

# StudMap

## Indoor Navigation

### Projektdokumentation

im Fach Fortgeschrittene Internetanwendungen



# Westfälische Hochschule

Gelsenkirchen Bocholt Recklinghausen

vorgelegt von: Thomas Buning, Marcus Büscher,  
Daniel Hardes, Christoph Inhestern,  
Dennis Miller, Fabian Paus,  
Christian Schlütter

Studienbereich: Informationstechnik

Gutachter: Prof. Dr. Martin Schulten

Abgabetermin: 21.01.2014

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Projektorganisation . . . . .	1
<b>2. Architektur</b>	<b>2</b>
2.1. Bestandteile . . . . .	2
2.2. Kommunikation . . . . .	3
<b>3. Domänenmodell</b>	<b>4</b>
3.1. Anwendungsstruktur . . . . .	4
3.2. Benutzerrollen . . . . .	4
3.3. Grundbegriffe . . . . .	5
<b>4. Datenbank</b>	<b>7</b>
4.1. Kartendaten . . . . .	7
4.2. Benutzerdaten . . . . .	11
4.3. Views . . . . .	12
4.4. Stored Procedures . . . . .	13
<b>5. Service</b>	<b>14</b>
5.1. Allgemeine Struktur . . . . .	14
5.2. HTTP-Schnittstelle . . . . .	14
5.3. Assembly-Schnittstelle . . . . .	15
5.4. Caching . . . . .	15
5.5. Sicherheit . . . . .	15
<b>6. Recherche</b>	<b>16</b>
6.1. QR-Codes . . . . .	16
6.2. NFC-Tags . . . . .	17
6.3. OCR der Raumschilder . . . . .	17
6.4. WLAN Fingerprinting . . . . .	19
<b>7. Core Bibliothek</b>	<b>21</b>
7.1. Map Webview . . . . .	21
<b>8. Collector</b>	<b>22</b>
8.1. Positionsermittlung . . . . .	22
8.2. Benutzeroberfläche . . . . .	23

<b>9. Admin</b>	<b>24</b>
9.1. Allgemeine Struktur . . . . .	24
9.2. Benutzeroberfläche . . . . .	26
9.3. Admin Spezifisches . . . . .	30
<b>10. Client</b>	<b>32</b>
10.1. Allgemeine Struktur . . . . .	32
10.2. Benutzeroberfläche . . . . .	33
<b>11. Fazit</b>	<b>34</b>
<b>12. Ausblick</b>	<b>35</b>
<b>A. Server</b>	<b>36</b>
A.1. Software . . . . .	36
A.2. Einrichtung IIS . . . . .	36
A.3. Einrichtung SQL Server . . . . .	36
<b>B. Benutzerverwaltung</b>	<b>60</b>
<b>C. Collector - Bedienungsanleitung</b>	<b>63</b>
C.1. Startbildschirm . . . . .	63
C.2. Stockwerkansicht . . . . .	64
<b>D. Webservice</b>	<b>67</b>
D.1. Allgemeine Objekte . . . . .	67
D.2. Rückgabe Objekte . . . . .	73
D.3. MapsController . . . . .	75
D.4. UsersController . . . . .	82
D.5. Verwendung der Benutzerschnittstelle . . . . .	83
<b>E. Admin-Bedienungsanleitung</b>	<b>84</b>
E.1. Registrieren . . . . .	84
E.2. LogIn / LogOut . . . . .	84
E.3. Passwort ändern . . . . .	85
E.4. Neue Map anlegen . . . . .	85
E.5. Neuen Floor anlegen . . . . .	85
E.6. Menühandhabung des Layers . . . . .	86
<b>F. Client Bedienungsanleitung</b>	<b>89</b>
F.1. Karte . . . . .	89
F.2. ActionBar . . . . .	92
F.3. Menü . . . . .	94



<b>G. Javascript Schnittstelle</b>	<b>99</b>
G.1. setStartPoint(nodeId) . . . . .	99
G.2. setEndPoint(nodeId) . . . . .	99
G.3. highlightPoint(nodeId, radius) . . . . .	99
G.4. clearMap() . . . . .	99
G.5. resetMap() . . . . .	100
G.6. resetZoom() . . . . .	100
G.7. zoomToNode(nodeId) . . . . .	100



## 1. Einleitung

Im Fach Fortgeschrittene Internetanwendungen haben wir uns im Rahmen einer studentischen Projektarbeit mit dem Thema Navigation und Lokalisierung innerhalb von Gebäuden beschäftigt. Dabei beschränkte sich das Ziel unseres Projekts auf die Navigation im Gebäude der Westfälischen Hochschule am Campus Bocholt. Dazu stellten wir uns zu Projektbeginn eine Karte des Gebäudes vor, auf der alle möglichen Navigationsziele eingezeichnet sind. Bei der Auswahl eines Navigationsziels sollte nach unseren Vorstellungen eine entsprechende Wegbeschreibung eingeblendet werden, die uns von unserem aktuellen Standpunkt zum gewünschten Ziel führt.

Um dieses Ziel zu erreichen, mussten wir uns mit verschiedenen Problemstellungen auseinandersetzen. Zunächst einmal war es nötig das Gebäude vollständig in unserem System zu erfassen. Zum anderen mussten wir die aktuelle Position innerhalb des Gebäudes ermitteln, um diese ebenfalls auf der Karte abbilden zu können.

### 1.1. Projektorganisation

Als Plattform für unser Projekt verwenden wir Google Code:

<https://code.google.com/p/studmap/>

Dort nutzen wir das SVN Repository zur Quellcode Ablage und den Issue Tracker zur Verwaltung von Benutzeranforderungen und Fehlern. Wir haben uns in unserem Projekt für eine agile Projektorganisation nach dem Vorbild von Scrum entschieden und den Issue Tracker entsprechend konfiguriert. So stehen uns die Issue Typen User Story, Task und Bug zur Verfügung. Zusätzlich haben wir noch vier Kategorien eingeführt: ProductBacklog, SprintBacklog, OpenBugs und OpenTasks. Mittels der Kategorien können wir die verschiedenen Issues besser strukturieren.

Kurz nach Beginn des Projektes haben wir die Benutzeranforderungen in Form von User Stories angelegt und dem ProductBacklog zugewiesen. Für dieses Projekt haben wir uns auf Sprints mit einer Dauer von jeweils zwei Wochen geeinigt. Zu Beginn eines jeden Sprints haben wir entsprechende User Stories in den SprintBacklog übertragen und abgearbeitet.

## 2. Architektur

Die Architektur unseres Projekts sieht an der Benutzeroberfläche neben dem Navigations-Client einen Collector-Client und eine Admin-Oberfläche vor. Die Informationen in einer zentralen Datenbank gespeichert und mit Hilfe eines Webservice bereitgestellt.

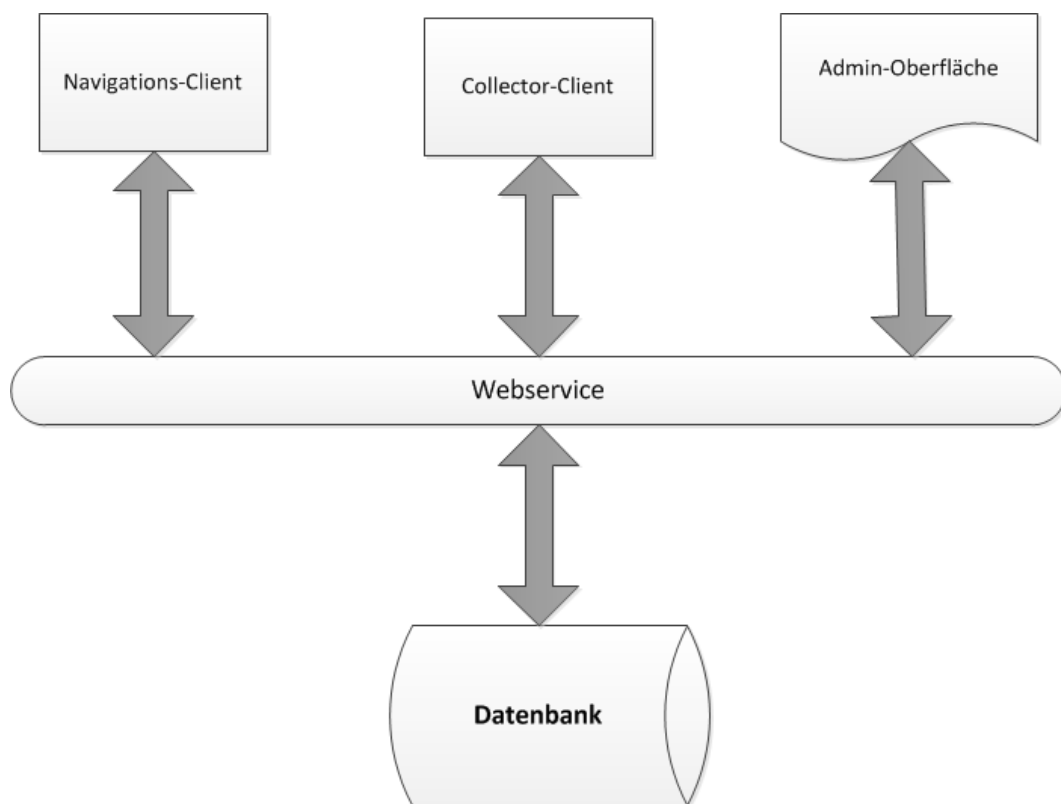


Abbildung 2.1.: Grafische Darstellung der Architektur

### 2.1. Bestandteile

Der **Navigations-Client** ist eine Anwendung für den Benutzer. Hier wird die Karte des Gebäudes mit allen Wegpunkten und die Navigation angezeigt. Dem Benutzer werden zusätzlich zur Navigation Komfortfunktionen zur einfacheren und schnelleren Bedienung bereit gestellt.

Die **Admin-Oberfläche** dient der Verwaltung von Karten und Benutzern. Der Administrator kann alte Karten bearbeiten oder neues Kartenmaterial einstellen. Zu



diesen Karten wird auch der Graph mit allen Knoten und Kanten erstellt und anschließend verwaltet. Mit der Benutzerverwaltung können neue Benutzer angelegt und vorhandene bearbeitet werden oder aber alte Benutzerkonten gelöscht werden.

Mit dem **Collector-Client** soll der Datenbestand des Projektes fortlaufend erweitert werden. Für die Knoten, bzw. Wegpunkte, können verschiedene, für die Navigation benötigte Informationen hinterlegt werden. Um fehlerhafte Eingaben zu vermeiden wird dieses Tool nicht vom Benutzer, sondern nur von Administratoren eingesetzt.

In der **Datenbank** werden sämtliche Informationen der Anwendung gespeichert. Sie enthält die Benutzerdaten und das gesamte Kartenmaterial. Das Kartenmaterial umfasst dabei die Kartengrafiken wie auch den zugehörigen Graphen. Über einen **Webservice** können die Daten abgerufen werden. Dieser stellt Funktionen zur Abfrage und Ablage von Navigations- und Benutzerdaten bereit.

Alternativ wurde uns ein Server mit einer öffentlichen IP-Adresse an der Hochschule bereitgestellt, auf welchem das System jetzt betrieben wird.

## 2.2. Kommunikation

Die externe Kommunikation des Navigations-Clients, Collector-Clients und der Admin-Oberfläche mit dem Webservice basiert auf HTTP. Als Austauschformat wird JSON verwendet. Die interne Kommunikation des Webservices mit der Datenbank basiert auf dem Entity-Framework.

Die westfälische Hochschule hat einen Server mit einer festen IP-Adresse zur Verfügung gestellt. Auf diesem betreiben wir einen IIS 7.5<sup>1</sup> mit unserem Projekt.

---

<sup>1</sup><http://www.iis.net/>



## 3. Domänenmodell

Durch das Domänenmodell legen wir Begriffe fest, mit denen die Kommunikation im Projektteam vereinheitlicht und damit einfacher wird.

### 3.1. Anwendungsstruktur

#### **Webservice (StudMap.Service)**

Stellt Funktionen zur Ablage und Abfrage von Navigationsinformationen und Benutzerdaten öffentlich bereit.

#### **Admin-Oberfläche (StudMap.Admin)**

Weboberfläche zum Anlegen, Bearbeiten von Navigationsinformationen und Benutzerdaten. Die Weboberfläche kann nur von der Benutzerrolle Administrator bedient werden.

#### **Navigations-Client (StudMap.Navigator)**

Eine Anwendung zur Anzeige von Karten und Navigation zwischen Wegpunkten. Diese wird durch den Anwender bedient.

#### **Collector-Client (StudMap.Collector)**

Eine Anwendung zur Eingabe von Navigationsinformationen. Diese wird von Administratoren verwendet.

### 3.2. Benutzerrollen

#### **Anwender (User)**

Der Anwender verwendet den Navigations-Client, um die kürzeste Route zu einem gewünschten Ziel zu erhalten.





## **Administrator**

Verwendet Admin-Oberfläche und den Collector-Client. Dazu muss dieser registriert sein.

## **3.3. Grundbegriffe**

### **Karte (Map)**

Beschreibt das gesamte Gebäude mit allen Stockwerken.

### **Stockwerk (Floor)**

2-dimensionale Ansicht mit allen Layern der Ebene.

### **Schicht (Layer)**

Es gibt mehrere Schichten, die jeweils Detailinformationen zu einem Stockwerk enthalten.

- Bild-Layer: Enthält grafische Darstellung des Stockwerks.
- Graph-Layer: Enthält Kanten und Knoten für Routen.
- POI-Layer: Zusatzinformationen zu speziellen Orten.
- Routen-Layer: Darstellung grafischer Elemente zur Navigation.

### **Route**

Hat einen Start- und einen Endknoten. Verbindet diese beiden Knoten über Zwischenknoten und Kanten.

### **Graph**

Gesamtheit aller Knoten und Kanten der Karte (Stockwerk-übergreifend).



### **Knoten (Node)**

Besteht aus eindeutigem Identifier, X- und Y-Koordinate und Stockwerk. Zu dem Knoten können zusätzliche Informationen hinterlegt werden: Name, Raumnummer, NFC-Tag, QR-Tag und Verweis auf PoI.

### **Kante (Edge)**

Verbindung zweier Knoten. Bedeutet, dass man von einem Punkt zum anderen laufen kann.

### **Point of Interest (PoI)**

Ort besonderen Interesses (z.B. Bibliothek, Mensa, ...)

## 4. Datenbank

Die Daten für StudMap werden in einer zentralen Datenbank gespeichert. In diesem Kapitel werden die einzelnen Tabellen thematisch gruppiert und Besonderheiten erläutert.

### 4.1. Kartendaten

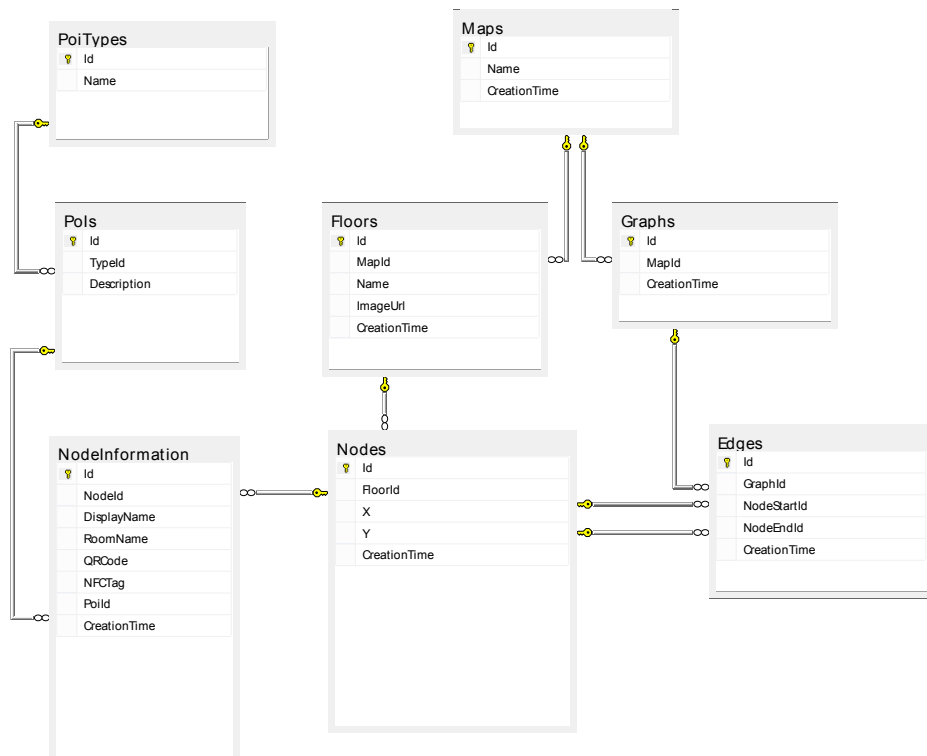


Abbildung 4.1.: Datenbankmodell für die Kartenobjekte

#### 4.1.1. Maps

Für jede Karte wird ein Eintrag in dieser Tabelle erzeugt. Zu jeder Karte wird ein frei vergebener Name und der Erstellungszeitpunkt gespeichert.

#### 4. Datenbank

---

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID der Karte
Name	NVARCHAR(255)	Name der Karte
CreationTime	DATETIME	Erstellungszeitpunkt

##### 4.1.2. Floors

Für jedes Stockwerk wird ein Eintrag in dieser Tabelle angelegt. Zu jedem Stockwerk wird ein frei vergebenen Name, eine URL auf ein Bild des Stockwerks und ein Erstellungszeitpunkt gespeichert. Ein Stockwerk ist genau einer Karte zugeordnet.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID des Stockwerks
MapId	INTEGER (FK)	ID der zugeordneten Karte
Name	NVARCHAR(255)	Name des Stockwerks
ImageUrl	NVARCHAR(MAX)	URL des Bilds
CreationTime	DATETIME	Erstellungszeitpunkt

##### 4.1.3. Graphs

Ein Graph beschreibt die Knoten- und Kantenstruktur auf einer Karte. Dazu werden alle Kanten mit dem Graphen verknüpft. Über die Kanten sind auch die Knoten mit dem Graphen indirekt verknüpft.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID des Graphen
MapId	INTEGER (FK)	ID der zugeordneten Karte
CreationTime	DATETIME	Erstellungszeitpunkt

##### 4.1.4. Edges

Eine Kante verknüpft zwei Knoten in einem Graphen.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID der Kante
GraphId	INTEGER (FK)	ID der zugeordneten Graphen
NodeStartId	INTEGER (FK)	ID des Startknotens
NodeEndId	INTEGER (FK)	ID des Endknotens
CreationTime	DATETIME	Erstellungszeitpunkt

#### 4.1.5. Nodes

Die Position eines Knoten wird durch die Zuordnung zu einem Stockwerk und seine X/Y-Koordinaten auf diesem Stockwerk bestimmt. Außerdem wird der Erstellungszeitpunkt eines Knotens gespeichert.

Die X- und Y-Koordinaten werden im Bereich 0.0 bis 1.0 gespeichert. Dabei bedeutet 0.0 ganz links (X) oder ganz oben (Y) und 1.0 ganz rechts (X) bzw. ganz unten (Y) auf dem Bild des Stockwerks.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID der Knotens
FloorId	INTEGER (FK)	ID der zugeordneten Stockwerks
X	DECIMAL(18,17)	X-Koordinate auf dem Stockwerk
Y	DECIMAL(18,17)	Y-Koordinate auf dem Stockwerk
CreationTime	DATETIME	Erstellungszeitpunkt

#### 4.1.6. NodeInformation

Zu einem Knoten können noch weitere Informationen hinterlegt werden. Diese sind optional und werden nur zu wichtigen Knoten wie Seminarräumen, Büros und Toiletten hinzugefügt. Über die Knoteninformationen kann auch ein PoI mit dem Knoten verknüpft werden.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID der Knoteninformation
NodeId	INTEGER (FK)	ID der zugeordneten Knotens
DisplayName	NVARCHAR(50)	Name der angezeigt werden soll
RoomName	NVARCHAR(255)	Offizieller Raumname (z.B. B4.0.1.11)
QRCode	NVARCHAR(255)	Hinterlegter QR-Code
NFCTAG	NVARCHAR(50)	Hinterlegtes NFC-Tag
PoiId	INTEGER (FK)	Optional zugeordneter PoI
CreationTime	DATETIME	Erstellungszeitpunkt

#### 4.1.7. Pols

Ein PoI (Point of Interest) kategorisiert für den Benutzer relevante Knoten. Hier kann z.B. nach Dozentenbüros, Mensa, Bibliothek und Toiletten gefiltert werden.



#### 4. Datenbank

---

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID des Knotens
TypeId	INTEGER (FK)	ID des PoI-Typs
Description	NVARCHAR(MAX)	Zusätzliche Beschreibung des PoIs

##### 4.1.8. PoiTypes

Diese Tabelle enthält die möglichen Typen von PoIs.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID des PoI-Typs
Name	NVARCHAR(255)	Name des PoI-Typs

## 4.2. Benutzerdaten

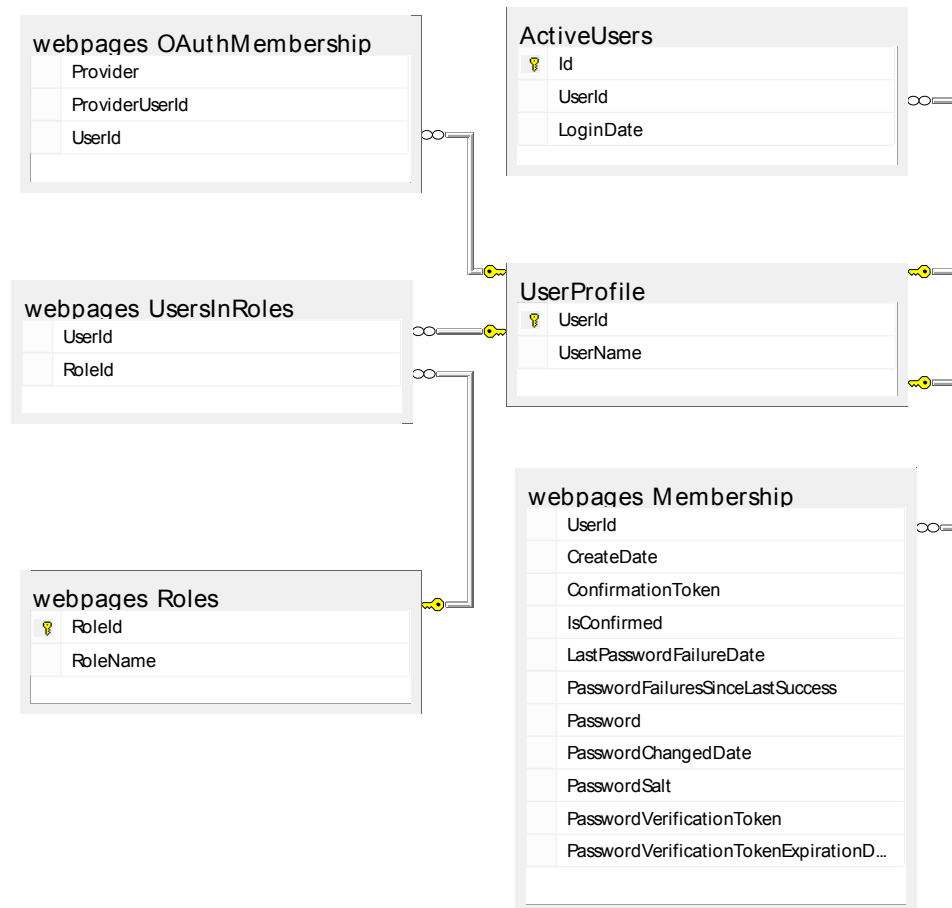


Abbildung 4.2.: Datenbankmodell für Benutzerobjekte

### 4.2.1. MVC spezifische Tabellen

Alle Tabellen aus dem Diagramm 4.2 mit Ausnahme der ActiveUsers-Tabelle werden von MVC generiert und verwaltet. Sie dienen zur Benutzerauthentifizierung und zur Autorisierung.

Relevant für StudMap sind die Tabellen “webpages\_Roles” und “webpages\_UserInRoles“. Hier werden bestehenden Nutzern Rollen zugewiesen. Um die Admin-Oberfläche nutzen zu können, muss einem Nutzer die Rolle “Admin“ zugewiesen worden sein.



#### 4.2.2. ActiveUsers

In dieser Tabelle wird festgehalten, welche Benutzer innerhalb eines festgelegten Zeitraums unsere App benutzt haben. Diese Tabelle kann z.B. um die letzte Position des Nutzers erweitert werden. Dadurch könnten andere Nutzer sehen, wer gerade in ihrer Nähe ist.

Spaltenname	Datentyp	Bedeutung
Id	INTEGER (PK)	ID des Eintrags
UserId	INTEGER (FK)	ID des Nutzers
LoginDate	DATETIME	Zeitpunkt der letzten Anmeldung bzw. Nutzung der App

### 4.3. Views

Um kompliziertere Abfragen nicht im Service abhandeln zu müssen, haben wir für diese in der Datenbank Views angelegt. Diese Views vereinfachen meist Joins über mehrere Tabellen.

#### 4.3.1. NodeInformationForMap

Diese View vereinigt die Informationen zu einem Knoten mit der zugehörigen Karte. Es werden nur Knoten berücksichtigt, zu denen ein Raum- oder Anzeigename hinterlegt wurde. Siehe [Maps](#) und [NodeInformation](#).

#### 4.3.2. PoisForMap

In dieser View werden die Knoteninformationen zusätzlich mit PoIs verknüpft. Es werden zudem alle Knoten herausgefiltert, zu denen kein PoI existiert. Siehe [PoIs](#).

#### 4.3.3. RoomsForMap

Ähnlich der View [NodeInformationForMap](#) liefert diese View allerdings nur den Raum- und Anzeigenamen eines Knoten.





## 4.4. Stored Procedures

Das Löschen von Tabellen kann auf Grund von Fremdschlüsselbeziehungen u.U. aufwendig und komplex sein. Daher haben wir für das Löschen des Teilgraphen auf einem Stockwerk (`sp_DeleteGraphFromFloor`), eines gesamten Stockwerks (`sp_DeleteFloor`) und einer gesamten Karte (`sp_DeleteMap`) Stored Procedures angelegt. Diese löschen zunächst alle zugehörigen Einträge in abhängigen Tabellen und entfernen dann den Eintrag in der Haupttabelle.



## 5. Service

Der Webservice stellt die Logik für alle Clients bereit. Dies schließt den Navigations-Client, den Collector-Client und die Admin-Oberfläche mit ein. Er kapselt in erster Linie die Datenbankzugriffe und nutzt Caching für häufige Anfragen. Es werden Funktionen zur Benutzerverwaltung sowie zur Navigation bereitgestellt.

### 5.1. Allgemeine Struktur

Als Grundlage verwendet der Webservice ein ASP.NET Web API Projekt<sup>2</sup>. Der Datenbankzugriff erfolgt über das Entity Framework von Microsoft<sup>3</sup>.

In einem Web API Projekt können die Funktionen des Webservices auf Basis von selbst definierten Datenstrukturen erstellt werden. Damit muss sich der Entwickler nicht damit beschäftigen, wie diese Objekte vom Client geliefert werden und in welchem Format die Antwort gesendet wird. Das Serialisieren erfolgt meist in den Formaten XML oder JSON. Unser Webservice verwendet im Standard das JSON-Format, da es leichtgewichtiger ist und somit besser für mobile Clients geeignet ist.

Die Funktionalität teilt sich in drei Bereiche auf. Das sind die Karten- und Navigationsinformationen, die Benutzerverwaltung und das WLAN-Fingerprinting (nur experimentell). Diese Bereiche werden in drei Controllern abgebildet. Ein Controller ist eine Klasse die Anfragen gegen den Webservice verarbeitet.

### 5.2. HTTP-Schnittstelle

Der Webservice bietet eine HTTP-Schnittstelle, die von den Clients genutzt wird. Diese bietet Funktionen über die HTTP-Methoden GET und POST auf bestimmte URLs an.

Der Client sendet einfache Anfrageparameter über die URL-Query-Parameter (GET). Bei komplexeren Anfragen kann er ein JSON-Dokument in den Body der Anfrage setzen (POST). Als Antwort erhält er ein JSON-Dokument<sup>4</sup>.

---

<sup>2</sup><http://www.asp.net/web-api>

<sup>3</sup>[http://msdn.microsoft.com/de-de/library/bb399567\(v=vs.110\).aspx](http://msdn.microsoft.com/de-de/library/bb399567(v=vs.110).aspx)

<sup>4</sup>Web API unterstützt auch XML, dies wird von unseren Clients allerdings nicht verwendet



Eine komplette Referenz der Webservice-Funktionen befindet sich im Anhang dieser Dokumentation (siehe [Webservice](#)).

### 5.3. Assembly-Schnittstelle

Da Anwendungen wie die Admin-Oberfläche und ein Teil der Client-Oberfläche immer auf demselben Server laufen wie der Webservice, ist ein Umweg über die HTTP-Schnittstelle aufwendiger als notwendig. Daher verwenden diese Anwendungen auch die Assembly-Schnittstelle des Webservices. Dabei werden die Anfragen direkt an die entsprechenden Controller gestellt. Das Serialisieren und Deserialisieren der Anfrage- und Antwortobjekte entfällt somit. Für dynamische Anfragen, z.B. in Javascript, nutzen diese Anwendungen allerdings weiterhin die HTTP-Schnittstelle.

Die Assembly-Schnittstelle bietet die gleiche Funktionalität wie die HTTP-Schnittstelle. Alle Funktionen des Webservice stehen als .NET Methoden der entsprechenden Controller-Klasse zur Verfügung.

### 5.4. Caching

Der Webservice nutzt das Caching der HTTP-Laufzeitumgebung, um häufige Abfragen (z.B. allgemeine Karten- und Stockwerkinformationen) und komplexe Berechnungen (z.B. Routen zwischen allen Knoten) effizienter durchzuführen. Die Cache-Objekte sind nur für eine bestimmte Dauer gültig (aktuell eine Stunde) und werden invalidiert, falls sich die Stammdaten ändern.

### 5.5. Sicherheit

Aus Sicherheitsgründen macht es Sinn die HTTP-Schnittstelle vor der Veröffentlichung des Programms auf HTTPS umzustellen, da bisher jede Kommunikation unverschlüsselt erfolgt.

## 6. Recherche

Für die Navigation innerhalb von Gebäuden konnten wir nicht auf die gängigen Standards zurückgreifen, sondern mussten uns andere Wege überlegen, wie wir die Nutzer innerhalb der Gebäude lokalisieren.

Außerhalb von Gebäuden ist die Lokalisierung mittels GPS sehr verbreitet und auch einfach und akkurat. Um eine ähnliche Lokalisierung unserer Nutzer zu ermöglichen haben wir uns die folgenden Möglichkeiten überlegt.

### 6.1. QR-Codes

QR-Codes sind weit verbreitet, einfach zu erstellen und mit vielen Geräten einzulesen. Dadurch bieten QR-Codes die Möglichkeit Informationen einfach an Orten anzubringen und von Maschinen einzulesen.

Wir haben uns zwei Konzepte überlegt, QR-Codes in unserem Projekt einzubauen. Dazu haben wir einmal die Möglichkeit betrachtet die Auswertung des QR-Codes am Server zu realisieren und mit der Möglichkeit verglichen die Auswertung direkt am Client des Nutzers zu implementieren.

#### 6.1.1. Serverseitig

Für die serverseitige Umsetzung hat gesprochen, dass das Smartphone keine Rechenleistung benötigt um die QR-Codes zu dekodieren. Des Weiteren wird durch eine serverseitige Implementierung vermieden, dass der Nutzer weitere Apps auf seinem Smartphone installieren muss.

Für die Implementierung in den Webservice haben wir uns für die offene Bibliothek [ZXing.NET](https://zxing.net/) entschieden. Allerdings ist uns bei der Implementierung und Testen der Bibliothek direkt aufgefallen, dass die Bilder zuvor am Smartphone verkleinert werden müssen um Bandbreite zu sparen und die Laufzeit der Bibliothek zu verringern. Entgegen unserer Absicht wird dafür jedoch clientseitig Rechenleistung benötigt. Darüber hinaus mussten wir feststellen, dass die Bibliothek keine zuverlässige Dekodierung der QR-Codes bietet.

#### 6.1.2. Clientseitig

Im Gegensatz zur serverseitigen Umsetzung wird bei dieser Implementierung die gesamte Dekodierung am Client vorgenommen. Dadurch wird die Netzlast verringert,

der Aufwand am Client aber erhöht.

Hier bot sich zum einen an eine Bibliothek in den Client aufzunehmen, oder eine externe Anwendung zum Dekodieren der QR-Codes zu benutzen. Wir haben uns schlussendlich dazu entschieden den QR-Code Reader „Barcode Scanner“ von ZXing zu verwenden. Dieser wird von Google empfohlen und kann zudem einfach eingebunden werden.

## 6.2. NFC-Tags

NFC ist eine Technologie, auf die wir im Alltag immer häufiger stoßen, sei es bei Werbung, Bezahl- oder Ticketsystemen. NFC steht für Near Field Communication und besteht aus zwei Komponenten, der passiven, dem NFC-Tag, und der aktiven Komponente, beispielsweise dem Smartphone.

Die als Datenspeicher fungierenden NFC-Tags werden immer preiswerter und sind zudem relativ klein, was eine Anbringung an den gewünschten Orten problemlos ermöglicht. Auf der anderen, der aktiven Seite stehen immer mehr Smartphones bereit, die diese Technologie unterstützen.

Die steigende Beliebtheit der NFC-Technologie verdankt diese dem Komfort. Wie der Name bereits aussagt, reicht schon die Nähe der aktiven Komponente zur passiven um Daten zu kommunizieren. Diesen Komfort bieten wir dem Benutzer, um seine Position dem Navigator mitzuteilen.

Des Weiteren lassen sich auf dem NFC-Tag zusätzliche Informationen hinterlegen, die Unwissende auf die Navigationsmöglichkeit aufmerksam machen.

### 6.2.1. Umsetzung

Der gesamte Prozess der Positionsermittlung findet clientseitig statt. Zum einen ist das Client-Gerät unumgänglich für die Kommunikation mit dem NFC-Tag und zum anderen sind die Datenmengen und der Aufwand der Interpretation sehr gering.

Die Android NFC API ermöglicht uns die native Umsetzung der Positionsermittlung für Android-Geräte.

## 6.3. OCR der Raumschilder

Anstatt QR- oder NFC-Tags an den Räumen der FH anzubringen, besteht die Möglichkeit die bereits angebrachten Türschilder zu verwenden. Hierzu ist eine Erkennung der Raumnummer auf dem Türschild notwendig. Die OCR (Optical Character Recognition) kann über eine Bibliothek sowohl auf Client-, als auch auf Serverseite

erfolgen. Erfolgt die Erkennung auf dem Client, dann könnten ebenfalls bestehende Apps zur Texterkennung verwendet werden.

### 6.3.1. Bibliothek tesseract

Tesseract<sup>5</sup> ist eine native Bibliothek für die Erkennung von Text in Bilddateien. Es gibt sowohl für .NET als auch für Java (Android) entsprechende Wrapper, die von uns verwendet werden können. Bei den Tests zu der OCR-Bibliothek haben sich allerdings einige Schwächen gezeigt. Tesseract ist für die Texterkennung von gescannten Dokumenten gedacht und arbeitet deshalb nur stabil, wenn sich auf dem Bild ausschließlich Text befindet. Dies wird an folgenden Beispielbildern deutlich.

Auf dem Bild mit Raumschild und Wand wird nur unzuverlässig Text erkannt (siehe Bild 6.1). Die Ergebnisse variieren von "Kein Text erkannt" bis hin zu "Buchstaben-salat mit Raumnummer". Schneidet man den relevanten Teil per Hand aus (siehe Bild 6.2), wird der Text einwandfrei erkannt.



Abbildung 6.1.: Gesamtes Bild Raumschild

---

<sup>5</sup><https://code.google.com/p/tesseract-ocr/>

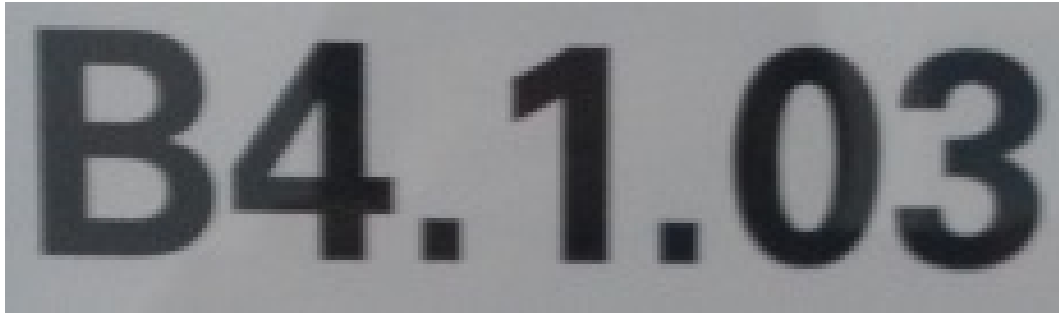


Abbildung 6.2.: Ausschnitt Raumschild

Damit die Raumschilder zuverlässig erkannt werden, ist es notwendig, dass aufgenommene Bilder zunächst auf den Bereich mit der Raumnummer zugeschnitten werden. Dies erfordert einen hohen Entwicklungsaufwand.

### 6.3.2. App Google Goggles

Google Goggles<sup>6</sup> ist eine Android-App, die zur Erkennung von Text, Symbolen und QR-Tags verwendet werden kann. Diese App lieferte in Tests auch bei suboptimalen Bildern gute Ergebnisse. Außerdem ist die App gut in das Android-Umfeld eingebettet und lässt sich leicht bedienen.

Allerdings gibt es für die Verwendung von Google Goggles noch keine öffentliche API<sup>7</sup>. Diese ist zwar von Google geplant, aber nie umgesetzt worden. Auch wenn die App eine komfortable Möglichkeit zur OCR-Erkennung bietet, kann diese ohne API nicht in unserem Projekt verwendet werden.

## 6.4. WLAN Fingerprinting

Bei der Positionierung des Nutzers mittels WLAN haben wir über eine für den Nutzer passive Positionierung recherchiert. Alle anderen Positionierungsmethoden benötigten eine Eingabe des Nutzers. Wir haben hier die Eingabe der Position auf einer Karte und das Einlesen von QR-Codes oder NFC-Tags behandelt. Die Positionierung ermöglicht es allerdings im Hintergrund zu laufen und ohne Eingabe des Nutzers die Position zu bestimmen.

WLAN ist zur Positionierung innerhalb von Gebäuden geeignet, da es zum einen eine weit verbreitete Infrastruktur ist, auf vielen mobilen Plattformen verfügbar ist, Wände durchdringt und Standard WLAN Access Points bereits eine Lokalisierung

---

<sup>6</sup><https://play.google.com/store/apps/details?id=com.google.android.apps.unveil>

<sup>7</sup><http://stackoverflow.com/questions/2080731/google-goggles-api>



auf Raum-Genauigkeit ermöglicht.

Aus all diesen Gründen haben wir uns mit der Positionierung mittels WLAN beschäftigt.

#### 6.4.1. Sammeln von WLAN Fingerprints

Um später Vergleiche im Client anstellen zu können mussten wir zuerst Daten des Netzwerkes sammeln. Ein Access Point wird dabei eindeutig durch eine **BSSID** gekennzeichnet und der Client gibt Auskunft über die empfangene Signalstärke (**RSS**<sup>8</sup>), welche beobachtet und aufgezeichnet werden kann.

Ziel dieser Phase war es an möglichst vielen Punkten in der Hochschule die *RSS* zu messen und diese zu einem Punkt auf der Karte der Hochschule zu speichern.

#### 6.4.2. Kalibrierung

Da das Sammeln der WLAN Fingerprints mit einem Smartphone realisiert wird und wir davon ausgehen mussten, dass nicht jeder Nutzer das gleiche Smartphone besitzt, mussten wir uns eine Möglichkeit der Kalibrierung überlegen. Dazu haben wir überlegt, dass der Nutzer zuerst in einer Kalibrierungsphase selbst einen Fingerprint erstellt von einem von uns festgelegten Ort und diesen mit dem von uns gemessenen Fingerprint vergleicht. Dadurch bekommen wir einen Faktor um den das Smartphone des Nutzers von unserem Gerät abweicht. Da wir vermuten, dass die WLAN Antennen der Smartphones auch in verschiedenen Bereichen, hohe, mittlere und niedrige Signalstärke, sich stark unterscheiden berechnen wir diesen Faktor für die gerade genannten Bereiche.

#### 6.4.3. Positionierung mittels WLAN Fingerprints

Um die Position eines Nutzers ermitteln zu können muss dieser, wie der *Collector* einen Fingerprint des WLANs an seiner aktuellen Position erstellen. Diesen Fingerprint und seine Faktoren, welche während der Kalibrierung ermittelt wurden, schickt der Client zum Server, welcher durch Vergleiche den Standpunkt ermittelt und zurückgibt.

Fabian:  
Wie werden die  
FP verglichen.

---

<sup>8</sup>RSS: received signal strength wird in dBm gemessen.





## 7. Core Bibliothek

Wir haben schon früh festgestellt, dass es eine Vielzahl an Strukturen und Funktionalitäten geben wird, die sowohl in der Collector-Applikation als auch in unserer Navigator-Applikation für den Endbenutzer Anwendung finden. Dem entsprechend haben wir diese in eine eigene Android-Bibliothek ausgelagert, die wir wiederum in unsere Android-Applikationen einbinden konnten. Zu den Kernfunktionalitäten und -Strukturen gehören unter Anderem folgende:

- Grundlegende Definitionen einer Map, eines Floors oder eines Knoten
- Konstanten für die Kommunikation mit dem Webservice
- Ein ErrorHandler für alle grundlegenden Fehler
- Snippets zur einfachen Kommunikation mit dem Benutzer mittels Dialogen
- Abbildung des Webservices zur vereinfachten Nutzung
- Javascript-Schnittstellendefinitionen für die Interaktion auf der Karte
- Asynchrone Tasks für Webservicekommunikation inkl. entsprechender Listener

### 7.1. Map Webview

Zur Anzeige der Karte nutzen wir die Bibliothek d3 und Floormap. Dadurch ist es möglich in einem Bild, mithilfe von Touch-Gesten, hinein und heraus zu zoomen. D3 nutzt dazu ein SVG Image welches auch erweiterbar ist. Wir haben dazu noch Funktionen ergänzt die es uns ermöglicht einen Graphen mit Knoten und Kanten anzuzeigen. Zur Anzeige nutzen wir Kreis und Linien Elemente von SVG. Zusätzlich ist es möglich Punkte zu verändern um diese deutlicher zu machen, einen Start und Endpunkt zur Navigation fest zu legen, zu einem Punkt zu zoomen und alles zurück zu setzen.

Eine genauere Übersicht über die Schnittstellenbeschreibung ist im Anhang: [Javascript Schnittstelle](#) enthalten.



## 8. Collector

Die Collector Anwendung wurde als Android-Applikation umgesetzt. Es wird Android 3.0 auf einem Smartphone vorausgesetzt, um alle Funktionen nutzen können. Der Collector ist eine Android-Applikation, die dazu dient, Daten zu vorher vom Administrator definierten Knoten/Punkten zu sammeln und diese entsprechend in der Datenbank zu hinterlegen. Damit ist die Applikation ein Werkzeug der Administratoren und nicht der Endbenutzer.

Mit der Entwicklung des Collectors haben wir eine intensive Recherche betrieben, wie wir eine Indoor-Navigation ermöglichen können.

### 8.1. Positionsermittlung

In diesem Kapitel wird beschrieben, wie die Position eines Anwenders auf der Karte bestimmt wird. Dazu werden die im Kapitel [Recherche](#) beschriebenen Verfahren verwendet.

#### 8.1.1. QR-Codes

An den Räumen werden QR-Tags angebracht, die eine Zuordnung zu einem Knoten auf der ermöglicht. Hier ist zu beachten, dass die Informationen auf dem QR-Tag auch für andere Anwendungen nützlich sein sollen. Deshalb kann hier nicht nur eine Knoten-ID hinterlegt werden.

Folgendes JSON-Format ist ein möglicher Kandidat:

```
1 {  
2   "General": {  
3     "Label": "A2.1.10",  
4     "Name": "Aquarium"  
5   },  
6   "StudMap": {  
7     "NodeId": "12",  
8     "Url": "https://code.google.com/p/studmap/"  
9   }  
10 }
```



### 8.1.2. NFC-Tags

Neben den QR-Tags werden auch NFC-Tags an den Räumen angebracht. Auf diese werden aktuell keine Informationen geschrieben. Die Knoten werden über die NFC-ID des Chips zugeordnet.

Um Benutzer, die die StudMap-App nicht installiert haben zu informieren wird eine URL auf den NFC-Tags gespeichert. Diese leitet auf eine Seite mit Informationen über den gescannten Raum und einen Link auf unsere Projektseite. In Zukunft kann hier auch ein Link auf den Google Play Store hinterlegt werden, um eine einfache Installation zu ermöglichen.

Die URL sieht z.B. so aus (für das Aquarium):

```
1 http://193.175.199.115/StudMapAdmin/Admin/NodeInfo?nodeId=847
```

## 8.2. Benutzeroberfläche

Anhand der Recherche-Ergebnisse haben wir uns entschieden einen Positionserkennung mittels QR-Codes, NFC-Tags und Wlan-Fingerprinting zu implementieren. QR-Codes werden mittels eines Tools auf dem Server generiert. Es bedarf hierbei keiner weiteren Behandlung durch die Collector-Applikation. Dem gegenüber stehen das Zuordnen von NFC-Tags zu Knoten und das Sammeln von Wlan-Fingerprints durch die Collector-Applikation. Zur Erfüllung dieser beiden Aufgaben bietet die Applikation eine schlichte Benutzeroberfläche. Eine Bedienungsanleitung der Collector Anwendung befindet sich im Anhang: [Collector - Bedienungsanleitung](#).



## 9. Admin

Wesentlicher Bestandteil unseres Projektes war die Administrationsumgebung. Genauer musste eine Anwendung geschaffen werden, die dem jeweiligen Endanwender Navigationsdaten zur Verfügung stellt. Also eine Umgebung, die einen Upload für Kartenmaterial bereitstellt. Des Weiteren muss die Administrationsanwendung Features besitzen um Routen zu definieren und Meta-Informationen zu besonderen Örtlichkeiten festhalten zu können. Die Meta-Informationen können durch Points of Interest (PoI) Informationen erweitert werden.

Im folgenden Abschnitt wird zunächst die allgemeine Struktur erläutert und anschließend liegt das Hauptaugenmerk auf der Benutzeroberfläche. Eine ausführliche Bedienungsanleitung der Administrationsumgebung ist im Anhang [E](#) beigelegt.

### 9.1. Allgemeine Struktur

#### ASP.NET MVC 4

Basis unserer Projektstruktur war das **ASP.NET MVC Framework**, welches ein Web Application Framework ist, und ein Model-View-Controller-Pattern implementiert.

Dies ermöglichte uns, eine Webanwendung zu entwickeln, bei der die Daten (*Model*) gekapselt von der Ausgabe (*View*) und dem *Controller* vorliegen. Die *View* repräsentiert unsere Daten und der *Controller* reagiert auf Zustandsänderungen und ist sozusagen das Bindeglied oder die Schnittstelle zwischen *View* und *Model*.

##### 9.1.1. Model

###### AccountModels

Kapselt die benutzerspezifischen Daten in einem Model zur Authentifizierung von Benutzerprofilen. Dieses Model findet Verwendung beim Registrieren sowie beim LogIn- / LogOut-Verfahren.

###### AdminModels

Model indem der MapName gekapselt vorliegt.



### **FloorViewModel**

Dieses Model enthält Daten zu einem Floor. FloorId, MapId und FloorImageFile sind diese Daten.

### **MapViewModel**

MapId und Name werden für die MapView in diesem Model gekapselt.

## **9.1.2. View**

Die Views in unserem Projekt wurden in Views die den Account-, den Admin- und den Home-Bereich betreffen unterteilt.

### **Account**

Enthält html-Seiten zum registrieren, anmelden und verwalten den eigenen Benutzerprofils.

### **Admin**

Unter dieser Rubrik fallen die Webseiten, mit dem der Administrator Maps, Floors und Knoteninformationen zu einer Map hinzufügen kann. Sozusagen die Arbeitsfläche des Administrators.

### **Home**

Einstiegspunkt oder oberstes Element (Index) unserer Webseite.

## **9.1.3. Controller**

### **HomeController**

Speziell für unser Projekt bedeutet es, dass wir drei Controller angelegt haben. Der Einstiegspunkt unserer Web-Anwendung ist der sogenannte *HomeController*. Dies ist der Controller, der zum Zuge kommt, sofern die anderen beiden Controller eine Interaktion oder Controller-Aufrufe mit gewissen Parametern nicht unterstützen.

### AccountController

Der *AccountController* verarbeitet die Ereignisse, die vom Registrierungs-, LogIn- und LogOut-Verhalten eines Benutzers ausgelöst werden. Im Fokus stehen hierbei die **HTTP GET-** und **HTTP POST-Methoden**, die von der Klasse *AccountController* implementiert werden.

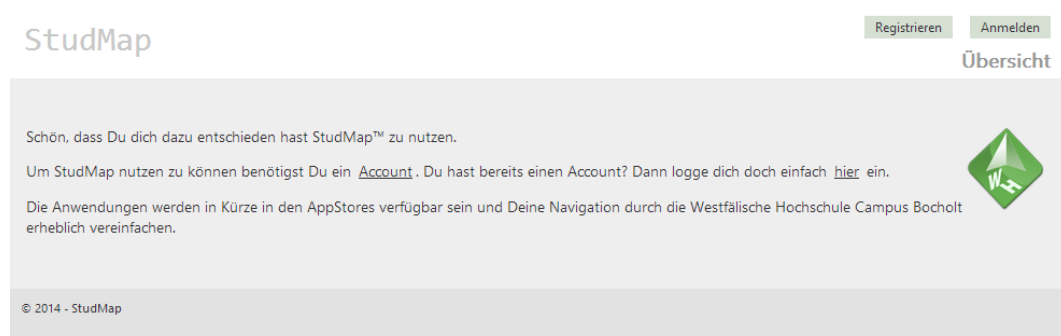
### AdminController

Der *AdminController* reagiert auf Zustandsänderungen, die beim Anlegen, Bearbeiten und Löschen von Datenmaterial in Form von Karten oder Informationen (Meta-Informationen), ausgelöst werden.

Relevante Funktionen sind die Methoden zum erstellen und löschen von Maps und Floors. Des weiteren werden über diesen Controller die Routeninformationen als Graph verarbeitet und in einer Datenbank gespeichert. Detailinformationen zu einem Knoten aus einem Graphen verarbeitet dieser Controller und speichert sie ab. Diese Detailinformationen ermöglichen besondere Orte als *Points of Interest* zu kennzeichnen.

## 9.2. Benutzeroberfläche

Die Administrator Benutzeroberfläche ist über folgenden Link <http://193.175.199.115/StudMapAdmin/> erreichbar.



Der Anwender enthält die Möglichkeit sich anzumelden oder sich zu registrieren.

## Registrieren

<http://193.175.199.115/StudMapAdmin/Account/Register>

StudMap

Registrieren Anmelden

Übersicht

Name

Passwort

Bestätige Passwort

Registrieren

© 2014 - StudMap

## Anmelden

<http://193.175.199.115/StudMapAdmin/Account/Login>

StudMap

Registrieren Anmelden

Übersicht

Name

Passwort

☐ Angemeldet bleiben

Anmelden

[Registriere](#) Dich wenn Du keinen Account hast.

© 2014 - StudMap

## Admin

Den Einstiegspunkt zum verwalten von Maps finden Sie unter folgenden Link  
<http://193.175.199.115/StudMapAdmin/Admin>. Er enthält eine Auflistung aktuell

## 9. Admin

vorhandener Maps. Es können neue Maps erstellt, bzw. vorhandene entfernt werden.



StudMap

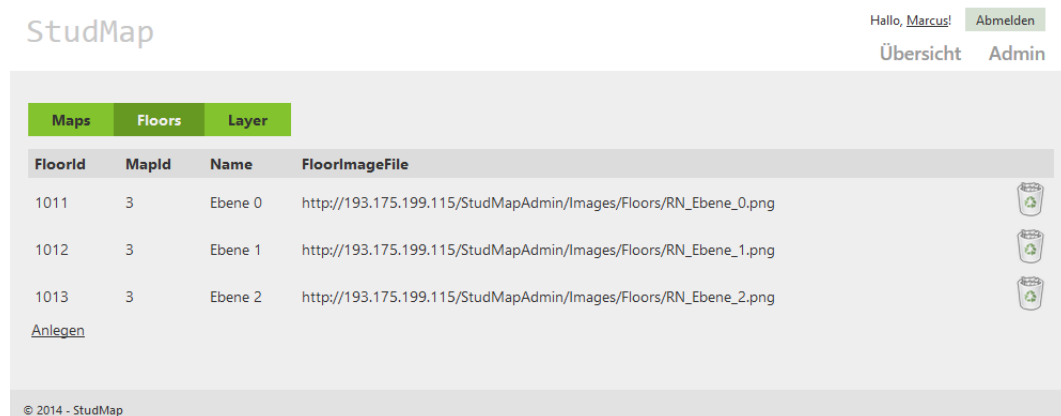
Hallo, Marcus! [Abmelden](#)

[Übersicht](#) [Admin](#)

Maps	Floors	Layer
MapId	Name	
3	Westfälische Hochschule	
<a href="#">Anlegen</a>		

© 2014 - StudMap




Die Verwaltung der einzelnen Floors zu einer Map werden in der nachfolgenden Grafik gezeigt. Durch einen Klick auf *Anlegen* kann eine neue Floor, mit der zugehörigen Kartengrundlage hinzugefügt werden.



StudMap

Hallo, Marcus! [Abmelden](#)

[Übersicht](#) [Admin](#)

Maps	Floors	Layer		
FloorId	MapId	Name	FloorImageFile	
1011	3	Ebene 0	http://193.175.199.115/StudMapAdmin/Images/Floors/RN_Ebene_0.png	
1012	3	Ebene 1	http://193.175.199.115/StudMapAdmin/Images/Floors/RN_Ebene_1.png	
1013	3	Ebene 2	http://193.175.199.115/StudMapAdmin/Images/Floors/RN_Ebene_2.png	
<a href="#">Anlegen</a>				

© 2014 - StudMap

Der Layer zeigt die Karte und einen gegebenenfalls erstellten Graphen mit Knoten und Kanten zu genau einem Floor an. Durch Verwendung spezieller Features kann der Administrator hier einen Graphen erstellen und Knoteninformationen hinterlegen. Des weiteren besteht die Möglichkeit Knoten mit Knoten anderer Floors zu verknüpfen. Basis für eine ergonomische Navigation der Karte ist die JavaScript-Bibliothek D3.



StudMap

Hallo, Marcus! [Abmelden](#)


[Übersicht](#) [Admin](#)

Maps

Floors

Layer

**Ebene 0**



**Knoteninformationen (1030)**

**Knoteninformationen**

**DisplayName:**

**RoomName:**

**QR-Code:**

**NFC Tag:**


**Pol Typ:**

**Pol Beschreibung:**

**Knoten**

Speichern

Abbrechen



Ebene 0



Über den Button *Abmelden* gelangen Sie zum Index unserer Webseite zurück.

## 9.3. Admin Spezifisches

### Neuen Benutzer mit Administrator Rechten versehen

Nachdem ein neuer Benutzer die Registratur erfolgreich abgeschlossen hat, ist dieser im Benutzerprofil noch nicht als Administrator gekennzeichnet. Derzeit ist es so, dass ein Datenbankadministrator in der Tabelle *webpages\_UsersInRoles* den Eintrag von 1 für Users auf 2 für Admins abändern muss. Nur Admins haben die Rechte Maps und Floors zu erstellen, sowie das Kartenmaterial mit Metainformationen zu bereichern.

### ELMAH

Als Logging Werkzeug haben wir das auf der .NET Plattform sehr verbreitete und OpenSource Tool *ELMAH* eingesetzt. *ELMAH*<sup>9</sup> loggt auftretende Fehler und Exceptions. Nach kurzem Installationsaufwand und Konfiguration einer Datenbank, in der die Fehler gespeichert werden, kann das Tool bereits genutzt werden. Genaue Installationsanleitungen findet Sie zu genüge im Internet<sup>10</sup>.

### Maintenance Tool

Während der Entwicklung des StudMap-Projektes ist es uns aufgefallen, dass wir zur Wartung und Ausführung bestimmter Aufgaben ein einfaches Tool hilfreich wäre. Daher haben wir eine WPF-Anwendung entwickelt, welche die folgenden Aufgaben erledigt.

### QR-Codes generieren

Eine Möglichkeit der Positionierung innerhalb des StudMap-Projektes ist das Einlesen von QR-Codes. In diesen QR-Codes stehen die im Kapitel *QR-Codes* beschriebenen Daten.

In der Anwendung werden alle *NodeInformation* ausgelesen und in einer Tabelle angezeigt. Anschließend kann in dieser Tabelle noch nach einem *Floor*, oder dem Namen des Raums gefiltert werden.

---

<sup>9</sup>ELMAH steht für "Error Logging Modules and Handlers"

<sup>10</sup><http://blog.thomasbandt.de/39/2380/de/blog/elmah-mit-aspnet-mvc-nutzen-und-fehler-loggen.html>



### *9. Admin*

---

Abschließend kann für alle ausgewählten Knoten ein QR-Code als PNG generiert werden. Zur Generierung der QR-Codes nutzen wir die Bibliothek [QrCode.Net](#).



## 10. Client

Der Client ist eine Android-Applikation für den Endbenutzer. Es handelt sich hierbei um einen Navigator. Wie jedes handelsübliche Navigationsgerät beinhaltet auch unsere Applikation eine Karte, in unserem Fall ein Gebäudeplan, da wir uns in unserem Projekt mit Indoor-Navigation beschäftigen. Darüber hinaus kann man einen Zielpunkt wählen und mit Hilfe eines QR-Codes, NFC-Tags oder des Wlans seine Position bestimmen. Sind Start und Zielpunkt bekannt, berechnet der Server eine Route und teilt diese dem Client mit, welcher diese daraufhin graphisch auf der Karte zur Anzeige bringt.

Des Weiteren gibt es Suchfunktionen und Auflistungen von besonders interessanten Orten (Points of Interest). Zur Benutzung der Wlan-Positionserkennung ist eine Registrierung und Anmeldung an unserem Server von Nöten.

### 10.1. Allgemeine Struktur

Die Navigator-Applikation wurde wie auch der Collector in Android umgesetzt. Es wird mindestens die Version 4.1 vorausgesetzt. Es galt folgende Hauptaufgaben zu erfüllen:

#### 10.1.1. Positionserkennung

Wie bereits im Kapitel [Collector](#) beschrieben, haben wir uns für Positionserkennung anhand von QR-Codes, NFC-Tags und Wlan-Fingerprinting entschieden. Für die Positionserkennung mittels NFC und WLAN greifen wir auf unsere Android-Bibliothek der Kernfunktionalitäten zurück. Während für die Interpretation des QR-Codes eine Fremd-Applikation zum Einsatz kommt. Diese muss bei der ersten Benutzung gegebenenfalls installiert werden.

Die letztendlich ermittelten Daten interpretieren wir mit Hilfe des Webservices, der wiederum Bestandteil unserer Kernfunktionalitäten ist.

#### 10.1.2. Anzeige und Navigation auf einer Karte

Wir haben uns für die Benutzung einer freien Bibliothek zur Anzeige von Karten entschieden. Nach intensiver Recherche fiel unsere Wahl dabei auf die Javascript Bibliothek [Floor Plan](#). Diese Bibliothek setzt auf der bekannten Javascript Bibliothek [D3.js](#) auf, die Daten in HTML Dokumenten visualisiert.

Das hat zur Folge, dass unsere Karte lediglich in einer Website platziert ist, die mittels einer WebView zur Anzeige gebracht wird. Dies hat für uns und den Endbenutzer den großen Vorteil, dass eine Änderung an der Karte nicht ein Update der Applikation nach sich zieht. Mittels einer definierten Schnittstelle lässt sich das Javascript aus unserer Android-Applikation heraus bedienen, so dass beispielsweise Positionsermittlungen automatisch auf der Karte nachgehalten werden. Dadurch ist die Navigation bei gewählten Zielpunkt vollständig.

### 10.1.3. Visuell ansprechende und intuitiv bedienbare Applikation

Eine Applikation sollte heute intuitiv bedienbar, übersichtlich und ansprechend sein, damit sie gerne und viel benutzt wird. Wir bauen dabei auf moderne Möglichkeiten des Android Betriebssystems. Im folgenden Kapitel werden wir diese näher erläutern.

## 10.2. Benutzeroberfläche

Neben der bereits erwähnten WebView mit einer hübschen Karte auf Basis der d3 Bibliothek, ist unsere gesamte Applikation in dunklen Tönen gehalten und bietet klare Linien bei einer Highlight-Farbe, die dem Grün der Westfälischen Hochschule sehr nahe kommt. Im Vordergrund der Applikation steht selbstverständlich die Karte, während grundlegende Funktionen wie die Suche oder der Aufruf des QR-Code-Scanners in der ActionBar geboten werden. Weitergreifende Funktionalitäten wie das Wechseln der Ebene oder Anmelden befinden sich in einem Drawer auf der linken Seite der Applikation, der auf Wunsch in das Bild hinein gezogen werden kann.

Zusätzliche Fenster wie z.B. die Auflistung der Points of Interest werden grundsätzlich in Dialogfenstern zur Anzeige gebracht, was der Übersichtlichkeit und Navigation durch die Applikation zu Gute kommt. Die Suche ist ein besonderes Event, welches in der ActionBar ausgeführt wird und dort die Anzeige verändert, sodass grundsätzlich Ordnung in der Applikation herrscht. Neben dem intuitiven Design der Applikation ist selbstverständlich auch die Bedienung der Karte äußerst benutzerfreundlich. Neben den bekannten Möglichkeiten des Multitouch, beispielsweise zum Zoomen, ist auch die Steuerung der Navigation selbsterklärend. Einen gewünschten Punkt angeklickt, schon kann es los gehen. Optisch ansprechend wird eine Route eingezeichnet und mit der Zielflagge gekennzeichnet.



## 11. Fazit

Im Verlauf unseres Projektes ist deutlich geworden, dass die Navigation und die Lokalisierung innerhalb von Gebäuden ein schwieriges Thema ist. Daher wählten wir für das Problem der Lokalisierung gleich mehrere Ansätze. Jedoch mussten wir erkennen, dass sowohl die Positionierung mittels Texterkennung der Raumschilder, als auch die Positionsermittlung über WLAN Fingerprints für unsere Anforderungen nicht bzw. nicht gut genug funktioniert haben. Aus diesem Grund beschäftigten wir uns mit alternativen Möglichkeiten und entschieden uns dafür die Positionsdaten auf NFC Tags und QR Codes zu speichern und diese im Gebäude zu platzieren. So ist die zuverlässige Positionsbestimmung immer durch Scannen eines NFC Tags oder eines QR Codes möglich.

Bei der Umsetzung dieser Ideen haben wir uns für ein Backend basierend auf Microsoft Technologien entschieden. Für die Entwicklung der Datenbankstruktur verwendeten wir das Entity Framework, mit dem die Datenbank automatisch aus unserem Datenmodell generiert werden konnte. Zusätzlich konnten komplexe Datenbankabfragen einfach umgesetzt und ausgewertet werden. Dadurch haben wir gerade zu Beginn des Projektes eine Menge Arbeit und Zeit gespart. Des Weiteren haben wir uns bei der administrativen Oberfläche für eine ASP.NET Webapplikation entschieden, in der wesentliche Bestandteile der Benutzeroberfläche und eine Benutzerverwaltung bereits integriert waren. Auch dadurch haben wir uns viel Arbeit erspart und konnten uns auf die wesentlichen Probleme konzentrieren.

Begeistert von diesen vielen Möglichkeiten, entschieden wir uns unsere Anwendung in der Windows Azure Cloud zu hosten und meldeten daher einen entsprechenden Studenten Account bei Microsoft an. Leider erhielten wir über Wochen kein eindeutiges Feedback von Microsoft weshalb wir uns letztendlich für einen eigenen Windows Server innerhalb der Hochschule entschieden haben.

Abschließend blicken wir auf ein komplexes Projekt mit vielen Herausforderungen zurück. Durch die unterschiedlichen Technologien haben wir alle etwas Neues kennengelernt und weitere wichtige Erfahrung sammeln können. Innerhalb des Projekts gab es allerdings nicht nur technische Herausforderungen, auch die Projektorganisation selbst, sowie die Zusammenarbeit im Team ist in jedem Projekt eine Herausforderung. Gemeinsam haben wir es, trotz der vielen Schwierigkeiten, geschafft unser Projektziel zu erreichen. So sind insgesamt drei Anwendungen zur Navigation innerhalb unserer Hochschule entstanden, die mit entsprechenden administrativen Aufwand auch wirklich eingesetzt werden könnten.



## 12. Ausblick

Nach Abschluss unserer Projektarbeit bleiben noch viele innovative Ideen offen. Dazu gehören Dinge, die in handelsüblichen Navigationsgeräten zum Standard gehören genauso wie kreative Ideen, die das heutige Internetzeitalter ermöglicht.

Entsprechend der Erwartung eines Benutzers, sollte eine Navigation fließend stattfinden, das heißt anhand der aktuellen Position werden richtungsweisende Pfeile eingeblendet, eine Sprachausgabe unterstützt die Wegfindung und eine Angabe über die noch zu gehende Distanz informiert den Benutzer. Diese und ähnliche Funktionen kennt man bereits aus handelsüblichen Navigationsgeräten und könnte sich eben bei diesen Denkanstöße für Erweiterungen einholen.

Grundsätzlich besteht die Möglichkeit unser System auch in anderen Gebäuden, wie Universitäten oder öffentlichen Einrichtungen, einzusetzen. Es erfordert lediglich eine entsprechende Administration und Einrichtung des Systems.

Des Weiteren bietet es sich an die in einem Knoten gespeicherten Informationen deutlich zu erweitern. Da gäbe es zum einen die Idee Stundenpläne oder Sprechstundenzeiten mit den Knoten zu verknüpfen oder zumindest auf externe Anwendungen, die den gewünschten Zweck erfüllen, zu verlinken. Zum anderen könnte man eine virtuelle Pinnwand den Knoten zu weisen, auf denen jegliche Art von öffentlich oder auch privaten Nachrichten hinterlassen werden können. Private Nachrichten ließen sich ermöglichen, wenn man eine Art Freundesliste pflegt. Die bereits registrierten Mitglieder bieten dazu eine gute Grundlage. Darüber hinaus würde eine solche Freundesliste den Weg ebnen, den Aufenthaltsort von Freunden in dem Gebäude sehen und sich dorthin navigieren lassen zu können.

Die bereits vorgestellten Erweiterungsmöglichkeiten sind selbstredend nicht alle vorstellbaren. Der Fantasie und dem Erfindergeist der Entwickler sind keine Grenzen gesetzt mit Ausnahme derer, dass die Anwendung benutzbar und benutzerfreundlich bleiben sollte.

Um eine letzte Vision mit auf den Weg zu geben, wäre dem Benutzer vielleicht daran gelegen Informationen in sozialen Netzwerken teilen zu können. Eine Möglichkeit dies mittels eines einfachen Knopfdruckes zu erledigen, ist eine von vielen verbleibenden Komfortmöglichkeiten.



## A. Server

Der Webserver ist unter 193.175.199.115 erreichbar:

- StudMap.Admin: <http://193.175.199.115/StudMapAdmin>
- StudMap.Client: <http://193.175.199.115/StudMapClient>
- StudMap.Service: <http://193.175.199.115/StudMapService>

### A.1. Software

Auf dem Server sind folgende Softwarekomponenten installiert:

- Internet Information Services (Version 7.5)
- Microsoft SQL Server 2012

### A.2. Einrichtung IIS

Der IIS kann sowohl auf einem Windows Server als auch auf einer normalen Windows Installation eingerichtet werden. Dazu muss unter „Programme“ „Windows-Features installieren oder deinstallieren“ ausgewählt werden. In dem Dialog kann der IIS installiert werden. Da wir in unserem Projekt die ASP.NET Technologie verwenden muss diese am IIS installiert werden.

Nach der Installation ist der IIS-Manager installiert worin Webseiten eingerichtet werden können.

Da wir auf dem Server eine normale Windows Installation laufen haben, kann kein Web Deploy aufgespielt werden. Daher müssen in Visual Studio Deployment Packages erstellt werden, welche über einen Rechtsklick auf „StudMap“ > „Bereitstellen“ > „Anwendung importieren“ importiert werden können.

### A.3. Einrichtung SQL Server

Im Projekt wird ein Microsoft SQL Server 2012 verwendet. Da unsere Anwendung auf komplexe Managementfunktionen verzichtet kann auch die kostenlose Express Version genutzt werden.



Der Aufbau der Datenbank kann aus den unten aufgeführten Datenbankskripten hergestellt werden.

```
1 USE [master]
2 GO
3 /***** Object: Database [StudMap] Script Date: 07.01.2014 16:09:00 *****/
4 CREATE DATABASE [StudMap]
5     CONTAINMENT = NONE
6     ON PRIMARY
7     ( NAME = N'StudMap_Data', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.
        STUDMAPDB\MSSQL\DATA\StudMap_Data.mdf' , SIZE = 10112KB , MAXSIZE = UNLIMITED, FILEGROWTH
        = 10%)
8     LOG ON
9     ( NAME = N'StudMap_Log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.STUDMAPDB
        \MSSQL\DATA\StudMap_Log.ldf' , SIZE = 2048KB , MAXSIZE = 2048GB , FILEGROWTH = 1024KB )
10 GO
11 ALTER DATABASE [StudMap] SET COMPATIBILITY_LEVEL = 110
12 GO
13 IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
14 begin
15 EXEC [StudMap].[dbo].[sp_fulltext_database] @action = 'enable'
16 end
17 GO
18 ALTER DATABASE [StudMap] SET ANSI_NULL_DEFAULT OFF
19 GO
20 ALTER DATABASE [StudMap] SET ANSI_NULLS OFF
21 GO
22 ALTER DATABASE [StudMap] SET ANSI_PADDING OFF
23 GO
24 ALTER DATABASE [StudMap] SET ANSI_WARNINGS OFF
25 GO
26 ALTER DATABASE [StudMap] SET ARITHABORT OFF
27 GO
28 ALTER DATABASE [StudMap] SET AUTO_CLOSE OFF
29 GO
30 ALTER DATABASE [StudMap] SET AUTO_CREATE_STATISTICS ON
31 GO
32 ALTER DATABASE [StudMap] SET AUTO_SHRINK OFF
33 GO
34 ALTER DATABASE [StudMap] SET AUTO_UPDATE_STATISTICS ON
35 GO
36 ALTER DATABASE [StudMap] SET CURSOR_CLOSE_ON_COMMIT OFF
37 GO
38 ALTER DATABASE [StudMap] SET CURSOR_DEFAULT GLOBAL
39 GO
40 ALTER DATABASE [StudMap] SET CONCAT_NULL_YIELDS_NULL OFF
41 GO
42 ALTER DATABASE [StudMap] SET NUMERIC_ROUNDABORT OFF
43 GO
44 ALTER DATABASE [StudMap] SET QUOTED_IDENTIFIER OFF
45 GO
46 ALTER DATABASE [StudMap] SET RECURSIVE_TRIGGERS OFF
47 GO
48 ALTER DATABASE [StudMap] SET ENABLE_BROKER
49 GO
50 ALTER DATABASE [StudMap] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
51 GO
```

```
52 ALTER DATABASE [StudMap] SET DATE_CORRELATION_OPTIMIZATION OFF
53 GO
54 ALTER DATABASE [StudMap] SET TRUSTWORTHY OFF
55 GO
56 ALTER DATABASE [StudMap] SET ALLOW_SNAPSHOT_ISOLATION OFF
57 GO
58 ALTER DATABASE [StudMap] SET PARAMETERIZATION SIMPLE
59 GO
60 ALTER DATABASE [StudMap] SET READ_COMMITTED_SNAPSHOT OFF
61 GO
62 ALTER DATABASE [StudMap] SET HONOR_BROKER_PRIORITY OFF
63 GO
64 ALTER DATABASE [StudMap] SET RECOVERY FULL
65 GO
66 ALTER DATABASE [StudMap] SET MULTI_USER
67 GO
68 ALTER DATABASE [StudMap] SET PAGE_VERIFY CHECKSUM
69 GO
70 ALTER DATABASE [StudMap] SET DB_CHAINING OFF
71 GO
72 ALTER DATABASE [StudMap] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
73 GO
74 ALTER DATABASE [StudMap] SET TARGET_RECOVERY_TIME = 0 SECONDS
75 GO
76 EXEC sys.sp_db_vardecimal_storage_format N'StudMap', N'ON'
77 GO
78 USE [StudMap]
79 GO
80 /***** Object: User [StudMapDBUser] Script Date: 07.01.2014 16:09:01 *****/
81 CREATE USER [StudMapDBUser] FOR LOGIN [StudMapDBUser] WITH DEFAULT_SCHEMA=[db_datareader]
82 GO
83 /***** Object: User [StudMapDBAdmin] Script Date: 07.01.2014 16:09:01 *****/
84 CREATE USER [StudMapDBAdmin] FOR LOGIN [StudMapDBAdmin] WITH DEFAULT_SCHEMA=[dbo]
85 GO
86 ALTER ROLE [db_datareader] ADD MEMBER [StudMapDBUser]
87 GO
88 ALTER ROLE [db_datawriter] ADD MEMBER [StudMapDBUser]
89 GO
90 ALTER ROLE [db_owner] ADD MEMBER [StudMapDBAdmin]
91 GO
92 /***** Object: StoredProcedure [dbo].[sp_DeleteFloor] Script Date: 07.01.2014 16:09:02 *****/
93 SET ANSI_NULLS ON
94 GO
95 SET QUOTED_IDENTIFIER ON
96 GO
97
98 -- =====
99 -- Author: Thomas Buning
100 -- Create date: 08.11.2013
101 -- Description: Loescht einen Floor aus der Datenbank
102 -- =====
103 CREATE PROCEDURE [dbo].[sp_DeleteFloor]
104     @FloorId int = 0
105 AS
106 BEGIN
107     SET NOCOUNT ON;
```



A. Server

---

```
108
109 IF @FloorId > 0
110 BEGIN
111
112     exec sp_DeleteGraphFromFloor @FloorId = @FloorId
113
114     DELETE FROM Floors
115     WHERE Id = @FloorId
116
117 END
118 END
119
120
121 GO
122 /***** Object: StoredProcedure [dbo].[sp_DeleteGraphFromFloor] Script Date: 07.01.2014
123      16:09:02 *****/
123 SET ANSI_NULLS ON
124 GO
125 SET QUOTED_IDENTIFIER ON
126 GO
127
128 -- =====
129 -- Author:   Thomas Buning
130 -- Create date: 19.11.2013
131 -- Description: L scht alle Informationen eines Graphen zu einem Floor
132 -- =====
133 CREATE PROCEDURE [dbo].[sp_DeleteGraphFromFloor]
134     @FloorId int = 0
135 AS
136 BEGIN
137     SET NOCOUNT ON;
138
139     IF @FloorId > 0
140     BEGIN
141
142         DELETE FROM NodeInformation
143         WHERE NodeId IN
144         (
145             SELECT n.Id
146             FROM Nodes n
147             WHERE n.FloorId = @FloorId
148         )
149
150         DELETE FROM PoIs
151         WHERE Id IN
152         (
153             SELECT ni.PoiId
154             FROM NodeInformation ni
155             LEFT JOIN Nodes n ON n.Id = ni.NodeId
156             WHERE n.FloorId = @FloorId
157         )
158
159         DELETE FROM Edges
160         WHERE NodeStartId IN
161         (
162             SELECT n.Id
163             FROM Nodes n
```



A. Server

```
164     WHERE n.FloorId = @FloorId
165   ) OR NodeEndId IN
166   (
167     SELECT n.Id
168     FROM Nodes n
169     WHERE n.FloorId = @FloorId
170   )
171
172   DELETE FROM Nodes
173   WHERE FloorId = @FloorId
174 END
175 END
176
177
178 GO
179 /***** Object: StoredProcedure [dbo].[sp_DeleteMap] Script Date: 07.01.2014 16:09:02 *****/
180 SET ANSI_NULLS ON
181 GO
182 SET QUOTED_IDENTIFIER ON
183 GO
184
185 -- =====
186 -- Author:   Thomas Buning
187 -- Create date: 08.11.2013
188 -- Description: Loescht zu einer Karte alle Elemente aus der Datenbank
189 -- =====
190 CREATE PROCEDURE [dbo].[sp_DeleteMap]
191
192     @MapId int = 0
193
194 AS
195 BEGIN
196     SET NOCOUNT ON;
197
198     IF @MapId <> 0
199     BEGIN
200         --Es muss der Graph geloescht werden
201         DELETE FROM Edges
202         WHERE GraphId IN
203         (
204             SELECT g.Id
205             FROM Graphs g
206             WHERE g.MapId = @MapId
207         )
208
209         DELETE FROM Graphs
210         WHERE MapId = @MapId
211
212         --Floors, Nodes und NodeInformationen loeschen
213         DECLARE @id int;
214
215         DECLARE floors_cursor CURSOR FAST_FORWARD FOR
216         SELECT f.Id
217         FROM Floors f
218         WHERE f.MapId = @MapId
219
220         OPEN floors_cursor
```



```
221     FETCH NEXT FROM floors_cursor
222     INTO @id
223
224     WHILE @@FETCH_STATUS = 0
225     BEGIN
226         exec sp_DeleteFloor @FloorId = @id
227
228         FETCH NEXT FROM floors_cursor
229         INTO @id
230     END
231
232     CLOSE floors_cursor
233
234     DELETE FROM Maps
235     WHERE Id = @MapId
236 END
237
238
239 END
240
241
242 GO
243 /***** Object: Table [dbo].[AccessPoints] Script Date: 07.01.2014 16:09:02 *****/
244 SET ANSI_NULLS ON
245 GO
246 SET QUOTED_IDENTIFIER ON
247 GO
248 SET ANSI_PADDING ON
249 GO
250 CREATE TABLE [dbo].[AccessPoints](
251     [Id] [int] IDENTITY(1,1) NOT NULL,
252     [MAC] [char](12) NOT NULL,
253     CONSTRAINT [PK_AccessPoints] PRIMARY KEY CLUSTERED
254 (
255     [Id] ASC
256 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
257     ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
258
259 GO
260 SET ANSI_PADDING OFF
261 GO
262 /***** Object: Table [dbo].[AccessPointScans] Script Date: 07.01.2014 16:09:02 *****/
263 SET ANSI_NULLS ON
264 GO
265 SET QUOTED_IDENTIFIER ON
266 GO
267 CREATE TABLE [dbo].[AccessPointScans](
268     [Id] [int] IDENTITY(1,1) NOT NULL,
269     [FingerprintId] [int] NOT NULL,
270     [AccessPointId] [int] NOT NULL,
271     [RecievedSignalStrength] [int] NOT NULL,
272     CONSTRAINT [PK_AccessPointScans] PRIMARY KEY CLUSTERED
273 (
274     [Id] ASC
275 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
276     ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```



```
276 ) ON [PRIMARY]
277
278 GO
279 /***** Object: Table [dbo].[ActiveUsers] Script Date: 07.01.2014 16:09:02 *****/
280 SET ANSI_NULLS ON
281 GO
282 SET QUOTED_IDENTIFIER ON
283 GO
284 CREATE TABLE [dbo].[ActiveUsers](
285     [Id] [int] IDENTITY(1,1) NOT NULL,
286     [UserId] [int] NOT NULL,
287     [LoginDate] [datetime] NOT NULL,
288     CONSTRAINT [PK_ActiveUsers] PRIMARY KEY CLUSTERED
289     (
290         [Id] ASC
291     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
292         ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
293 ) ON [PRIMARY]
294 GO
295 /***** Object: Table [dbo].[Edges] Script Date: 07.01.2014 16:09:02 *****/
296 SET ANSI_NULLS ON
297 GO
298 SET QUOTED_IDENTIFIER ON
299 GO
300 CREATE TABLE [dbo].[Edges](
301     [Id] [int] IDENTITY(1,1) NOT NULL,
302     [GraphId] [int] NOT NULL,
303     [NodeStartId] [int] NOT NULL,
304     [NodeEndId] [int] NOT NULL,
305     [CreationTime] [datetime] NOT NULL,
306     CONSTRAINT [PK_Edges] PRIMARY KEY CLUSTERED
307     (
308         [Id] ASC
309     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
310         ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
311 ) ON [PRIMARY]
312 GO
313 /***** Object: Table [dbo].[Fingerprints] Script Date: 07.01.2014 16:09:02 *****/
314 SET ANSI_NULLS ON
315 GO
316 SET QUOTED_IDENTIFIER ON
317 GO
318 CREATE TABLE [dbo].[Fingerprints](
319     [Id] [int] IDENTITY(1,1) NOT NULL,
320     [NodeId] [int] NOT NULL,
321     CONSTRAINT [PK_Fingerprints] PRIMARY KEY CLUSTERED
322     (
323         [Id] ASC
324     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
325         ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
326 ) ON [PRIMARY]
327 GO
328 /***** Object: Table [dbo].[Floors] Script Date: 07.01.2014 16:09:02 *****/
329 SET ANSI_NULLS ON
```



```
330 GO
331 SET QUOTED_IDENTIFIER ON
332 GO
333 CREATE TABLE [dbo].[Floors](
334     [Id] [int] IDENTITY(1,1) NOT NULL,
335     [MapId] [int] NOT NULL,
336     [Name] [nvarchar](255) NOT NULL,
337     [ImageUrl] [nvarchar](max) NOT NULL,
338     [CreationTime] [datetime] NOT NULL,
339     CONSTRAINT [PK_Floors] PRIMARY KEY CLUSTERED
340 (
341     [Id] ASC
342 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
343 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
344
345 GO
346 /***** Object: Table [dbo].[Graphs] Script Date: 07.01.2014 16:09:02 *****/
347 SET ANSI_NULLS ON
348 GO
349 SET QUOTED_IDENTIFIER ON
350 GO
351 CREATE TABLE [dbo].[Graphs](
352     [Id] [int] IDENTITY(1,1) NOT NULL,
353     [MapId] [int] NOT NULL,
354     [CreationTime] [datetime] NOT NULL,
355     CONSTRAINT [PK_Graphs] PRIMARY KEY CLUSTERED
356 (
357     [Id] ASC
358 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
359 ) ON [PRIMARY]
360
361 GO
362 /***** Object: Table [dbo].[Maps] Script Date: 07.01.2014 16:09:02 *****/
363 SET ANSI_NULLS ON
364 GO
365 SET QUOTED_IDENTIFIER ON
366 GO
367 CREATE TABLE [dbo].[Maps](
368     [Id] [int] IDENTITY(1,1) NOT NULL,
369     [Name] [nvarchar](255) NULL,
370     [CreationTime] [datetime] NOT NULL,
371     CONSTRAINT [PK_Maps] PRIMARY KEY CLUSTERED
372 (
373     [Id] ASC
374 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
375 ) ON [PRIMARY]
376
377 GO
378 /***** Object: Table [dbo].[NodeInformation] Script Date: 07.01.2014 16:09:02 *****/
379 SET ANSI_NULLS ON
380 GO
381 SET QUOTED_IDENTIFIER ON
382 GO
383 CREATE TABLE [dbo].[NodeInformation](
```



```
384 [Id] [int] IDENTITY(1,1) NOT NULL,
385 [NodeId] [int] NOT NULL,
386 [DisplayName] [nvarchar](50) NULL,
387 [RoomName] [nvarchar](255) NULL,
388 [QRCode] [nvarchar](255) NULL,
389 [NFCTag] [nvarchar](50) NULL,
390 [PoiId] [int] NULL,
391 [CreationTime] [datetime] NOT NULL,
392 CONSTRAINT [PK_NodeInformation] PRIMARY KEY CLUSTERED
393 (
394 [Id] ASC
395 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
396 ) ON [PRIMARY]
397
398 GO
399 /***** Object: Table [dbo].[Nodes] Script Date: 07.01.2014 16:09:02 *****/
400 SET ANSI_NULLS ON
401 GO
402 SET QUOTED_IDENTIFIER ON
403 GO
404 CREATE TABLE [dbo].[Nodes](
405 [Id] [int] IDENTITY(1,1) NOT NULL,
406 [FloorId] [int] NOT NULL,
407 [X] [decimal](18, 17) NOT NULL,
408 [Y] [decimal](18, 17) NOT NULL,
409 [CreationTime] [datetime] NOT NULL,
410 CONSTRAINT [PK_Nodes] PRIMARY KEY CLUSTERED
411 (
412 [Id] ASC
413 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
414 ) ON [PRIMARY]
415
416 GO
417 /***** Object: Table [dbo].[PoIs] Script Date: 07.01.2014 16:09:02 *****/
418 SET ANSI_NULLS ON
419 GO
420 SET QUOTED_IDENTIFIER ON
421 GO
422 CREATE TABLE [dbo].[PoIs](
423 [Id] [int] IDENTITY(1,1) NOT NULL,
424 [TypeId] [int] NOT NULL,
425 [Description] [nvarchar](max) NULL,
426 CONSTRAINT [PK_PoIs] PRIMARY KEY CLUSTERED
427 (
428 [Id] ASC
429 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
430 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
431
432 GO
433 /***** Object: Table [dbo].[PoiTypes] Script Date: 07.01.2014 16:09:02 *****/
434 SET ANSI_NULLS ON
435 GO
436 SET QUOTED_IDENTIFIER ON
437 GO
```





A. Server

---

```
438 CREATE TABLE [dbo].[PoiTypes](
439     [Id] [int] IDENTITY(1,1) NOT NULL,
440     [Name] [nvarchar](255) NULL,
441     CONSTRAINT [PK_PoiTypes] PRIMARY KEY CLUSTERED
442     (
443         [Id] ASC
444     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
         ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
445 ) ON [PRIMARY]
446
447 GO
448 /***** Object: Table [dbo].[UserProfile] Script Date: 07.01.2014 16:09:02 *****/
449 SET ANSI_NULLS ON
450 GO
451 SET QUOTED_IDENTIFIER ON
452 GO
453 CREATE TABLE [dbo].[UserProfile](
454     [UserId] [int] IDENTITY(1,1) NOT NULL,
455     [UserName] [nvarchar](56) NOT NULL,
456     CONSTRAINT [PK_UserProfile] PRIMARY KEY CLUSTERED
457     (
458         [UserId] ASC
459     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
         ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
460 ) ON [PRIMARY]
461
462 GO
463 /***** Object: Table [dbo].[webpages_Membership] Script Date: 07.01.2014 16:09:02 *****/
464 SET ANSI_NULLS ON
465 GO
466 SET QUOTED_IDENTIFIER ON
467 GO
468 CREATE TABLE [dbo].[webpages_Membership](
469     [UserId] [int] NOT NULL,
470     [CreateDate] [datetime] NULL,
471     [ConfirmationToken] [nvarchar](128) NULL,
472     [IsConfirmed] [bit] NULL,
473     [LastPasswordFailureDate] [datetime] NULL,
474     [PasswordFailuresSinceLastSuccess] [int] NOT NULL,
475     [Password] [nvarchar](128) NOT NULL,
476     [PasswordChangedDate] [datetime] NULL,
477     [PasswordSalt] [nvarchar](128) NOT NULL,
478     [PasswordVerificationToken] [nvarchar](128) NULL,
479     [PasswordVerificationTokenExpirationDate] [datetime] NULL
480 ) ON [PRIMARY]
481
482 GO
483 /***** Object: Table [dbo].[webpages_OAuthMembership] Script Date: 07.01.2014 16:09:02 *****/
484 SET ANSI_NULLS ON
485 GO
486 SET QUOTED_IDENTIFIER ON
487 GO
488 CREATE TABLE [dbo].[webpages_OAuthMembership](
489     [Provider] [nvarchar](30) NOT NULL,
490     [ProviderUserId] [nvarchar](100) NOT NULL,
491     [UserId] [int] NOT NULL
```



A. Server

---

```
492 ) ON [PRIMARY]
493
494 GO
495 /***** Object: Table [dbo].[webpages_Roles] Script Date: 07.01.2014 16:09:02 *****/
496 SET ANSI_NULLS ON
497 GO
498 SET QUOTED_IDENTIFIER ON
499 GO
500 CREATE TABLE [dbo].[webpages_Roles](
501     [RoleId] [int] NOT NULL,
502     [RoleName] [nvarchar](256) NOT NULL,
503     CONSTRAINT [PK_webpages_Roles] PRIMARY KEY CLUSTERED
504     (
505         [RoleId] ASC
506     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
507         ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
508 ) ON [PRIMARY]
509 GO
510 /***** Object: Table [dbo].[webpages_UsersInRoles] Script Date: 07.01.2014 16:09:02 *****/
511 SET ANSI_NULLS ON
512 GO
513 SET QUOTED_IDENTIFIER ON
514 GO
515 CREATE TABLE [dbo].[webpages_UsersInRoles](
516     [UserId] [int] NOT NULL,
517     [RoleId] [int] NOT NULL
518 ) ON [PRIMARY]
519 GO
520 GO
521 /***** Object: View [dbo].[NodeInformationForMap] Script Date: 07.01.2014 16:09:02 *****/
522 SET ANSI_NULLS ON
523 GO
524 SET QUOTED_IDENTIFIER ON
525 GO
526
527 CREATE VIEW [dbo].[NodeInformationForMap]
528 AS
529 SELECT      dbo.Maps.Id AS MapId, dbo.NodeInformation.DisplayName, dbo.NodeInformation.
530             RoomName, dbo.Nodes.Id AS NodeId, dbo.Floors.Id AS FloorId
531 FROM        dbo.Floors INNER JOIN
532             dbo.Maps ON dbo.Floors.MapId = dbo.Maps.Id INNER JOIN
533             dbo.Nodes ON dbo.Floors.Id = dbo.Nodes.FloorId INNER JOIN
534             dbo.NodeInformation ON dbo.Nodes.Id = dbo.NodeInformation.NodeId
535 WHERE       (dbo.NodeInformation.DisplayName <> N'') AND (dbo.NodeInformation.RoomName <> N'')
536             AND (dbo.NodeInformation.DisplayName IS NOT NULL) AND
537             (dbo.NodeInformation.RoomName IS NOT NULL)
538 GO
539 /***** Object: View [dbo].[PoisForMap] Script Date: 07.01.2014 16:09:02 *****/
540 SET ANSI_NULLS ON
541 GO
542 SET QUOTED_IDENTIFIER ON
543 GO
544 CREATE VIEW [dbo].[PoisForMap]
545 AS
```



```

546 SELECT      dbo.Maps.Id AS MapId, dbo.PoiTypes.Id AS PoiTypeId, dbo.PoiTypes.Name AS
              PoiTypeName, dbo.PoIs.Id AS PoiId, dbo.PoIs.Description AS PoiDescription,
547              dbo.NodeInformation.DisplayName, dbo.NodeInformation.RoomName, dbo.
              Nodes.Id AS NodeId, dbo.Floors.Id AS FloorId
548 FROM          dbo.PoIs INNER JOIN
549              dbo.PoiTypes ON dbo.PoIs.TypeId = dbo.PoiTypes.Id INNER JOIN
550              dbo.NodeInformation ON dbo.PoIs.Id = dbo.NodeInformation.PoiId INNER
              JOIN
551              dbo.Floors INNER JOIN
552              dbo.Nodes ON dbo.Floors.Id = dbo.Nodes.FloorId INNER JOIN
553              dbo.Maps ON dbo.Floors.MapId = dbo.Maps.Id ON dbo.NodeInformation.
              NodeId = dbo.Nodes.Id
554 WHERE         (dbo.NodeInformation.Id IS NOT NULL) AND (dbo.PoIs.Id IS NOT NULL)
555
556 GO
557 /***** Object: View [dbo].[RoomsForMap] Script Date: 07.01.2014 16:09:02 *****/
558 SET ANSI_NULLS ON
559 GO
560 SET QUOTED_IDENTIFIER ON
561 GO
562 CREATE VIEW [dbo].[RoomsForMap]
563 AS
564 SELECT      n.Id AS NodeId, ni.DisplayName, ni.RoomName, f.Id AS FloorId
565 FROM          dbo.Maps AS m LEFT OUTER JOIN
566              dbo.Floors AS f ON f.MapId = m.Id LEFT OUTER JOIN
567              dbo.Nodes AS n ON n.FloorId = f.Id LEFT OUTER JOIN
568              dbo.NodeInformation AS ni ON n.Id = ni.NodeId
569 WHERE         (ni.DisplayName <> '') AND (ni.Id IS NOT NULL) AND (ni.DisplayName IS NOT NULL) OR
570              (ni.Id IS NOT NULL) AND (ni.RoomName <> '') AND (ni.RoomName IS NOT
              NULL)
571
572 GO
573 /***** Object: View [dbo].[RSSDistribution] Script Date: 07.01.2014 16:09:02 *****/
574 SET ANSI_NULLS ON
575 GO
576 SET QUOTED_IDENTIFIER ON
577 GO
578 CREATE VIEW [dbo].[RSSDistribution]
579 AS
580 SELECT      f.NodeId, aps.AccessPointId, AVG(aps.RecievedSignalStrength) AS AvgRSS, STDEV(aps.
              RecievedSignalStrength) AS StDevRSS
581 FROM          dbo.AccessPointScans AS aps INNER JOIN
582              dbo.Fingerprints AS f ON f.Id = aps.FingerprintId
583 GROUP BY f.NodeId, aps.AccessPointId
584 HAVING (STDEV(aps.RecievedSignalStrength) > 0.5) AND (COUNT(aps.Id) > 3)
585
586 GO
587 ALTER TABLE [dbo].[AccessPointScans] WITH CHECK ADD CONSTRAINT [
              FK_AccessPointScans_AccessPoints] FOREIGN KEY([AccessPointId])
588 REFERENCES [dbo].[AccessPoints] ([Id])
589 GO
590 ALTER TABLE [dbo].[AccessPointScans] CHECK CONSTRAINT [FK_AccessPointScans_AccessPoints]
591 GO
592 ALTER TABLE [dbo].[AccessPointScans] WITH CHECK ADD CONSTRAINT [
              FK_AccessPointScans_Fingerprints] FOREIGN KEY([FingerprintId])
593 REFERENCES [dbo].[Fingerprints] ([Id])
594 GO

```



```
595 ALTER TABLE [dbo].[AccessPointScans] CHECK CONSTRAINT [FK_AccessPointScans_Fingerprints]
596 GO
597 ALTER TABLE [dbo].[ActiveUsers] WITH CHECK ADD CONSTRAINT [FK_ActiveUsers_UserProfile]
    FOREIGN KEY([UserId])
598 REFERENCES [dbo].[UserProfile] ([UserId])
599 GO
600 ALTER TABLE [dbo].[ActiveUsers] CHECK CONSTRAINT [FK_ActiveUsers_UserProfile]
601 GO
602 ALTER TABLE [dbo].[Edges] WITH CHECK ADD CONSTRAINT [FK_Edges_Graphs] FOREIGN KEY([GraphId])
603 REFERENCES [dbo].[Graphs] ([Id])
604 GO
605 ALTER TABLE [dbo].[Edges] CHECK CONSTRAINT [FK_Edges_Graphs]
606 GO
607 ALTER TABLE [dbo].[Edges] WITH CHECK ADD CONSTRAINT [FK_Edges_Nodes] FOREIGN KEY([NodeStartId
    ])
608 REFERENCES [dbo].[Nodes] ([Id])
609 GO
610 ALTER TABLE [dbo].[Edges] CHECK CONSTRAINT [FK_Edges_Nodes]
611 GO
612 ALTER TABLE [dbo].[Edges] WITH CHECK ADD CONSTRAINT [FK_Edges_Nodes1] FOREIGN KEY([NodeEndId
    ])
613 REFERENCES [dbo].[Nodes] ([Id])
614 GO
615 ALTER TABLE [dbo].[Edges] CHECK CONSTRAINT [FK_Edges_Nodes1]
616 GO
617 ALTER TABLE [dbo].[Fingerprints] WITH CHECK ADD CONSTRAINT [FK_Fingerprints_Nodes] FOREIGN
    KEY([NodeId])
618 REFERENCES [dbo].[Nodes] ([Id])
619 GO
620 ALTER TABLE [dbo].[Fingerprints] CHECK CONSTRAINT [FK_Fingerprints_Nodes]
621 GO
622 ALTER TABLE [dbo].[Floors] WITH CHECK ADD CONSTRAINT [FK_Floors_Maps] FOREIGN KEY([MapId])
623 REFERENCES [dbo].[Maps] ([Id])
624 GO
625 ALTER TABLE [dbo].[Floors] CHECK CONSTRAINT [FK_Floors_Maps]
626 GO
627 ALTER TABLE [dbo].[Graphs] WITH CHECK ADD CONSTRAINT [FK_Graphs_Maps] FOREIGN KEY([MapId])
628 REFERENCES [dbo].[Maps] ([Id])
629 GO
630 ALTER TABLE [dbo].[Graphs] CHECK CONSTRAINT [FK_Graphs_Maps]
631 GO
632 ALTER TABLE [dbo].[NodeInformation] WITH CHECK ADD CONSTRAINT [FK_NodeInformation_Nodes]
    FOREIGN KEY([NodeId])
633 REFERENCES [dbo].[Nodes] ([Id])
634 GO
635 ALTER TABLE [dbo].[NodeInformation] CHECK CONSTRAINT [FK_NodeInformation_Nodes]
636 GO
637 ALTER TABLE [dbo].[NodeInformation] WITH CHECK ADD CONSTRAINT [FK_NodeInformation_PoIs]
    FOREIGN KEY([PoiId])
638 REFERENCES [dbo].[PoIs] ([Id])
639 GO
640 ALTER TABLE [dbo].[NodeInformation] CHECK CONSTRAINT [FK_NodeInformation_PoIs]
641 GO
642 ALTER TABLE [dbo].[Nodes] WITH CHECK ADD CONSTRAINT [FK_Nodes_Floors] FOREIGN KEY([FloorId])
643 REFERENCES [dbo].[Floors] ([Id])
644 GO
645 ALTER TABLE [dbo].[Nodes] CHECK CONSTRAINT [FK_Nodes_Floors]
```



```

646 GO
647 ALTER TABLE [dbo].[PoIs] WITH CHECK ADD CONSTRAINT [FK_PoIs_PoiTypes] FOREIGN KEY([TypeId])
648 REFERENCES [dbo].[PoiTypes] ([Id])
649 GO
650 ALTER TABLE [dbo].[PoIs] CHECK CONSTRAINT [FK_PoIs_PoiTypes]
651 GO
652 ALTER TABLE [dbo].[webpages_Membership] WITH CHECK ADD CONSTRAINT [
        FK_webpages_Membership_UserProfile] FOREIGN KEY([UserId])
653 REFERENCES [dbo].[UserProfile] ([UserId])
654 GO
655 ALTER TABLE [dbo].[webpages_Membership] CHECK CONSTRAINT [FK_webpages_Membership_UserProfile]
656 GO
657 ALTER TABLE [dbo].[webpages_OAuthMembership] WITH CHECK ADD CONSTRAINT [
        FK_webpages_OAuthMembership_UserProfile] FOREIGN KEY([UserId])
658 REFERENCES [dbo].[UserProfile] ([UserId])
659 GO
660 ALTER TABLE [dbo].[webpages_OAuthMembership] CHECK CONSTRAINT [
        FK_webpages_OAuthMembership_UserProfile]
661 GO
662 ALTER TABLE [dbo].[webpages_UsersInRoles] WITH CHECK ADD CONSTRAINT [
        FK_webpages_UsersInRoles_UserProfile] FOREIGN KEY([UserId])
663 REFERENCES [dbo].[UserProfile] ([UserId])
664 GO
665 ALTER TABLE [dbo].[webpages_UsersInRoles] CHECK CONSTRAINT [
        FK_webpages_UsersInRoles_UserProfile]
666 GO
667 ALTER TABLE [dbo].[webpages_UsersInRoles] WITH CHECK ADD CONSTRAINT [
        FK_webpages_UsersInRoles_webpages_Roles] FOREIGN KEY([RoleId])
668 REFERENCES [dbo].[webpages_Roles] ([RoleId])
669 GO
670 ALTER TABLE [dbo].[webpages_UsersInRoles] CHECK CONSTRAINT [
        FK_webpages_UsersInRoles_webpages_Roles]
671 GO
672 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane1', @value=N'[OE232FF0-B466-11cf-A24F
        -00AA00A3EFFF, 1.00]
673 Begin DesignProperties =
674     Begin PaneConfigurations =
675         Begin PaneConfiguration = 0
676             NumPanes = 4
677             Configuration = "(H (1[40] 4[20] 2[20] 3) )"
678         End
679         Begin PaneConfiguration = 1
680             NumPanes = 3
681             Configuration = "(H (1 [50] 4 [25] 3))"
682         End
683         Begin PaneConfiguration = 2
684             NumPanes = 3
685             Configuration = "(H (1 [50] 2 [25] 3))"
686         End
687         Begin PaneConfiguration = 3
688             NumPanes = 3
689             Configuration = "(H (4 [30] 2 [40] 3))"
690         End
691         Begin PaneConfiguration = 4
692             NumPanes = 2
693             Configuration = "(H (1 [56] 3))"
694     End

```



*A. Server*

---

```
695 Begin PaneConfiguration = 5
696     NumPanes = 2
697     Configuration = "(H (2 [66] 3))"
698 End
699 Begin PaneConfiguration = 6
700     NumPanes = 2
701     Configuration = "(H (4 [50] 3))"
702 End
703 Begin PaneConfiguration = 7
704     NumPanes = 1
705     Configuration = "(V (3))"
706 End
707 Begin PaneConfiguration = 8
708     NumPanes = 3
709     Configuration = "(H (1[56] 4[18] 2) )"
710 End
711 Begin PaneConfiguration = 9
712     NumPanes = 2
713     Configuration = "(H (1 [75] 4))"
714 End
715 Begin PaneConfiguration = 10
716     NumPanes = 2
717     Configuration = "(H (1[66] 2) )"
718 End
719 Begin PaneConfiguration = 11
720     NumPanes = 2
721     Configuration = "(H (4 [60] 2))"
722 End
723 Begin PaneConfiguration = 12
724     NumPanes = 1
725     Configuration = "(H (1) )"
726 End
727 Begin PaneConfiguration = 13
728     NumPanes = 1
729     Configuration = "(V (4))"
730 End
731 Begin PaneConfiguration = 14
732     NumPanes = 1
733     Configuration = "(V (2))"
734 End
735 ActivePaneConfig = 0
736 End
737 Begin DiagramPane =
738     Begin Origin =
739         Top = 0
740         Left = 0
741     End
742     Begin Tables =
743         Begin Table = "Floors"
744             Begin Extent =
745                 Top = 6
746                 Left = 38
747                 Bottom = 135
748                 Right = 208
749             End
750             DisplayFlags = 280
751             TopColumn = 0
```



*A. Server*

---

```
752     End
753     Begin Table = "Maps"
754         Begin Extent =
755             Top = 6
756             Left = 246
757             Bottom = 118
758             Right = 416
759         End
760         DisplayFlags = 280
761         TopColumn = 0
762     End
763     Begin Table = "Nodes"
764         Begin Extent =
765             Top = 6
766             Left = 454
767             Bottom = 135
768             Right = 624
769         End
770         DisplayFlags = 280
771         TopColumn = 0
772     End
773     Begin Table = "NodeInformation"
774         Begin Extent =
775             Top = 120
776             Left = 246
777             Bottom = 286
778             Right = 416
779         End
780         DisplayFlags = 280
781         TopColumn = 0
782     End
783 End
784 End
785 Begin SQLPane =
786 End
787 Begin DataPane =
788     Begin ParameterDefaults = ""
789     End
790 End
791 Begin CriteriaPane =
792     Begin ColumnWidths = 11
793         Column = 1860
794         Alias = 1425
795         Table = 1170
796         Output = 720
797         Append = 1400
798         NewValue = 1170
799         SortType = 1350
800         SortOrder = 1410
801         GroupBy = 1350
802         Filter = 1350
803         Or = 1350
804         Or = 1350
805         Or = 1350
806     End
807 End
808 End
```



```

809 ' , @level0type=N'SHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'
      NodeInformationForMap'
810 GO
811 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPaneCount', @value=1 , @level0type=N'SHEMA
      ',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'NodeInformationForMap'
812 GO
813 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F
      -00AA00A3EFFF, 1.00]
814 Begin DesignProperties =
815     Begin PaneConfigurations =
816         Begin PaneConfiguration = 0
817             NumPanes = 4
818             Configuration = "(H (1[40] 4[20] 2[20] 3) )"
819         End
820         Begin PaneConfiguration = 1
821             NumPanes = 3
822             Configuration = "(H (1 [50] 4 [25] 3))"
823         End
824         Begin PaneConfiguration = 2
825             NumPanes = 3
826             Configuration = "(H (1 [50] 2 [25] 3))"
827         End
828         Begin PaneConfiguration = 3
829             NumPanes = 3
830             Configuration = "(H (4 [30] 2 [40] 3))"
831         End
832         Begin PaneConfiguration = 4
833             NumPanes = 2
834             Configuration = "(H (1 [56] 3))"
835         End
836         Begin PaneConfiguration = 5
837             NumPanes = 2
838             Configuration = "(H (2 [66] 3))"
839         End
840         Begin PaneConfiguration = 6
841             NumPanes = 2
842             Configuration = "(H (4 [50] 3))"
843         End
844         Begin PaneConfiguration = 7
845             NumPanes = 1
846             Configuration = "(V (3))"
847         End
848         Begin PaneConfiguration = 8
849             NumPanes = 3
850             Configuration = "(H (1[56] 4[18] 2) )"
851         End
852         Begin PaneConfiguration = 9
853             NumPanes = 2
854             Configuration = "(H (1 [75] 4))"
855         End
856         Begin PaneConfiguration = 10
857             NumPanes = 2
858             Configuration = "(H (1[66] 2) )"
859         End
860         Begin PaneConfiguration = 11
861             NumPanes = 2
862             Configuration = "(H (4 [60] 2))"

```





*A. Server*

---

```
863     End
864     Begin PaneConfiguration = 12
865         NumPanes = 1
866         Configuration = "(H (1) )"
867     End
868     Begin PaneConfiguration = 13
869         NumPanes = 1
870         Configuration = "(V (4))"
871     End
872     Begin PaneConfiguration = 14
873         NumPanes = 1
874         Configuration = "(V (2))"
875     End
876     ActivePaneConfig = 0
877 End
878 Begin DiagramPane =
879     Begin Origin =
880         Top = -192
881         Left = 0
882     End
883     Begin Tables =
884         Begin Table = "PoIs"
885             Begin Extent =
886                 Top = 92
887                 Left = 630
888                 Bottom = 204
889                 Right = 800
890             End
891             DisplayFlags = 280
892             TopColumn = 0
893         End
894         Begin Table = "PoiTypes"
895             Begin Extent =
896                 Top = 86
897                 Left = 856
898                 Bottom = 181
899                 Right = 1026
900             End
901             DisplayFlags = 280
902             TopColumn = 0
903         End
904         Begin Table = "NodeInformation"
905             Begin Extent =
906                 Top = 179
907                 Left = 463
908                 Bottom = 308
909                 Right = 633
910             End
911             DisplayFlags = 280
912             TopColumn = 0
913         End
914         Begin Table = "Floors"
915             Begin Extent =
916                 Top = 200
917                 Left = 185
918                 Bottom = 329
919                 Right = 355
```



A. Server

```

920         End
921         DisplayFlags = 280
922         TopColumn = 0
923     End
924     Begin Table = "Nodes"
925         Begin Extent =
926             Top = 27
927             Left = 299
928             Bottom = 156
929             Right = 469
930         End
931         DisplayFlags = 280
932         TopColumn = 0
933     End
934     Begin Table = "Maps"
935         Begin Extent =
936             Top = 27
937             Left = 34
938             Bottom = 139
939             Right = 204
940         End
941         DisplayFlags = 280
942         TopColumn = 0
943     End
944 End
945 End
946 Begin SQLPane =
947 End
948 Begin DataPane =
949     Begin ParameterDefaults = ""
950     End
951 End
952 Begin CriteriaPane =
953     Begin ColumnWidths = 11
954         Column = 1440
955         ' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'
            PoisForMap'
956 GO
957 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane2', @value=N'Alias = 900
958     Table = 1170
959     Output = 720
960     Append = 1400
961     NewValue = 1170
962     SortType = 1350
963     SortOrder = 1410
964     GroupBy = 1350
965     Filter = 1350
966     Or = 1350
967     Or = 1350
968     Or = 1350
969     End
970 End
971 End
972 ' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'PoisForMap'
973 GO
974 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPaneCount', @value=2 , @level0type=N'SCHEMA
    ',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'PoisForMap'

```



```
975 GO
976 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F
    -00AA00A3EFFF, 1.00]
977 Begin DesignProperties =
978     Begin PaneConfigurations =
979         Begin PaneConfiguration = 0
980             NumPanes = 4
981             Configuration = "(H (1[40] 4[20] 2[20] 3) )"
982         End
983         Begin PaneConfiguration = 1
984             NumPanes = 3
985             Configuration = "(H (1 [50] 4 [25] 3))"
986         End
987         Begin PaneConfiguration = 2
988             NumPanes = 3
989             Configuration = "(H (1 [50] 2 [25] 3))"
990         End
991         Begin PaneConfiguration = 3
992             NumPanes = 3
993             Configuration = "(H (4 [30] 2 [40] 3))"
994         End
995         Begin PaneConfiguration = 4
996             NumPanes = 2
997             Configuration = "(H (1 [56] 3))"
998         End
999         Begin PaneConfiguration = 5
1000             NumPanes = 2
1001             Configuration = "(H (2 [66] 3))"
1002         End
1003         Begin PaneConfiguration = 6
1004             NumPanes = 2
1005             Configuration = "(H (4 [50] 3))"
1006         End
1007         Begin PaneConfiguration = 7
1008             NumPanes = 1
1009             Configuration = "(V (3))"
1010         End
1011         Begin PaneConfiguration = 8
1012             NumPanes = 3
1013             Configuration = "(H (1[56] 4[18] 2) )"
1014         End
1015         Begin PaneConfiguration = 9
1016             NumPanes = 2
1017             Configuration = "(H (1 [75] 4))"
1018         End
1019         Begin PaneConfiguration = 10
1020             NumPanes = 2
1021             Configuration = "(H (1[66] 2) )"
1022         End
1023         Begin PaneConfiguration = 11
1024             NumPanes = 2
1025             Configuration = "(H (4 [60] 2))"
1026         End
1027         Begin PaneConfiguration = 12
1028             NumPanes = 1
1029             Configuration = "(H (1) )"
1030         End
```



*A. Server*

---

```
1031 Begin PaneConfiguration = 13
1032     NumPanes = 1
1033     Configuration = "(V (4))"
1034 End
1035 Begin PaneConfiguration = 14
1036     NumPanes = 1
1037     Configuration = "(V (2))"
1038 End
1039 ActivePaneConfig = 0
1040 End
1041 Begin DiagramPane =
1042     Begin Origin =
1043         Top = 0
1044         Left = 0
1045     End
1046     Begin Tables =
1047         Begin Table = "m"
1048             Begin Extent =
1049                 Top = 6
1050                 Left = 246
1051                 Bottom = 118
1052                 Right = 416
1053             End
1054             DisplayFlags = 280
1055             TopColumn = 0
1056         End
1057         Begin Table = "f"
1058             Begin Extent =
1059                 Top = 6
1060                 Left = 38
1061                 Bottom = 135
1062                 Right = 208
1063             End
1064             DisplayFlags = 280
1065             TopColumn = 1
1066         End
1067         Begin Table = "n"
1068             Begin Extent =
1069                 Top = 138
1070                 Left = 38
1071                 Bottom = 267
1072                 Right = 208
1073             End
1074             DisplayFlags = 280
1075             TopColumn = 0
1076         End
1077         Begin Table = "ni"
1078             Begin Extent =
1079                 Top = 120
1080                 Left = 246
1081                 Bottom = 305
1082                 Right = 416
1083             End
1084             DisplayFlags = 280
1085             TopColumn = 0
1086         End
1087     End
```



### A. Server

```

1088 End
1089 Begin SQLPane =
1090 End
1091 Begin DataPane =
1092     Begin ParameterDefaults = ""
1093     End
1094 End
1095 Begin CriteriaPane =
1096     Begin ColumnWidths = 11
1097         Column = 1440
1098         Alias = 900
1099         Table = 1170
1100         Output = 720
1101         Append = 1400
1102         NewValue = 1170
1103         SortType = 1350
1104         SortOrder = 1410
1105         GroupBy = 1350
1106         Filter = 1350
1107         Or = 1350
1108         Or = 1350
1109         Or = 1350
1110     End
1111 End
1112 End
1113 ', @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'RoomsForMap'
1114 GO
1115 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA
    ',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'RoomsForMap'
1116 GO
1117 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F
    -00AA00A3EFFF, 1.00]
1118 Begin DesignProperties =
1119     Begin PaneConfigurations =
1120         Begin PaneConfiguration = 0
1121             NumPanes = 4
1122             Configuration = "(H (1[40] 4[20] 2[20] 3) )"
1123         End
1124         Begin PaneConfiguration = 1
1125             NumPanes = 3
1126             Configuration = "(H (1 [50] 4 [25] 3))"
1127         End
1128         Begin PaneConfiguration = 2
1129             NumPanes = 3
1130             Configuration = "(H (1 [50] 2 [25] 3))"
1131         End
1132         Begin PaneConfiguration = 3
1133             NumPanes = 3
1134             Configuration = "(H (4 [30] 2 [40] 3))"
1135         End
1136         Begin PaneConfiguration = 4
1137             NumPanes = 2
1138             Configuration = "(H (1 [56] 3))"
1139         End
1140         Begin PaneConfiguration = 5
1141             NumPanes = 2
1142             Configuration = "(H (2 [66] 3))"

```



*A. Server*

---

```
1143 End
1144 Begin PaneConfiguration = 6
1145     NumPanes = 2
1146     Configuration = "(H (4 [50] 3))"
1147 End
1148 Begin PaneConfiguration = 7
1149     NumPanes = 1
1150     Configuration = "(V (3))"
1151 End
1152 Begin PaneConfiguration = 8
1153     NumPanes = 3
1154     Configuration = "(H (1[56] 4[18] 2) )"
1155 End
1156 Begin PaneConfiguration = 9
1157     NumPanes = 2
1158     Configuration = "(H (1 [75] 4))"
1159 End
1160 Begin PaneConfiguration = 10
1161     NumPanes = 2
1162     Configuration = "(H (1[66] 2) )"
1163 End
1164 Begin PaneConfiguration = 11
1165     NumPanes = 2
1166     Configuration = "(H (4 [60] 2))"
1167 End
1168 Begin PaneConfiguration = 12
1169     NumPanes = 1
1170     Configuration = "(H (1) )"
1171 End
1172 Begin PaneConfiguration = 13
1173     NumPanes = 1
1174     Configuration = "(V (4))"
1175 End
1176 Begin PaneConfiguration = 14
1177     NumPanes = 1
1178     Configuration = "(V (2))"
1179 End
1180 ActivePaneConfig = 0
1181 End
1182 Begin DiagramPane =
1183     Begin Origin =
1184         Top = 0
1185         Left = 0
1186     End
1187     Begin Tables =
1188         Begin Table = "aps"
1189             Begin Extent =
1190                 Top = 6
1191                 Left = 38
1192                 Bottom = 135
1193                 Right = 267
1194             End
1195             DisplayFlags = 280
1196             TopColumn = 0
1197         End
1198         Begin Table = "f"
1199             Begin Extent =
```



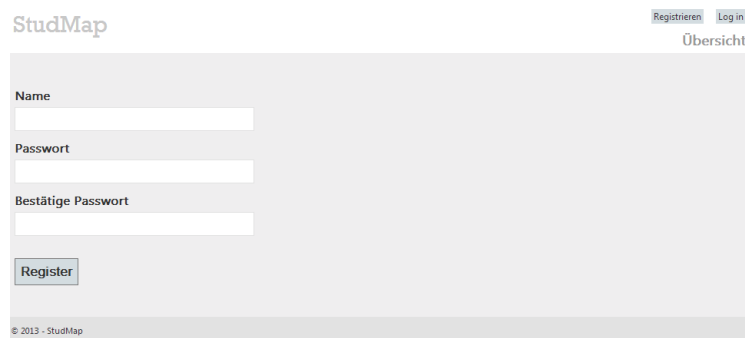
A. Server

---

```
1200         Top = 6
1201         Left = 305
1202         Bottom = 101
1203         Right = 491
1204     End
1205     DisplayFlags = 280
1206     TopColumn = 0
1207 End
1208 End
1209 End
1210 Begin SQLPane =
1211 End
1212 Begin DataPane =
1213     Begin ParameterDefaults = ""
1214     End
1215 End
1216 Begin CriteriaPane =
1217     Begin ColumnWidths = 12
1218         Column = 1440
1219         Alias = 900
1220         Table = 1176
1221         Output = 720
1222         Append = 1400
1223         NewValue = 1170
1224         SortType = 1356
1225         SortOrder = 1416
1226         GroupBy = 1350
1227         Filter = 1356
1228         Or = 1350
1229         Or = 1350
1230         Or = 1350
1231     End
1232 End
1233 End
1234 ' , @level0type=N'SCHEMA',@level0name=N'dbo' , @level1type=N'VIEW',@level1name=N'
RSSDistribution'
1235 GO
1236 EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA
',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'RSSDistribution'
1237 GO
1238 USE [master]
1239 GO
1240 ALTER DATABASE [StudMap] SET READ_WRITE
1241 GO
```

## B. Benutzerverwaltung

In einem ASP.NET MVC 4 Projekt ist bereits eine vollständige Benutzerverwaltung integriert, die wir auch in unserem Projekt benutzen wollen. Durch die integrierte Benutzerverwaltung sind Webseiten zur Registrierung und für den Login / Logout bereits fertig.



The screenshot shows the 'StudMap' Admin interface for user registration. At the top right, there are links for 'Registrieren', 'Log in', and 'Übersicht'. The main form contains three input fields: 'Name', 'Passwort', and 'Bestätige Passwort'. Below these fields is a 'Register' button. The footer of the page indicates '© 2013 - StudMap'.

Abbildung B.1.: Webseite zur Registrierung im StudMap Admin

Für die Benutzerverwaltung verwendet das ASP.NET MVC 4 Projekt folgende Datenbankstruktur:



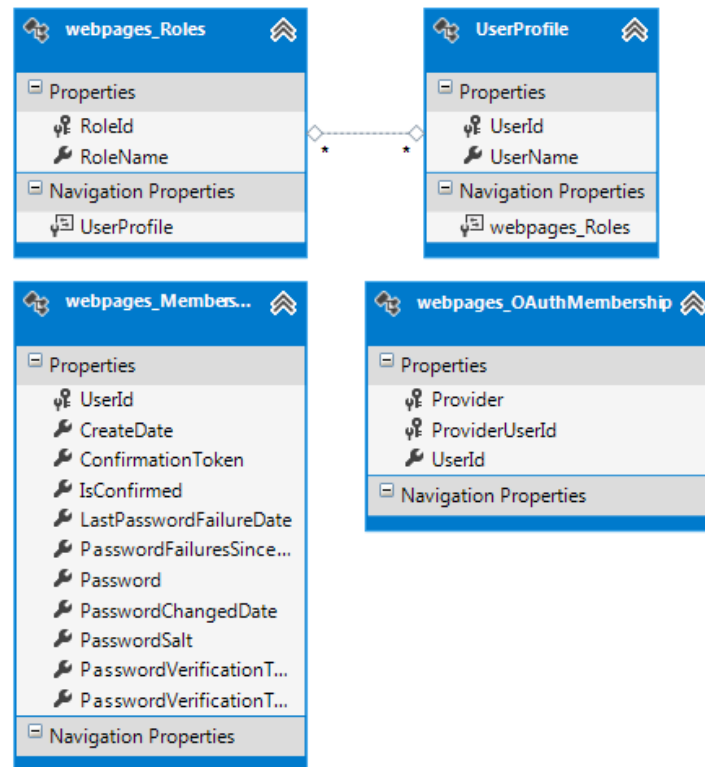
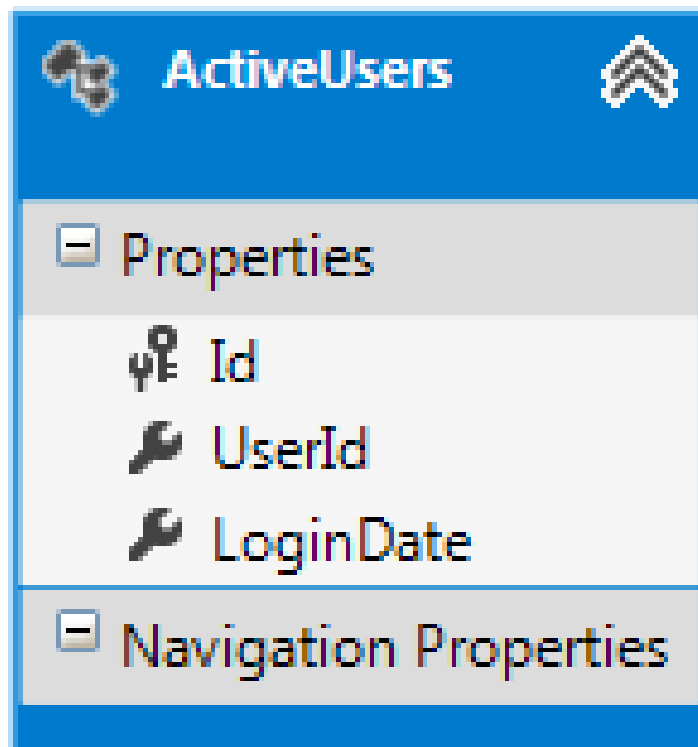


Abbildung B.2.: Datenbankstruktur der integrierten Benutzerverwaltung

Für unser Projekt sind nur die drei Tabellen `UserProfile`, `webpages_Roles` und `webpages_Membership` relevant. Wie im Domain Model bereits beschrieben unterscheiden wir zwischen den Benutzerrollen Benutzer und Administrator. Jeder Anwender kann sich in mehreren Benutzerrollen befinden. Zusätzlich sind in der Tabelle `webpages_membership` weitere Anwenderdaten wie beispielsweise das Datum der Registrierung das Passwort hinterlegt. Damit ist die Benutzerverwaltung für den Administrationsbereich vollständig.

Um die (beispielsweise über das Smartphone) am System angemeldeten Clients zu verwalten haben wir eine weitere Tabelle `ActiveUsers` hinzugefügt:



ActiveUsers	
[-] Properties	
[key icon]	Id
[wrench icon]	UserId
[wrench icon]	LoginDate
[-] Navigation Properties	

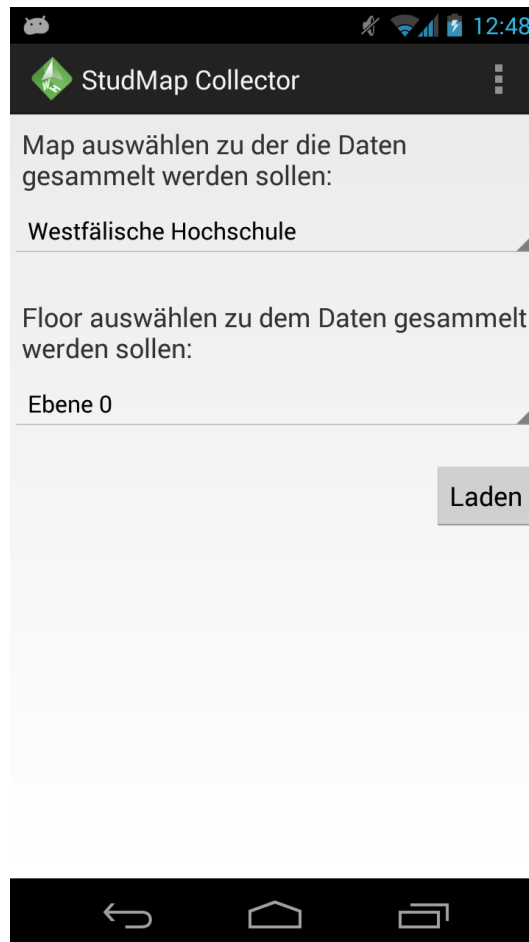
Abbildung B.3.: Tabelle ActiveUsers

Für den StudMap Admin genügt die bereits integrierte Benutzerverwaltung. Allerdings benötigen wir noch eine Schnittstelle, damit auch die Web bzw. Smartphone Clients auf die Benutzerverwaltung zugreifen können. Siehe dazu Kapitel [Verwendung der Benutzerschnittstelle](#).

## C. Collector - Bedienungsanleitung

### C.1. Startbildschirm

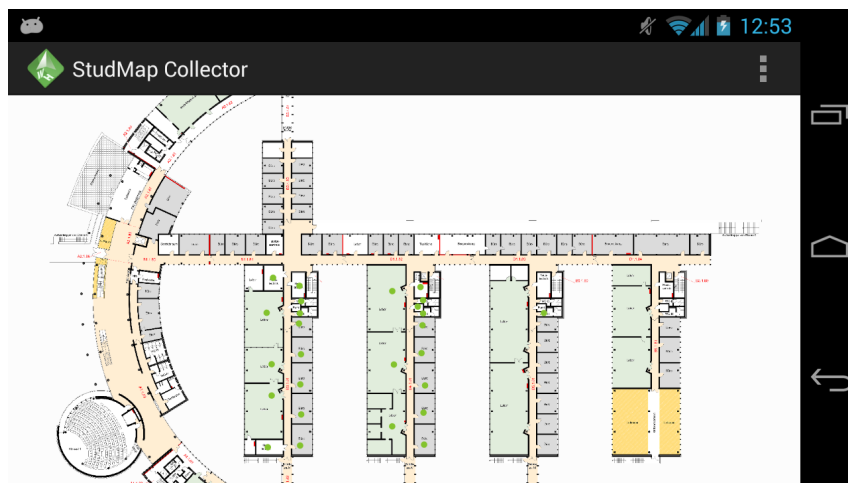
Nach Start der Anwendung befindet man sich auf dem Startbildschirm:



Hier können Karte und Stockwerk für die Datenerfassung ausgewählt werden. Über den Button Laden wird die entsprechende Auswahl geladen.

## C.2. Stockwerkansicht

Hier sieht man das ausgewählte Stockwerk mit den zur Verfügung stehenden Datenpunkten, die grün dargestellt sind.



### C.2.1. Datenpunktauswahl

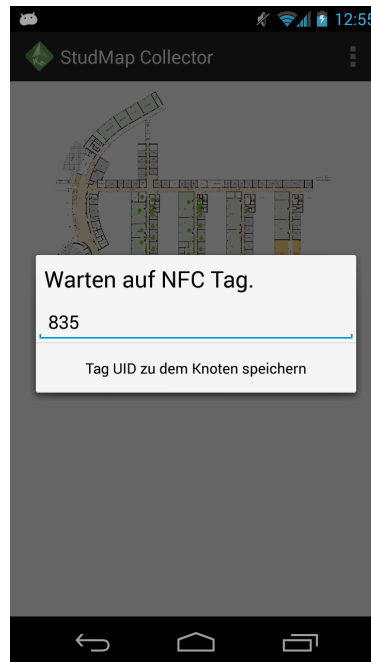
Bei der Auswahl eines Datenpunktes erscheint folgender Dialog:



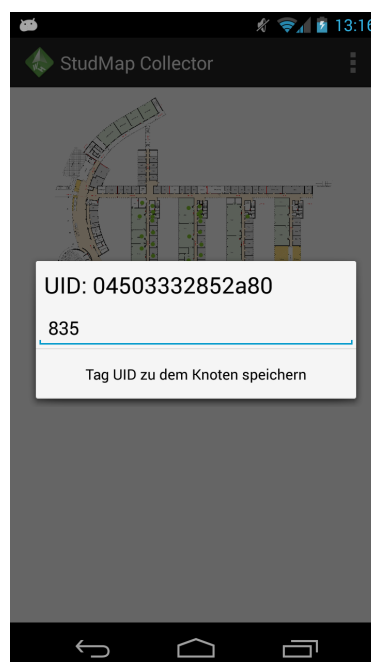
Über diesen Dialog können für den ausgewählten Datenpunkt Positionsdaten hinterlegt werden. Dazu stehen die beiden Techniken NFC und Wifi zur Verfügung.

### C.2.1.1. NFC-Tag Ansicht

Solange das Gerät keinen Kontakt zu einem NFC Tag hat, wird nur die ID des Datenpunkts ausgegeben.

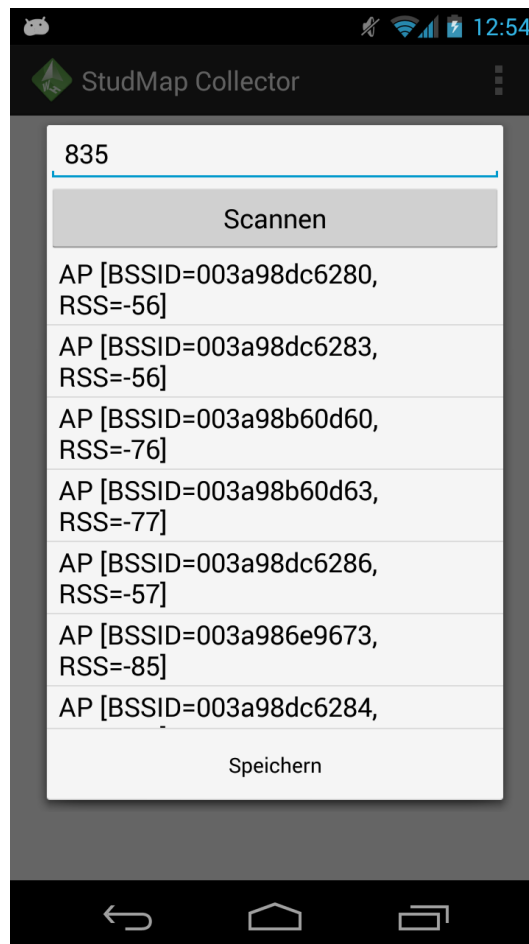


Sobald ein NFC Tag gefunden wurde, wird auch die ID des NFC-Tags im Dialog angezeigt und der Datenpunkt kann mit dem NFC-Tag verknüpft werden.



### C.2.1.2. Wifi

Über diesen Dialog kann für den ausgewählten Datenpunkt ein WLAN-Fingerprint erstellt und gespeichert werden.





## D. Webservice

Für den Zugriff auf die Daten stellen wir den Webservice `StudMap.Service` zur Verfügung. Der Webservice besteht dabei aus zwei Schnittstellen in Form von sogenannten Controller Klassen, die jeweils von der Klasse `ApiController`<sup>11</sup> abgeleitet sind:

- `MapsController`: Verwaltung von Karten- und Routeninformationen
- `UsersController`: Verwaltung von Benutzerinformationen

Bevor nun die Funktionen der jeweiligen Controller Klasse erläutert werden, folgt eine Übersicht, über die verschiedenen Rückgabe Werte und ihre Bedeutung.

### D.1. Allgemeine Objekte

Im folgenden werden die im Webservice verwendeten Objekte aufgeführt. Für jedes Objekt werden Eigenschaften und die Repräsentation der Daten im JSON-Format aufgelistet.

#### D.1.1. Edge

Repräsentiert eine Kante in einem Graphen.

Eigenschaften:

StartNodeId	ID des Start-Knotens der Kante.
EndNodeId	ID des End-Knotens der Kante.

Beispiel:

```
1 {  
2   "StartNodeId": 12,  
3   "EndNodeId": 6  
4 }
```

---

<sup>11</sup>siehe: [MSDN Dokumentation](#)

### D.1.2. Node

Repräsentiert einen Knoten in einem Graphen.

Eigenschaften:

Id	ID des Knotens.
X	X-Koordinate auf dem Bild des Stockwerks. Wertebereich: 0.0 - 1.0. 0.0 bedeutet linker Bildrand. 1.0 bedeutet rechter Bildrand.
Y	Y-Koordinate auf dem Bild des Stockwerks. Wertebereich: 0.0 - 1.0. 0.0 bedeutet oberer Bildrand. 1.0 bedeutet unterer Bildrand.
FloorId	ID des Stockwerks auf dem sich der Knoten befindet.
HasInformation	true, wenn es Raum- oder PoI-Daten zu dem Knoten gibt. Ansonsten false.

Beispiel:

```
1 {  
2   "Id": 12,  
3   "X": 0.45,  
4   "Y": 0.76,  
5   "FloorId": 2,  
6   "HasInformation": true  
7 }
```

### D.1.3. Graph

Repräsentiert für ein Stockwerk den entsprechenden Teilgraphen. Eigenschaften:

FloorId	ID des Stockwerks, das der entsprechende Graph repräsentiert.
Edges	Eine Liste von Kanten (s. <a href="#">Edge</a> ).
Nodes	Eine Liste von Knoten (s. <a href="#">Node</a> ).

Beispiel:

```
1 {  
2   "FloorId": 12,  
3   "Edges": { {...}, {...}, {...} },  
4   "Nodes": { {...}, {...}, {...} }  
5 }
```

### D.1.4. FloorPlanData

Repräsentiert Daten die auf einem Bild dargestellt werden können.

Eigenschaften:

Graph	
-------	--



Beispiel:

```
1 {  
2   "Graph": {...}  
3 }
```

### D.1.5. Room

Repräsentiert Daten die für einen Raum relevant sind.

Eigenschaften:

NodeId	ID des Knotens an dem sich der Raum befindet.
DisplayName	Der Anzeigename für den Raum (z.B. Aquarium)
RoomName	Der eigentliche Raumname (z.B. A 2.0.11).
FloorId	Id des Floors, auf welchen sich der Room befindet.

Beispiel:

```
1 {  
2   "NodeId": 12,  
3   "DisplayName": "Aquarium",  
4   "RoomName": "A2.0.11",  
5   "FloorId": 1  
6 }
```

### D.1.6. PoiType

Repräsentiert einen **Poi** Typen, wie beispielsweise Mensa oder Bibliothek.

Eigenschaften:

Id	ID des Poi Typs.
Name	Der Name des Poi Tpes (z.B. Mensa).

Beispiel:

```
1 {  
2   "Id": 12,  
3   "Name": "Mensa"  
4 }
```

### D.1.7. Poi

Repräsentiert einen Point Of Interest, wie beispielsweise eine Mensa oder eine Bibliothek.

#### D. Webservice

---

Eigenschaften:

Type	Typ des PoIs (s. <a href="#">PoiType</a> ).
Description	Beschreibung des PoIs.

Beispiel:

```
1 {  
2   "Type": 1,  
3   "Description": "In der Mensa kann man essen."  
4 }
```

#### D.1.8. RoomAndPoI

Enthält Informationen zu einem Raum und dem zugeordneten PoI.

Eigenschaften:

Room	Rauminformationen (s. <a href="#">Room</a> ).
PoI	PoI-Informationen (s. <a href="#">PoI</a> ).

Beispiel:

```
1 {  
2   "Room": {...},  
3   "PoI": {...}  
4 }
```

#### D.1.9. NodeInformation

Repräsentiert die für einen Knoten relevanten Daten.

Eigenschaften:

DisplayName	Anzeigename für den Knoten (z.B. Dr. Schulten, Martin).
RoomName	Raumname für den Knoten (z.B. B2.0.03).
Node	NodeInformation repräsentiert diesen Knoten.
PoI	PoI Informationen zu diesem Knoten.
QRCode	Dem Knoten zugeordnetem QR Code.
NFCTag	Dem Knoten zugeordnetem NFC Tag.

Beispiel:

```
1 {  
2   "DisplayName": "Dr. Schulten, Martin",  
3   "RoomName": "B2.0.03",  
4   "Node": {...},  
5   "PoI": {...},  
6   "QRCode": "...",  
7   "NFCTag": "..."  
8 }
```

### D.1.10. QRCode

Repräsentiert die QRCode-Daten für den Knoten.

Eigenschaften:

General	Grundsätzliche Informationen zu einem Room.
StudMap	Allgemeine Informationen zum Projekt sind in StudMap hinterlegt.

Beispiel:

```
1 {  
2   "General": {...},  
3   "StudMap": {...}  
4 }
```

### D.1.11. FullNodeInformation

Repräsentiert die für einen Knoten relevanten Daten.

Eigenschaften:

Map	Informationen zur gesamten Karte.
Floor	Repräsentiert die für dieses Stockwerk relevanten Daten.
Info	NodeInformation zum aktuellen Knoten.

Beispiel:

```
1 {  
2   "Map": {...},  
3   "Floor": {...},  
4   "Info": {...}  
5 }
```

### D.1.12. Floor

Repräsentiert die für ein Stockwerk relevanten Daten.

Eigenschaften:

Id	ID des Stockwerks.
MapId	ID der Karte zu dem das Stockwerk gehört.
Name	Name des Stockwerks.
ImageUrl	Der Dateipfad auf dem Server zum Bild des Stockwerks.
CreationTime	Zeitstempel, an dem das Stockwerk erstellt wurde.

Beispiel:

```
1 {  
2   "Id": 1011,  
3   "MapId": 3,  
4   "Name": "Ebene 0",  
5   "ImageUrl": "Images/Floors/RN_Ebene_0.png",  
6   "CreationTime": "2013-11-18 14:36:24.607"  
7 }
```

### D.1.13. Map

Repräsentiert die für eine Karte relevanten Daten.

Eigenschaften:

Id	ID der Karte.
Name	Name der Karte.

Beispiel:

```
1 {  
2   "Id": 3,  
3   "Name": "Westfälische Hochschule",  
4 }
```

### D.1.14. User

Repräsentiert die für einen Benutzer relevanten Daten.

Eigenschaften:

Name	Name des Benutzers.
------	---------------------

Beispiel:

```
1 {  
2   "Name": "Daniel",  
3 }
```

### D.1.15. SaveGraphRequest

Repräsentiert die Änderungen an dem Teilgraph für ein Stockwerk.

Eigenschaften:

FloorId	ID des Stockwerks.
NewGraph	Der hinzugefügte Teilgraph (s. <a href="#">Graph</a> ).
DeletedGraph	Der gelöschte Teilgraph (s. <a href="#">Graph</a> ).

Beispiel:



```
1 {  
2   "FloorId": 2,  
3   "NewGraph": {...},  
4   "DeletedGraph": {...}  
5 }
```

## D.2. Rückgabe Objekte

### D.2.1. BaseResponse

Allgemeine Rückgabe vom Service, die einen Status und ggf. einen Fehler enthält.  
Die Daten werden im JSON Format zurück gegeben.

**Beispiel:**

```
1 {  
2   "Status":1,  
3   "ErrorCode":0  
4 }
```

#### D.2.1.1. ResponseStatus

Wir unterscheiden zwischen:

- **None** = 0: Defaultwert
- **Ok** = 1: Funktion erfolgreich ausgeführt
- **Error** = 2 Fehler bei Funktionsausführung

#### D.2.1.2. ResponseError

Ist beim **ResponseStatus Error** gesetzt. Es werden folgende Fehlerszenarien unterschieden:

**Allgemein:**

- **001 - DatabaseError:**  
Fehler bei der Ausführung einer Datenbankabfrage.
- **002 - Unknown:**  
Unbekannter Fehler.

Marcus:  
Alle ge-  
nauen  
Fehler-  
typen be-  
schreiben.

**Registrierung:**

- 101 - `UserNameDuplicate`:  
Der Benutzername ist bereits vergeben.
- 102 - `UserNameInvalid`:  
Der Benutzername ist ungültig.
- 103 - `PasswordInvalid`:  
Das Passwort ist ungültig.

**Anmeldung:**

- 110 - `LoginInvalid`:  
Die Logindaten (Name oder Passwort) sind ungültig.

**Maps:**

- 201 - `MapIdDoesNotExist`:  
Zur angeforderten MapId existiert keine Map.
- 202 - `FloorIdDoesNotExist`:  
Zur angeforderten FloorId existiert kein Floor.
- 203 - `NodeIdDoesNotExist`:  
Zur angeforderten NodeId existiert kein Node.

**Navigation:**

- 301 - `NoRouteFound`:  
Es konnte keine Route gefunden werden.
- 302 - `StartNodeNotFound`:  
Der angegebene Startknoten existiert nicht.
- 303 - `EndNodeNotFound`:  
Der angegebene Endknoten existiert nicht.

**Information:**

- 401 - `PoiTypeIdDoesNotExist`:  
Zur angegebenen PoiTypeId existiert kein PoiType.
- 402 - `NFCTagDoesNotExist`:  
Das angegebene NFC-Tag wurde nicht gefunden.
- 403 - `QRCodeDoesNotExist`:  
Der angegebene QR-Code wurde nicht gefunden.
- 404 - `PoiDoesNotExist`:  
Zur angegebenen PoiId existiert kein PoI.



- 405 - QRCodeIsNullOrEmpty:  
Es wurde kein QR-Code angegeben.
- 406 - NFCTagIsNullOrEmpty:  
Es wurde kein NFC-Tag angegeben.
- 407 - NFCTagAlreadyAssigned:  
Das NFC-Tag ist bereits einem anderen Knoten zugeordnet.

### D.2.2. ObjectResponse

Eine generische Klasse die `BaseResponse` um ein Feld `Object` erweitert, indem die Nutzdaten gespeichert werden.

Beispiel:

```
1 {  
2   "Status": 1,  
3   "ErrorCode": 0,  
4   "Object": {...}  
5 }
```

### D.2.3. ListResponse

Eine generische Klasse die `BaseResponse` um eine Liste `List` erweitert, indem Nutzdaten in Form einer Collection gespeichert werden.

Beispiel:

```
1 {  
2   "Status": 1,  
3   "ErrorCode": 0,  
4   "List": [...]  
5 }
```

## D.3. MapsController

### D.3.1. CreateMap

Erstellt eine neue Karte mit dem vorgegebenen Namen.

POST [/api/Maps/CreateMap?mapName=WHS](#)

Parameter:

mapName	Sprechender Name der Karte.
---------	-----------------------------



Rückgabewert: **ObjectResponse**, **Map**

### D.3.2. DeleteMap

Löscht eine Karte mit der angegebenen ID.

POST [/api/Maps/DeleteMap?mapId=2](#)

Parameter:

mapId	ID der Map, die gelöscht werden soll.
-------	---------------------------------------

Rückgabewert: **BaseResponse**

### D.3.3. GetMaps

Liefert eine Liste aller Karten zurück.

GET [/api/Maps/GetMaps](#)

Rückgabewert: **ListResponse**, **Map**

### D.3.4. CreateFloor

Erstellt einen Floor zu einer Map mit einem sprechenden Namen.

POST [/api/Maps/CreateFloor?mapId=2&name=Erdgeschoss](#)

Parameter:

mapId	ID der Map, zu der ein Floor angelegt wird.
name	Sprechender Name des Floors.

Rückgabewert: **ObjectResponse**, **Floor**

### D.3.5. DeleteFloor

Löscht einen Floor mit der angegebenen ID.

POST [/api/Maps/DeleteFloor?floorId=3](#)

Parameter:

floorId	ID des Floors, der gelöscht werden soll.
---------	--

Rückgabewert: **BaseResponse**



### D.3.6. GetFloorsForMap

Liefert alle Stockwerke einer Karte.

GET </api/Maps/GetFloorsForMap?mapId=2>

Parameter:

mapId	ID des Karte, von der die Stockwerke abgefragt werden sollen.
-------	---

Rückgabewert: [ListResponse](#), [Floor](#)

### D.3.7. GetFloor

Liefert Informationen zu einem Stockwerk.

GET </api/Maps/GetFloor?floorId=3>

Parameter:

floorId	ID des Stockwerks.
---------	--------------------

Rückgabewert: [ObjectResponse](#), [Floor](#)

### D.3.8. GetFloorplanImage

Liefert die URL des Bilds zu einem Stockwerk.

GET </api/Maps/GetFloorplanImage?floorId=3>

Parameter:

floorId	ID des Stockwerks.
---------	--------------------

Rückgabewert: String

### D.3.9. SaveGraphForFloor

Speichert den Graphen zu einem Stockwerk ab.

POST </api/Maps/SaveGraphForFloor>

POST Body: [SaveGraphRequest](#)

Rückgabewert: [ObjectResponse](#), [Graph](#)



### D.3.10. DeleteGraphForFloor

Löscht den Teilgraphen auf einem Stockwerk. Die Teilgraphen auf anderen Stockwerken werden nicht verändert. Kanten die Stockwerke mit diesem Stockwerk verbinden, werden ebenfalls entfernt.

POST </api/Maps/DeleteGraphForFloor?floorId=2>

Parameter:

floorId	ID des Stockwerks, dessen Teilgraph gelöscht werden soll.
---------	---

Rückgabewert: [BaseResponse](#)

### D.3.11. GetGraphForFloor

Liefert den Teilgraphen für ein Stockwerk.

GET </api/Maps/GetGraphForFloor?floorId=2>

Parameter:

floorId	ID des Stockwerks.
---------	--------------------

Rückgabewert: [ObjectResponse](#), [Graph](#)

### D.3.12. GetFloorPlanData

Liefert die verschiedenen Schichten auf einem Stockwerk.

GET </api/Maps/GetFloorPlanData?floorId=2>

Parameter:

floorId	ID des Stockwerks.
---------	--------------------

Rückgabewert: [ObjectResponse](#), [FloorPlanData](#)

### D.3.13. GetConnectedNodes

Liefert die Knoten zurück, die mit dem übergebenen Knoten verbunden sind.

GET </api/Maps/GetConnectedNodes?nodeId=1011>

Parameter:

nodeId	ID des Knotens.
--------	-----------------

Rückgabewert: [ListResponse](#), [Node](#)

#### D.3.14. GetRouteBetween

Liefert die Route zwischen zwei Knoten, wenn diese existiert.

GET </api/Maps/GetRouteBetween?mapId=2&startNodeId=12&endNodeId=46>

Parameter:

mapId	ID der Karte, auf der die Route bestimmt werden soll.
startNodeId	ID des Startknotens.
endNodeId	ID des Zielknotens.

Rückgabewert: [ListResponse](#), [Node](#)

#### D.3.15. GetNodeInformationForNode

Liefert weitere Informationen zu einem Knoten. Diese umfassen Raumnummern, zugeordnete NFC- und QR-Tags, usw.

GET </api/Maps/GetNodeInformationForNode?nodeId=12>

Parameter:

nodeId	ID des Knotens.
--------	-----------------

Rückgabewert: [ObjectResponse](#), [Graph](#)

#### D.3.16. GetFullNodeInformationForNode

Liefert alle Informationen zu einem Knoten. Diese umfassen die Objekte [Map](#), [Floor](#) und [NodeInformation](#).

GET </api/Maps/GetFullNodeInformationForNode?nodeId=1011>

Parameter:

nodeId	ID des Knotens.
--------	-----------------

Rückgabewert: [ObjectResponse](#), [FullNodeInformation](#)

#### D.3.17. GetNodeInformaion

Liefert Informationen zu allen Knoten auf einem Stockwerk.

GET </api/Maps/GetNodeInformaion?mapId=2&floorId=12>

Parameter:

mapId	ID der Karte.
floorId	ID des Stockwerks.

Rückgabewert: [ListResponse](#), [NodeInformation](#)

### D.3.18. SaveNodeInformation

Speichert zusätzliche Informationen zu einem Knoten ab.

POST [/api/Maps/SaveNodeInformation](#)

POST Body: [NodeInformation](#)

Rückgabewert: [ObjectResponse](#), [NodeInformation](#)

### D.3.19. GetPoiTypes

Liefert eine Liste aller Typen von PoIs zurück.

GET [/api/Maps/GetPoiTypes](#)

Rückgabewert: [ListResponse](#), [PoiType](#)

### D.3.20. GetPolsForMap

Liefert eine Liste aller PoIs auf einer Karte zurück.

GET [/api/Maps/GetPoIsForMap?mapId=2](#)

Parameter:

mapId	ID der Karte.
-------	---------------

Rückgabewert: [ListResponse](#), [RoomAndPoi](#)

### D.3.21. GetRoomsForMap

Liefert eine Liste aller Räume auf einer Karte zurück.

GET [/api/Maps/GetRoomsForMap?mapId=2](#)

Parameter:

mapId	ID der Karte.
-------	---------------

Rückgabewert: [ListResponse](#), [Room](#)



### D.3.22. GetNodeForNFC

Sucht auf einer Karte nach einem Knoten mit einem bestimmten NFC-Tag.

GET </api/Maps/GetNodeForNFC?mapId=2&nfcTag=46A6CG739ED9>

Parameter:

mapId	ID der Karte.
nfcTag	NFC-Tag, nach dem gesucht werden soll.

Rückgabewert: [ObjectResponse](#), [Node](#)

### D.3.23. GetNodeForQRCode

Sucht auf einer Karte nach einem Knoten mit einem bestimmten QR-Code.

GET </api/Maps/GetNodeForQRCode?mapId=2&qrCode=46A6CG739ED9>

Parameter:

mapId	ID der Karte.
qrCode	QR-Code, nach dem gesucht werden soll.

Rückgabewert: [ObjectResponse](#), [Node](#)

### D.3.24. SaveNFCForNode

Speichert ein NFC-Tag zu einem bestimmten Knoten.

GET </api/Maps/SaveNFCForNode?nodeId=1011&nfcTag=46A6CG739ED9>

Parameter:

nodeId	ID des Knotens.
nfcTag	NFC-Tag, der abgespeichert werden soll.

Rückgabewert: [BaseResponse](#)

### D.3.25. SaveQRCodeForNode

Speichert einen QR-Code zu einem bestimmten Knoten.

GET </api/Maps/SaveQRCodeForNode?nodeId=1011&qrCode=46A6CG739ED9>

Parameter:

nodeId	ID des Knotens.
qrCode	QR-Code, der abgespeichert werden soll.

Rückgabewert: [BaseResponse](#)



## D.4. UsersController

Der `UserController` stellt klassische Funktionen zur Benutzerverwaltung zur Verfügung.

### D.4.1. Register

Registriert einen neuen Anwender in der Benutzerrolle Benutzer.

POST </api/Users/Register?userName=test&password=geheim>

Parameter:

userName	Der Benutzername.
password	Passwort im Klartext.

Rückgabewert: `BaseResponse`

### D.4.2. Login

Meldet einen bereits registrierten Anwender am System an.

POST </api/Users/Login?userName=test&password=geheim>

Parameter:

userName	Der Benutzername.
password	Passwort im Klartext.

Rückgabewert: `BaseResponse`

### D.4.3. Logout

Meldet einen angemeldeten Anwender vom System ab.

GET </api/Users/Logout?userName=test>

Parameter:

userName	Der Benutzername.
----------	-------------------

Rückgabewert: `BaseResponse`



#### **D.4.4. GetActiveUsers**

Ermittelt eine Liste der aktuell am System angemeldeten Anwender.

GET `/api/Users/GetActiveUsers`

Rückgabewert: `ListResponse`, `User`

### **D.5. Verwendung der Benutzerschnittstelle**

#### **D.5.1. Registrierung**

Bevor sich ein Benutzer am StudMap System anmelden kann, muss er sich zunächst über die Funktion `Register` registrieren.

#### **D.5.2. Aktive und inaktive Benutzer**

Im StudMap System wird zwischen aktiven und inaktiven Benutzern unterschieden. Nachdem sich ein Benutzer am System registriert hat gilt dieser als inaktiv. Über die Funktion `Login` kann er sich am System anmelden und gilt somit als aktiv.

Damit der angemeldete Benutzer auch aktiv bleibt, sollte sich dieser in einem Zeitintervall von fünf Minuten über die Methode `Login` am System aktiv melden. Nach einer Inaktivität von 15 Minuten wird der Benutzer automatisch inaktiv.

Über die Funktion `Logout` kann sich ein Benutzer wieder vom System abmelden und wird somit inaktiv.

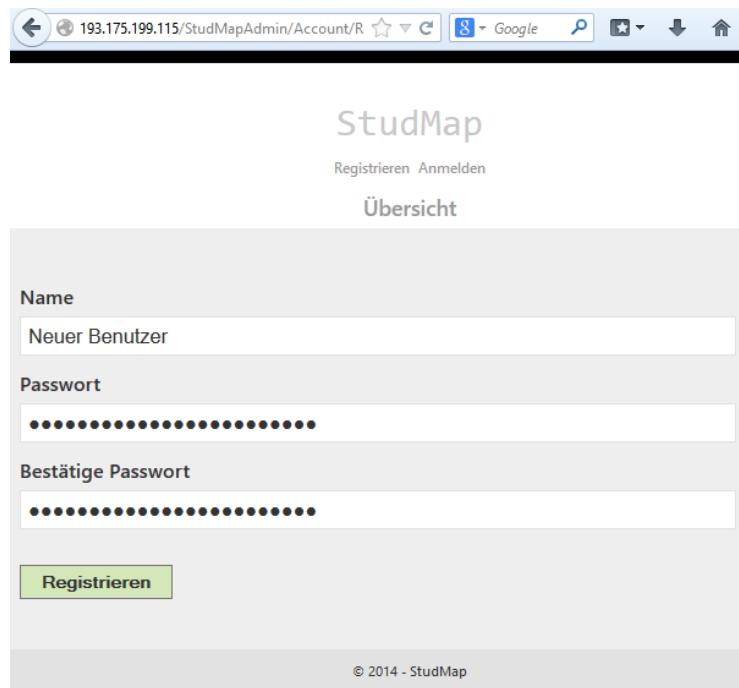
#### **D.5.3. Aktive Benutzer abfragen**

Die aktiven Benutzer können über die Funktion `GetActiveUsers` abgefragt werden. Damit die Anzeige der aktiven Benutzer im Client möglichst aktuell ist, sollte diese Abfrage ebenfalls in regelmäßigen Zeitabständen erfolgen.

## E. Admin-Bedienungsanleitung

### E.1. Registrieren

1. Aufruf der Webseite: <http://193.175.199.115/StudMapAdmin/Account/Register>.



The screenshot shows a web browser window with the address bar displaying [193.175.199.115/StudMapAdmin/Account/R](http://193.175.199.115/StudMapAdmin/Account/R). The page title is "StudMap" with links for "Registrieren" and "Anmelden". Below the title is a section labeled "Übersicht" containing a registration form. The form has three input fields: "Name" with the placeholder text "Neuer Benutzer", "Passwort", and "Bestätige Passwort", all filled with dots. A green "Registrieren" button is at the bottom of the form. The footer of the page reads "© 2014 - StudMap".

2. Button *Registrieren* betätigen.
3. Datenbankadministrator muss den **Neuen Benutzer mit Administrator Rechten** **versehen**.

### E.2. Login / Logout

1. Aufruf der Webseite: <http://193.175.199.115/StudMapAdmin/Account/Login>.
2. Benutzername und Passwort eingeben.
3. Button *Anmelden* betätigen.
4. Sie können sich jederzeit über den Button *Abmelden* ausloggen.





### E.3. Passwort ändern

1. Aufruf der Webseite: <http://193.175.199.115/StudMapAdmin/Account/Manage> oder auf eigenen Benutzernamen klicken.
2. Aktuelles Kennwort und gewünschtes neues Kennwort eingeben. Dieses zusätzlich bestätigen.
3. Button *Passwort ändern* betätigen.

### E.4. Neue Map anlegen

1. Aufruf der Webseite: <http://193.175.199.115/StudMapAdmin/Admin/CreateMap> oder auf der Admin Seite auf *anlegen* klicken.
2. Anschließend Map-Namen eingeben und bestätigen.
3. Map kann über das Papierkorb-Symbol gelöscht werden.

### E.5. Neuen Floor anlegen

1. Aufruf der Webseite: <http://193.175.199.115/StudMapAdmin/Admin>.
2. Anschließend auf den Map-Namen klicken.
3. Über den Button *Anlegen* gelangen Sie nun zur Seite auf der Sie einen Floor hinzufügen können.
4. Den Eintrag MapId wird automatisch übernommen.
5. Sie geben den Floor-Namen ein und ergänzen einen Link zu Ihrer Kartengrundlage <sup>12</sup>.
6. Abschließend betätigen Sie den Button *Anlegen*.

---

<sup>12</sup>Bevorzugtes Format der Karte sind PDF und PNG Files.

## E.6. Menühandhabung des Layers

1. Aufruf der Webseite: <http://193.175.199.115/StudMapAdmin/Admin>.
2. Anschließend auf den Map-Namen klicken.
3. Dann auf den Floor-Namen klicken.
4. Sie können über das Mausrad in die Karte, bzw. aus der Karte hinaus zoomen und das Kartenwerk verschieben indem Sie die linke Maustaste gedrückt halten und die Maus hin und her bewegen.
5. Des weiteren können Sie folgende Features ausüben, um die Karte mit Meta-Informationen zu ergänzen:
  - Knoten anlegen
  - Kanten anlegen
  - Knoten löschen
  - Knoteninformationen hinterlegen
  - Knoten mit Knoten auf anderem Floor verbinden

### Knoten anlegen

1. Sie befinden sich auf der Layer Webseite und haben das Kartenwerk vor Augen.
2. Sie zoomen an die entsprechende Stelle an der Sie einen Knoten erzeugen möchten mit dem Mausrad.
3. Dann betätigen Sie in Kombination mit der *Strg-Taste* die *linke Maustaste*.
4. Abschließend müssen Sie den Button *Speichern* betätigen, den Sie sehen müssen wenn sie komplett raus gezoomt haben.

### Kanten anlegen

1. Sie befinden sich auf der Layer Webseite und haben das Kartenwerk vor Augen.
2. Zwei Knoten werden mit einander verbunden, indem Sie zuerst einen der beiden Knoten mit einem einfachen Klick mit der *linken Maustaste* markieren.
3. Anschließend navigieren Sie zum zweiten Knoten. Durch einen Klick mit der *linken Maustaste* werden beide Knoten mit einer Kante miteinander verbunden.

4. Speichern Sie Ihr Ergebnis über den entsprechenden Button ab.

## Knoten löschen

1. Sie befinden sich auf der Layer Webseite und haben das Kartenwerk vor Augen.
2. Sie zoomen zu den Knoten, den es zu entfernen gilt.
3. Markieren Sie den Knoten mit der *linken Maustaste*.
4. Betätigen Sie die Taste *Entf* auf Ihrer Tastatur. Der Knoten und anliegende Kanten sind entfernt worden.
5. Vergessen Sie nicht den aktuell vorliegenden Graphen zu speichern.

## Knoteninformationen hinterlegen

1. Sie befinden sich auf der Layer Webseite und haben das Kartenwerk vor Augen.
2. Durch einen einfachen Klick mit der *rechten Maustaste* erhalten Sie gezieltere Informationen zum Knoten.

**Knoteninformationen (1030)**

**Knoteninformationen**

DisplayName: Prof. Dr. Martin Schul

RoomName: B4.0.06

QR-Code: {"General": {"RoomNa

NFC Tag:

Pol Typ: Büro Dozent

Pol Beschreibung: Tel.: 02871 2155-822  
e-Mail: martin.schulzen@w-hs.de

**Knoten**

Speichern Abbrechen

3. Sie können die den Knoten um Informationen erweitern, indem Sie entsprechende Vermerke in den Textboxen vornehmen.
  - Display Name: Sprechender Name des Raumes
  - RoomName: Einmalige Raumbezeichnung
  - QR-Code: Von uns automatisierter QR-Code für diesen Raum
  - NFC-Tag: NFC-Tag ID für diesen Raum

- PoI-Type: Art des Raumes, z.B. Labor
  - PoI-Beschreibung: Informationen die diesen Raum genauer beschreiben oder die Sie diesem Knoten zusätzlich hinterlegen wollen.
4. Die Knoten Id und die genauen Koordinaten des Knotens, in Prozent, werden angezeigt nachdem Sie auf das Stichwort **Knoten** mit der Maus navigieren.
  5. Info: Ergänzen Sie einen Knoten um Informationen erst, nachdem Sie den Graphen gespeichert haben. Ansonsten gehen Ihre Eingaben verloren.

### Knoten mit Knoten auf anderem Floor verbinden

1. Sie befinden sich auf der Layer Webseite und haben das Kartenwerk vor Augen.
2. Zuerst benötigen Sie die Knoten Id's, mit den Sie Ihren Knoten verbinden möchten. Also müssen Sie sich möglicherweise die Id eines Knotens einer anderen Floor notieren.
3. Betätigen Sie die *Shift-Taste + Linke Maustaste* auf den Knoten:

4. Tragen Sie die Id ein.
5. Drücken Sie den Button *Verbinden*
6. Abschließend betätigen Sie den Button *Speichern* und zwei Knoten wurden über diesen Weg miteinander verbunden.

## F. Client Bedienungsanleitung

Die Applikation startet mit der Anzeige der untersten Ebene der geladenen Karte (hier Ebene0).

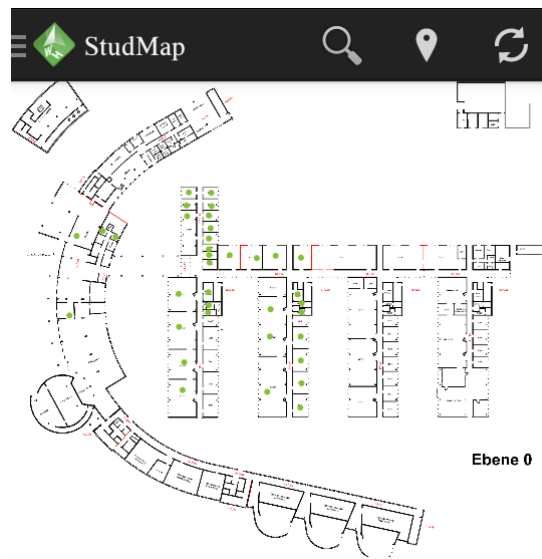


Abbildung F.1.: Ansicht der App nach erfolgreichem Start

### F.1. Karte

Die Karte wird über verschiedene Touch-Funktionen bedient. So kann mit der bekannten „Pitch and Zoom Geste“ in die Karte hinein gezoomt werden.



Abbildung F.2.: Vergrößerte Ansicht von Ebene 1

### **F.1.1. Start-/Zielpunkt wählen**

Um einen Punkt als Start- oder Zielpunkt zu markieren öffnet sich nach der Auswahl eines Punktes ein Dialog mit entsprechenden Auswahlmöglichkeiten.



Abbildung F.3.: Vergrößerte Ansicht des Dialogs zur Entscheidung Start- oder Zielpunkt

Markierte Punkte werden zunächst auch vergrößert, rot dargestellt. Der eingeblendete Dialog am Boden der Karte ist jedoch leicht verändert.

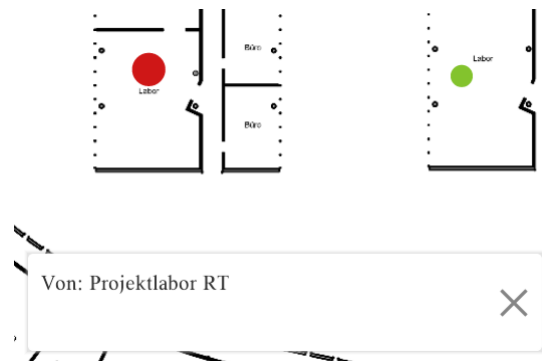


Abbildung F.4.: Vergrößerte Ansicht des Dialogs für einen Startpunkt

### F.1.2. Navigation

Wurde ein Start- oder Zielpunkt gewählt (siehe [Start-/Zielpunkt wählen](#)), wird der nächste gewählte Punkt automatisch zum Ziel-, bzw. Startpunkt, und eine Route eingezeichnet. Start und Ziel bekommen jeweils ein ansprechendes Symbol und unten im Bild ist eine Einblendung über Start und Ziel.

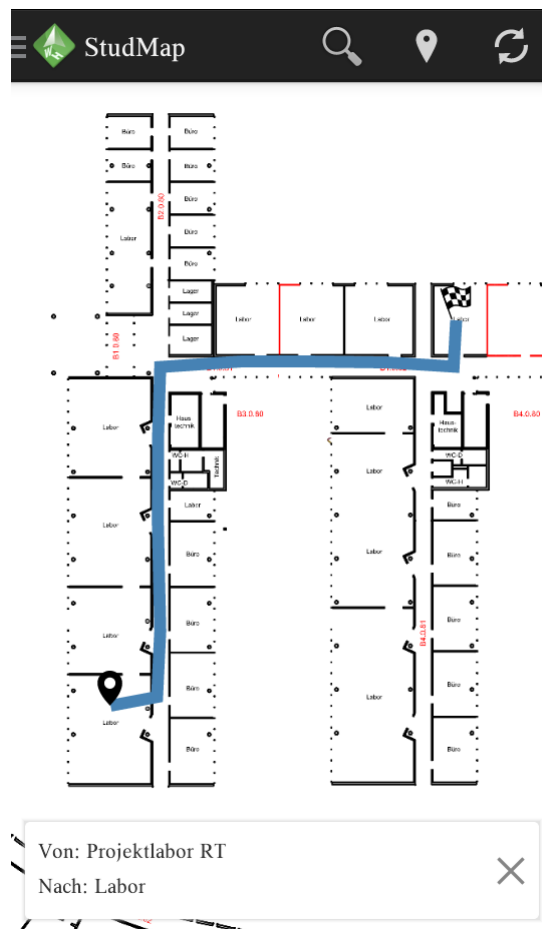


Abbildung F.5.: Ansicht der Karte mit eingeblendeter Route

## F.2. ActionBar

Die Icon in der ActionBar erlauben den Zugriff auf die Suche, die Positionierung und das neu laden der Karte.



Abbildung F.6.: Vergrößerte Ansicht der ActionBar

### F.2.1. Suche

Bei der Suchfunktion kann über ein Suchfeld, welches sich selbst vervollständigt (F.7, rechtes Bild), ein gesuchter Raum gefunden werden.



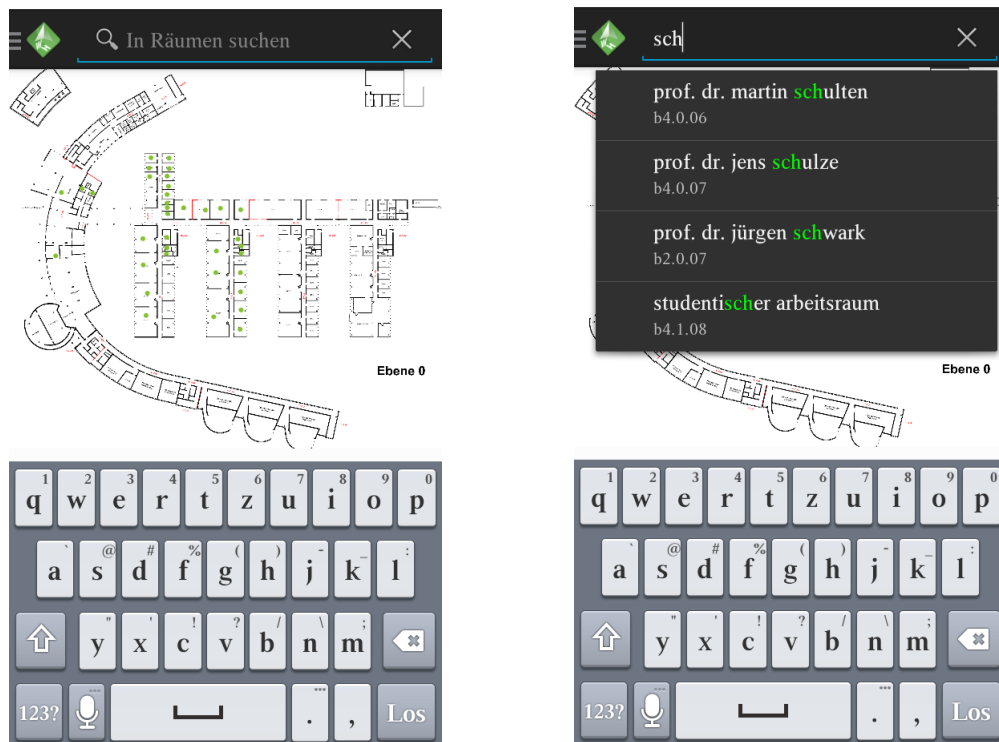


Abbildung F.7.: Ansicht der Suche und der AutoComplete-Funktion

Wird ein Punkt durch die Suche, oder auch den PoI-Dialog (siehe [Point of Interest](#)) markiert, so wird dieser leicht vergrößert und rot dargestellt. Im unteren Bildschirm wird zusätzlich ein Dialog mit den Knoteninformationen eingeblendet.

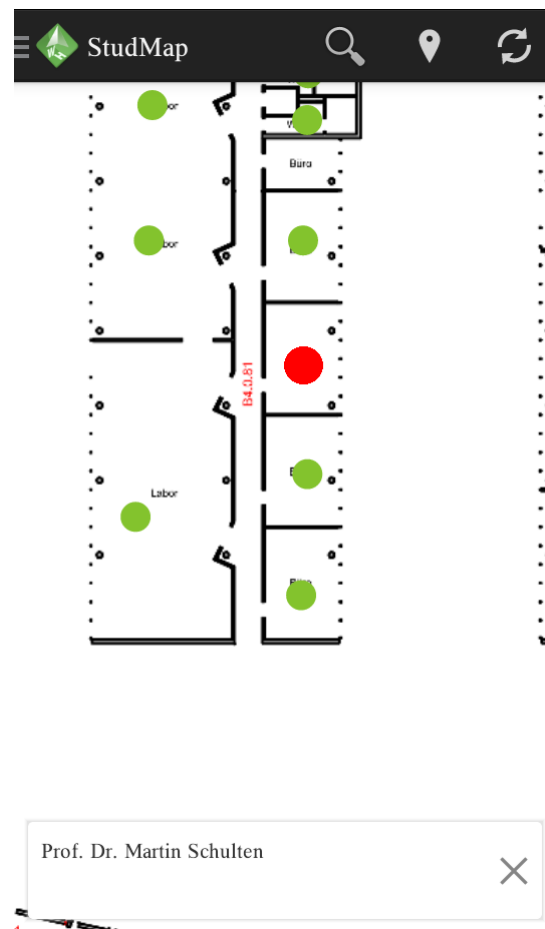


Abbildung F.8.: Ansicht der Karte mit einem Markierten Raum

### **F.2.2. Positionierung**

Die Positionierung kann auf zwei Arten erfolgen. Zum einen kann ein NFC-Tag eingescannt werden, dies erfolgt im Hintergrund und es ist keine Benutzeraktion notwendig. Zum anderen kann ein QR-Tag eingescannt werden, wozu zunächst über das entsprechende Icon der Scanner gestartet werden muss.

### **F.2.3. Neu laden**

Sollte die Karte einmal nicht erreichbar sein oder nur fehlerhaft geladen werden, so kann hier das Laden der Karte manuell gestartet werden.

## **F.3. Menü**

Über einen Wisch vom linken Rand nach rechts lässt sich das Menü aufrufen.

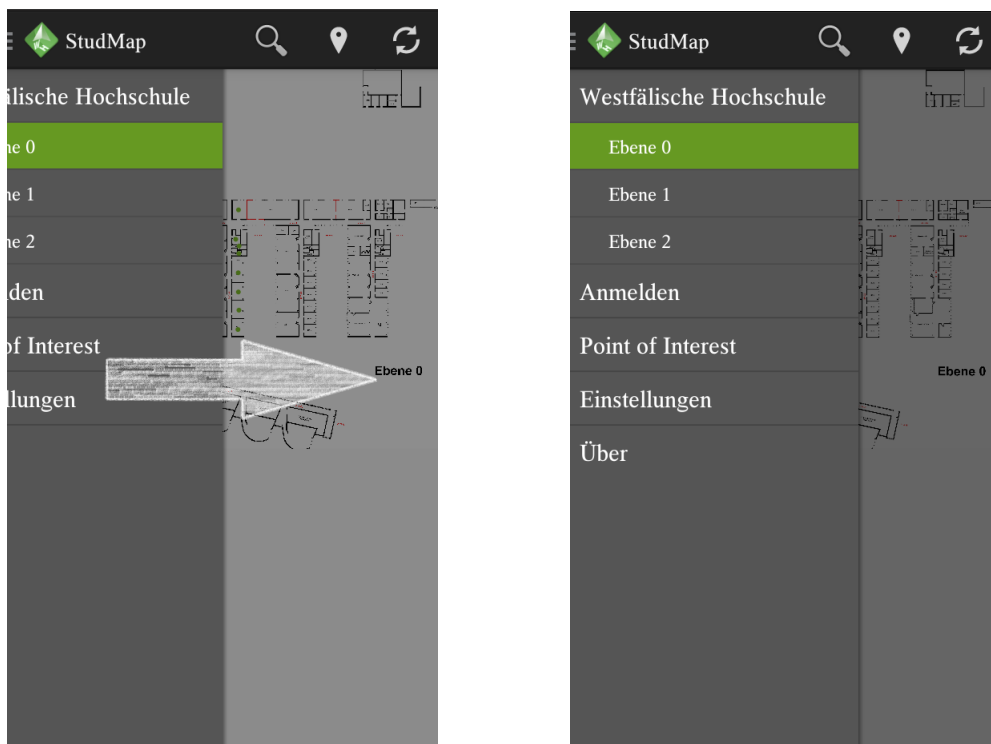


Abbildung F.9.: Ansicht des Menüs

### F.3.1. Ebene auswählen

Hier lässt sich die anzuzeigende Ebene auswählen. Die momentan aktive Ebene wird hellgrün markiert.

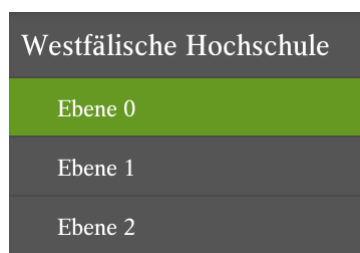


Abbildung F.10.: Ausschnitt der Auswahl für die Ebenen

### F.3.2. Anmelden

In diesem Dialog kann sich der User anmelden erstmalig registrieren.

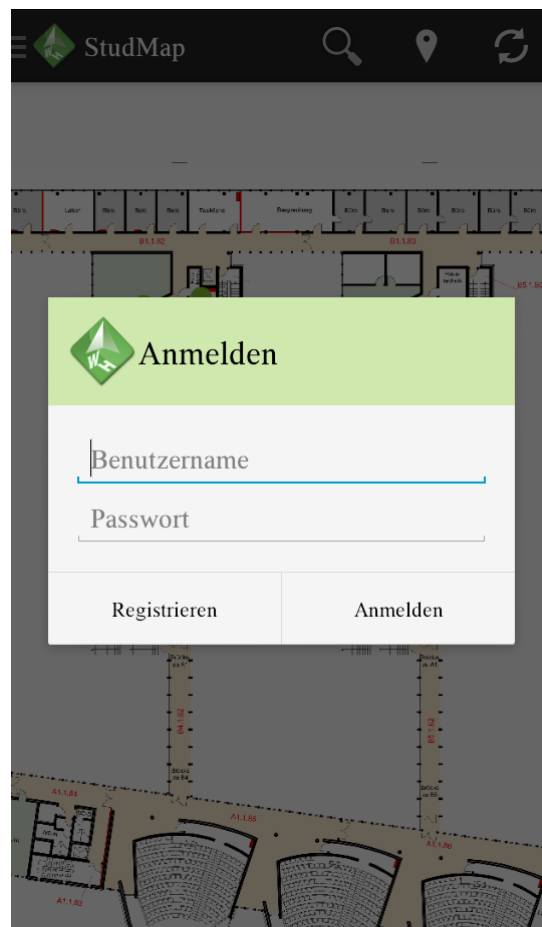


Abbildung F.11.: Ansicht des Dialogs für die Anmeldung

### **F.3.3. Point of Interest**

Der Dialog gibt einen Überblick über interessante Punkte. Über ein Suchfeld lässt sich die Auswahl beschränken.

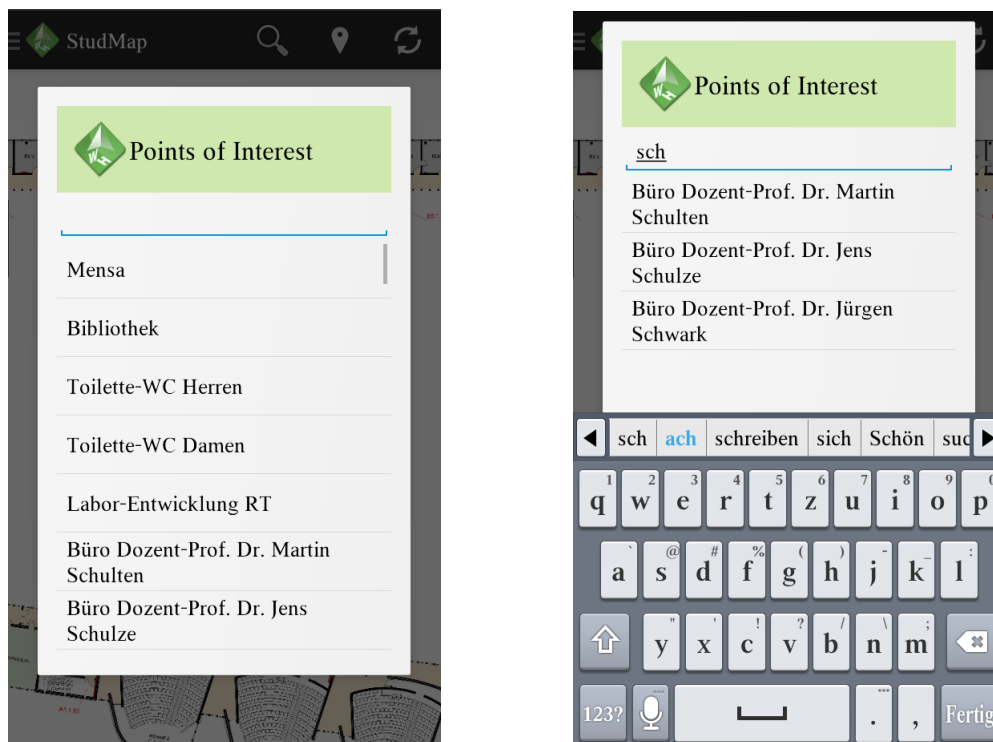


Abbildung F.12.: Ansicht des Point of Interest Dialog

### F.3.4. Einstellungen

In den Einstellungen können die anzuzeigende Karte und die IP-Adresse des Host eingestellt, bzw. geändert, werden.



Abbildung F.13.: Ansicht der Einstellungen

### F.3.5. About

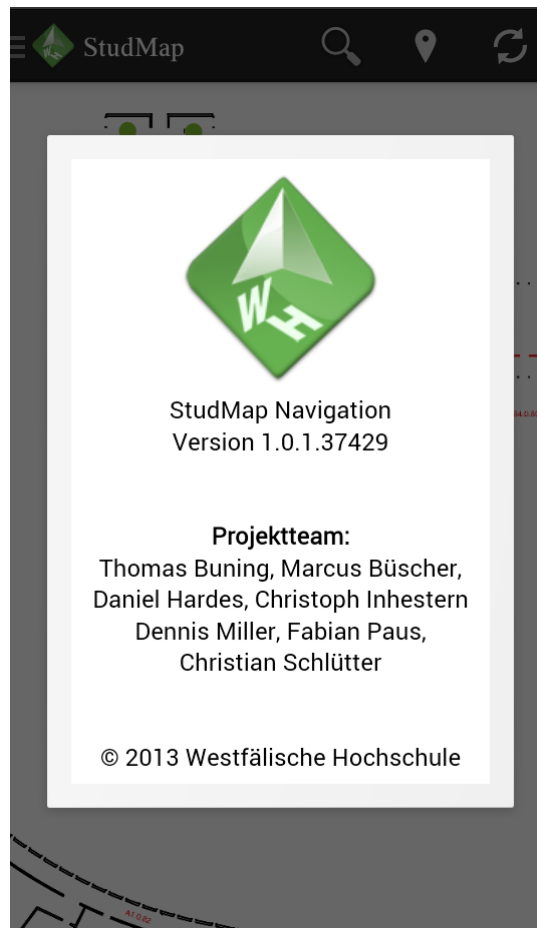


Abbildung F.14.: Ansicht des About Dialogs

## G. Javascript Schnittstelle

Die Javascript Schnittstelle der Karte bietet mehrere Funktionen zur Steuerung. Darunter Methoden zur Veränderung von Punkten, Setzen von Start- und Endpunkten, Zoomen zu Punkten und das Zurücksetzen der Karte.

Im folgenden werden diese hier beschrieben.

### G.1. **setStartPoint(nodeId)**

Setzt einen Startpunkt auf die Karte und markiert diesen in blau. Sind Start- und Endpunkt gesetzt, wird der Weg angezeigt. Eigenschaften:

nodeId	ID des Knotens der Kante.
--------	---------------------------

### G.2. **setEndPoint(nodeId)**

Setzt einen Endpunkt auf die Karte und markiert diesen in rot. Sind Start- und Endpunkt gesetzt, wird der Weg angezeigt.

Eigenschaften:

nodeId	ID des Knotens der Kante.
--------	---------------------------

### G.3. **highlightPoint(nodeId, radius)**

Highlighted den übergebenen Knoten und verändert den Radius.

Eigenschaften:

nodeId	ID des Knotens der Kante.
radius	Radius den der Knoten haben soll.

### G.4. **clearMap()**

Setzt alle Elemente auf der Karte zurück.



## **G.5. resetMap()**

Setzt alle Elemente auf der Karte und die Start-und-Endpunkte zurück.

## **G.6. resetZoom()**

Zoomt die Karte auf Bildschirmfüllende Größe.

## **G.7. zoomToNode(nodeId)**

Zoomt zum übergebenen Knoten.

Eigenschaften:

nodeId	ID des Knotens der Kante.
--------	---------------------------