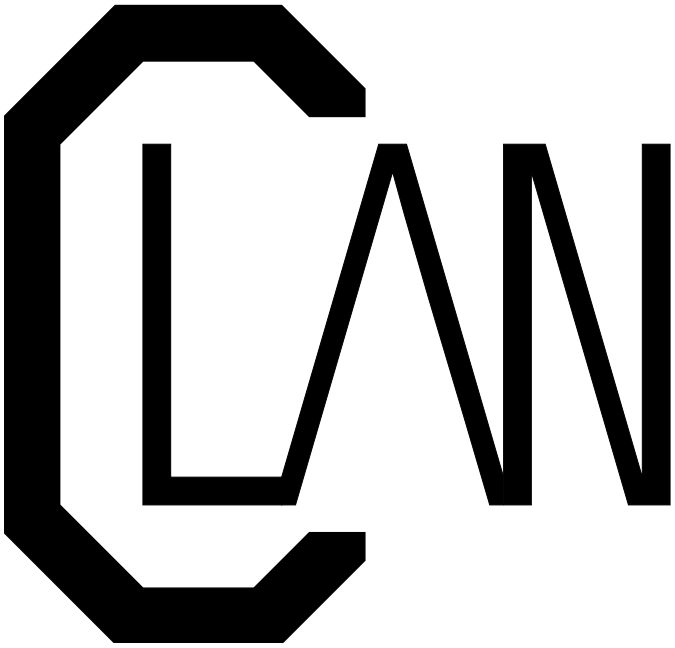

Campus Location and Navigation

Functional Specifications Document

Vincent Diener, Kenan Ibrahimov,
Sebastian Friebe, Sarah Lutteropp,
Melanie Magin, Kris Reiling

15th June 2013

Contents

1 Introduction

It is often difficult for KIT visitors and new students to find the buildings, rooms or the ways between them on the campus. Our product will simplify this task, leading to better time efficiency and less stress for everyone.

The general goal of this project is the design and implementation of a routing system for the KIT campus. The system will be able to compute and to display the shortest path between a starting point and a destination. It will be possible to route from and to buildings and rooms inside buildings. For the latter, the system will display a floor plan map of the building. It will also be possible to search just for the destination, without computing a route from some other point.

On the administration side, the system will provide a graphical use interface (*GUI*) to modify the routing graph if needed (in case of construction works or if some more details are needed) or create it from scratch. It will be possible to easily add/delete buildings with or without routing information of the interior. Moreover, it will be possible to add vertices and edges with information that is necessary to do the routing.

The product will be shipped with *floor plan maps* of at least two buildings of "KIT Campus Süd" and a *map* of the campus exterior.

2 Objectives

In this section, we describe in detail the features of the routing tool and the admin tool. This includes the functionality and the different ways for the user to input data into the tools, as well as how the tools process that information and shows it to the user.

2.1 Mandatory Criteria – Routing Tool

Displaying the map

- An exterior *map* of the campus is displayed.
- Interior *floor plan maps* can be displayed.

  Apart from showing the campus *map*, the system is also capable of displaying the *floor plan map* of a building.
- Stairs can be displayed.

  If the route contains stairs, they will be displayed as an arrow pointing upwards or downwards.
- If the user searches for a location, it is highlighted on the *map*.
- Ways passing under/through buildings are displayed in another color.

Navigation in the map

- The user can change his viewpoint by zooming in/out and panning.
- The user can enter a building, switch between floors and leave a building.

  When selecting a building, the ground *floor plan map* of the selected building will be displayed. Once this happens, it is possible for the user to switch between the different floors of the selected building and leave the building again.

User input

- The user can search for a location.

  Locations can be searched for by entering identifying information. This information includes building names and numbers, room names and addresses.

- The user can search for a route.

  The user can provide a start and end point for searching a route between them.

- Search suggestions (auto-completion) are provided in the *search fields*.

- Start and destination for routing can be selected by clicking on the *map*.

Computing the route

- The shortest path between two locations (in- and outdoor) is computed.

  It is possible to search for routes on the whole campus *map*. Those routes may or may not start or end in buildings.

- Routing for pedestrians is provided.

- A fast algorithm is used for routing.

  For further explanation, see "Nonfunctional Requirements".

- Shortcuts through other buildings are considered.

  When computing shortest path, the system will take into account routes that pass through other buildings.

- When routing to a building (or lecture hall) with more than one entrance, the path to the nearest entrance is calculated instead of the path to the main entrance.

Output

- A path between two places can be displayed.

  After the user has specified a start and end point, the calculated route is shown on the *map*. The route will be clearly visualized and easy to understand.

- Textbased route output is provided.

  Show a written form of the route. E.g., "Walk for 100m. Turn right. Walk for 25m. You have reached your destination."

2.2 Mandatory Criteria – Administration Tool

Modifying the map The *map* will be displayed in the same way as in the routing tool and the navigation in it (e.g. zooming, panning) is also the same. There are the following additional features:

- The *routing graph* is displayed on top of the *map*.

- A set of vertices and edges can be highlighted.

  Vertices and edges of the *routing graph* will be drawn in another color when selected with the mouse.

- The outline of a building can be highlighted.

  The outline of a building will be drawn in another color when selected with the mouse.

- The administrator can create, open and save a *map*.

  It is possible to open and save a *map* of the campus or a *floor plan map* of a building and its underlying *routing graph*.

- The *background image* of the *map* can be exchanged.
- Buildings, vertices and edges can be added to or removed from a *map*.
- Vertices can be moved and edges can be modified on the *map*.
- *Properties* attached to buildings, vertices and edges can be edited.

  These *properties* include for vertices: Name, address, building number and it's entrance. And for edges: Contains stairs (needed for wheelchair users) and opening hours.

- *Floor plan maps* can be added to a building. For example, if you want to add a building with multiple floors, they can be added one after another witch an attached *routing graph*.
- The administrator can add edges representing links between different floors of a building (e.g. stairs, elevators).

2.3 Facultative Criteria – Routing Tool

Filtering By filtering nodes and edges, we can restrict our *routing graph* to a subgraph of it (e.g. a *graph* that uses no edges with the *property* "contains stairs"). This will allow for the following additional modes:

- Routing for wheelchair users is provided.
- Routing for cyclists is provided.
- It is possible to search only inside a building.

  It is possible to search for routes within certain buildings only.

- The user can search for points of interest.

  It is possible to search for certain places within buildings, (e.g. locate restrooms inside a building) as well as to search for points of interest outside of buildings (e.g. locate bicycle stands on the *map*).

Searching

- When the user types in the *search fields*, spellchecking (using all possible locations as dictionary) is done.
- The user can bookmark favourite search requests.

Computing the routes

- The user can enter his own average speed to obtain the calculated time needed to walk the route.
- The routing destination can be one of multiple places (e.g. nearest restroom, bicycle stand, coffee machine or exit).

Android App

- An Android app is provided.
- The tool can use *GPS* positions to find the nearest vertex in the underlying *routing graph* of the exterior *map*.
- Live routing on a smartphone or tablet with *GPS* is provided.

Java Applet

- A Java Applet is provided.

2.4 Facultative Criteria – Administration Tool

- The user of the administration tool can define his own set of filters for routing.
- New categories of *properties* (e.g. "coffee machine") can be added.
- It is possible to close some roads temporarily (e.g. roads that are under construction).

2.5 Excluded Criteria – Routing Tool

- No routing for cars will be provided.
- No *map* data outside "KIT Campus Süd" will be provided by the project.
- Speech input will not be supported.
- Public transportation will not be considered when calculating the route.
- No photos of locations/places will be provided.
- No alternative routes will be calculated or shown.
- The tool is not a server based application.

2.6 Excluded Criteria – Administration Tool

- There will be no auto generation of the *routing graph* for a *map* from images.
- No functionality to import/export OpenStreetMap data will be provided.

3 Product Usage

This section describes how and by whom the product will be used and what they need to use it. As they share a lot of their functionalities, the routing tool and the administration tool will be implemented within the same application. This allows the user of the administration tool to switch to the routing tool at any time to test the map he just created.

3.1 Routing

Scopes Nowadays, it is not unusual for universities to be situated within extensive campuses. While this is of course convenient, it can also be quite confusing for students, especially for freshmen. They have to look up lecture halls on big, confusing plans that are often not even up to date. As a result, a lot of time is wasted. We seek to change this with our campus routing tool. The tool provides a well-arranged, simple yet powerful map application that can be intuitively used by students or people working on the campus to easily find any building or calculate the shortest route between any two points. The location or route is then visualized in a way that enables users to fully understand where they are or where they are supposed to go with just one quick look.

Target Audiences As mentioned above, the main target audience for the routing application are university students, with the focus being on the students who are new to the campus. No previous knowledge is needed to use it, so everyone who knows how to use a computer and has access to one will also be able to navigate with the routing tool. Another target audience is the people working on the campus such as professors, security and cleaning personnel. They might as well have problems finding their way on the campus when they start to work there.

3.2 Administration Tool

Scopes The administration tool is used to create the *map* that the routing tool uses. It allows to create it from scratch or modify all the data that is shown by the routing tool. This includes loading a *background image*, creating the corresponding *routing graph* and adding information to its vertices and edges. The main goal of the administration tool is to make this difficult, tedious task simple and quick by providing an intuitive interface and automating many of the time-consuming steps. For example, when adding the building number to one vertex of a building outline, all other vertices will automatically receive this *property* as well.

Target Audiences The *map* will most likely be provided by the universities for their students. Thus, the target audience for the administration tool are people working for the university. While the tool is mainly self-explanatory, it is also very versatile. The *routing graph* can be edited in a lot of ways, so it takes some time to completely grasp the whole functionality of the administration tool. However, this learning curve is not very steep so even the university students will not have a problem learning how to use the tool in a short amount of time. This makes them the second target audience: students at universities where no good, easily accessible *map* exists can use the administration tool of our application to create one and share it with their colleagues.

4 Product Environment

4.1 Environment for Mandatory Criteria

Both the routing tool and the administration tool will run on any decent Linux-PC with Java 1.7 installed on it.

4.2 Environment for Facultative Criteria

- **Android version of the routing tool**
  The minimum Android version needed to run the app-version of the routing tool is Android 2.0.
- **Java applet version of the routing tool**
  The Java applet version of the routing tool will run on any PC as described in the "Environment for Mandatory Criteria"-section that has a browser with the Java applet plugin installed.

5 Functional Requirements

5.1 Main Functions

5.1.1 Routing Tool

Displaying the map

/FMR010/ Show Campus Map
A *map* of the campus is displayed.

/FMR020/ Show Floor plan map
A *floor plan map* of a building can be displayed.

/FMR030/ Show Route
A calculated route is displayed on the *background image* in a red color.

/FMR040/ Show Way under Building
A way passing under a building is displayed in a blue color.

/FMR050/ Select Point
The user is able to select a point on the *map*, which will be marked. This can be used to select start and destination by clicking on the *map*.

Navigation in the map

/FMR060/ Zoom
The user can upscale and downscale the displayed section of the *background image* with the corresponding buttons.

/FMR070/ Pan
The user can move the displayed section of the *background image* by using the mouse.

/FMR080/ Enter Building
The user can enter a building by performing a mouse double-clicking on it.

/FMR090/ Switch Floor plan map inside a Building
If the user has entered a building, he can switch his view between different *floor plan maps* of the building by moving a special slider provided.

/FMR100/ Leave Building
The user can leave a building by clicking the provided "outside map" icon in the corner of the *floor plan map* or by double-clicking outside of the *floor plan map*.

Searching and Routing

/FMR110/ Search Location
The user can search for a location without requesting a route to it by typing the data of it either in the start or in the destination *search field* (/FMR140/).

/FMR120/ Find Search Suggestions
When the user types into a *search field*, the current input is compared to the *searchable information* of each building, to find possible completions. The resulting list is displayed.

/FMR130/ Compute Route
The user can type in a starting point and an end point into two different text fields (/FMR140/). By clicking a button, the computation of the shortest route (for pedestrians) between these two points starts. When computation is completed the computed route is displayed. If the destination is a building, the route will end at the nearest entrance of that building.

/FMR140/ Search Field Input
The user can search a location by name, building number, room number and address.

/FMR150/ Output Textbased Description
The route can be shown in a textual form like: "Walk for 100m. Turn right. Walk for 25m. You have reached your destination." This can be done by clicking on the corresponding option.

/FMR160/ Discard Route
If the user wants to discard the calculated route from the *map*, he can either calculate a new one that will then be shown instead the previous or simply use the "Remove route" function that the tool provides.

5.1.2 Administration Tool

/FMA010/ Display Routing Graph
The *routing graph* is displayed on the *map*.

/FMA020/ Create a Map or Floor plan map
By clicking on the corresponding buttons, the administrator can create a new *map* or *floor plan map* by loading a *background image*. At the beginning there is an empty *routing graph* on it.

/FMA030/ Open a Map or Floor plan map
The administrator can open a *map* or a *floor plan map*.

/FMA040/ Save a Map or Floor plan map
The administrator can save a *map* or a *floor plan map*.

/FMA050/ Exchange Background Image
The administrator can exchange the *background image* of the *map* or *floor plan map*.

/FMA060/ Set Scale
The administrator can set the scale (e.g. the size of 100px in meter) of the *background image* by defining two points on the image and entering their distance in meter.

/FMA070/ Create Vertex
The administrator can create a vertex by clicking on the current *background image*.

/FMA080/ Create Edge
The administrator can create an edge by clicking on two vertices.

/FMA090/ Create Building
The administrator can create a building by drawing a polygon. This polygon represents the outline of the building.

/FMA100/ Add Edges between Floor plan maps
The administrator can add edges between two different *floor plan maps* of a building. This can be used to add stairs to a *floor plan map* by defining a special edge between two vertices.

/FMA110/ Select Vertex or Edge
When the administrator selects a vertex or an edge, this vertex or edge is shown in a different color.

/FMA120/ Select Building
When the administrator selects a building, the outline of the building is shown in a different color.

/FMA130/ Modify Properties
If an edge, vertex or building is selected, *properties* of the current selection are shown in a seperate part of the *GUI* where the user can edit them.

These *properties* include: Name, address, building number, building entrance, contains stairs (needed for wheelchair users) and opening hours. Some of them can only be attached to vertices and some only to edges. For example, an edge cannot be a "building entrance".

/FMA140/ Move Vertex
The administrator can move a vertex by selecting it and dragging it.

/FMA150/ Move Edge
The administrator can move an edge by moving its end-vertices.

/FMA160/ Move Building
The administrator can move a building by selecting it and dragging it.

/FMA170/ Remove Vertex or Edge
The administrator can remove a vertex or an edge by selecting it and pressing the delete key on the keyboard.

/FMA180/ Remove Building
The administrator can remove a building by selecting it and pressing the delete key on the keyboard.

5.2 Facultative Functions

5.2.1 Routing

/FFR010/ Show Current Position
It is possible to show the current position of the user on the *map* by using a *GPS* signal.

/FFR020/ Search by GPS Coordinates
To search for a location, the user can also type its *GPS* coordinates into the text field.

/FFR030/ Navigation Using GPS
The user can navigate to a destination from his current position by using *GPS* to get his position.

/FFR040/ Switch Routing Mode
The user can switch the routing mode between pedestrian, bicycle and wheelchair.

/FFR050/ Show Estimated Travel Time
The user can enter his own average speed in a provided dialog. After he has done this, the routing tool will calculate and display the estimated time it will take the user to traverse the routes. If no average speed is given no time will be displayed.

/FFR060/ Bookmark Routes
The user can mark a route as a favorite, so the corresponding search query will be saved and can be reused later.

/FFR070/ Select Waypoints by Clicking
The user can mark different points on the *map* and then request a route from a start point to a destination over these waypoints.

/FFR080/ Search Point of Interest
There is list of categories from points of interests (e.g., restrooms). The user can select one categorie and then he will see the nearest to him marked on the map.

/FFR090/ Filters
The tool offers more efficient routing options using filters (e.g., use no stairs in the route).

5.2.2 Administration Tool

/FFA010/ Define Properties
The user can define new a category of *properties* (e.g., "has lamps", "road width"), its default value and its type (boolean, integer or string).

/FFA020/ Define Filters
The user can define a new set of filters for edges and vertices and save it

These filters can be of the form "has property", "has not property", "property smaller" or "property greater".

6 Nonfunctional Requirements

6.1 Routing

/NF010/ Performance: Compute the shortest route in less than one second.

/NF020/ Performance: *GUI* reacts to the user-input in less than one second.

/NF030/ Reliability: When entering an existing destination, the correct destination is found every time.

/NF040/ Flexibility: Using the Java-technology ensures the runnability of the product on all common operating systems.

/NF050/ Flexibility: The user interface is exchangeable. This means that the *GUI* can be altered without changing the internal implementation of the functionalities.

/NF060/ Usability: A high user-friendliness is achieved through a simple and intuitive *GUI*.

/NF070/ Usability: All menus have a consistent format.

/NF080/ Maintainability: Maintenance and expansion of the software is possible without high cost.

/NF090/ Stability: Arbitrary keypresses does not lead to a system crash or unintended behavior.

/NF100/ Stability: There can be no unhandled exceptions from incorrect user input.

6.2 Administration tool

/NF110/ Flexibility: Different file formats can be loaded to use as *background image*.

/NF120/ Flexibility: The user interface is exchangeable. This means that the *GUI* can be altered without changing the internal implementation of the functionalities.

/NF130/ Reliability: A software crash does not cause any loss of saved data.

/NF140/ Maintainability: Maintenance and expansion of the software is possible without high cost.

/NF150/ Usability: The product uses symbols and words that are intuitively understandable by the target audience.

7 Product Data

7.1 Essential Data

/D010/ The graph data in XML using our file structure

We chose *XML* because it is human-readable, easy to understand and easy to modify. We loosely based our format on OSM XML (see `http://wiki.openstreetmap.org/wiki/OSM_XML`), but made some major changes to make it suit our needs. The basic structure of our format can be described as follows:
First is the underlying *background image* with a scale. The scale defines how many pixel on the screen are hundred meters Then is a list of all nodes. Each node can have multiple *properties*. Those *properties* can include building IDs and entrance IDs that link the entrance-nodes of buildings to their counterparts in the interior. Second is a list of all edges between the nodes, representing the ways on our *map*. The edges can also have a set of *properties* attached to them. Last is a list of buildings, which are defined as polygons and include a link to the *XML*-File containing the *map* of the building interior. The following example shows the representation of a *routing graph* consisting of four nodes and one building. Two of those nodes are entrances to the building.

```
<?xml version="1.0"?>
<campusmap version="1.0" image="map/kitmap.png" name="KIT Campus Sued" scale="163.37">
        <node id="4" x="1253" y="332">
                <tag key="building_id" value="3219" />
                <tag key="entrance_id" value="0" />
        </node>
        <node id="2" x="998" y="711">
                <tag key="building_id" value="3219" />
                <tag key="entrance_id" value="1" />
        </node>
        <node id="3" x="1200" y="345">
                <tag key="bicycle_stand" value="true"/>
        </node>
        <node id="1" x="963" y="734" />
        <edge id="7928" node_1="1" node_2="2" />
        <edge id="7122" node_1="2" node_2="3">
                <tag key="length" value="15" />
        </edge>
        <edge id="3228" node_1="4" node_2="3" />
        <building file="buildings/infobau.xml" id="3219">
                <tag key="display_name" value="Infobau, Geb. 50.34" />
                <tag key="name" value="50.34" />
                <tag key="name" value="Infobau" />
                <outline coords="93,26;234,23;323,74;23,9;8,90" id="0" />
        </building>
</campusmap>
```

The following are the contents of “infobau.xml”, representing the interior of the building. The building *XML* file is split into different *floor plan maps* with special stair-edges linking them together.

```
<?xml version="1.0"?>
<building version="1.0">
	<defaultfloor floor="0" />
	<floorplan floor="0" image="buildings/infobau_f0.png">
		<node id="0" x="90" y="11">
			<tag key="entrance_id" value="0" />
		</node>
		<node id="1" x="83" y="18" />
		<node id="2" x="71" y="2" />
		<edge id="1" node_1="0" node2="1" />
		<edge id="2" node_1="1" node2="2" />
	</floorplan>
	<floorplan floor="-1" image="buildings/infobau_f-1.png">
		<node id="3" x="21" y="13">
			<tag key="entrance_id" value="1" />
		</node>
	</floorplan>
	<stairs>
		<edge id="0" node_1="2" node_2="3">
			<tag key="elevator" value="true" />
			<tag key="length" value="6" />
		</edge>
	</stairs>
</building>
```

As shown in the example, key/value pairs can be assigned to nodes or ways as node-properties and way-properties, e.g. node 3 of the exterior *map* has been defined as a bicycle stand. The details of our format will be presented in the next document.

/D020/ The background image of the map
Supported file formats are: PNG, JPG, GIF, BMP and PDF.

/D030/ The session data
This includes information about which data has to be automatically loaded when the program starts. For example, when the user loads a *map* for routing and ends the program, that *map* will be automatically loaded the next time they start the program.

/D040/ The help file
A help document for the program is provided, describing the functions of the routing and administration tool.

7.2 Facultative Data

/FD010/ Filters added by the user
They can be used to restrict the *routing graph* to a subgraph. For more information, see "Facultative Criteria – Routing Tool".

/FD020/ Queries
This allows the user to bookmark favourite routes and places.

/FD030/ Properties
The user can add *properties* that can subsequently be used for filtering.

8 Global Test Cases and Scenarios

8.1 Routing - Test cases

Displaying the map

/**RTC010**/ Show Campus *map* (/FMR010/).
Precondition: The user starts the program.
Expected outcome: The *map* of the campus is displayed.

/**RTC020**/ Select Point (/FMR050/).
Precondition: The user views the campus map and he selects any point on the map.
Expected outcome: The point selected on the *map* is marked.

/**RTC030**/ Show *Floor plan map* (/FMR020/).
Precondition: The user views the campus map and then selects a building on it
Expected outcome: The ground *floor plan map* of the selected building will be displayed

/**RTC040**/ Show Route (/FMR030/).
Precondition: The user has typed a starting point and destination in the corresponding text fields and clicks the "Go" button (see *GUI*, Routing view).
Expected outcome: A calculated route is displayed on the *background image* in a red color.

/**RTC050**/ Show way under building (/FMR040/).
Precondition: The user searched for the shortest path between "Audimax" and "H.S.a.F".
Expected outcome: The shortest path between selected buildings will be displayed (see /RTC040/), especially the way passing under "Infobau" is displayed in a blue color.

Navigation in the map

/**RTC060**/ Upscale and downscale the displayed section of the *background image* with corresponding buttons (/FMR060/).

/**RTC070**/ Pan the displayed section of the background image by using the mouse (/FMR070/)

/**RTC080**/ Enter Building (/FMR080/).
Preconditions: The user views the exterior map and then he double-clicks on a building.
Expected outcome: The ground *floor plan map* of the selected building replaces the exterior view.

/**RTC090**/ Switch *Floor plan map* inside a building (/FMR090/).
Preconditions: The user has entered the building and he moves the slider in order to switch to another floor.
Expected outcome: The current *floor plan map* view is replaced by selected floor plan's map view.

/**RTC100**/ Leave Building (/FMR100/).
Precondition: The user has entered a building and he clicks on "outside map" icon in the corner of the *floor plan map*.
Expected outcome: The current *floor plan map* view is replaced by map view of the exterior.

Searching and Routing

/**RTC110**/ Search Location (/FMR110/).
Preconditions: The user has typed the name of a location in the *text field* and clicks the "Go" button (see *GUI*, Routing view).
Expected outcome: The found location is highlighted on the *map*.

/**RTC120**/ Find Search Suggestions (/FMR120/).
Preconditions: The user types into a *search field* the initial letters of a building's *searchable information*.
Expected outcome: The resulting list of possible completions is displayed.

/**RTC130**/ Compute Route (/FMR130/).
Preconditions: The user has typed a starting point and destination in the corresponding text fields and clicks the "Go" button (see *GUI*, Routing view).
Expected outcome: The shortest path between these two points is computed and and displayed, in case the destination is a building, the route ends at the nearest entrance of that building.

/**RTC140**/ Text Field Input (/FMR140/).
Preconditions: The user has typed *searchable information* into a *search field* and clicks the "Go" button (see *GUI*, Routing view).
Expected outcome: The typed information is searched and displayed.

/**RTC150**/ Output text based Description (/FMR150/).
Preconditions: The user has typed a starting point and destination in the corresponding text fields and clicks the "Go" button (see *GUI*, Routing view). After the shortest path has been displayed, he selects "Show text output" form the menu.
Expected outcome: The route is shown in a textual form. See example in /FMR150/.

/**RTC160**/ Discard Route (/FMR160/)
Preconditions: The calculated shortest route between entered points is displayed on the *map* and user selects "Remove route".
Expected outcome: The displayed route is discarded and the *map* of the campus is displayed.

8.2 Administration tool - Test cases

/**ATC010**/ Display *Routing Graph* (/FMA010/).
Preconditions: The administrator loads a map.
Expected outcome: The *routing graph* is displayed on top of the *background image*.

/**ATC020**/ Create a *map* (/FMA020/)
Preconditions: The administrator selects "Create Map" from the menu in order to create a new *map* and selects a *background image* to be loaded.
Expected outcome: *The selected background image* is loaded and at the beginning there is an empty *routing graph* on it.

/**ATC030**/ Open a *map* (/FMA030/).
Preconditions: The administrator selects "Open" from the menu and selects the *map*.
Expected output: The selected *map* is displayed.

/**ATC040**/ Save a *map* (/FMA040/).
Preconditions: The administrator views a *map* and selects "Save" from the menu or presses CTRL + S.
Expected outcome: The current *map* is saved.

/**ATC050**/ Exchange *Background Image* (/FMA050/).
Preconditions: The administrator views a *map* and selects "Load new background image" from the menu and selects his desired *background image*.
Expected outcome: The selected *background image* is loaded and the *background image* of the currently viewed *map* is replaced with the new one.

/**ATC060**/ Set Scale (/FM060/).

Preconditions: The administrator marks two points on the *background image* and after right-clicking he selects "Set Scale" from menu-list and enters the distance of the marked points in meter.

Expected outcome: The entered information is saved and set as the scale of the current viewed *map*.

/**ATC070**/ Create Vertex (/FMA070/).

Preconditions: The administrator right-clicks on the *background image* and selects "Create new Vertex" from arising menu-list.

Expected outcome: The new vertex is created and is available at the selected point on the current *map*.

/**ATC080**/ Create Edge (/FMR080/).

Preconditions: The administrator selects two vertices and then after right-clicking selects "Add new Edge" from arising menu-list.

Expected outcome: The new edge is created.

/**ATC090**/ Create Building (/FMA090/).

Preconditions: The administrator draws a polygon on a map and then after right-clicking on it selects "Save as Building" from arising menu-list.

Expected outcome: The new building is created and drawn polygon represents the outline of this building.

/**ATC100**/ Add Edges between *Floor plan maps* (/FMA100/).

Preconditions: The administrator has created a building with two *floor plan maps*. He selects the "Add stairs" tool and clicks on one vertex on each floor.

Expected outcome: An edge between the selected floor plans is created.

/**ATC110**/ Select Vertex or Edge (/FMA110/).

Preconditions: The administrator selects any vertex or any edge (test both cases) by left-clicking on it.

Expected outcome: The selected vertex or edge is highlighted in another color.

/**ATC120**/ Select Building (/FMA120/).

Preconditions: The administrator selects any building by left-clicking on it.

Expected outcome: The outline of the selected building is highlighted in another color.

/**ATC130**/ Modify *Property* values (/FMA130/).

Preconditions: The administrator selects a vertex, edge or building (test all cases) and adds new properties or edits existing properties in the arising *GUI* Window (see *Figure 6*).

Expected outcome: The added properties or edited properties for the selected vertex, edge or building are updated.

/**ATC140**/ Move Vertex (/FMA140/).

Preconditions: The administrator selects a vertex and drags it.

Expected outcome: The selected vertex is moved to the position the administrator drags it to.

/**ATC150**/ Move Edge (/FMA150/).

Preconditions: The administrator moves end-vertices of an edge (see /ATC140/).

Expected outcome: The edge is moved in accordance with the movement of the end-vertices of this edge.

/**ATC160**/ Move Building (/FMA160/).
Preconditions: The administrator selects a building and drags it to the desired position.
Expected outcome: The selected building is moved to the position the administrator drags it to.

/**ATC170**/ Remove Vertex or Edge (/FMA170/).
Preconditions: The administrator selects a vertex or edge (test both cases) and presses the delete key on the keyboard.
Expected outcome: The selected vertex or edge is removed and is not on the currently viewed *map* anymore.

/**ATC180**/ Remove Building (/FMA180/).
Preconditions: The administrator selects a building and presses the delete key on the keyboard.
Expected outcome: The selected building is remove and is not on the currently viewed *map* anymore.

8.3 Routing - Test scenarios

Test scenario 1, covering all product functions of the routing tool

The freshman KIT student Jonas Schneider has to attend a lecture at 11:30 AM. Unfortunately, it is now 11:15 and he has no idea where the lecture hall is. He remembers he downloaded the KIT Campus Routing System *CLAN* the other day.
He boots his laptop and starts the routing program, which presents him with a *map* of the KIT campus (/FMR010/). He remembers the lecture hall was called something like "Hörsaal am Fa...", so he types that into the text field that says "To:" (/FMR140/). He is immediately presented with a list of search suggestions, the first one being "Hörsaal am Fasanengarten" (/FMR120/). He clicks that suggestion, resulting in it getting put into the "To:"-field. With the "From:"-field still being empty, he clicks the button captioned "Go" to start the search (/FMR110/). Now the *map* view shows the highlighted outline of the "Hörsaal am Fasanengarten" on the *map*.
He now decides to use the program to calculate the shortest route to the lecture hall. He knows he is standing right in front of the "Audimax" lecture hall, so he wants to set this as the starting point. To do that, he looks for the "Audimax" on the presented campus *map*, right-clicks it and selects "From here" (/FMR050/), putting "Audimax" into the "From:"-field. Now, with "Audimax" in the "From:"-field and "Hörsaal am Fasanengarten" in the "To:"-field, he clicks the "Go"-button again to start the calculation of the route (/FMR130/). Almost immediately, a route is displayed on the campus *map* (/FMR030/ and /FMR160/). He zooms (/FMR060/) and pans (/FMR070/) the *map* a bit to study the details of the route.
By doing that, he notices that while most of the route is colored red, a small portion is blue, indicating that this part is going under or through a building (/FMR040/). He selects "Route in text form" from the program menu. For that part of the route, it says "Pass under the building" (/FMR150/).
After that, he wants to know how to get into the lecture hall. For this, he double-clicks the lecture hall to make the program enter the building (/FMR080/). The floor plan of "Hörsaal am Fasanengarten" is shown (/FMR020/), along with the route, now starting at the entrance of the building's lobby and ending at the entrance of the lecture hall which makes up almost all of the building. Earlier, he got a text from his colleagues, informing him that they are on the second floor in the lecture hall. He uses the slider next to the *map* to change the floor plan view to show the second floor (/FMR090/) and notices that he can easily get up there by using the stairs located in the lobby.
He clicks the iconized outside-*map* to exit the building (/FMR100/) and checks the route again. He then exits the program, shuts down his laptop and starts heading to the lecture hall, still barely making it on time.

8.4 Administration tool - Test scenarios

Test scenario 2, *map* creation and editing

The third-year KIT student Andrew Ryan noticed that many freshmen have trouble finding their way on the campus because the university administration did not provide a good map. He decides that he is going to make one, using the administration tool of the KIT Campus Routing System *CLAN*.
First, he starts the program and chooses "Create new map" from the program menu, after which he is prompted to choose an image file to serve as the *background image* for the *map* (/FMA020/). He chooses the pdf-map of the campus that he found on the university website. The main view now shows the pdf-map as *background image*. He chooses "Set scale" from the program menu and clicks on two points on the *background image* that he knows are 100 meters apart (/FMA060/). With the scale set, he selects the "Create vertex" tool and places ten vertices on the *background image* to represent waypoints of the *graph* (/FMA070/). He chooses the "Create edge"-tool and clicks two vertices to connect them with an edge and repeats that until he has mapped out all the ways between the ten vertices he just placed (/FMA080/).
Since one of the edges he placed contains a small set of stairs, he wants to somehow add that information to the *graph*. To do this, he uses the "Select"-tool and clicks the vertex, resulting in the vertex being highlighted in a different color (/FMA110/). Immediately, the properties of the vertex are shown. He selects the "Contains stairs" *property* and sets it to "TRUE" (/FMA130/).
Then he notices that he misplaced one vertex and connected two vertices with an edge where there shouldn't be one. He selects the "Move"-tool and drags the vertex (and also the edges connected to it) to the correct position (/FMA140/ and /FMA150/). To remove the misplaced edge, he selects the "Delete edge" tool and clicks it, resulting in the edge vanishing from the *graph* (/FMA170/).
He is done for now, so he wants to save the *map*. To do this, he chooses "Save" from the program menu, types "kitmap.xml" as filename and clicks the corresponding button to save the *map* (/FMA040/). He then exits the program by closing the window.

Test scenario 3, building creation and editing

Andrew wants to continue working on the *map* and wants to add the building "Hörsaal am Fasanengarten". He starts the admin tool of the KIT Campus Routing System *CLAN*, chooses "Load" from the menu and selects "kitmap.xml", the file he saved earlier (see Test scenario 2) (/FMA030/).
The *background image* and routing *graph* of the *map* he created are shown (/FMA010/). He selects the "Create building" tool and is prompted for a building name and a file name (for the XML file where the building data will be stored). He types "Hörsaal am Fasanengarten" and "hsaf_map.xml". Then he clicks on the *background image* to place vertices that define the outline of the building as a polygon (/FMA090/). He clicks the first building-vertex again, closing the polygon.
Upon closer inspection, however, he notices that he completely mis-placed most of the vertices and decides to try again. He selects the building by clicking it (/FMA120/) and presses the delete key: the building is gone (/FMA180/).
He then repeats the steps to create a building, this time paying more attention. With that, the building is created.
He now wants to add an entrance to the building. To do that, he selects "Add entrance" from the program menu and clicks the outline, creating a vertex with the "entrance" *property*.
Then he notices that the building is not exactly at the right position, which he corrects by dragging it (/FMA160/).
When he enters the building by double clicking it, there is already a vertex present, representing the counterpart to the entrance outside he just created. There is no *background image* yet, so only that single vertex is shown.
He selects "Change background image" from the program menu and enters "HSaF_FLOOR_0.png". The file is loaded and now shown as the new *background image* behind the vertex (/FMA050/).
Before mapping out the ground floor, he wants to add the 1st floor. He selects "Add floor plan map" and when prompred for the floor level and *background image* he selects "1" and "HSaF_FLOOR_1.png".

Using the slider next to the *map*, he can now switch between the floors.
After mapping out most of both floors, he wants to add stairs to connect them. For that, he chooses the "Add stairs"-tool and clicks two vertices, one on floor 0 and one on floor 1 (/FMA100/). A small arrow pointing up and down respectively is now shown next to the stair-vertices.
Done with his work on the *map* for now, he choses "Save" from the program menu to save his progress and exits the program by closing the window.

9 System Model

9.1 MVC-Modell

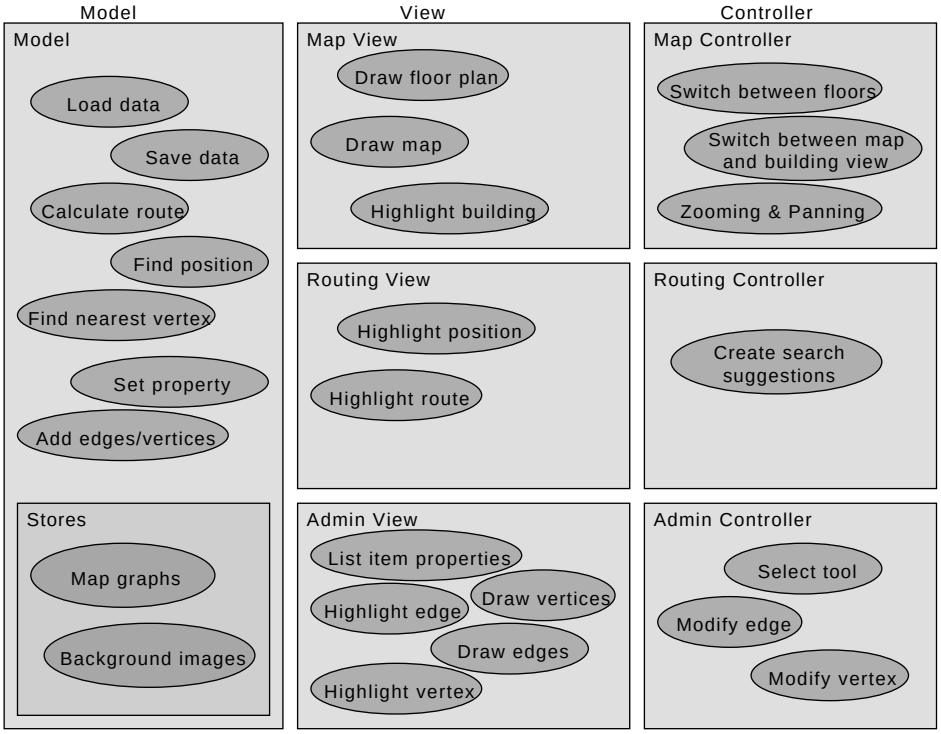

Figure 1: The parts of the application in the MVC-Architecture

The *MVC-architecture* is applied three times inside the project: The map component, the routing tool and the administration tool all use separate views and controllers but share the same model. However, since the routing tool and the administration tool both need to work with the map, they extend the map view and the map controller.
The model has to do the work of the program. It stores the opened *routing graph* and the image that should be displayed as background of the *map* for the exterior and every building floor. It also does the actual routing inside the *routing graph* and offers some functions like finding the nearest vertex for a position.
The view of the map component has to display the currently loaded *map*. This *map* consists of the *background image* either for an interior or an exterior *map*. When the exterior *map* is displayed, a building might be highlighted if one is selected. The routing and administration tools extend this *map* view with their own functionality, for example the routing tool needs a way to display the calculated routes. The administration view shows besides the *map* a panel with the *properties* of the selected edge or vertex.
The controller has to handle the user input. In the case of the map component this means panning and zooming of the *map*, as well as switching between exterior and interior. The controller of the routing tool supports the user to enter his destination since it compares the input with the names of all locations and creates a list of search suggestions. The administration tool controller mainly parses the user input and passes it on to the model to modify the *routing graph*.

9.2 Use Case Diagram

After starting the routing tool and loading the *map*, a zoomed out view of the area is displayed. Now the user can look at the *background image* without entering positions. It is possible to zoom into the

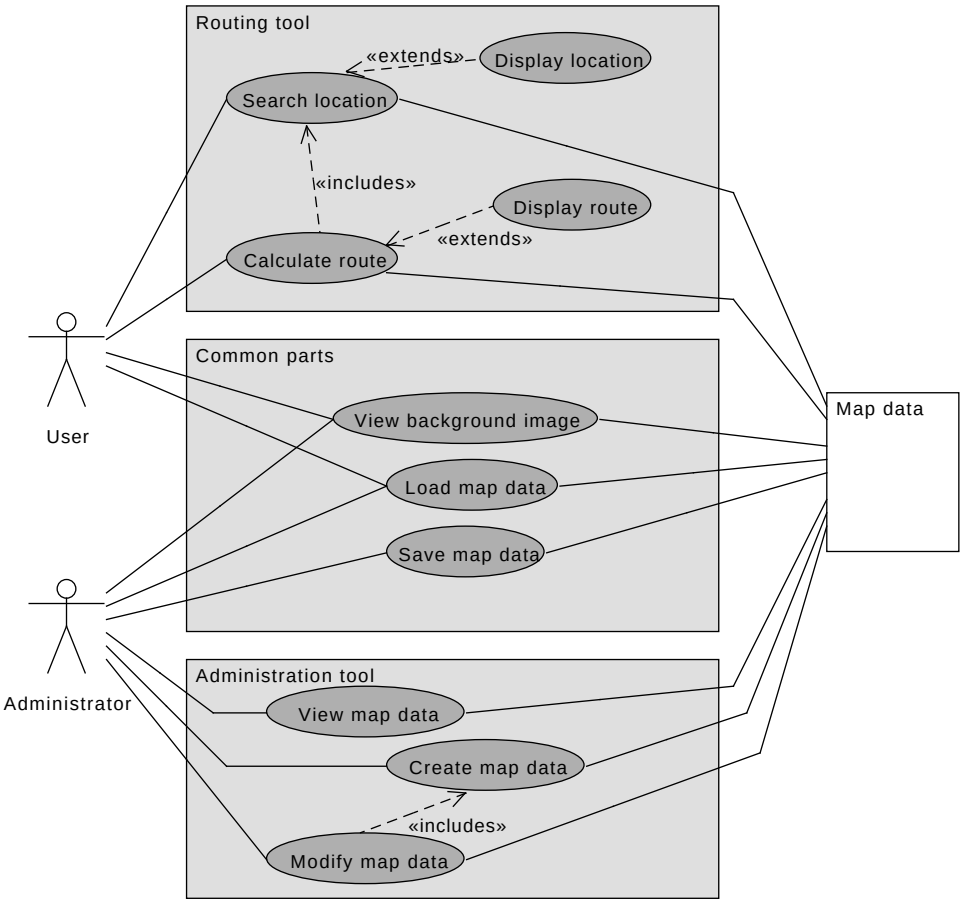

Figure 2: Possible use-cases of the program

map, move the displayed section and click on buildings to show their floor plan maps.
If he wants to, he can enter a position. This position can be given as coordinates, address, the name of a place or by clicking on the wanted position. By entering only one position, the position will be displayed at the *map* but when he inputs a second position a route between the locations can be calculated and is displayed. The search criteria can be further specified by filtering for some *properties*, for example only stair-free routes can be allowed.

To provide this functionality, both program parts needs to have access to the map. This data consists of the *routing graph* that can be modified with the administration tool and the *background image* which has been provided by the user and is linked to the *routing graph*.

9.3 Activitiy Diagrams

Routing Tool

The routing tool can be used in three different ways: The user can look at the map, he can search for a location and he can calculate a route between two positions. When he enters a position, the program displays it. After a second position has been entered, the program displays a route between the positions, if one could be found. While calculating this route, the program will follow the restrictions defined by the selected filters. When the user is finished with looking at his route or location, he can either enter new positions or exit the program.

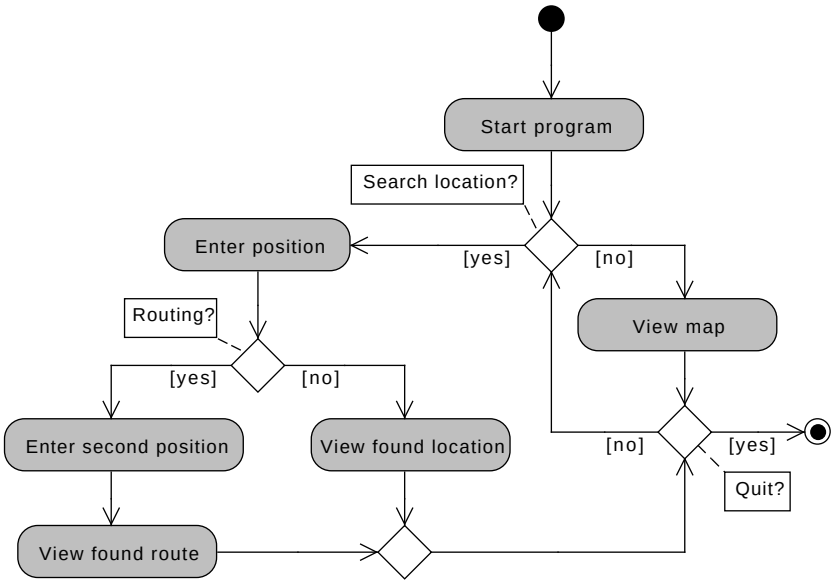

Figure 3: A possible work-flow when using the routing tool

Administration Tool

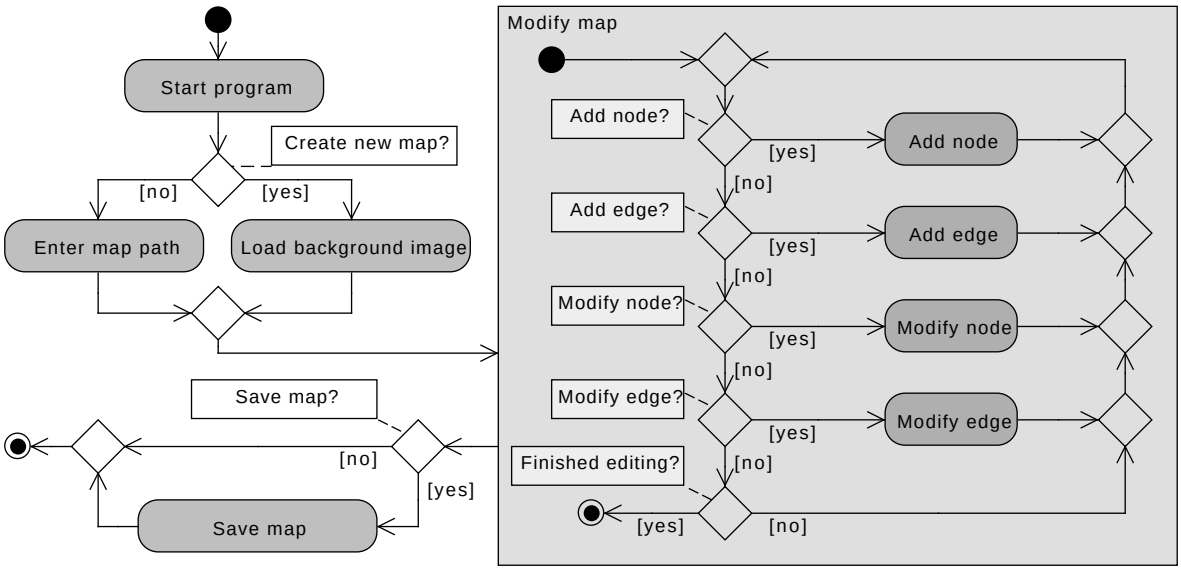

Figure 4: A possible work-flow when using the administration tool

The first decision after starting the editor is whether to load an existing map or to load a *background image* to create a new *map*. After doing one of these, the *map* can be modified. To do this the user can add, edit or remove nodes or edges. The *properties* of existing nodes and edges can be edited to attach further information to them, e.g. adding the name of a building to a node. When the user has finished editing the *map*, he can save his changes or drop them if they are not useful.

9.4 Graphical User Interface

Administration View

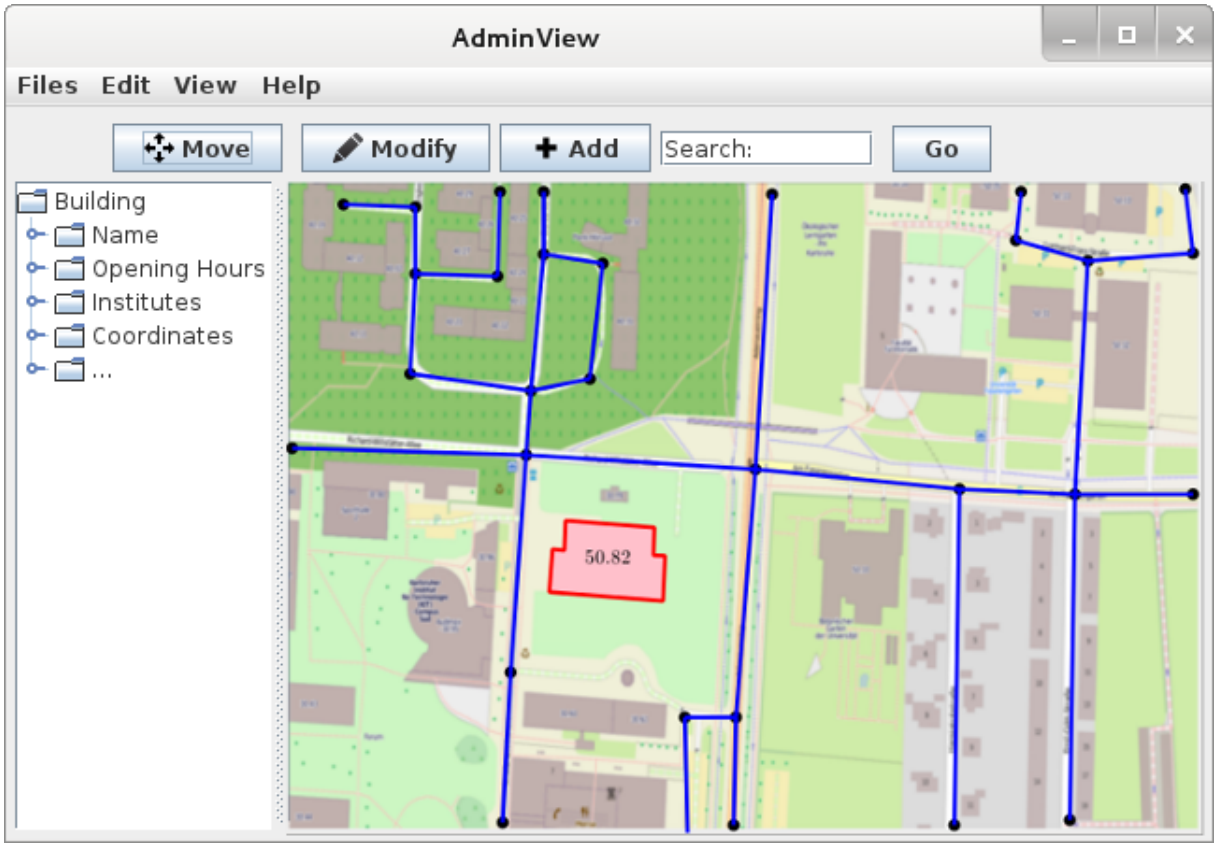

Figure 5: Administration view with displayed *routing graph*.

The administrator has the option to load and save the *routing graphs* and background images via the "Files" dialogue. In the toolbar below the main menu will be a collection of tools for modifications of the *routing graph* and a *search field* for specific locations. On the right side will be a list of the selected nodes and edges with sub-entries of their *properties*. The main area shows the editable *routing graph* of the *map* and the *background image*.

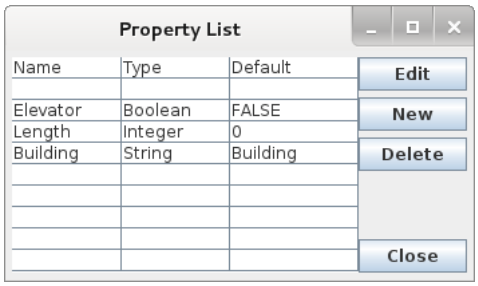

Figure 6: List of configured *properties*.

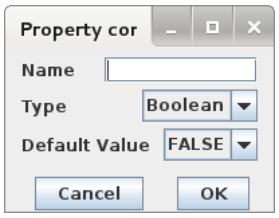

Figure 7: Configure a new *property*.

The *property* list (Figure 6) can be accessed through "Edit –> Properties" and will show a list of the names, datatypes and default values of all available *properties*, which can be added to vertices and edges.
To edit an entry in this list it has to be selected and then the "Edit" button pressed. To add a new entry to the list the "New" button has to be pressed. In each case a configuration dialogue (Figure 7) will pop up, where the name, datatype and default value can be entered.

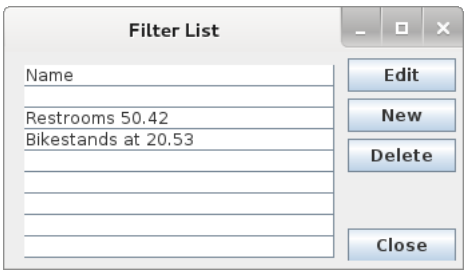

Figure 8: List of configured filters.

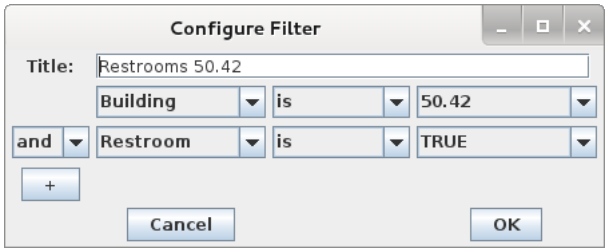

Figure 9: Configure a filter.

Similar to the *properties* filters will be shown and edited (Figures 8 and 9).

Routing View

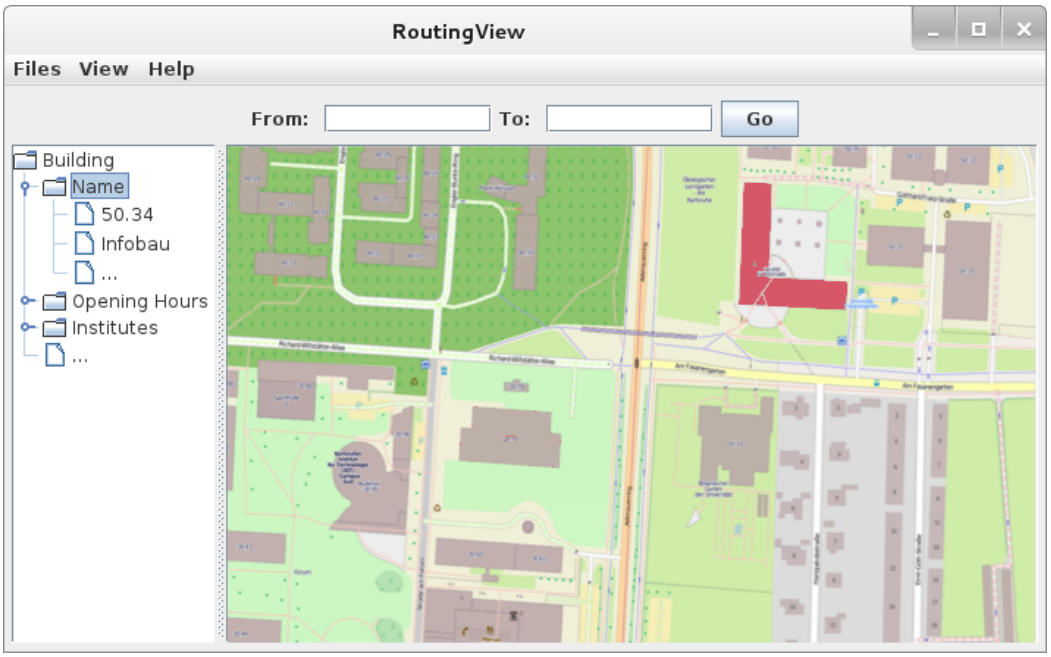

Figure 10: Routing view.

The routing view has two text fields labeled “From” and “To” for inserting the start and endpoint of a route. The main area below shows the *map* and the calculated path of the route. On the side of the *map* will be a list of *properties* of the current selection (building, way, route, etc).

View inside a building

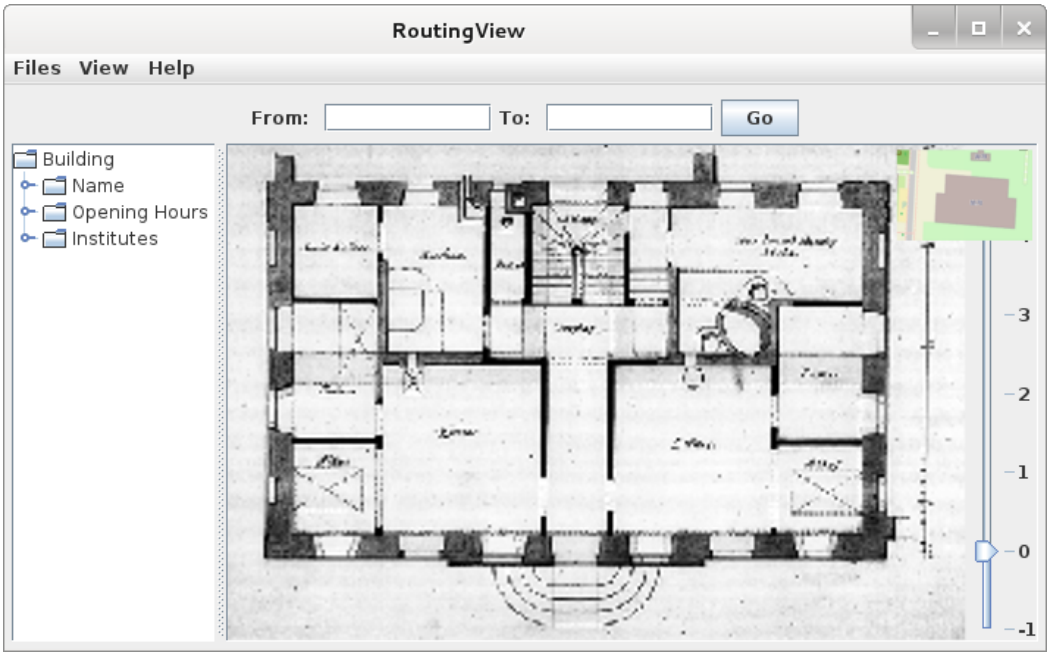

Figure 11: Routing view inside a building.

The inside view of a building shows the *floor plan map* of a building with an iconized outside-map in a corner of the main map area. To change the displayed floor a selection bar will be displayed at the right side.

To enter the building view a building has to be selected while a click on the mini-map or outside the outline of the building exits the building view.

10 Glossary

Background image An image that is either a geographic map of the campus or a floor-plan of a building.

CLAN "Campus Location and Navigation". The project name.

Floor plan map *Map* of one floor of the interior of a building.

GPS "Global Positioning System", a global satellite system to find out the own position.

GUI "Graphical user interface" is a human-computer interface (i.e., a way for humans to interact with computers) that uses windows, icons and menus and which can be manipulated by a mouse (and often to a limited extent by a keyboard as well).

Map Combination of an background image and a routing graph.

MVC-Architecture Model-View-Controller. A software architecture principle that describes the separation of the presentation (View), input handling (controller) and data storage and manipulation (model).

Property Data attached to edges and nodes which contain additional information. This data is used to allow routing after certain criteria (e.g. without stairs) and to display information about the found route.

Routing Graph A graph used for routing. In this application each vertex represents a searchable entity, e.g entrances of a building, an office or a coffee machine, while edges represents the ways between the vertices, e.g. roads, walk paths, stairs or corridors.

Search field The *From:* or the *To:* text field (See *GUI*, Routing view).

Searchable information The name, number or address of a building or room.

XML "Extensible Markup Language", a human readable file format to store hierarchical data. It describes elements (e.g. a vertex) which may contain additionally information (e.g. the position of the vertex).