7] Alin Zamfiroiu Cătălin Boja. "Input Methods in Mobile Learning Environments". In: (Dec. 2013).

[8] ArcGIS Resource Center. *Esri ASCII Grid format*. URL: https://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/esri-ascii-raster-format.htm.

[9] A.Sanchez E.Redondo D.Fonseca and I.Navarro. "Mobile learning in the field of Architecture and Building construction. A case study analysis". In: *Revista de Universidad y Sociedad del Conocimiento* 11.1 (2014), pp. 152–174. DOI: http://rusc.uoc.edu/rusc/ca/index.php/rusc/article/view/v11n1-redondo-fonseca-sanchez-navarro.html.

[10] Erasmus+. *What is Erasmus+?* URL: https://ec.europa.eu/programmes/erasmus-plus/about_en. (accessed: 18.06.2020).

[11] M.Emanuel F.Schipper and R.Oldenziel. "Sustainable Urban Mobility in the Present, Past, and Future." In: *Technology and Culture 61* 1 (2020), pp. 307–317. DOI: 10.1353/tech.2020.0004.

[12] Roy Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_2.

[13] H.Hochmair. "Towards a Classification of Route Selection Criteria for Route Planning Tools". In: *Developments in Spatial Data Handling* (2005), pp. 481–492. DOI: https://doi.org/10.1007/3-540-26772-7_37.

[14] Govert Strang van Hees. *Globale en lokale geodetische systemen*. URL: https://ncgeo.nl/index.php/nl/publicaties/groene-serie/item/2349-gs-30-g-strang-van-hees-globale-en-lokale-geodetische-systemen.

[15] J.Ortúzar. "Sustainable Urban Mobility: What Can Be Done to Achieve It?" In: *Journal of the Indian Institute of Science* (Oct. 2019). DOI: 10.1007/s41745-019-00130-y.

[16] James R. Lewis Jeff Sauro. *Quantifying the User Experience*. 2012. ISBN: 9780123849687. DOI: https://www.sciencedirect.com/book/9780123849687/quantifying-the-user-experience.

[17] Matthew Keegan. *Shenzhen's silent revolution: world's first fully electric bus fleet quietens Chinese megacity*. URL: https://www.theguardian.com/cities/2018/dec/12/silence-shenzhen-world-first-electric-bus-fleet. (accessed: 18.06.2020).

[18] Samuli Laato, A.K.M. Najmul Islam, and Teemu Henrikki Laine. "Did location-based games motivate players to socialize during COVID-19?" In: *Telematics and Informatics* 54 (2020), pp. 101458–101458.

[19] Dale Leorke. *Location-Based Gaming*. 2019. ISBN: 978-981-13-0683-9. DOI: https://doi.org/10.1007/978-981-13-0683-9.

[20] M.Ally and J.P.Blasquez. "What is the future of mobile learning in education?" In: *Revista de Universidad y Sociedad del Conocimiento* 11.1 (2014), pp. 142–151. DOI: http://rusc.uoc.edu/rusc/ca/index.php/rusc/article/view/v11n1-ally-prieto.html.

[21] Öner Özdemir. "Coronavirus Disease 2019 (COVID-19): Diagnosis and Management". In: *Erciyes Med. J.* 42(3) (2020), pp. 242–247.

[22] Janene Pieters. *DUTCH RAILWAY HANDLES 1.3 MILLION TRAVELERS PER WORKING DAY*. URL: https://nltimes.nl/2019/07/04/dutch-railway-handles-13-million-travelers-per-working-day. (accessed: 18.06.2020).

[23] Michael Pratt. *Difference Between REST and HTTP*. URL: https://www.baeldung.com/cs/rest-vs-http.

[24] S.Taneja and A.Goel. "Mobile applications in educational institutions". In: *Proceedings - 2015 IEEE International Conference on Computational Intelligence and Communication Technology* 1 (2015), pp. 789–794. DOI: 10.1109/CICT.2015.148.

[25] Team SEEMLESS. *Project MEES*. 2018-2019.

[26] Margaret Simpson. *Locomotion No. 1, George Stephenson and the world's first public railway.* 2014. URL: https://maas.museum/inside-the-collection/2014/12/24/locomotion-no-1-george-stephenson-and-the-worlds-first-public-railway/. (accessed: 18.06.2020).

[27] Karl L. Wuensch. *What is a Likert Scale? and How Do You Pronounce 'Likert?'* 2005. URL: http://core.ecu.edu/psyc/wuenschk/StatHelp/Likert.htm.

[28] B.Magoutas .Anagnostopoulou .Bothos and J.Schrammel. "Persuasive Technologies for Sustainable Urban Mobility". In: (Apr. 2016).

APPENDIX

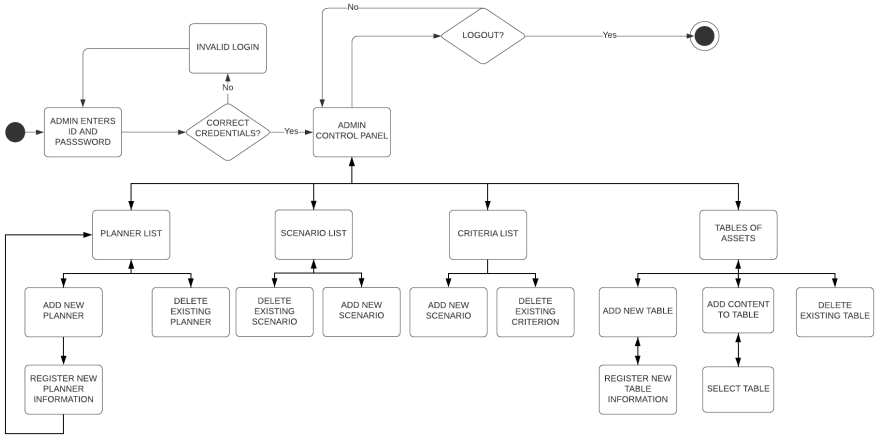


Figure 26: Administrator activity diagram.

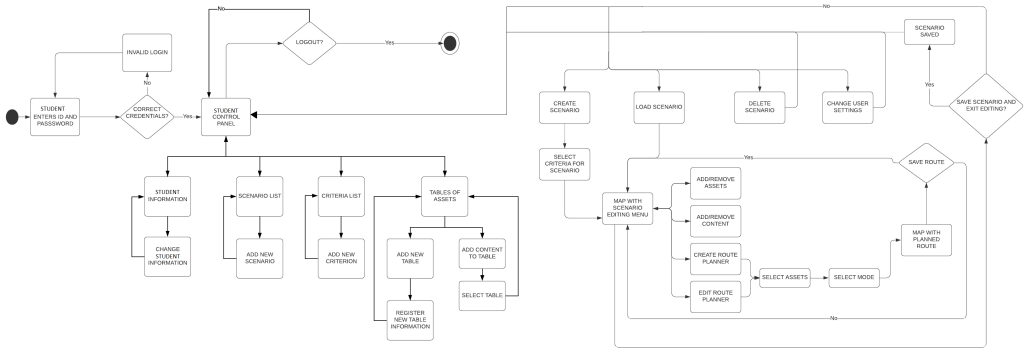


Figure 27: Planner activity diagram.

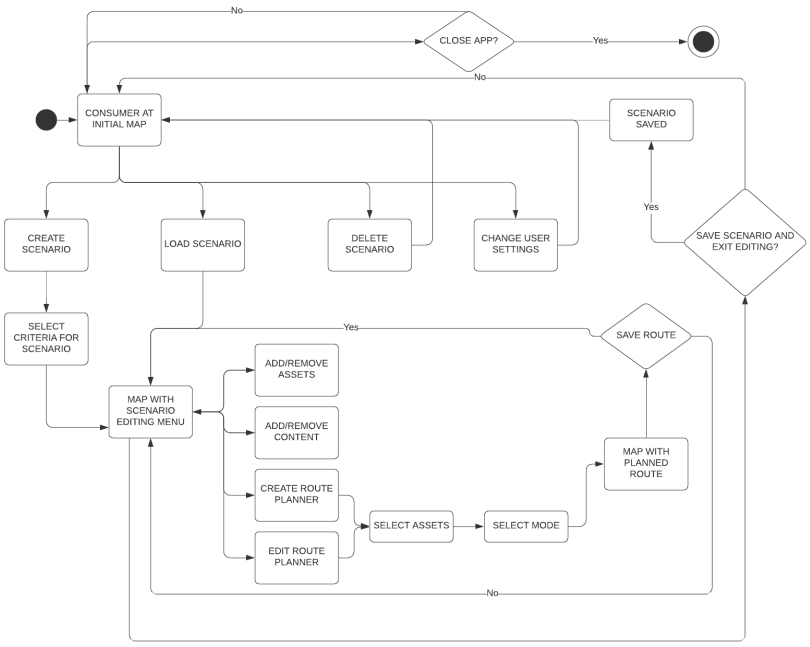


Figure 28: Consumer activity diagram.

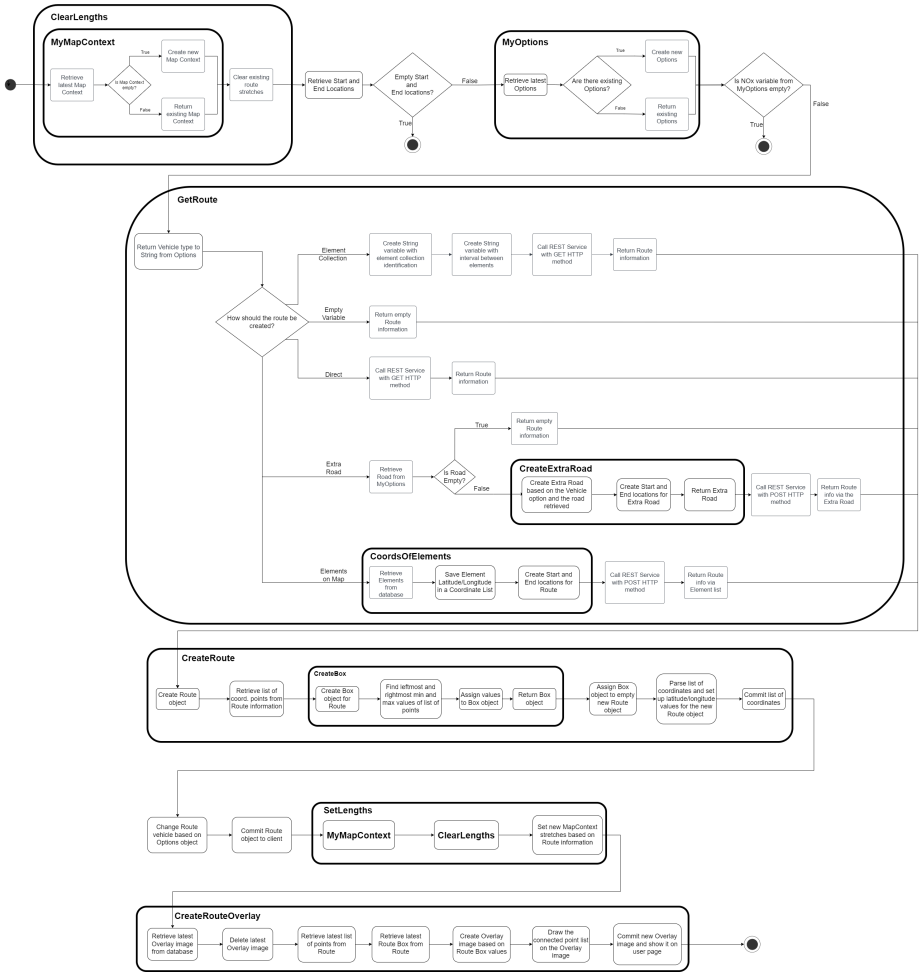


Figure 29: Diagram of the MakeRoute microflow, an essential microflow for the route planning process.

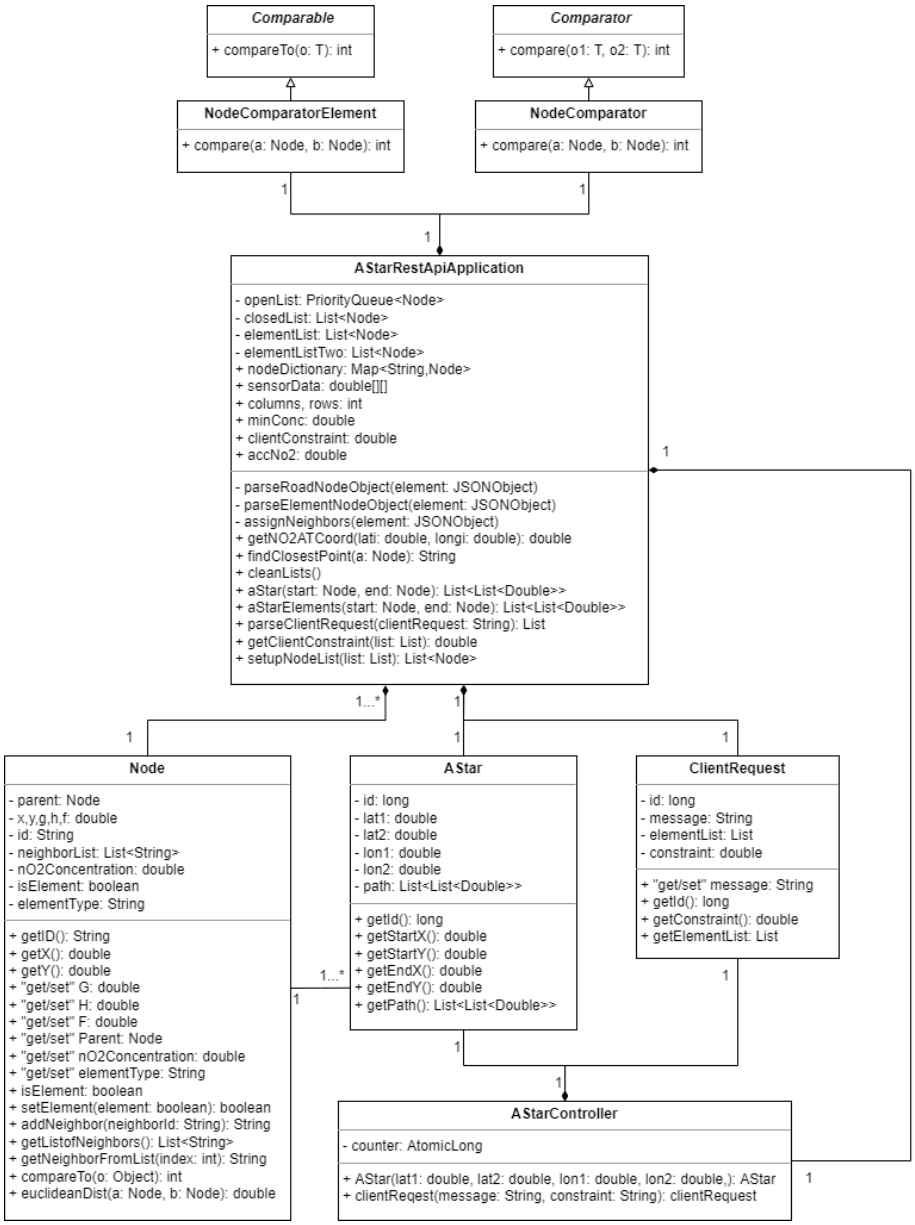


Figure 30: UML diagram of the initial implementation of the pathfinding web service.